Towards Modular Controlled Text Generation: Attribute-Specialised PEFT Modules and Comparative Evaluation

Michela Lorandi

BSc, MSc.

A dissertation submitted in fulfilment of the requirements for the award of Doctor of Philosophy (PhD)

to the



School of Computing, Dublin City University

Supervisors:

Prof. Anya Belz

Prof. Sarah Jane Delany - TU Dublin

August, 2026

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of *Doctor of Philosophy (PhD)* is entirely my own work, and that I have exercised reasonable care to ensure that the work is original and has conformed to the regulations on the use and declaration of Generative AI, and does not, to the best of my knowledge, breach any law of copyright, and has not been taken from the work of others, save to the extent that such work has been cited and acknowledged within the text of my work.

Signed: **Michela Lorandi**

ID No.: 22269312

Date: August 26, 2026

Generative AI Declaration

Generative AI (GenAI) tools were used in this thesis for: (i) proofreading, grammar checking, and paraphrasing; and (ii) text generation as part of the experimental methodology (see Chapters 4, 5 and 6 and Section 2.5).

Tools (T): For proofreading and paraphrasing, we used the DCU-whitelisted tool Google Gemini. For experimental purposes, multiple open-source large language models (LLMs) were evaluated, including LLaMA 3 8B, LLaMA 3.1 8B, LLaMA 3.2 3B, LLaMA 3.3 70B, Mistral 7B, Falcon 40B (details and version references are provided within the relevant chapters), as well as closed-source LLMs, including GPT-3.5 and GPT-4.

Outputs (O): Proofreading outputs consisted of grammar suggestions, spelling corrections, and paraphrased text. These were manually reviewed, edited, and partially discarded to preserve authorial intent and voice. In the experimental methodology, GenAI models were used both to generate controlled text outputs and to evaluate the effectiveness of the proposed control mechanisms (details are provided within the relevant chapters).

Prompts (P): Proofreading prompts were short task-specific queries such as "*Check grammar and suggest improvements while preserving meaning*". Experimental prompts followed standardised task definitions and templates as described in the relevant chapters.

Iterations (I): Proofreading outputs were typically refined through one to two iterative edits using human judgement. Experimental text generation involved multiple iterations across different model configurations, datasets, and hyperparameter settings (details appear within the relevant chapters).

Critical Reflection (C): All GenAI outputs were critically reviewed for accuracy, suitability, and consistency with the intended meaning. GenAI tools did not contribute conceptual content, analytical reasoning, or interpretation. All synthesis and argumentation remain the author's own. The use of GenAI is consistent with DCU's academic integrity guidelines.

Acknowledgements

First, I would like to thank my supervisor Prof. Anya Belz. Your guidance was invaluable and I feel extremely grateful to have found a supervisor who trusted me to find my own path while always knowing when to step in with the right question at the right moment.

To my lovely husband, Michele, thank you for always being by my side in every situation, thank you for being my rock especially in the difficult moments. Thank you for always make me laugh and simply walk trough life with me.

To my parents, Loredana and Renato, without you nothing of this would have been possible, thank you for always believing in me. To my sisters, Erika and Linda, and to everyone in my big messy family, I love you all, thank you for being patient with me and always been there for a chat.

To Sara, thank you for always listening to me, to celebrate with me, to cry and laugh together, your support was priceless and I'm deeply grateful I found you.

I am grateful to all my co-authors for the possibility to collaborate on amazing projects. To my amazing lab mates and friends, Kanishk, Charlie, Tytti, Mohammed, Simon, Craig and Joao, for the technical discussions and sharing knowledge, but even more for the long talks, the nights out and to always be there in all ups and downs.

To Ale, Z and NDR, our online coffee nights were priceless breaks, thank you for being there even for just a laugh.

Finally, to everyone I didn't name but was part of this journey. I am very grateful for this journey and for everyone I met, for every lesson learned and all the support received.

Contents

1 Introduction **2**
- 1.1 Thesis Topic . . . 2
- 1.2 Research Questions . . . 7
- 1.3 Formal Foundations . . . 11
  - 1.3.1 Language Modelling . . . 11
  - 1.3.2 Text Generation . . . 11
  - 1.3.3 Controlled Text Generation . . . 12
  - 1.3.4 Single-Attribute and Multiple-Attribute Control . . . 12
  - 1.3.5 Control Evaluation . . . 13
  - 1.3.6 Notation Summary . . . 15
- 1.4 Thesis Structure . . . 15
- 1.5 Publications . . . 17

2 A Categorisation Scheme for Controlled Text Generation Techniques **20**
- 2.1 Chapter Overview . . . 20
- 2.2 Motivation . . . 22
- 2.3 CTG Categorisation Scheme . . . 24
  - 2.3.1 Preliminaries . . . 24
  - 2.3.2 Dimension 1: Control Attributes . . . 26
  - 2.3.3 Dimension 2: Control Integration . . . 29
  - 2.3.4 Dimension 3: Control Implementation . . . 31
- 2.4 Development of the Categorisation Scheme . . . 34
  - 2.4.1 Paper Selection Process . . . 35
  - 2.4.2 Manual Annotation Procedure . . . 37
- 2.5 Automatic Annotation of CTG Papers . . . 39
  - 2.5.1 Task Definition and Formulation . . . 42
  - 2.5.2 Methodology . . . 45
  - 2.5.3 Experimental setup . . . 57
  - 2.5.4 Main Results . . . 60
  - 2.5.5 Ablation Studies . . . 62
  - 2.5.6 Manual Evaluation . . . 65
  - 2.5.7 Automated Annotation of 2024–2025 Papers . . . 67
  - 2.5.8 Limitations . . . 78
- 2.6 Dataset Statistics and Composition . . . 79
  - 2.6.1 Temporal Distribution and Control Type . . . 80
  - 2.6.2 Control attributes . . . 82
  - 2.6.3 Control Integration Strategies . . . 86

2.6.4 Control Implementation Methods . . . 88
2.6.5 Multi-dimensional Analysis . . . 91
2.6.6 Model Architectures, Evaluation Datasets and Evaluation Metrics . . . 93
2.7 Discussion and Framework Utility . . . 99
2.8 Chapter Summary . . . 101

3 Related Research 102
3.1 Surveys on Controlled Text Generation . . . 102
3.2 Controlled Text Generation Techniques . . . 105
3.2.1 Training-Free Implementation Methods . . . 106
3.2.2 Training Implementation Methods . . . 112
3.2.3 Hybrid Implementation Methods . . . 117
3.3 Parameter-Efficient Fine-Tuning Techniques . . . 120
3.4 Combining Multiple PEFT Modules . . . 124
3.5 Portability of PEFT Modules . . . 126

4 Evaluation Challenges and Framework for Controlled Text Generation 129
4.1 Background . . . 129
4.2 Chapter Overview . . . 130
4.3 Semantic Consistency Assessment in Data-to-Text Generation . . . 132
4.3.1 Overview and Motivation . . . 133
4.3.2 Task Definition . . . 135
4.3.3 Methodology . . . 135
4.3.4 Experimental Setup . . . 141
4.3.5 Results . . . 142
4.3.6 Discussion and Limitations . . . 144
4.4 Fair Comparison of Controlled Text Generation Techniques . . . 145
4.4.1 Overview and Motivation . . . 145
4.4.2 Comparative Study Design . . . 146
4.4.3 Evaluated Controlled Text Generation Techniques . . . 159
4.4.4 Experimental Setup . . . 163
4.4.5 Evaluation Metrics Analysis . . . 164
4.4.6 Results . . . 173
4.4.7 Discussion and Limitations . . . 187
4.5 Chapter Summary . . . 189

5 Enable Content Control with Data-to-Text Generation 191
5.1 Background . . . 191
5.2 Chapter Overview . . . 193
5.3 Out-of-the-Box Pre-trained LLMs for Data-to-Text Generation . . . 197
5.3.1 WebNLG Shared Tasks Overview . . . 198
5.3.2 System Description . . . 199
5.3.3 Experimental Setup . . . 203
5.3.4 Results . . . 208
5.3.5 Discussion . . . 217
5.3.6 Limitations and Motivation for PEFT . . . 219
5.4 Data-to-Text Generation using PEFT Techniques . . . 220

5.4.1 Data . . . 222
5.4.2 System Description . . . 223
5.4.3 Experimental Setup . . . 229
5.4.4 Results . . . 236
5.4.5 Discussion . . . 247
5.5 Chapter Summary . . . 251

6 Attribute-specialised PEFT modules for Controlled Text Generation 255
6.1 Background . . . 255
6.2 Chapter Overview . . . 256
6.3 Experimental Setup . . . 259
6.3.1 Datasets . . . 259
6.3.2 Task Construal . . . 262
6.3.3 Evaluation Metrics . . . 263
6.4 Training Single-Attribute Specialised Modules . . . 264
6.4.1 Composite dataset construction . . . 266
6.4.2 Plug-and-play QLoRA . . . 267
6.4.3 Models . . . 269
6.4.4 Experimental Setup . . . 270
6.4.5 Results . . . 271
6.4.6 Summary of Single-Attribute Modules Training . . . 278
6.5 Multiple-Attribute Control via Module Composition . . . 279
6.5.1 Composition Techniques . . . 281
6.5.2 Study Overview . . . 284
6.5.3 Experimental Setup . . . 285
6.5.4 Results . . . 287
6.5.5 Validation of Control Effectiveness Metric for Topic Control . 298
6.5.6 Limitations . . . 300
6.5.7 Summary of Module Composition . . . 301
6.6 Cross-Model Module Portability . . . 302
6.6.1 Study Overview . . . 303
6.6.2 Portability Framework . . . 305
6.6.3 Experimental Setup . . . 309
6.6.4 Results . . . 311
6.6.5 Limitations . . . 318
6.6.6 Summary of Cross-Model Portability Experiments . . . 319
6.7 Chapter Summary . . . 320

7 Conclusion 323
7.1 Contributions . . . 323
7.2 Research Questions Revisited . . . 327
7.3 Limitations . . . 331
7.4 Future Work . . . 333
7.5 Concluding Remarks . . . 335

A Appendix of the Categorisation Scheme for CTG 366
A.1 Automatic Annotation of CTG Papers . . . 366
A.1.1 Hyperparameters . . . 366
A.1.2 CTG Papers Prompts and Questions Definition . . . 367
A.1.3 Evaluation Papers Prompts and Questions Definition . . . 368
A.2 QCET Evaluation Methodology Annotation . . . 369
A.2.1 Task Description . . . 369
A.2.2 Evaluation categorisation scheme . . . 370
A.2.3 Data and Experimental Setup . . . 372
A.2.4 Prompts and questions definition . . . 375
A.2.5 Results . . . 376

B A Fair Comparison of Controlled Text Generation Techniques Appendix 384
B.1 LLMs Prompts . . . 384
B.2 Hyperparameters . . . 384
B.2.1 CTG Techniques . . . 384
B.2.2 Evaluation Metrics . . . 391
B.3 Scientific artifacts and licensing . . . 391

C Enable Content Control with Data-to-Text Generation Appendix 393
C.1 Out-of-the-Box LLMs for Data-to-Text Generation . . . 393
C.1.1 Hyperparameters and APIs . . . 393
C.1.2 Prompts . . . 394
C.2 Data-to-Text Generation using PEFT Techniques . . . 395
C.2.1 Prompts . . . 395
C.2.2 Hyperparameters . . . 395

D Appendix of Attribute-specialised PEFT modules for CTG 401
D.1 Design of Single-Attribute Specialised Modules . . . 401
D.1.1 Hyperparameters . . . 401
D.2 Scientific artifacts and licensing . . . 404

List of Figures

2.1 Taxonomy of **Control Attribute Types** in CTG systems. The hierarchy distinguishes between Content Control (*what* is said) and Non-Content Control (*how* it is said). Each control attribute is accompanied by example control signals and attribute values observed across the analysed papers. 27

2.2 Architectural diagram of control integration levels in CTG systems. Control signals (shown with dashed arrows) indicate where control values enter the system. Different integration strategies can be combined within a single system. 29

2.3 Taxonomy of control implementation methods in CTG systems, organised by training requirements. Methods range from training-free approaches (***LM Prompting***, ***Text Selection***, ***Modification of Token Distribution***) through training-based methods (***Fine-Tuning***, ***Complete Training***) to ***Hybrid*** approaches that combine multiple methods. 32

2.4 Paper selection process for all papers analysed in our systematic survey. The consistent process across all time periods ensures comparable coverage and selection criteria. 36

2.5 Task formulation for automatic paper categorisation. The model receives paper text (x), a property-value question (q), and task description (t) as input, and generates a reasoning trace (r) explaining the decision process followed by structured annotations (a) in JSON format containing categorisation labels for all fields. 44

2.6 Example of a generated reasoning trace in the training set using LLaMa3.3 70B Instruct. 48

2.7 Prompt used to generate reasoning traces with LLaMa 3.3 70B Instruct. Placeholders are shown in {braces}. 49

2.8 Overview of the automatic annotation pipeline. (1) Input preprocessing converts PDFs to cleaned text via Docling and text cleaning (see 2.5.2.1 Dataset Creation). (2) Synthetic reasoning traces are generated using LLaMA 3.3 70B Instruct and validated against gold annotations (see 2.5.2.1 Dataset Creation). (3) Training proceeds in two phases: supervised fine-tuning (SFT) with or without reasoning traces (see 2.5.2.3 SFT with long CoT), followed by reinforcement learning (RL) refinement using GRPO with DAPO loss (see 2.5.2.4 RL for structured annotation generation). The best SFT checkpoint is selected for RL training, and the best RL checkpoint is used for inference. (4) The final model generates reasoning traces and structured annotations for new papers at inference time. All training uses Qwen3 8B with LoRA (r=32, α=64). 51

2.9 Annotation comparison for the same paper. Gold annotation (left), our best multi-field model (middle: Qwen3 8B SFT+RL with EM+F+JSON rewards), and LLaMa3.3 70B Instruct baseline (right). 67

2.10 Temporal distribution of papers in our dataset (N=318). 81

2.11 Distribution of single-attribute versus multiple-attribute control papers. 82

2.12 Distribution of control attribute categories. 83

2.13 Control attributes usage across CTG papers. 85

2.14 Control integration usage across CTG papers. 87

2.15 Distribution of Control Integration strategies across all papers. The horizontal bar chart (left) shows the frequency of each individual method; the matrix below indicates which methods co-occur; the vertical bar chart (top) shows the total number of papers for each combination. 88

2.16 Control implementation usage across CTG papers. 89

2.17 Distribution of Control Implementation methods across all papers. The bar chart (left) shows the frequency of each individual method; the matrix (bottom) indicates which methods appear in combination; the bar chart (top) shows the size of each combination. 91

2.18 Control Implementation x Integration patterns for single-attribute versus multiple-attribute control systems. Cell values show percentage of papers within each control type. 92

2.19 Model architecture usage across CTG papers showing the overall top 15 most frequently used models. 94

2.20 Automatic evaluation metrics usage across CTG papers showing the frequency of the top 15 most used evaluation metrics. 96

2.21 Evaluation datasets usage across CTG papers showing the frequency of the top 15 most used evaluation datasets. 97

3.1 Taxonomy of parameter-efficient fine-tuning methods from Lialin et al. (2023), organised into three broad categories: addition-based (left), reparametrisation-based (right), and selection-based (top right). Methods appearing in overlapping regions combine characteristics of multiple categories. 120

3.2 Placement of different PEFT modules within the Transformer architecture from Sabry and Belz (2023). Addition-based methods (Adapters, Compacters, Tiny-Attention Adapters, Prefix Tuning, Prompt Tuning) introduce new components either sequentially after Transformer sub-layers or in parallel with attention projections. Reparametrisation-based methods (LoRA) are inserted in parallel with the query and value projection matrices within the attention block. 122

4.1 CTG techniques selection process from ACL Anthology to which we added 2 open-source LLMs and 2 relevant techniques not in the ACL Anthology. 148
4.2 Diagrammatic view of where in the model architecture the control is injected at inference time, for each of our 12 CTG techniques evaluated. 160
4.3 Correlation matrix between classifiers used in sentiment and topic control. Darker color means less correlation. 168
4.4 Evaluation results for Sentiment control showing the average system-level Average Control Effectiveness for each combination of dataset and control attribute values (positive/negative). Standard deviation depicted as error bars. 174
4.5 Evaluation results for Topic control showing the average Control Effectiveness across the 4 used datasets and control attribute values. Standard deviation depicted as error bars. 176
4.6 Evaluation results for Keywords control showing the average Control Effectiveness across the 4 used datasets and control attribute values. Standard deviation depicted as error bars. 179
4.7 Evaluation results for Multiple-attribute control showing the average Control Effectiveness across the 4 used datasets and control attribute values. Standard deviation reported with error bars. 182

5.1 Example in English from WebNLG 2020 dataset showing input structured triples and corresponding reference output. The system must verbalise all information from the structured input while producing grammatically correct and coherent text. 192
5.2 Categorisation of the two system types investigated in Chapter 5 according to the three-dimensional framework established in Chapter 2. **Dimension 1 (top):** Both systems use ***multiple-attribute control***, coordinating ***content control*** (via structured RDF triples specifying knowledge elements) with ***non-content control*** (via language labels). **Dimensions 2 and 3 (middle and bottom):** The systems share the same control integration strategy (***input addition***) but differ in control implementation, with Section 5.3 using ***training-free LM prompting*** and Section 5.4 implementing ***training-based PEFT fine-tuning***. 194
5.3 Two generation strategies for data-to-text generation in under-resourced languages. (**top**) *Direct Generation*: The model generates text directly in the target language from the input prompt. (**bottom**) *Generate-then-Translate*: The model first generates English text from the input prompt, which is then translated to the target language using a machine translation system. 200

5.4 The two prompt types used in the final systems. Input triples are highlighted in yellow, model output in blue. The *zero-shot minimal instruction prompt* (**left**) consists of a brief task description followed by the input triples. The *few-shot in-context learning prompt* (**right**) additionally incorporates three example input-output pairs (highlighted in green) before the test input. 201
5.5 Screenshot of the human evaluation interface. 207
5.6 The two systems used in the final evaluation with input and output structure examples. Given Input (triples) highlighted in yellow, model output in blue. The few-shot in-context prompt also incorporates examples (highlighted green). 224

6.1 Categorisation of the approach presented in this chapter according to the scheme introduced in Chapter 2. Dimension 1 covers the control attributes targeted (***topic*** and ***sentiment***); Dimensions 2 and 3 cover the control integration (***input addition***) and control implementation (***fine-tuning*** via LoRA/QLoRA) strategies respectively. . 259
6.2 A single QLoRA block (orange) shown attached to its corresponding model weights. 268
6.3 Diagram of our three QLoRA module composition techniques: (Top) Element-wise Weights Average; (Bottom-left) Element-wise Outputs Summing; (Bottom-right) Element-wise Outputs Averaging. 281

A.1 Definition of the multi-field question for CTG categorisation. The question defines all 4 fields for the CTG categorisation: *Control Type*, *Control Attributes*, *Control Implementation*, and *Control Integration*. 379
A.2 Prompt used for training and inference of CTG papers annotation. Placeholders are shown in {braces}. 380
A.3 Example of generated output using our best model (multi-field SFT+RL with EM+F+JSON rewards). The output contains the reasoning trace and the final answer expressed as JSON object. 381
A.4 Definition of the measure-level evaluation type question for evaluation categorisation. 382
A.5 Definition of the measure-level QCET node question for evaluation categorisation. The question defines all 3 fields for the QCET node categorisation: QCET node level 1, QCET node level 2, and QCET node level 3. 383

List of Tables

2.1 Characteristics of annotation tasks. **Field**: The annotation dimension being classified. **Class**: Single=single-label (one value per instance), Multi= multi-label (multiple values per instance). **Label**: Binary=2 possible values, Multi=3+ possible values. **Set**: Closed=fixed taxonomy, Open=unbounded vocabulary. 44

2.2 Dataset statistics for the CTG annotation task. Paper text and multi-field reasoning trace lengths are measured in tokens using the Qwen3 8B tokenizer. Test set reasoning traces are not used (model generates them during inference). 50

2.3 Performance on CTG system annotation across four dimensions: Control Type (binary classification), Control Attributes, Control Integration, and Control Implementation (multi-label). Results are averaged over 5 runs. **Bold** indicates best result, *italics* second-best. R=reasoning trace; EM=Exact Match; J=Jaccard score (intersection-over-union). 61

2.4 Ablation study on reward components for RL training. We compare training with RL only (from base model) vs. SFT+RL (from SFT checkpoint), using different reward combinations: Exact Match (EM) vs. Jaccard (J) for correctness, with optional Format (F) and JSON format rewards. # Avg Reason=average reasoning trace length. Results averaged over 5 runs. **Bold** indicates best results; *italic*: second-best. 63

2.5 Comparison of single-field vs. multi-field models on CTG annotation. Single-field trains four models (one per dimension), requiring four sequential inferences. Multi-field predicts all dimensions jointly in a single forward pass. Time=average inference time per paper (lower is better). Results averaged over 5 runs. **Bold** = best result. 65

2.6 Manual evaluation results on 30 unseen papers from 2024 using our best model (SFT+RL with EM+F+JSON). Partial=some labels correct. No partial results for Control Type, as it is binary classification. 67

2.7 Annotation accuracy on 2024–2025 papers. Exact Match is strict accuracy (100% correct predictions). F1-score applies to multi-value fields only; Control Type is a single-value binary field. 72

2.8 Error type breakdown for multi-value fields. Omission errors occur when the model predicts a subset of correct values. Hallucination errors occur when the model predicts values not in the reference annotation. Mixed errors contain both. 73

2.9 Most commonly hallucinated values (top 3 per field). Hallucinations occur when the model predicts a value not present in the gold annotation. 75
2.10 Most commonly omitted values (top 3 per field). Omissions occur when the model fails to predict a value present in the gold annotation. 76
2.11 Top 5 most frequently used models by year. Numbers in parentheses indicate paper count. 94
2.12 Top 5 most frequently used automatic evaluation metrics by year. Numbers in parentheses indicate paper count. 96
2.13 Top 5 most frequently used evaluation datasets by year. Numbers in parentheses indicate paper count. 98

4.1 Pearson's r for Datasets A and B, sorted (within Automatic Metrics and LLM-as-Judge) by Dataset A's r (+Textification). Top three Pearson's r in each group bolded. 143
4.2 Overview of the selected CTG techniques. FT=Free Text Generation, C=Control, Si=Single-attribute control, M=Multiple-attribute control, S=Sentiment, T=Topic, K=Keywords, C Intr=Control Integration, I Add=Input Addition, S Add=System Addition, M Inj=Model Injection. 149
4.3 Number of samples for each dataset used for the execution of the experiments. 154
4.4 Input format for each CTG technique. 156
4.5 Control attributes and respective values used in the experiments. . . 156
4.6 Mapping to translate the used topic values into topic values available in each CTG technique. Our Topic value → Technique Topic value. A dash (—) indicates that the technique does not support the corresponding topic and was therefore not evaluated on that control value. 158
4.7 Postprocessing for each CTG technique. 159
4.8 Accuracy, F1, Precision and Recall of different sentiment and topic classifiers on the Yelp polarity and AGnews datasets. 165
4.9 Comparison between Control Effectiveness computed as the Average of the used classifiers and Majority Voting between the classifiers for Sentiment (Pearson's r = 0.99 with Spearman's ρ 0.98). Techniques are grouped by control implementation category. Standard deviation in round brackets. System rank in square brackets. 166
4.10 Comparison between Control Effectiveness computed as Average of the used classifiers and Majority Voting between the classifiers for Topic (Pearson's r = 0.98 with Spearman's ρ 0.97). Standard deviation in round brackets. System rank in square brackets. 167
4.11 Comparison between Perplexity and SLOR for Sentiment control. Standard deviation in round brackets. System rank in square brackets. 170
4.12 Comparison between Perplexity and SLOR for Topic control. Standard deviation in round brackets. System rank in square brackets. . . 171
4.13 Comparison between Perplexity and SLOR for Keywords control. Standard deviation in round brackets. System rank in square brackets. 172

4.14 Comparison between Perplexity and SLOR for Multiple-attribute control. Standard deviation in round brackets. System rank in square brackets. 172

4.15 Evaluation results for Sentiment control. Each row contains the weighted average over all datasets and 3 seeds. The Control Effectiveness columns show the accuracy of each classifier separately and the average between them. Best score per column in bold. Standard deviation in round brackets. ZS=Zero-Shot, FS=Few-Shot, Ov=Overall results averaged across ZS and FS. 174

4.16 Results for Topic control. Control Effectiveness shows the accuracy of the 3 classifiers and the average between them. Numbers are weighted averages over all datasets and 3 seeds. Best score per column in bold. Standard deviation in round brackets. ZS=Zero-Shot, FS=Few-Shot, Ov=Overall results averaged across ZS and FS. 175

4.17 Evaluation results for Keywords control. Control Effectiveness shows the Word Inclusion Coverage considering exact match and lemma match. Each row contains the weighted average over all datasets and 3 seeds. Best score per column in bold. Standard deviation in round brackets. Cov=Word Inclusion Coverage with lemmas extracted with Spacy, ExtCov=Word Inclusion Coverage with lemmas extracted with Spacy and Lemmingflect, ZS=Zero-Shot, FS=Few-Shot, Ov=Overall results averaged across ZS and FS. 178

4.18 Results for Multiple-attribute control. Control Effectiveness is calculated via majority voting between three classifiers. The 'Both' column shows the proportion of outputs where both control attributes are correct. The **Avg CE**$^{(!)}$ column shows the average of the single-attribute CE scores and is included for reference only. Each row contains the weighted average over all datasets and 3 seeds. Best score per column in bold. Standard deviation in round brackets. S=Sentiment, T=Topic, ZS=Zero-Shot, FS=Few-Shot, Ov=Overall results averaged across ZS and FS. 180

4.19 Memory used to store each CTG technique in GB. 183

4.20 Inference execution time in seconds per one sample for each CTG technique using the different control attributes (sentiment, topic, keywords, and multiple-attribute). Each row contains the average over all datasets and 3 seeds with the standard deviation. In bold the fastest technique per column. Cells marked with '–' indicate that the technique does not support the corresponding control attribute and was therefore not executed for that setting. 184

4.21 Side by side comparison of Control Effectiveness scores for sentiment control and topic control systems as assessed in original work (Original) and in LPF where it is the average of three classifier scores (DistilBERT, BERT, DeBERTa for topic; DistilBERT, T5, DeBERTa for sentiment). All scores express the proportion of outputs that are classified as having the property they are intended to have. 185

5.1 Number of training, development, and test samples per language in the WebNLG datasets. English and Russian from WebNLG 2020+ (Ferreira et al., 2020); Breton, Welsh, Irish, and Maltese from WebNLG 2023 (Cripwell et al., 2023). 199
5.2 Automatic evaluation results for **English**. Best score per column in bold, best score per model in italics. Number of model parameters in brackets. COMET scores are between 0 and 1. ZS MI=Zero-Shot Minimal Instruction, FS IC=Few-Shot In Context. 210
5.3 Automatic evaluation results for **Irish** and **Welsh**. Highest score in each column for each language in **bold**, highest score for each model in *italics*. Number of parameters in brackets in column 1. COMET scores are between 0 and 1. M=Model, ZS MI=Zero-Shot Minimal Instruction, FS IC=Few-Shot In Context, GT=Google Translate, AT=Alibaba Translate, NLLB=No Language Left Behind system, COM=COMET. 211
5.4 Automatic evaluation results for **Maltese** and **Breton**. Highest score in each column for each language in bold, highest score for each model in italics. Number of parameters in brackets in column 1. M=Model, ZS MI=Zero-Shot Minimal Instruction, FS IC=Few-Shot In Context, NLLB=No Language Left Behind system, COM=COMET. 212
5.5 Human evaluation results for **English** for human-authored references, GPT-3.5 zero-shot, GPT-3.5 few-shot), and best WebNLG20 system (Amazon AI). Means and homogeneous subsets from Tukey HSD (alpha = .05); a system spanning multiple letters does not differ significantly from either subset. 215
5.6 Mean **WebNLG 2023** human scores for **Welsh**, **Maltese** and **Irish**, per system for Fluency, Addition and Omission. Means and homogeneous subsets from Tukey HSD (alpha = .05); a system spanning multiple letters does not differ significantly from either subset. 215
5.7 Human evaluation results of **English** from **WebNLG 2020**. Means and homogeneous subsets from Tukey HSD (alpha = .05). 217
5.8 Overview of evaluation configurations performed across development (WebNLG validation), internal testing (WebNLG 2020 test), and shared task (GEM 2024 datasets). Human and LLM-as-judge evaluations used the same 180-sample subset with four quality criteria rated on a 7-point scale. 234
5.9 Preliminary automatic evaluation results of our testing phase on the validation set of WebNLG 2020 in English. Best overall system in bold, best prompting system in italics. 238
5.10 Automatic evaluation results comparison between our system and Lorandi and Belz (2023a) best system (GPT-3.5) on the test set of WebNLG 2020 (D2T-1 FA) in English. Best overall system in bold. Few-Shot dynamic = Few-shot prompt with dynamic selection of examples based on length and predicates. 239
5.11 Automatic evaluation results on the English test set of WebNLG 2020 (D2T-1 FA) using all references, comparing the performance of participating systems in the GEM 2024 shared task. Best overall system in bold. 242

5.12 Automatic evaluation results across all 6 GEM 2024 data-to-text test sets (Section 5.4.1): D2T-1 (WebNLG-based) and D2T-2 (Wikidata-based), each with three variants, namely FA (Factual), CFA (Counterfactual), FI (Fictional). Best system per metric in bold, second-best in italic. 243

5.13 Human and LLM evaluation results for DCU-NLG-PBN compared with best-performing team and mean performance across all shared task participants, across languages and criteria (scale 0-7, higher better). Scores computed across all six dataset variants (D2T-1 and D2T-2, each with Factual/Counterfactual/Fictional). Best performing system in bold. 245

6.1 Overview of datasets used in this chapter. Train and test sizes are after sampling (see Section 6.4.1). Avg words are approximate. SST-2 avg words varies between pre- and post-sampling versions; see Table 6.3 for full statistics. The "Used in" column indicates the section(s) in which each dataset is used for training or evaluation. 260

6.2 Mapping of DBPedia topic labels into AG News topic labels 262

6.3 Statistics of the datasets used for each data split (train, validation, test), showing the number of samples, average number of words per text, and average number of characters per text, both before and after dataset sampling. 267

6.4 **Sentiment Control:** Diversity (Distinct-n) and Fluency (SLOR) for the model + QLoRA module. All values are averages over 3 runs. Bold = highest score in column/section. 272

6.5 **Sentiment Control:** Control Effectiveness for the model + QLoRA module. All values are averages over 3 runs. Bold = highest score in column/section; underline = train/test on same dataset. 273

6.6 Diversity (Distinct-n) and Fluency (SLOR) for **Topic Control**, training on *single* and *combined* datasets. All values are averages over 3 runs. Bold = highest score in column and section. 275

6.7 Control Effectiveness for **Topic Control**, training on *single* and *combined* datasets. All values are averages over 3 runs. Bold = highest score in column and section; underline = train and test set from same dataset. 276

6.8 Study structure in terms of raw model and QLoRA (composed) modules. D_i = individual datasets; C_i = composition techniques. For other details, see text. 285

6.9 **Sentiment Control:** Diversity and Fluency for the model + QLoRA module combinations explained in Table 6.8. Here, e.g. Output Summing(data1, data2) refers to the output summation module composition technique. All values are averages over 3 runs. Bold (shaded) = (second) highest score in column/section. 288

6.10 **Sentiment Control:** Control Effectiveness for the model + QLoRA module combinations explained in Table 6.8. Here, e.g. Output Summing(data1, data2) refers to the output summation module composition technique. All values are averages over 3 runs. Bold (shaded) = (second) highest score in column/section; underline = train/test on same dataset. 289
6.11 Sentiment control effectiveness (%) comparing single-dataset training with multi-module output summing. Bold=best; grey=second-best. . 291
6.12 Diversity and Fluency for **Topic Control**, training on *single* and *combined* datasets, and composition of modules trained on single datasets, e.g. Output Summing(data1, data2). All values are averages over 3 runs. Bold (shaded) = (second) highest score in column and section. 293
6.13 Control Effectiveness for **Topic Control**, training on *single* and *combined* datasets, and composition of modules trained on single datasets, e.g. Output Summing(data1, data2). All values are averages over 3 runs. Bold (shaded) = (second) highest score in column and section; underline = train and test set from same dataset. 294
6.14 Diversity and Fluency for **Multi-attribute Control** alongside single-attribute control results for comparison. All values are averages over 3 runs. S=the Combined Sentiment dataset, T=the Combined Topic dataset, M=Multiple, p = processed, Ind mod=composition is on all 5 individually trained modules. Bold (shaded) = (second) highest score in column and section. 296
6.15 Control Effectiveness for **Multi-attribute Control** alongside single-attribute control results for comparison. All values are averages over 3 runs. S=the Combined Sentiment dataset, T=the Combined Topic dataset, M=Multiple, p = processed, Ind mod=composition is on all 5 individually trained modules. Bold (shaded) = (second) highest score in column and section; underline = train and test set from same dataset. 297
6.16 Prompt used for the instruction-tuned LLM classifier (Command R+ quantised to 4-bit precision). 298
6.17 Control Effectiveness calculated using an LLM as classifier for **Topic Control**, training on *single* and *combined* datasets, and composition of modules trained on single datasets Sum(data1, data2), Average(data1, data2), and Weights Average(data1, data2). Bold (shaded) = (second) highest score in column and section; underline = train and test set from same dataset. 299
6.18 Pearson's and Spearman's correlations between the LLM as classifier and the other three classifiers used; p-value in parentheses. 299
6.19 Portability results with 0 post-porting learning steps for sentiment control, grouped by target model. Each block shows the raw base model (no module), the identity upper bound, and all ported configurations. CE values are percentages averaged over seeds; Distinct-n and SLOR are averaged over test sets and seeds. Params = size of the target model relative to the source model. **Bold** = best; *Italic* = second best within each block. 312

6.20 Portability results with 20 post-porting learning steps (10% of training data) for sentiment control grouped by target model. Each block shows the raw base model (no module), the identity upper bound, and all source models ported into the target. All values are averaged over 3 seeds. CE values are percentages; Distinct-n and SLOR are weighted averages across test sets and seeds. **Bold** = best within each transfer condition, *Italic* = second best. LFT = post-porting Lightweight Fine-Tuning. 314
6.21 Portability results with 0 post-porting learning steps for topic control grouped by target model. Each block shows the raw base model (no module), the identity upper bound, and all source models ported into the target. All values are averaged over 3 seeds. CE values are percentages; Distinct-n and SLOR are weighted averages across test sets and seeds. **Bold** = best within each transfer condition, *Italic* = second best. 316
6.22 Portability results with 100 post-porting learning steps (10% of training data) for topic control grouped by target model. Each block shows the raw base model (no module), the identity upper bound, and all source models ported into the target. All values are averaged over 3 seeds. CE values are percentages; Distinct-n and SLOR are weighted averages across test sets and seeds. **Bold** = best within each transfer condition, *Italics* = second best. LFT = post-porting Lightweight Fine-Tuning. 317

A.1 Dataset statistics for the evaluation annotation task. Paper text and reasoning trace lengths are measured in tokens using the Qwen3 8B tokeniser. Test set reasoning traces are not used (model generates them during inference). 374
A.2 Paper-level evaluation type classification results, averaged over 5 runs. **Bold** indicates best result, *italics* second-best. R=Reasoning trace; EM=Exact Match; J=Jaccard score (intersection-over-union). 377
A.3 Measure-level evaluation type classification results, averaged over 5 runs. **Bold** indicates best result, *italics* second-best. R=Reasoning trace; EM=Exact Match; J=Jaccard score (intersection-over-union). . 377
A.4 Measure-level QCET node classification results, averaged over 5 runs. **Bold** indicates best result, *italics* second-best. R=Reasoning trace; EM=Exact Match; J=Jaccard score (intersection over union). 378

B.1 Template of the Zero-Shot prompt for Falcon 40B Instruct model. . . 385
B.2 Template of the Few-Shot prompt for Falcon 40B Instruct model. . . 386
B.3 Template of the Zero-Shot prompt for LLaMa2 70B chat model. . . . 387
B.4 Template of the Few-Shot prompt for LLaMa2 70B chat model. . . . 388

C.1 Template of the Zero-Shot Minimal Instruction prompt. 395
C.2 Few-Shot In Context prompt. **Top:** Template of the prompt. **Centre:** Examples' triple set and texts in English, Irish, and Breton. **Bottom:** Examples' triple set and texts in Maltese and Welsh. . . . 396
C.3 Input template for our fine-tuned Mistral 7B. **Top:** Template of the input. **Bottom:** Complete example in English. 397

C.4 Zero-shot prompt for Mistral 7B Instruct. **Top:** Template of the input. **Bottom:** Complete example in English. 397
C.5 Zero-shot prompt for Falcon 40B Instruct. 397
C.6 Few-Shot prompt for Mistral 7B Instruct. **Top:** Template of the prompt. **Middle:** Fixed examples used during testing. **Bottom:** Complete example in English. 398
C.7 Few-Shot prompt for Falcon 40B Instruct. **Top:** Template of the prompt. **Bottom:** Fixed examples used during testing. 399

D.1 Hyperparameters of the best-performing modules for sentiment control, selected through grid search. The modules were trained on the combined datasets. 402
D.2 Hyperparameters of the best-performing modules for topic control, selected through grid search. The modules were trained on the combined datasets. 403

Towards Modular Controlled Text Generation: Attribute-Specialised PEFT Modules and Comparative Evaluation

Michela Lorandi

Abstract

Controlled Text Generation (CTG) aims to produce text that reliably adheres to specified attributes, such as sentiment, topic, or factual content. Despite the impressive generation capabilities of large language models (LLMs), standardised evaluation protocols are largely absent, simultaneous control over multiple attributes remains unreliable, and existing fine-tuning approaches are either computationally prohibitive or too rigid to accommodate new control requirements without retraining. This thesis addresses these challenges by developing principled evaluation methodology, systematically analysing the CTG landscape, and proposing modular, reusable control mechanisms.

Evaluation in CTG research lacks principled foundations. We develop a level-playing-field (LPF) framework that addresses four key sources of evaluation bias: system selection, metric selection, dataset selection, and output-generation protocols. Applied to 12 CTG systems, the framework demonstrates that reported scores and system rankings vary substantially with methodological choices.

We introduce a multi-dimensional categorisation scheme for CTG techniques along three dimensions: control attributes, control integration, and control implementation. Validated through manual annotation of 140 papers and extended to 178 further papers via an LLM-based annotation system, the scheme reveals the dominance of single-attribute control and the rise of prompting.

Comparing prompting-based and Parameter-Efficient Fine-Tuning (PEFT) approaches for content-controlled generation, we find that LoRA-adapted (Low Rank Adaptation) Mistral 7B matches GPT-3.5 within 1% BLEU despite a 25-fold difference in parameter count, showing that PEFT can largely compensate for model scale.

The central contribution is a plug-and-play framework of attribute-specialised LoRA modules, composable at inference time for multiple-attribute control without joint retraining. Experiments with LLaMA 3 and Mistral show that composing independently trained modules via output summing outperforms individually trained ones. Few-step adaptation enables module reuse across architectures at a fraction of retraining cost.

This thesis demonstrates that modular, attribute-specialised PEFT approaches offer a principled and efficient path towards flexible, reusable control in large language models.

Chapter 1

Introduction

1.1 Thesis Topic

Controlled Text Generation (CTG) focuses on generating natural language text with specific syntactic or semantic properties that can be controlled, such as sentiment, topic, style, toxicity level, or factual content. As a subfield of Natural Language Generation (NLG), CTG addresses a fundamental challenge: how to direct the capabilities of Large Language Models (LLMs) so that their outputs reliably satisfy specified requirements. This capability is essential across a diverse range of applications, from automated report generation and creative writing assistance to conversational systems and data-to-text verbalisation.

The importance of CTG is particularly evident when considering high-stakes domains. In medical report generation, for instance, systems must verbalise patient data accurately; hallucinations, instances where the system generates factually incorrect information not present in the source data, or omissions, instances where required information is absent from the output, could have serious clinical consequences (Ji et al., 2023; Asgari et al., 2025; Ahmad et al., 2023). Legal document generation (Dahl et al., 2024), financial reporting (Kang and Liu, 2023), and safety-critical systems similarly demand precise control over output properties, such as factual accuracy, completeness, and tone, that cannot be reliably achieved through prompting alone. While this thesis does not address formal verification of these

systems, these application scenarios demonstrate why developing effective, evaluable control mechanisms is crucial for enabling language models to be deployed in contexts where output quality has real consequences.

Current LLMs demonstrate impressive text generation capabilities, and powerful models such as GPT-4 can sometimes satisfy individual control requirements, generating text with a particular sentiment or adhering to a defined topic, when prompted appropriately (Pu and Demberg, 2023). However, this approach remains unreliable: control effectiveness is highly dependent on prompt formulation, sensitive to wording variations, and difficult to guarantee even for single attributes (Sclar et al., 2024). The challenge intensifies considerably for *multiple-attribute control*. Coordinating several requirements simultaneously, such as generating a factually grounded product description with specific sentiment and a defined stylistic register, introduces new complexities: attributes may conflict, certain requirements may be neglected, and overall output quality can degrade. Furthermore, prompting-based control with large proprietary models raises practical concerns about cost, reproducibility, and adaptability, particularly in resource-constrained settings where such models are unavailable or impractical to deploy. These challenges show the need for more principled, efficient, and modular approaches to controlled text generation.

Existing approaches to CTG present substantial limitations. Prompting-based methods require no training but prove brittle and difficult to use when multiple attributes must be controlled simultaneously (Huang et al., 2023; Gu et al., 2022b). Full fine-tuning of large models for each control requirement demands prohibitive computational cost and creates practical maintenance challenges (Hu et al., 2022; Lialin et al., 2023): introducing a new control attribute requires retraining the entire model from scratch. Alternative approaches employ small auxiliary language models to influence generation by modifying the base model's token probabilities at each decoding step (Dathathri et al., 2019; Krause et al., 2021; Liu et al., 2021). While offering some modularity, these methods can compromise output fluency and quality by overriding the base model's learned linguistic distributions.

Some existing parameter-efficient methods achieve a form of modularity. Prefix-tuning approaches, for example, learn separate prefixes, learnable vectors that are prepended to the model's internal representations, for each control value, trained independently (Li and Liang, 2021; Qian et al., 2022). However, these methods are limited in three important respects. First, the learned prefixes are typically model-specific and cannot be transferred to a different base model architecture, requiring full retraining whenever the underlying model changes. Second, prefixes are control *value*-specific rather than control *attribute*-specific: separate prefixes must be maintained for each possible value within an attribute (e.g., individual prefixes for positive, negative, and neutral sentiment, rather than a single module representing sentiment as a whole). Third, these prefixes must be dynamically loaded and combined every time control requirements change, creating practical deployment challenges. Taken together, these limitations mean that while such approaches offer modularity within a single model, they do not enable the cross-model reusability and attribute-level abstraction that would maximise resource efficiency and deployment flexibility.

The field also lacks systematic frameworks for evaluating control effectiveness. CTG research uses different evaluation metrics, metric implementations, and datasets, making it difficult to compare systems or to assess the performance of individual techniques in isolation. To give a concrete example: Krause et al. (2021), Yu et al. (2021), and Zhang and Song (2022) all propose methods for controlling binary sentiment and all evaluate using a sentiment classifier, yet they use different test corpora (Bookcorpus, PPLM Prompts, and OpenWebText, respectively) and different classifiers (RoBERTa fine-tuned on SST-2, BERT trained on IMDb, and DistilBERT fine-tuned on SST-2). As a result, their reported scores are not comparable: it is simply not possible to determine from the published results which approach achieves better sentiment control. This problem is not limited to sentiment; it extends to any control attribute for which no shared evaluation protocol exists. The absence of such standards represents a significant obstacle to scientific progress in the field,

and motivates the evaluation framework developed in Chapter 4.

These limitations point to a common underlying need: CTG systems that are modular, composable, and efficiently reusable, with evaluation against consistent and reproducible standards. Addressing this need implies asking whether control attributes can be learned independently and subsequently combined without joint retraining, whether the resulting modules can transfer across model architectures to avoid redundant computation, and whether fair evaluation frameworks can be established to measure effectiveness reliably.

This thesis addresses the challenges outlined above through a systematic investigation spanning four interconnected areas: categorisation and evaluation of CTG systems, comparison of control implementation strategies, development of modular control mechanisms, and cross-model module reusability.

First, we establish analytical and evaluative foundations by developing a multi-dimensional categorisation scheme for CTG approaches and proposing a level-playing-field evaluation framework that enables fair system comparison. Second, we investigate control implementation strategies for content-controlled generation, comparing prompting-based approaches with parameter-efficient fine-tuning (PEFT) methods to understand their effectiveness and deployment trade-offs.

These investigations motivate our core contribution: attribute-specialised PEFT modules for flexible, reusable control. We develop methods for training individual modules to control specific attributes (e.g., sentiment as a whole) rather than specific values (e.g., only positive sentiment), and investigate strategies for composing these independently trained modules to achieve multiple-attribute control without joint retraining. Finally, we explore module portability across different base model architectures, demonstrating that learned control mechanisms can be transferred and reused as underlying models evolve.

Unlike existing modular approaches, our attribute-specialised modules are designed to support cross-model portability, enabling reuse across different base model architectures without retraining. This approach offers four key advantages: **com-**

putational efficiency, as parameter-efficient adaptation reduces training cost substantially compared to full fine-tuning; **extensibility**, as new control attributes can be added by training additional modules without retraining existing ones; **reusability**, as trained modules can be reused across different base models, reducing overall resource requirements; and **attribute-level abstraction**, as control is captured at the level of attributes rather than individual values, enabling more general and transferable representations.

These contributions can be formally enumerated as follows:

1. **Multi-dimensional categorisation scheme for CTG techniques:** Development of a systematic framework for analysing and comparing controlled text generation approaches along three dimensions: control attributes (what is controlled), control integration (how control signals are represented), and control implementation (how control is achieved during generation). This scheme provides a structured analytical lens for the field.

2. **Systematic survey of CTG techniques:** Application of the categorisation scheme to systematically analyse existing controlled text generation methods, identifying relationships between approaches, under-explored research areas, and positioning our contributions within the broader landscape of CTG research.

3. **Evaluation framework for CTG:** Development of fair comparison methodology for CTG techniques using level-playing-field evaluation principles, and investigation of appropriate evaluation approaches for different control types, including semantic consistency assessment for content control and classifier-based metrics for stylistic attributes.

4. **Comparative analysis of control implementation strategies:** Comprehensive evaluation of prompting-based and state-of-the-art approaches for controlled generation, showing their relative effectiveness and computational trade-offs.

5. **Design of attribute-specialised PEFT modules:** Methods for creating performant single-attribute PEFT modules using supervised fine-tuning (SFT).

6. **Strategies for compositional multiple-attribute control:** Development and evaluation of strategies for combining independently trained single-attribute modules to achieve multiple-attribute control without joint retraining.

7. **Method for cross-model module transfer:** Investigation of PEFT module portability across different model architectures, identifying conditions under which learned control mechanisms can be reused to improve efficiency and reduce training requirements.

1.2 Research Questions

This thesis focuses on a high-level research question: **How can we achieve effective controlled text generation in large language models through systematic evaluation, implementation comparison, and modular, reusable control mechanisms?** This question encompasses both the challenge of coordinated multiple-attribute control and the practical concern of doing so efficiently, through module reusability rather than complete retraining for each new control configuration.

To address this high-level question systematically, we decompose it into four interconnected research questions that progress from foundational evaluation challenges through implementation strategies to modular composition and cross-model transfer.

RQ 1: How can we systematically categorise and understand existing controlled text generation approaches? To effectively advance CTG research, we require a structured framework for organising and comparing different approaches. This research question addresses both the development of a multi-dimensional cat-

egorisation scheme and its application to systematically analyse the literature.

RQ1.1 How can we develop a multi-dimensional framework for categorising CTG approaches?

RQ1.2 Can we effectively automate the categorisation of CTG papers using language models with human validation?

RQ 2: How can we effectively evaluate control in text generation systems? Evaluation of CTG systems presents two fundamental challenges: selecting metrics that accurately measure whether intended control has been achieved, and ensuring fair comparison across systems through standardised evaluation practices. As established in Section 1.1, the absence of shared evaluation protocols makes it impossible to determine which approaches genuinely advance the state of the art. This research question addresses both dimensions:

RQ2.1 What evaluation approaches are appropriate for measuring different types of control (content, sentiment, topic, keywords)?

RQ2.2 How can we design an evaluation framework that enables fair, reproducible comparison of CTG systems?

Different control attributes require different evaluation strategies: content control necessitates semantic consistency assessment between structured input and generated output, while sentiment and topic control can be measured through classifier-based accuracy. Beyond metric selection, fair comparison requires standardised experimental conditions and consistent implementations. The framework developed in response to RQ1 inform evaluation methodology throughout the thesis and are addressed in Chapter 4.

RQ 3: What control implementation strategies are effective for content-controlled text generation? Before investigating modular attribute-specialised approaches, we must understand the effectiveness and trade-offs of existing control

strategies. Using data-to-text generation as a testbed, where content specifications are explicitly provided through structured input, we examine:

RQ3.1 How effective are different prompting strategies with out-of-the-box LLMs for content-controlled generation tasks?

RQ3.2 How does parameter-efficient fine-tuning (PEFT) compare to prompting-based approaches for enabling content control?

This comparison addresses a practical deployment trade-off between two control implementation strategies. Prompting-based approaches require no training and can exploit powerful proprietary models, but lack customisability, incur ongoing costs, and offer limited control reliability. PEFT-based approaches require initial training resources, even if modest, but enable model customisation, cost-effective deployment, and potentially more reliable control through dedicated adaptation. Understanding these trade-offs informs the design of attribute-specialised modules in subsequent research questions and is addressed in Chapter 5.

RQ 4: Can attribute-specialised PEFT modules enable effective controlled text generation? Building on RQ3's findings that PEFT provides viable control, we investigate whether control attributes can be isolated into separate, specialised modules rather than requiring monolithic systems. This modularity is central to the thesis objective of reusable control:

RQ4.1 How can we create effective single-attribute PEFT modules?

RQ4.2 Can multiple single-attribute modules be composed to achieve coordinated multiple-attribute control without joint retraining?

Effective attribute-specialised modules would enable flexible CTG systems where new control attributes can be added without retraining existing components, reducing the resource burden of maintaining multiple task-specific models. Successful

composition would further enable fine-grained control over multiple attributes simultaneously, meeting the needs of complex generation tasks while maintaining modularity. These questions are addressed in Chapter 6.

RQ 5: Can attribute-specialised PEFT modules transfer across model architectures? Another important test of module reusability is whether learned control mechanisms can transfer across different base models, enabling reuse without retraining or with substantially reduced training when the underlying language model changes:

RQ5.1 Can PEFT modules trained on one model architecture be effectively applied to different models?

RQ5.2 How do architectural differences between the training model and the target model affect module transferability?

Successful cross-model transfer would enable greater resource efficiency and module reusability: rather than retraining control mechanisms whenever underlying models are updated, existing modules could be directly transferred to new model versions. Understanding the architectural conditions for successful transfer would additionally provide insights into whether learned control representations capture model-agnostic linguistic properties or remain tied to specific architectures. These questions are addressed in Chapter 6.

Together, these five research questions form a comprehensive and cohesive investigation from foundational landscape understanding through evaluation frameworks to implementation strategies, modular composition, and cross-model reusability. The progression begins with systematic categorisation of existing CTG approaches (RQ1), which provides the analytical foundation for the subsequent investigations. Rigorous evaluation frameworks (RQ2) then enable meaningful assessment of control implementation strategies (RQ3), which informs the design of attribute-specialised modules (RQ4), whose reusability is ultimately tested through cross-model transfer

(RQ5). Each question builds upon insights from previous ones, collectively addressing the high-level research question by developing and validating attribute-specialised PEFT modules for multiple-attribute control and demonstrating their reusability, composability, and transferability across different model architectures.

1.3 Formal Foundations

This section establishes the mathematical foundations for controlled text generation, providing formal definitions that are independent of any particular implementation approach.

1.3.1 Language Modelling

Let $\mathcal{V}$ denote a finite vocabulary of tokens. A language model M defines a probability distribution over sequences of tokens from $\mathcal{V}$. Given a sequence of tokens $\mathbf{x} = x_1, x_2, \ldots, x_n$ where $x_i \in \mathcal{V}$, an autoregressive language model models the probability of $\mathbf{x}$ as the joint probability of each token given the tokens preceding it:

$$P(\mathbf{x}) = \prod_{i=1}^{n} P(x_i \mid x_1, \ldots, x_{i-1}) \tag{1.1}$$

Modern neural language models implement this distribution using learned parameters θ. Throughout this thesis, M_θ denotes the model itself (the parameterised function), while $P_\theta(\cdot)$ denotes the probability distribution it defines.

1.3.2 Text Generation

Given an input context $\mathbf{x} = (x_1, \ldots, x_n)$, text generation is the process of producing a continuation $\mathbf{y} = (y_1, \ldots, y_m)$ by sampling from or maximising the conditional probability distribution:

$$P(\mathbf{y} \mid \mathbf{x}; \theta) = \prod_{j=1}^{m} P(y_j \mid x_1, \ldots, x_n, y_1, \ldots, y_{j-1}; \theta) \tag{1.2}$$

Common decoding strategies include greedy decoding (selecting the highest-probability token at each step), beam search (maintaining multiple hypotheses), and sampling-based methods (sampling from the probability distribution, potentially with temperature scaling or nucleus sampling).

1.3.3 Controlled Text Generation

Definition 1.3.1 (Control Attribute)**.** A *control attribute* is a measurable property of generated text that defines *what* aspect of the text is being controlled. Examples include sentiment, topic, length, style, formality, or toxicity level. Each control attribute a has an associated range of possible values $\mathcal{R}_a$. For example, Sentiment $= s$ where $s \in \{\text{positive, negative, neutral}\}$ (i.e., $\mathcal{R}_{\text{sent}} = \{\text{positive, negative, neutral}\}$), or Length $= \ell$ where $\ell \in \mathbb{N}$ (i.e., $\mathcal{R}_{\text{length}} = \mathbb{N}$).

Definition 1.3.2 (Controlled Text Generation)**.** Let M_θ be a language model, $\mathbf{x}$ an input context, and a control specification consisting of k attribute-value pairs: $(a_1, v_1), (a_2, v_2), \ldots, (a_k, v_k)$, where each a_i is a control attribute name and $v_i \in \mathcal{R}_{a_i}$ is the target value for that attribute. Controlled text generation is the task of generating text $\mathbf{y}$ that satisfies the specified attribute constraints while maintaining fluency and coherence:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}, (a_1, v_1), \ldots, (a_k, v_k); \theta) \tag{1.3}$$

The primary challenge in CTG lies in effectively conditioning the generation process on these attribute-value specifications while maintaining text quality and naturalness.

1.3.4 Single-Attribute and Multiple-Attribute Control

Definition 1.3.3 (Single-Attribute Control)**.** Single-attribute control is the task of generating text constrained by a single control attribute. Given a control attribute a with target value $v \in \mathcal{R}_a$, the generation objective is to maximise:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}, a{=}v; \theta) \tag{1.4}$$

where the generated text $\mathbf{y}$ must satisfy the constraint specified by the single attribute-value pair (a, v) while preserving fluency and coherence.

Definition 1.3.4 (Multiple-Attribute Control)**.** Multiple-attribute control is the task of generating text constrained by two or more control attributes simultaneously. Given $k > 1$ attribute-value pairs $(a_1, v_1), (a_2, v_2), \ldots, (a_k, v_k)$ where $v_i \in \mathcal{R}_{a_i}$, the generation objective is to maximise:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}, (a_1, v_1), \ldots, (a_k, v_k); \theta) \tag{1.5}$$

where the generated text must simultaneously satisfy all k attribute-value constraints. This task is substantially more challenging than single-attribute control because: (i) attributes may conflict or interact in complex ways (e.g., maintaining formal register while incorporating informal colloquial keywords), and (ii) since generation is modelled as probability maximisation, increasing the probability of text satisfying one attribute may reduce the probability of satisfying others, creating trade-offs that degrade overall generation quality.

1.3.5 Control Evaluation

To assess whether a CTG system achieves the desired control, we require methods to measure how well generated text satisfies attribute constraints.

Definition 1.3.5 (Attribute Evaluation Function)**.** For a control attribute a with value range $\mathcal{R}_a$, an evaluation function $f_a : \mathcal{V}^* \to \mathcal{R}_a$ is defined such that for a generated text sequence $\mathbf{y} \in \mathcal{V}^*$, the value $v = f_a(\mathbf{y}) \in \mathcal{R}_a$ represents the measured attribute value.

Evaluation functions can be implemented in various ways, depending on the attribute being measured. Common approaches include:

- Deterministic measurements (e.g., counting tokens for length)
- Trained classifiers (e.g., sentiment classification models)
- Semantic similarity metrics (e.g., comparing generated text to structured input)
- Human judgements (e.g., fluency or appropriateness ratings)

Definition 1.3.6 (Control Accuracy)**.** Control accuracy is defined as the proportion of generated texts for which a discrete control attribute achieves its target value. For a set of N generated texts $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, control attribute a, and target value $v \in \mathcal{R}_a$, control accuracy is:

$$\text{Acc}(a, v) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[f_a(\mathbf{y}_i) = v] \tag{1.6}$$

where $\mathbb{1}[\cdot]$ is the indicator function that equals 1 when the condition is true and 0 otherwise. The result ranges from 0 to 1, where 1 indicates perfect control (all texts achieve the target value) and 0 indicates complete control failure (no texts achieve the target value).

For multiple-attribute control with k attributes, overall control effectiveness is assessed by evaluating how many generated texts satisfy *all* specified attribute-value pairs simultaneously. Given a set of attribute-value pairs $(a_1, v_1), (a_2, v_2), \ldots, (a_k, v_k)$ and a set of N generated texts $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, multiple-attribute control accuracy is:

$$\text{Acc}((a_1, v_1), \ldots, (a_k, v_k)) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[f_{a_1}(\mathbf{y}_i) = v_1 \wedge \cdots \wedge f_{a_k}(\mathbf{y}_i) = v_k] \tag{1.7}$$

where $\wedge$ denotes logical AND. This metric reflects the true performance of multiple-attribute control by measuring the proportion of texts that satisfy *all* attribute constraints simultaneously, rather than aggregating individual attribute performance.

1.3.6 Notation Summary

Throughout this thesis, we use the following notation:

- $\mathcal{V}$: Vocabulary of tokens
- M or M_θ: Language model with parameters θ
- $\mathbf{x} = (x_1, \ldots, x_n)$: Input sequence or context
- $\mathbf{y} = (y_1, \ldots, y_m)$: Generated output sequence
- a or a_i: A control attribute
- $\mathbf{a} = \{a_1, \ldots, a_k\}$: Set of control attributes
- v or v_i: Target value for a control attribute
- $\mathbf{v} = \{v_1, \ldots, v_k\}$: Set of target values
- f_a: Evaluation function for control attribute a
- $\mathcal{R}_a$: Range of possible values for attribute a

1.4 Thesis Structure

This thesis is organised into seven chapters that progress from foundational frameworks through empirical investigations to modular approaches for controlled text generation. Below, we summarise each chapter's contribution and its relationship to the research questions introduced in Section 1.2.

Chapter 2 addresses RQ1 (how can we systematically categorise and understand existing CTG approaches?) by developing a multi-dimensional framework for systematically analysing and comparing CTG approaches. The scheme is organised along three dimensions: control attributes, control integration, and control implementation. This framework provides a structured lens for the literature review

in Chapter 3 and clarifies how our attribute-specialised PEFT approach relates to prior methods. The chapter additionally presents an automatic annotation system for CTG papers based on this categorisation scheme.

Chapter 3 surveys the state-of-the-art in controlled text generation, organised according to the categorisation scheme established in Chapter 2. We review parameter-efficient fine-tuning techniques, PEFT module merging approaches, and existing CTG methods, positioning our contributions within the broader research landscape.

Chapter 4 addresses RQ2 (how can we effectively evaluate control in text generation systems?) by developing an evaluation approach for different control types and proposing a level-playing-field framework for fair system comparison. Section 4.3 examines semantic consistency assessment for data-to-text generation, comparing textual, semantic, and model-based metrics for content control. Section 4.4 presents a comprehensive comparative study of CTG techniques under level-playing-field evaluation conditions. The methodology established here informs assessment throughout the remainder of the thesis.

Chapter 5 addresses RQ3 (what control implementation strategies are effective for content-controlled text generation?) through comparative evaluation of prompting-based and PEFT-based approaches for data-to-text generation. Section 5.3 evaluates prompting-based approaches with out-of-the-box LLMs across multiple languages, while Section 5.4 investigates PEFT methods for parameter-efficient model adaptation. The findings establish the practical trade-offs between these approaches and motivate the modular architecture developed in the following chapter.

Chapter 6 addresses RQ4 (can attribute-specialised PEFT modules enable effective controlled text generation?) and RQ5 (can attribute-specialised PEFT modules transfer across model architectures?), forming the central empirical contributions of this thesis by developing novel training methods for attribute-specialised

PEFT modules, composition strategies for multiple-attribute control, and cross-model transfer techniques. The chapter proceeds in three stages: the creation of single-attribute PEFT modules using supervised fine-tuning and reinforcement learning (RQ4.1); the composition of independently trained modules to achieve multiple-attribute control without joint retraining (RQ4.2); and an investigation of module portability across different model architectures, examining the conditions under which attribute-specialised modules can be transferred and reused (RQ5).

Chapter 7 concludes the thesis by summarising the main findings, reflecting on the limitations of the proposed approach, and identifying directions for future work.

1.5 Publications

This thesis has so far resulted in the following list of publications:

1. **Michela Lorandi** and Anya Belz. 2023. **How to Control Sentiment in Text Generation: A Survey of the State-of-the-Art in Sentiment-Control Techniques.** In *Proceedings of the 13th Workshop on Computational Approaches to Subjectivity, Sentiment, & Social Media Analysis*, pages 341–353, Toronto, Canada. Association for Computational Linguistics.

2. **Michela Lorandi** and Anya Belz. 2023. **Data-to-text Generation for Severely Under-Resourced Languages with GPT-3.5: A Bit of Help Needed from Google Translate (WebNLG 2023)**. In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 80–86, Prague, Czech Republic. Association for Computational Linguistics.

3. Liam Cripwell, Anya Belz, Claire Gardent, Albert Gatt, Claudia Borg, Marthese Borg, John Judge, **Michela Lorandi**, Anna Nikiforovskaya, William Soto-Martinez, and Craig Thomson. 2023. **The 2023 WebNLG Shared Task on Low Resource Languages. Overview and Evaluation Results**

(WebNLG 2023). In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 55–66, Prague, Czech Republic. Association for Computational Linguistics.

Author contribution: the present author collaborated on the implementation of the human evaluation interface and on the analysis of the human evaluation results.

4. **Michela Lorandi** and Anya Belz. 2024. **High-quality Data-to-Text Generation for Severely Under-Resourced Languages with Out-of-the-box Large Language Models**. *In Findings of the Association for Computational Linguistics: EACL 2024*, pages 1451–1461, St. Julian's, Malta. Association for Computational Linguistics.

5. Simon Mille, Massimiliano Pronesti, Craig Thomson, **Michela Lorandi**, Sophie Fitzpatrick, Rudali Huidrom, Mohammed Sabry, Amy O'Riordan, and Anya Belz. 2024. **Filling Gaps in Wikipedia: Leveraging Data-to-Text Generation to Improve Encyclopedic Coverage of Underrepresented Groups**. In *Proceedings of the 17th International Natural Language Generation Conference: System Demonstrations*, pages 16–19, Tokyo, Japan. Association for Computational Linguistics.

   Author contribution: the present author collaborated by providing the systems that generate text from the selected input triples, and on the overall design and deployment of the website (frontend and backend) on the server.

6. **Michela Lorandi** and Anya Belz. 2024. **DCU-NLG-PBN at the GEM'24 Data-to-Text Task: Open-Source LLM PEFT-Tuning for Effective Data-to-Text Generation**. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 76–83, Tokyo, Japan. Association for Computational Linguistics.

7. Rudali Huidrom, **Michela Lorandi**, Simon Mille, Craig Thomson, and Anya

Belz. 2025. **Assessing Semantic Consistency in Data-to-Text Generation: A Meta-Evaluation of Textual, Semantic and Model-Based Metrics**. In *Proceedings of the 18th International Natural Language Generation Conference*, pages 98–107, Hanoi, Vietnam. Association for Computational Linguistics.

Author contribution: the present author collaborated on the design of the experimental plan, contributed to the creation of the datasets, and provided the systems used in the evaluation.

8. **Michela Lorandi**, Anya Belz, Simone Mille, and Craig Thomson. 2026. **Automatic Paper Analysis and Categorisation for Systematic Reviews with Combined Reasoning-Augmented SFT and DAPO RL**. *In Findings of the Association for Computational Linguistics: ACL 2026*, San Diego, USA. Association for Computational Linguistics.

9. **Michela Lorandi** and Anya Belz. 2026. **A Comparative Study of Controlled Text Generation Systems Using Level-Playing-Field Evaluation Principles.** arXiv preprint arXiv:2605.12395.

10. **Michela Lorandi** and Anya Belz. 2026. **Output Composability of QLoRA PEFT Modules for Plug-and-Play Attribute-Controlled Text Generation.** arXiv preprint arXiv:2605.12345.

Chapter 2

A Categorisation Scheme for Controlled Text Generation Techniques

2.1 Chapter Overview

This chapter presents a multi-dimensional categorisation framework for CTG systems and demonstrates its utility through both manual and automatic annotation of the research literature. The chapter is structured around the two sub-questions of RQ1 (How can we systematically categorise and understand existing controlled text generation approaches?): Sections 2.2–2.4 address RQ1.1 (How can we develop a multi-dimensional framework for categorising CTG approaches?) by developing the categorisation framework and validating it through systematic manual annotation, while Section 2.5 addresses RQ1.2 (Can we effectively automate the categorisation of CTG papers using language models with human validation?) by investigating automatic annotation using large language models and evaluating its reliability.

Specifically, Section 2.2 motivates the need for such a framework, discussing the limitations of existing single-dimensional categorisation approaches. Section 2.3 introduces the categorisation scheme itself, organised along three orthogonal dimen-

sions: **control attributes** (what is being controlled), **control integration** (how control signals are represented and made available to the system), and **control implementation** (how control is achieved during generation). Section 2.4 describes the process of developing and validating the scheme through systematic analysis of the research literature, including the selection of papers reporting CTG techniques, manual annotation procedure, and resulting dataset statistics.

Section 2.5 investigates whether annotation according to this scheme can be automated, presenting an LLM-based approach with human validation and evaluating its performance against the manually annotated dataset. Section 2.5.7.2 provides detailed error analysis, demonstrating that automatic annotation with validation provides substantial efficiency gains while maintaining annotation quality. Section 2.6 analyses the resulting annotated dataset, reporting statistics across the three dimensions and additional annotations, and identifying trends and patterns in the CTG literature. Finally, Section 2.7 reflects on the framework's broader utility, including its role in organising the literature review in Chapter 3 and in positioning the contributions of this thesis within the CTG landscape.

Some of the content reported in this chapter was previously reported in the following publications, with overlapping sections indicated in parentheses:

1. **Michela Lorandi** and Anya Belz. 2023. **How to Control Sentiment in Text Generation: A Survey of the State-of-the-Art in Sentiment-Control Techniques.** In *Proceedings of the 13th Workshop on Computational Approaches to Subjectivity, Sentiment, & Social Media Analysis*, pages 341–353, Toronto, Canada. Association for Computational Linguistics. (Section 2.3)

2. **Michela Lorandi**, Anya Belz, Simon Mille, and Craig Thomson. 2026. **Automatic Paper Analysis and Categorisation for Systematic Reviews with Combined Reasoning-Augmented SFT and DAPO RL**. *In Findings of the Association for Computational Linguistics: ACL 2026*, St. Diego, USA. Association for Computational Linguistics. (Section 2.5)

2.2 Motivation

The controlled text generation literature has grown rapidly in recent years, with researchers developing increasingly sophisticated methods for manipulating diverse attributes of generated text. However, this growth has created significant organisational and analytical challenges. Existing categorisation schemes (Zhang et al., 2023; Liang et al., 2024b) typically focus on a single dimension, such as implementation method or architectural component, making it difficult to compare approaches that differ along multiple axes simultaneously. For instance, papers on sentiment control may employ completely different implementation strategies than those addressing formality control, while recent work on modular and compositional control does not fit neatly into existing categorisation schemes organised around monolithic control mechanisms.

Several recent surveys have examined CTG from different perspectives. Zhang et al. (2023) provide comprehensive coverage of CTG techniques based on pre-trained language models, categorising methods into fine-tuning, retraining, and post-processing approaches, and identifying control conditions as semantic, structural, or lexical. However, their survey restricts its scope to Transformer-based PLMs, limiting generalisability across the broader CTG landscape. More recently, Liang et al. (2024b) survey CTG with LLMs, dividing techniques by when control is applied (training-stage versus inference-stage) and organising attributes into content control and attribute control. While this temporal perspective is useful, their framework similarly focuses on large-scale Transformer models, and the absence of explicit paper selection criteria makes it difficult to assess coverage completeness or potential biases in the resulting survey.

These existing surveys share a common limitation: systems are categorised primarily along a single principal dimension, with other aspects treated as secondary characteristics. This obscures important relationships between different dimensions of control. For instance, the choice of control attribute can constrain which implementation strategies are viable, while the representation of control signals can

determine which architectures are appropriate. Single-dimensional categorisation makes such cross-cutting relationships difficult to identify and analyse.

What is needed is a multi-dimensional framework that treats different aspects of controlled generation as orthogonal dimensions, enabling systematic positioning of any CTG system and identification of under-explored combinations. We propose a categorisation scheme organised along three key dimensions:

1. **Control attributes** – *what* is being controlled: the properties of generated text being manipulated, ranging from surface-level features (e.g., sentiment, formality) to structural constraints (e.g., syntax, discourse structure) to semantic content.

2. **Control integration** – *where* control signals are represented and enter the system: the architectural level at which control values are specified and made available to the system, whether at the input level, through selection of internal model components, or via integration of external modules.

3. **Control implementation** – *how* control is achieved: the technical mechanism through which control is effected, such as architectural modification, training procedure adaptation, or decoding-time intervention.

This multi-dimensional framework allows us to position any CTG system systematically within a three-dimensional space, supporting direct comparability across different approaches, and to identify gaps in the literature: combinations of attribute types, control integration strategies, and implementation mechanisms that remain under-explored. Unlike those used in prior surveys, our framework is developed through a rigorous systematic process: we define explicit paper selection criteria (Section 2.4), manually annotate a substantial corpus of CTG papers, and validate the scheme on a held-out set of papers. We additionally investigate whether annotation can be automated through an LLM-based approach (Section 2.5), demonstrating the framework's scalability to the growing CTG literature.

2.3 CTG Categorisation Scheme

This section presents our categorisation scheme for controlled text generation systems. The scheme is organised along three orthogonal dimensions: control attributes, control integration, and control implementation. Each dimension is described in a dedicated subsection below, with definitions and illustrative examples. In addition to these three dimensions, our annotated dataset includes supplementary annotations covering the datasets and evaluation metrics used in each paper; these are described alongside the dataset statistics in Section 2.6 rather than as part of the scheme itself.

The full scheme is summarised in Figures 2.1, 2.2 and 2.3, which present (i) the taxonomy of control attributes, (ii) the taxonomy of control integration strategies, and (iii) the taxonomy of control implementation strategies, respectively. These figures are used to organise the related work survey in Chapter 3.

2.3.1 Preliminaries

We establish four key concepts that underpin the categorisation scheme: control attribute, control signal, and the distinction between single- and multiple-attribute control, alongside the architectural notions of control integration and control implementation.

2.3.1.1 Control Attribute and Control Signal.

Following Definition 1.3.1, a **control attribute** is a measurable property of generated text that defines *what* aspect is being controlled. Each control attribute a has an associated range of possible values $\mathcal{R}_a$. Examples include sentiment ($s \in \{\text{positive, negative, neutral}\}$), topic (discrete categories or continuous embeddings), length ($\ell \in \mathbb{N}$), style, formality, or discourse structure. The attribute name a and its target value $v \in \mathcal{R}_a$ together form an attribute-value pair (a, v) that specifies a control objective (see Definition 1.3.2).

A **control signal** is the concrete representation used to encode and convey a

control attribute's target value within the generation system. It defines *how* the control attribute value is represented, and may take various forms: discrete labels (e.g., the value *positive* for sentiment control), continuous vectors (e.g., learned embeddings), text sequences (e.g., prompt prefixes such as "Write in a formal tone:"), or structured representations (e.g., knowledge graphs, syntactic trees, or structured JSON).

The distinction is crucial: a control attribute such as sentiment is defined independently of any system implementation and has an associated range of values $\mathcal{R}_a$, whereas the corresponding control signal, the representation of a target value, can vary substantially across systems. For example, to control sentiment, one system might use a discrete label (a value from $\mathcal{R}_{\text{sentiment}}$), another a natural language instruction, and another a learned embedding vector. All three represent the same underlying attribute, but differ in their signal representation.

2.3.1.2 Single- vs. Multiple-Attribute Control.

Following Definitions 1.3.3 and 1.3.4, CTG systems are classified by the number of attribute-value pairs in the control specification. In **single-attribute control**, a single attribute-value pair (a, v) constrains generation: the system controls only one attribute per generation pass. For example, GeDi (Krause et al., 2021) steers generation towards a single target sentiment using class-conditional language models.

In **multiple-attribute control**, multiple attribute-value pairs constrain generation simultaneously $(a_1, v_1), (a_2, v_2), \ldots, (a_k, v_k)$ where $k > 1$ constrain generation simultaneously within a single pass. For instance, Gu et al. (2022b) control both sentiment and topic simultaneously by interpolating single-attribute control vectors learned independently for each attribute.

Multiple-attribute control introduces substantial additional challenges beyond single-attribute control. As noted in Definition 1.3.4, attributes may conflict or interact in complex ways (e.g., maintaining formal register while incorporating informal colloquial keywords), and trade-offs may emerge where increasing satisfaction

of one attribute-value constraint reduces satisfaction of another. These challenges make multiple-attribute control a substantially harder problem and a central focus of this thesis.

2.3.1.3 Control Integration and Control Implementation.

Our categorisation scheme examines two further dimensions, both architectural. **Control integration** refers to the architectural level at which a control signal enters the system: at the input level (*input addition*), through selection of internal model components (*model injection*), or via configuration of external modules (*system addition*). **Control implementation** refers to the mechanism by which control capabilities are enabled, for instance, through prompting, fine-tuning, or modification of token distributions at decoding time. These two dimensions are conceptually distinct but interdependent: the choice of implementation method can constrain which integration strategies are viable, and vice versa.

2.3.2 Dimension 1: Control Attributes

CTG methods control diverse attributes of generated text, ranging from high-level semantic properties, such as topic and sentiment, to low-level structural properties, such as length and syntactic form. To systematically characterise this landscape, we propose a taxonomy that distinguishes two high-level categories: ***Content Control*** and ***Non-Content Control***, as shown in Figure 2.1.

Content control attributes constrain the core semantic content of the generated text, that is, *what* is said. These attributes operate at two levels of granularity. ***Fine-grained*** content control specifies particular entities, events, or knowledge elements that must appear in the text. ***Coarse-grained*** content control operates at a higher level, focusing on general themes, topics, or abstract categories. Both levels can be realised through either ***structured*** representations (e.g., knowledge graphs, event sequences, structured outlines) or ***unstructured*** representations (e.g., keywords, topic labels, free-form definitions).

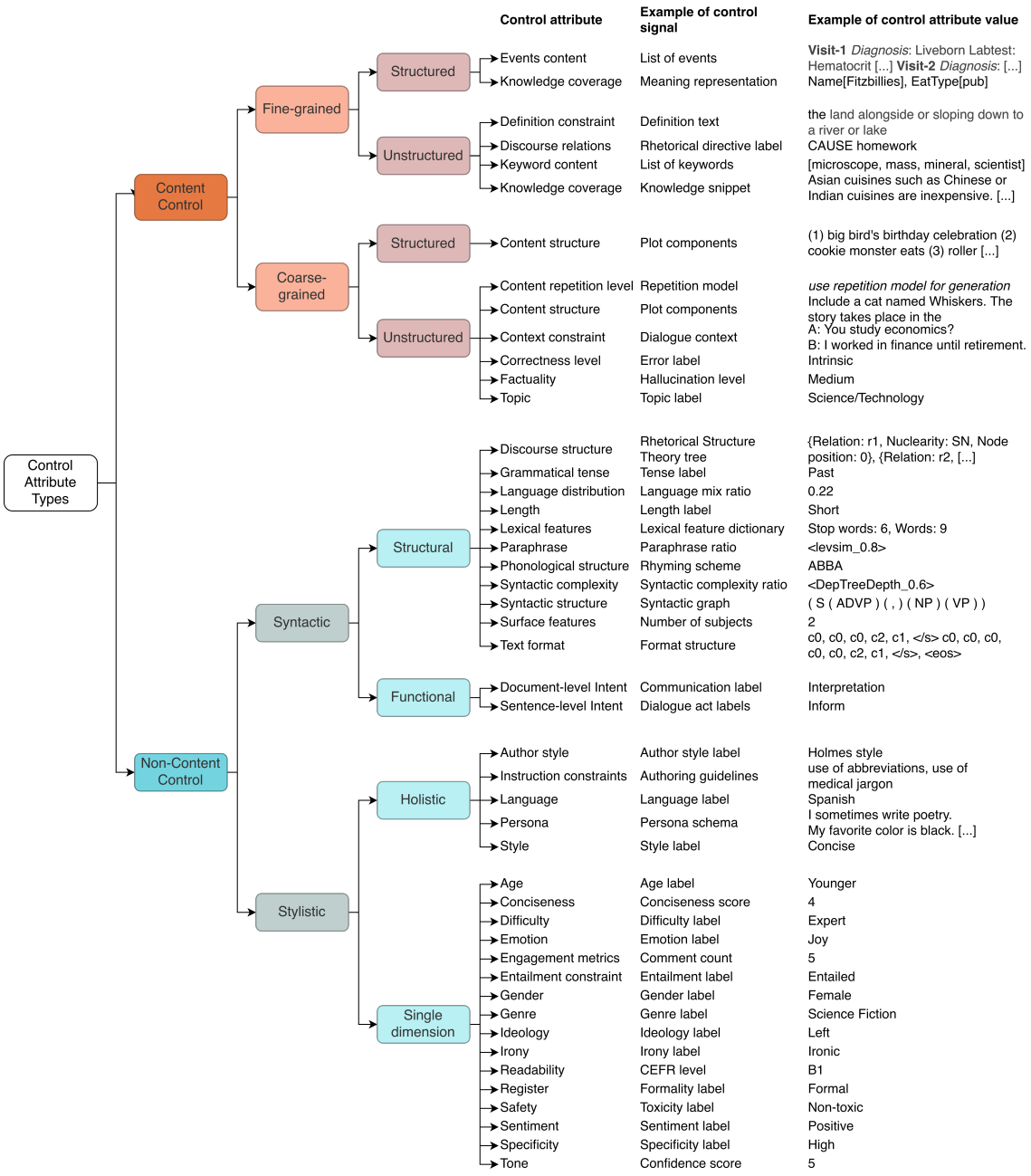


Figure 2.1: Taxonomy of **Control Attribute Types** in CTG systems. The hierarchy distinguishes between Content Control (*what* is said) and Non-Content Control (*how* it is said). Each control attribute is accompanied by example control signals and attribute values observed across the analysed papers.

Non-content control attributes shape the form, structure, and style of the generated text—that is, *how* something is said, without directly constraining *what* is said in terms of semantic content or factual information. This distinction is fundamental: non-content attributes can be manipulated while preserving the core message (e.g., expressing the same opinion with positive versus negative sentiment,

or conveying the same information formally versus informally), whereas content attributes determine the substantive meaning itself.

Non-content control is divided into two subtypes: Syntactic and Stylistic control, each with two further subdivisions. ***Syntactic*** control constrains the structural and grammatical form of the text: ***structural syntactic*** attributes focus on how sentences are constructed (e.g., length, tense, syntactic complexity), while ***functional syntactic*** attributes focus on the communicative or pragmatic role a sentence plays (e.g., discourse relations, speech acts). ***Stylistic*** control constrains the manner of expression and affective tone: ***holistic style*** attributes control style at the level of situational context or authorial persona (e.g., formality register, writing style), while ***single-dimension style*** attributes control individual stylistic dimensions independently. Single-dimension stylistic attributes include sentiment (positive, negative, neutral emotional valence), toxicity (presence of offensive or harmful language), emotion (specific affective states such as joy, anger, or sadness), and political ideology (liberal, conservative framing). These are classified as stylistic rather than content-based because they concern the affective presentation and framing of information rather than the factual content itself, the same factual claim can be expressed with different sentiment, toxicity levels, or ideological framing without changing its content.

A CTG system may control multiple attributes simultaneously, either from the same branch of the taxonomy (e.g., multiple single-dimension stylistic attributes such as sentiment and toxicity) or from different branches (e.g., combining stylistic control with content control). For instance, a system might jointly control topic (a content attribute: ***Content*** → ***Coarse-Grained*** → ***Unstructured***) and sentiment (a stylistic attribute: ***Non-Content*** → ***Stylistic*** → ***Single-dimension***), or simultaneously constrain length (structural syntactic: ***Non-Content*** → ***Syntactic*** → ***Structural***) and formality (holistic stylistic: ***Non-Content*** → ***Stylistic*** → ***Holistic***).

2.3.3 Dimension 2: Control Integration

Control integration refers to the architectural level at which control values are specified and enter the CTG system, capturing *where* control information is provided to influence generation. Importantly, we classify integration based on where the specific control signal (e.g., "positive sentiment", "formal tone") is provided by the user or system designer, not where the learned parameters or mechanisms that implement control reside in the model architecture. For example, a system might accept a sentiment label as additional input (***Input Addition***) even though the model has been fine-tuned throughout its parameters to respond to this input, the integration point is where the control signal enters, not where the fine-tuned weights are located. Figure 2.2 provides a diagrammatic overview of the three integration levels: ***Input Addition***, ***Model Injection***, and ***System Addition***.

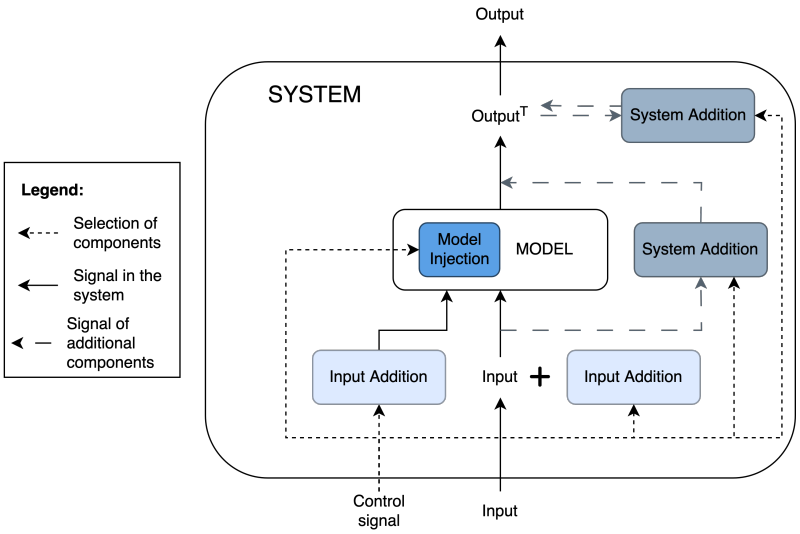


Figure 2.2: Architectural diagram of control integration levels in CTG systems. Control signals (shown with dashed arrows) indicate where control values enter the system. Different integration strategies can be combined within a single system.

Input Addition occurs when control values are specified as part of the input to the model, alongside any other input presented to the model for processing. Common approaches include prepending control codes or tokens to the input sequence (e.g., `[POSITIVE]` for sentiment control), adding structured prompts that specify desired attributes, or using special tokens representing control values. The key characteristic is that the control value is provided as part of the input, enabling dynamic control

specification at inference time without architectural (re)configuration. Note that, even when mechanisms such as LoRA adapters (Hu et al., 2022) are present in the model, if the control value is specified via input tokens processed by a generic adapter, this constitutes ***Input Addition***, since the control signal enters through the input channel.

Model Injection occurs when control values are determined by the selection or configuration of model components. This includes loading value-specific adapter modules (e.g., selecting a positive-sentiment LoRA module versus a negative-sentiment LoRA), activating control-specific attention layers, or configuring architectural components that encode fixed control values. The distinguishing feature is that changing the control value requires modifying the model configuration, not simply changing the input.

System Addition occurs when control values are specified through external components that operate alongside or after the base generation model. This includes post-processing classifiers that filter or re-rank outputs according to a target attribute value, plug-and-play modules that modify token distributions, and external discriminators configured with desired attribute values. The base model remains unmodified, allowing external components to be added, removed, or replaced independently.

Finally, many CTG systems (Shu et al., 2020; Chang et al., 2021; Majumder et al., 2022; Xu et al., 2023; Liang et al., 2024a) use ***Hybrid*** integration, combining multiple levels. For example, a system might prepend control tokens at the input level (***Input Addition***) while also selecting a specialised adapter (***Model Injection***), or combine input-level prompting with external output filtering (***Input Addition*** and ***System Addition***). Hybrid approaches leverage the complementary strengths of each strategy: input-level control for dynamic inference-time specification, model-internal control for deep architectural steering, and system-level control for flexible post-hoc refinement.

The three integration levels differ in their implications for flexibility and com-

putational cost. ***Input Addition*** enables runtime control variation through input alone but requires the model to be responsive to varied input signals. ***Model Injection*** embeds control deeply within the architecture but limits runtime flexibility, as changing control values requires loading different modules. ***System Addition*** offers maximal modularity but may incur additional inference-time cost through external processing.

2.3.4 Dimension 3: Control Implementation

Control implementation refers to the method by which control capabilities are enabled in a CTG system, capturing *how* the system acquires the ability to respond to control signals. As illustrated in Figure 2.3, implementation methods vary in their training requirements, computational costs, and the degree to which they modify the underlying language model. We divide them into three primary categories: training-free implementation methods, training-based implementation methods, and hybrid approaches.

2.3.4.1 Training-Free Methods

Training-free methods enable control without modifying the parameters of the underlying language model, leveraging existing model capabilities and requiring no task-specific training data for the control task. The choice of control implementation method constrains which integration strategies are feasible: training-free approaches typically use ***Input Addition*** (as in ***LM Prompting***), though ***System Addition*** is also possible when incorporating pre-existing external components such as off-the-shelf sentiment classifiers or toxicity detectors that require no further training. However, if a ***System Addition*** approach requires training new components specifically for the control task, the overall implementation is classified as Training-Based (if only the external system is trained) or Hybrid (if both the external system and base model are trained), rather than training-free.

LM Prompting uses the instruction-following and in-context learning capa-

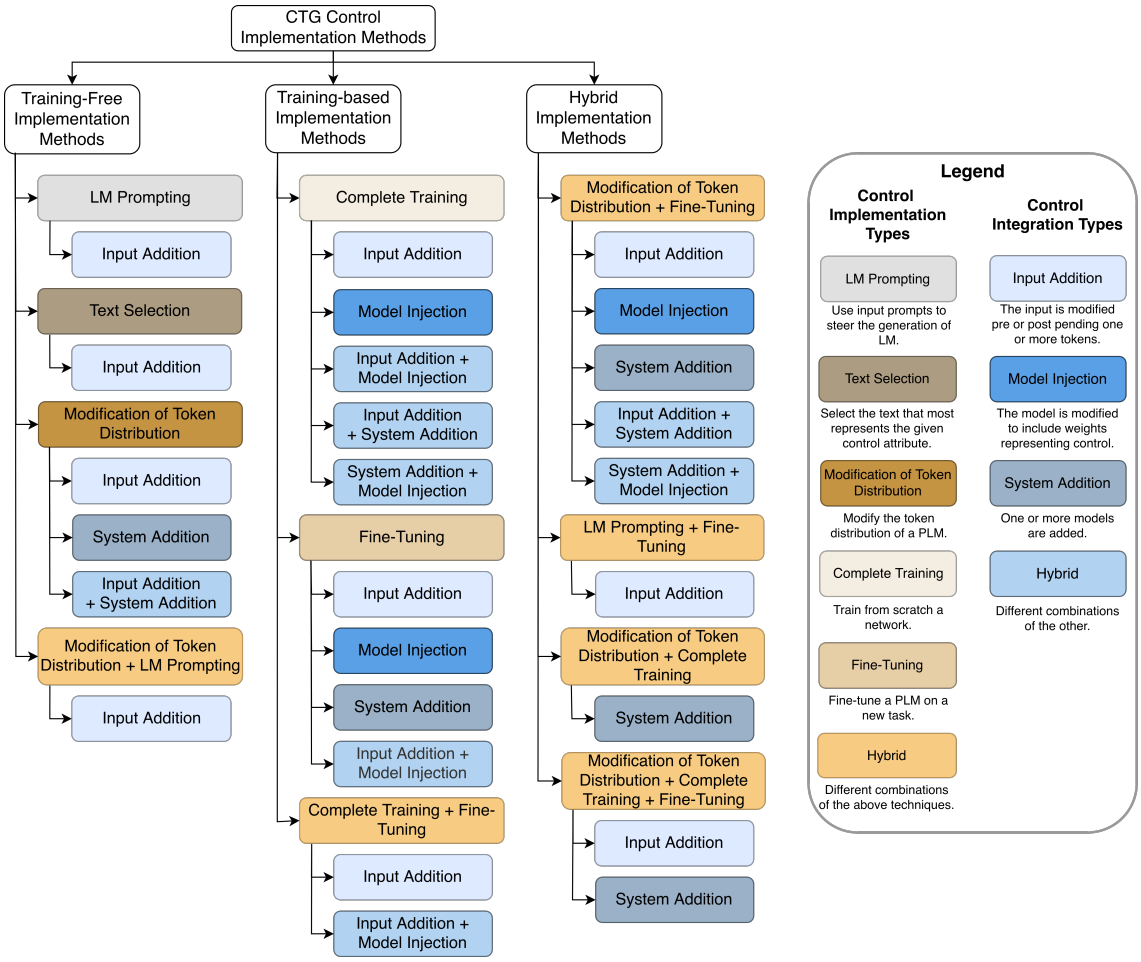


Figure 2.3: Taxonomy of control implementation methods in CTG systems, organised by training requirements. Methods range from training-free approaches (*LM Prompting*, *Text Selection*, *Modification of Token Distribution*) through training-based methods (*Fine-Tuning*, *Complete Training*) to *Hybrid* approaches that combine multiple methods.

bilities of pre-trained models through carefully crafted input prompts, instructions, or natural language descriptions. Prompting strategies range from simple zero-shot and few-shot approaches to more sophisticated techniques such as chain-of-thought prompting. Control is achieved entirely through input design, without modifying the model's parameters, architecture, or output distribution.

Text Selection implements control through a generate-and-filter paradigm: the system generates multiple candidate outputs and selects the one that best satisfies the desired control attributes. This might include best-of-N sampling, where candidates are scored by an attribute classifier, and rejection sampling, where generation is repeated until an acceptable output is produced.

Modification of Token Distribution adjusts the output probability distribution during decoding to favour tokens that align with target control attribute values, while leaving the base model's parameters unchanged. For example, PPLM (Dathathri et al., 2019) updates token probabilities at each generation step using gradients from attribute discriminators, while FUDGE (Yang and Klein, 2021) uses classifiers that predict future attribute values to guide current token selection. Control is applied dynamically at inference time through distribution manipulation rather than parameter modification.

2.3.4.2 Training-Based Methods

Training-based methods enable control capabilities by modifying the model's parameters through a training procedure, typically offering deeper integration of control mechanisms at the cost of requiring control-specific training data and computational resources.

Fine-Tuning adapts a pre-trained language model through continued training on control-specific data or with control-oriented objectives. Fine-tuning approaches vary along two key dimensions: (i) the training objective used, and (ii) which parameters are subject to training. Training objectives include supervised fine-tuning on attribute-annotated datasets (e.g., sentiment-labelled text) or reinforcement learning with attribute-specific reward signals that encourage desired control behaviours. Regarding parameters, fine-tuning may update all model parameters (full fine-tuning) or employ parameter-efficient methods such as LoRA (Hu et al., 2022) or adapter modules that introduce control capabilities by training only a small subset of additional or modified parameters. The key distinction from Complete Training is that fine-tuning starts from an existing pre-trained model, leveraging its language understanding while specialising it for controlled generation.

Complete Training builds control capabilities into a language model from the ground up, training from scratch with control mechanisms embedded in the training objective, architecture, or data. This includes models trained on control-code-

conditioned data from initialisation and models with control-specific architectural components designed into the network from the outset. The distinguishing feature is that the model is not adapted from an existing pre-trained model but trained end-to-end for controlled generation.

2.3.4.3 Hybrid Methods

Many CTG systems combine multiple implementation strategies to leverage their complementary strengths. Hybrid approaches can combine methods within the same category or across categories. Within training-free methods, a common pattern (Xu et al., 2020; Han et al., 2024; Hong et al., 2024) combines prompting for high-level control specification with fine-tuning for fine-grained steering. Across categories, a typical pattern pairs fine-tuning, to embed basic control capabilities, with token distribution modification at inference time for further refinement (Krause et al., 2021; Liu et al., 2021; Zhang and Song, 2022; Yu et al., 2024).

2.4 Development of the Categorisation Scheme

This section describes how we developed and validated our categorisation scheme through systematic analysis of the CTG literature, and how this scheme serves multiple purposes within the thesis. The categorisation scheme enables three key contributions: (i) it provides the analytical framework for our systematic review of CTG methods in Chapter 3, (ii) it allows us to position our attribute-specialised PEFT approach (Chapters 5 and 6) within the broader research landscape, and (iii) it facilitates identification of under-explored research opportunities through comprehensive analysis of field-wide patterns and trends.

The development process involved four separate rounds of paper selection from the ACL Anthology, each serving a distinct purpose, described in detail in Section 2.4.1. This staged approach ensures both the robustness of our categorisation scheme, validated on papers not used in its development, and the currency of our analysis, covering research through 2025.

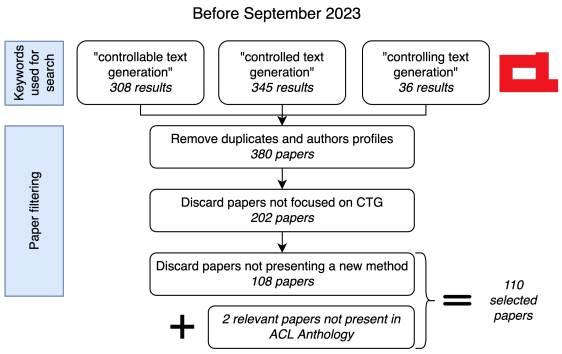
We begin by describing the systematic paper selection process (Section 2.4.1), followed by the manual annotation procedure used to develop and validate the scheme (Section 2.4.2), and conclude with statistics on the resulting dataset composition (Section 2.6).

2.4.1 Paper Selection Process

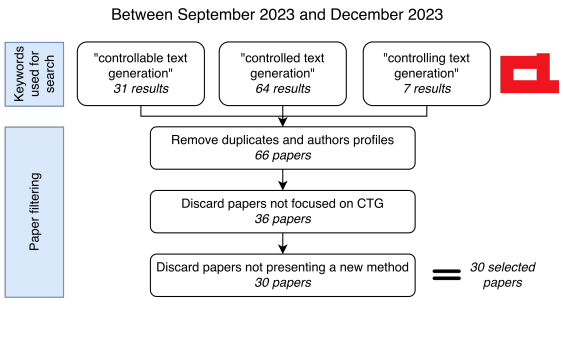
We conducted systematic literature searches of the ACL Anthology across four separate rounds, each serving a distinct purpose. All rounds applied consistent selection criteria: we identified papers presenting novel CTG techniques that enable explicit control over one or more attributes of generated text using control signals, whether explicit (e.g., control codes, attribute labels) or implicit (e.g., learnt representations). We included techniques across diverse control scenarios, from single-attribute to multiple-attribute control, and across different generation tasks, from structured data-to-text generation to open-ended creative writing. We manually excluded papers that merely mentioned controllable generation without proposing control techniques, as well as papers focused solely on dataset creation or evaluation methodologies.

Rounds 1–2: Development and Validation (up to December 2023). Our initial search (Figures 2.4a and 2.4b) covered publications up to December 2023, using three keyword strings: "controllable text generation", "controlled text generation", and "controlling text generation". After removing duplicates and non-paper resources, the search yielded 446 candidate papers. We applied a two-stage filtering process: first, removing papers not directly related to CTG (reducing to 238 papers); then retaining only papers presenting novel control methods or architectures (yielding 138 papers from the ACL Anthology). We additionally included two foundational papers (Dathathri et al., 2019; Keskar et al., 2019) not indexed in the ACL Anthology but widely cited within our corpus, resulting in 140 papers covering publications through December 2023.

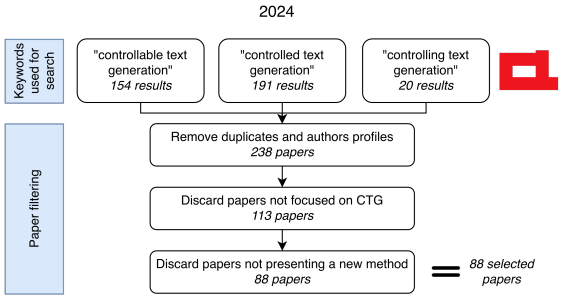
We divided these 140 papers into training and validation sets by publication date. Papers published up to August 2023 (110 papers) formed our training set for iterative development of the categorisation scheme (Figure 2.4a). Papers published between September and December 2023 (30 papers) served as a validation set to assess whether the scheme generalised beyond the papers used in its development (Figure 2.4b).



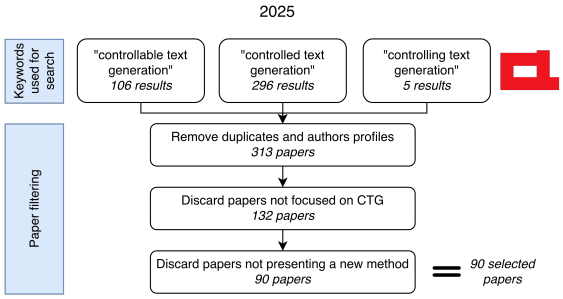
(a) Selection process for papers published up to August 2023, used to create the categorisation scheme.



(b) Selection process for papers published between September and December 2023, used to validate the scheme.



(c) Paper selection process for 2024 publications, following the same systematic process. These papers were annotated using semi-automated methods with manual verification (Section 2.5).



(d) Paper selection process for 2025 publications, following the same systematic process. These papers were annotated using semi-automated methods with manual verification (Section 2.5).

Figure 2.4: Paper selection process for all papers analysed in our systematic survey. The consistent process across all time periods ensures comparable coverage and selection criteria.

Rounds 3–4: Temporal Analysis (2024–2025). To assess whether the categorisation scheme remained applicable as the field evolved, we replicated the systematic selection process for papers published in 2024 and 2025, applying identical keyword strings and filtering criteria. This process identified 88 papers from 2024

(Figure 2.4c) and 90 papers from 2025 (Figure 2.4d). Successfully categorising these papers, published more than a year after the scheme's development, would demonstrate the scheme's temporal stability and continued relevance. These papers were annotated using a semi-automated approach combining LLM-based initial annotation with manual verification (Section 2.5). This approach served dual purposes: it enabled efficient processing of the substantial volume of recent papers while maintaining annotation quality, and it provided a practical demonstration of scalable systematic review process. If our categorisation scheme could be reliably automated, it would enable ongoing tracking of CTG research as the field continues to evolve, rather than requiring complete manual re-annotation with each new wave of publications.

Scope limitations of the keyword-based search. This keyword-based search strategy inherently limits the scope of papers identified: a substantial body of relevant work, including most data-to-text generation and text style transfer papers, as well as papers published before 2018 (when the term had not yet become established), does not use "controllable/controlled/controlling text generation" explicitly despite falling within our scheme's scope. We nonetheless adopted this approach to systematically scan the ACL Anthology using search terms that are as general as possible while remaining tractable for manual review, and and mitigated it partially by also including papers that did not match the keywords directly but cited a paper that did. Even so, relevant work that neither uses the target keywords nor cites a paper that does may have been excluded.

2.4.2 Manual Annotation Procedure

Scheme Development through Qualitative Analysis. The categorisation scheme was created through systematic qualitative analysis of the 110 training papers, following established approaches in qualitative data analysis. We began by reading papers and noting CTG properties and elements without imposing predetermined

categories. For instance, observing that some systems accepted sentiment labels as input while others modified decoder probabilities, or that some controlled sentiment while others controlled formality or length. Through close reading and comparison across papers, we identified recurring patterns and commonalities: papers varied in what they controlled (attributes), where control information entered the system (integration), and how control was technically achieved (implementation). These patterns formed the initial basis for our three-dimensional categorisation scheme.

The scheme underwent iterative refinement through multiple revision cycles, each triggered by encountering papers that revealed ambiguities or gaps in the current categories. The refinement process involved five recurring stages: (i) initial annotation of a subset of papers using preliminary categories; (ii) identification of unclear cases, boundary ambiguities, or missing categories not adequately captured; (iii) refinement of category definitions, decision criteria, and boundaries between related categories; (iv) re-annotation of previously coded papers under the updated scheme to ensure consistency; and (v) verification of consistency across the full training set. For instance, early iterations did not distinguish between Model Injection (selecting specialised internal components) and System Addition (external modules), but analysis of papers using both approaches revealed this as a meaningful distinction requiring separate categories. Similarly, the Control Attributes taxonomy underwent several revisions as we encountered papers controlling increasingly diverse properties, requiring us to develop a hierarchical structure distinguishing Content from Non-Content attributes, and further subdividing each branch.

This iterative process continued until we reached stability: no new edge cases in the training set required modification of the scheme, all 110 papers could be coherently categorised, and category definitions had sufficient precision to support consistent annotation decisions.

Annotation Dimensions. For each paper in the training set, we manually extracted information across six dimensions: (1) control attributes—the properties of generated text being manipulated; (2) control integration strategies—the architec-

tural level at which control signals enter the system; (3) control implementation methods—the technical mechanisms through which control is achieved; (4) model architectures employed; (5) datasets used for training and evaluation; and (6) evaluation metrics and methodologies. Dimensions (1)–(3) correspond directly to the three axes of our categorisation scheme (Section 2.3), while dimensions (4)–(6) constitute supplementary annotations enabling the broader field analysis reported in Section 2.6.

Validation on Held-Out Papers. We validated the scheme's generalisability by applying it to the 30 held-out validation papers published between September and December 2023. Importantly, no instances were encountered that required modification of the categorisation scheme: all validation papers could be classified using the existing categories without creating new ones or refining existing definitions. This result suggests that our scheme captures enduring dimensions of variation in CTG techniques rather than idiosyncrasies of the training set, and that it generalises to papers published after the development period without requiring ongoing revision.

2.5 Automatic Annotation of CTG Papers

Systematic reviews play a critical role in synthesising research evidence, identifying patterns, gaps, and trends that may not be apparent from individual studies. However, they are notoriously resource-intensive (Kitchenham et al., 2004): comprehensive literature search, paper selection, information extraction, and consistent categorisation all require substantial time and effort. This creates a practical challenge in fast-moving fields such as NLP, where new work is published at a pace that can render a review incomplete or outdated before it is finished (Rohatgi et al., 2023).

Having developed and validated our categorisation scheme on 140 papers published through December 2023 (Section 2.4), we identified two compelling reasons to extend the survey to more recent publications. First, bringing the survey up to date

through 2025 would enable temporal analysis of how the CTG field has evolved, revealing trends such as shifts in methodological approaches, the emergence of new control attributes, or changes in evaluation practices. Second, successfully categorising papers published in 2024–2025, more than a year after the scheme's development, would provide additional validation of the scheme's temporal stability and continued applicability as the field develops.

However, the systematic search identified 196 papers published in 2024–2025 alone, exceeding the entire manually annotated corpus of 140 papers spanning 2018–2023. This acceleration in publication rate made purely manual annotation of recent papers prohibitively time-consuming, while the need for up-to-date analysis remained pressing. These circumstances motivated the investigation of whether the annotation process could be reliably automated, enabling efficient processing of the substantial volume of recent publications while maintaining annotation quality and consistency with our categorisation scheme.

Automatic paper categorisation is not straightforward. Unlike open-ended generation or simple classification, it requires: (1) deep comprehension of technical content in domain-specific language; (2) reliable extraction of specific information elements according to a predefined schema; (3) consistent application of category definitions across papers with diverse writing styles and presentation formats; and (4) structured generation that respects categorical constraints and dependencies between annotation dimensions. Recent advances in LLMs make them promising candidates for this task, but whether they meet these requirements with sufficient reliability for use in systematic reviews remains an open question.

We investigate this question by formulating automatic annotation of CTG papers according to our categorisation scheme (Section 2.3) as a reasoning-augmented structured generation problem. This task is well-suited for evaluating automation: the scheme encompasses multiple orthogonal dimensions with complex category structures; we have gold-standard manual annotations for 140 papers; and successful automation would immediately enable analysis of the 196 recent papers not yet

manually annotated.

Main Contributions. This section makes the following contributions:

1. We formulate automatic paper categorisation as a reasoning-augmented structured generation problem, introducing a task formulation that is agnostic to the specific categorisation scheme and thus applicable to systematic reviews across different domains.

2. We evaluate multiple state-of-the-art methods for structured generation under realistic low-resource conditions (100 training samples), comparing reasoning-augmented approaches against baseline methods.

3. We conduct ablation studies examining the contribution of different components, including reasoning augmentation and training objectives, to final annotation quality.

4. We demonstrate the practical utility of our approach by applying the best-performing method to annotate the 196 papers published in 2024-2025, with manual verification confirming annotation quality sufficient for systematic literature analysis.

Generalisation to Other Categorisation Schemes. A key ambition of this work is to develop an annotation process that generalises beyond the specific CTG categorisation scheme, providing a reusable approach for systematic literature analysis across different NLP research topics. To test whether our approach achieves this generalisability within the NLP domain, we evaluate it on a second, independent task: classifying evaluation methods in NLP papers according to the Quality Criteria for Evaluations Taxonomy (QCET) (Belz et al., 2025). This task differs substantially from CTG categorisation in both the properties being classified (evaluation methodology rather than generation techniques) and the category structure, making it a rigorous test of whether our methodology succeeds only for the specific CTG framework or generalises to structured annotation tasks across NLP systematic

reviews. Success on both tasks would demonstrate that the approach could be applied to systematic reviews of other NLP research areas, enabling scalable literature analysis beyond CTG while remaining within the domain of NLP research.

Our approach generalises successfully to this second domain. On paper-level evaluation type classification, SFT+RL achieves 78.7% exact match, the best result overall and 7.4 points above the strongest pre-trained baseline (Qwen3 14B, 71.3%). On the harder measure-level variant, SFT+RL reaches 30.7% exact match, exceeding the strongest pre-trained baseline (LLaMA 3.3 70B Instruct, 19.3%) by 11.4 points. The gains are most pronounced on the most complex task, hierarchical QCET node classification: none of the pre-trained models achieve a single fully correct prediction (0.0% exact match across all three levels), whereas our SFT+RL approach achieves 18.0% exact match, with reinforcement learning improving over SFT alone at every hierarchy level. These results indicate that the benefit of our reasoning-augmented fine-tuning approach over pre-trained baselines increases with task complexity, supporting its applicability beyond the CTG domain. Full details of the QCET experiments are provided in Appendix A.2.

2.5.1 Task Definition and Formulation

We frame automatic paper annotation as a low-resource structured generation task, reflecting a realistic setting in which the value of automation would be substantially diminished if a large manually annotated dataset had to be constructed first. Our experimental setup uses papers from the first two collection rounds described in Section 2.4.1. From the 110 training papers (Round 1, published up to August 2023), we randomly selected 100 papers for training our models and 10 for validation during development. The 30 papers from Round 2 (published September–December 2023) served as our held-out test set for final evaluation. This division reflects realistic annotation budgets in academic research while allowing us to study the data efficiency of different approaches under constrained conditions, and crucially ensures that test papers were published after the training period, simulating real-

world deployment where models must categorise newly published research.

Our primary experimental task is CTG system categorisation: classifying control techniques employed in CTG systems according to the framework presented in Section 2.3. The task has a fixed four-field structure, in which three of the four fields involve multi-label predictions, requiring the model to identify all applicable categories rather than selecting a single best match.

To characterise the complexity of this task, we analyse each annotation field along three standard dimensions of classification problems. Fields may involve *single-label* classification (assigning one value per paper) or *multi-label* classification (assigning multiple values per paper). They may be *binary* (two possible classes) or *multi-class* (three or more possible classes). Finally, they may involve *closed-set* classification (selecting from a fixed, predefined taxonomy) or *open-set* classification (extracting labels from an unbounded vocabulary within the paper, requiring normalisation).

Table 2.1 presents the classification characteristics of each annotation field. The Control Type field requires a single binary decision from a closed set (single-attribute versus multiple-attribute). The Control Implementation and Control Integration fields both require multi-label, multi-class prediction from closed taxonomies, papers may employ multiple implementation methods or integration strategies simultaneously, selected from the predefined categories in our scheme. The Control Attributes field presents the greatest complexity: it requires multi-label, multi-class prediction from an open set, meaning the model must extract and normalise attribute names (e.g., "sentiment", "formality", "toxicity") directly from the paper text rather than selecting from a fixed list. Together, these properties make CTG categorisation a challenging structured generation task that tests model capabilities across multiple dimensions simultaneously.

2.5.1.1 Task formulation

We formulate each annotation task as a conditional text generation problem with reasoning augmentation. Given the text of a paper x, a property-value question q

Task	Field	Class	Label	Set
CTG	Control Type	Single	Binary	Closed
	Control Implementation	Multi	Multi	Closed
	Control Integration	Multi	Multi	Closed
	Control Attributes	Multi	Multi	Open

Table 2.1: Characteristics of annotation tasks. **Field**: The annotation dimension being classified. **Class**: Single=single-label (one value per instance), Multi= multi-label (multiple values per instance). **Label**: Binary=2 possible values, Multi=3+ possible values. **Set**: Closed=fixed taxonomy, Open=unbounded vocabulary.

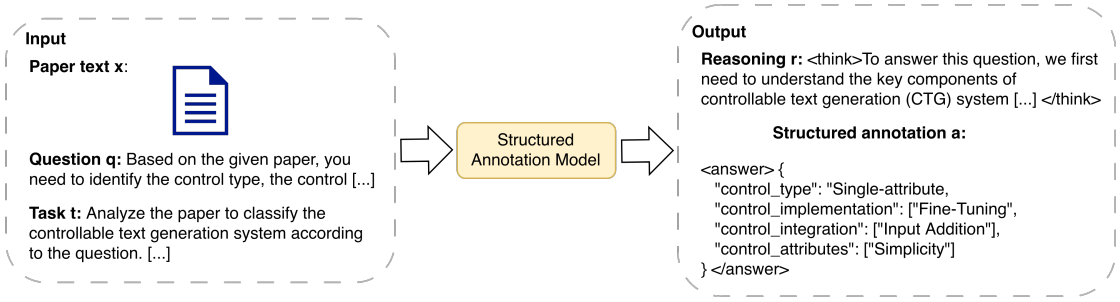


Figure 2.5: Task formulation for automatic paper categorisation. The model receives paper text (x), a property-value question (q), and task description (t) as input, and generates a reasoning trace (r) explaining the decision process followed by structured annotations (a) in JSON format containing categorisation labels for all fields.

about the system proposed in the paper, and a task description t, the model generates two components: a reasoning trace r and structured annotations a (Figure 2.5). The reasoning trace, enclosed within `<think>...</think>` tags, provides step-by-step justification for annotation decisions by explicitly referencing evidence within the paper, such as method descriptions, experimental results, or author statements. The structured annotations are produced in JSON format following the schema specified in q, containing the final categorisation decisions for all required fields, with multi-label fields represented as lists.

The reasoning trace serves three important purposes. First, it encourages the model to engage in systematic analysis of paper content rather than relying on superficial pattern matching. Second, it supports transparency by making the model's decision process explicit and examinable. Third, it enables efficient human verification: rather than re-reading entire papers to verify each annotation, reviewers can examine the reasoning traces to assess whether the model has identified appropriate

evidence and applied sound logic. Annotations whose reasoning references relevant paper sections and follows coherent argumentation can be accepted with higher confidence; those whose reasoning appears flawed or unsupported can be flagged for manual review. This verification pathway substantially reduces the effort required to maintain annotation quality while preserving the rigour essential for systematic literature analysis.

2.5.2 Methodology

Our experimental methodology consists of three phases: (i) dataset creation, comprising manual annotation of gold labels and automatic generation of synthetic reasoning traces; (ii) Supervised Fine-Tuning (SFT) using long Chain-of-Thought (CoT) reasoning, in which models are trained to produce explicit step-by-step justifications alongside structured annotations; and (iii) Reinforcement Learning (RL) refinement using Decoupled Clip and Dynamic sAmpling Policy Optimisation (DAPO) (Yu et al., 2025), which further aligns model outputs with annotation quality objectives under low-resource conditions. Each phase is described in detail below, with particular attention to the design decisions motivated by the constrained annotation budget of 100 training samples.

2.5.2.1 Dataset creation

To train and evaluate our annotation models, we created a dataset by preprocessing the papers collected during Rounds 1–2 (Section 2.4.1) and augmenting them with synthetically generated reasoning traces. These 140 papers, which had been manually annotated as described in Section 2.4.2, served as the source material but required substantial preprocessing before they could be used for model training. Specifically, we needed to: (i) extract and clean the full text of each paper, (ii) formulate the annotation task as a structured generation problem, and (iii) generate reasoning traces that would support the gold-standard annotations. The resulting dataset thus builds upon the manually annotated papers from Rounds 1–2 but trans-

forms them into a format suitable for training and evaluating LLM-based annotation models.

Paper collection and manual annotation. We use all 140 papers identified through the systematic screening process described in Section 2.4.1, covering publications up to December 2023. Respecting temporal ordering, we allocate papers published up to August 2023 to the training pool (110 papers) and papers published between September and December 2023 to the held-out test set (30 papers). From the 110 training-pool papers, we randomly select 10 for validation, leaving 100 for training. All papers were manually annotated according to the procedure described in Section 2.4.2. For the present experiments, we use all four fields from our categorisation scheme (Section 2.3): Control Type, Control Implementation, Control Integration, and Control Attributes. These fields fully characterise any CTG system's control approach. While we additionally annotate supplementary information such as model architectures, evaluation datasets, and evaluation metrics for the comprehensive field analysis in Section 2.6, these are not included in the automatic annotation task as they serve descriptive rather than categorisation purposes.

Paper preprocessing. We preprocess each PDF into cleaned text to enable models to focus on substantive content while managing computational constraints. PDFs are converted to text using Docling[1], after which we remove sections that are either uninformative for categorisation or excessively lengthy: reference lists, author lists, ethical statements, limitation discussions, acknowledgements, and appendices. While appendices occasionally contain methodological details, their inclusion would substantially increase input length and was not practical given available computational resources.

We additionally remove LaTeX artefacts not properly decoded during PDF extraction. These are identified using regex-based pattern matching for: (i) Docling placeholder tags such as `<!-- formula-not-decoded -->` and `<!-- image -->`,

[1] `https://ds4sd.github.io/docling/`

indicating failed conversion of mathematical expressions or figures; (ii) residual LaTeX syntax including escaped backslashes (\\), malformed dollar signs ($\), and orphaned commands; and (iii) lines containing only single characters, which typically result from corrupted image captions or formatting. This cleaning ensures the model receives coherent text without distracting artefacts.

The resulting text contains the title, abstract, and main body of each paper. Two papers were excluded after preprocessing revealed severe extraction errors: watermarks overlaid on the paper content interfered with PDF-to-text conversion, causing Docling to produce output consisting almost entirely of `<!-- image -->` placeholder tags with the underlying text content inaccessible. This yielded a final dataset of 99 training papers, 10 validation papers, and 29 test papers.

Synthetic reasoning trace generation. Reasoning-augmented SFT training requires examples that include not only correct annotations but also explicit reasoning traces explaining how those annotations derive from paper content. Manually writing such traces for all training papers would substantially offset the efficiency gains that motivate automation. We therefore generate synthetic reasoning traces automatically (see Figure 2.6) using LLaMA 3.3 70B Instruct (Grattafiori et al., 2024), which produces explanations referencing specific evidence within each paper and connecting that evidence to the final categorisation decisions.

Traces are generated with temperature 0.6, a maximum length of 4,096 tokens, and the prompt template shown in Figure 2.7. The prompt instructs the model to: (1) read the paper carefully; (2) identify relevant evidence for each categorisation field; (3) explain step-by-step how that evidence supports specific category assignments; and (4) output the reasoning within `<think>...</think>` tags followed by structured annotations within `<answer>...</answer>` tags as a JSON object.

Each generated trace undergoes automatic validation before inclusion in the training dataset, checking three requirements: (i) properly formatted `<think>...</think>` and `<answer>...</answer>` tags with non-empty content; (ii) valid JSON within `<answer>...</answer>` tags conforming to the expected schema; and (iii)

Example of generated reasoning trace

```
To extract the control type, control attributes, control implementation, and control integration from the given paper, let's follow the steps outlined in the instructions.

1. **Initial Analysis**: The question asks for the control type, control attributes, control implementation, and control integration used in the proposed system. The possible answer categories for control type are Single-attribute and Multiple-attribute. Control attributes could be various, such as sentiment, topic, length, etc. Control implementation could be LM Prompting, Text Selection, Modification of Token Distribution, Complete Training, Fine-Tuning, or Hybrid. Control integration could be Input Addition, Model Injection, System Addition, or Hybrid.

2. **Systematic Paper Review**:
    - The abstract and introduction indicate that the paper proposes a joint framework with a topic refinement mechanism for topic-aware dialogue response generation. This suggests that the system is focused on controlling the topic attribute.
    - The methodology section describes a three-pass mechanism involving response generation, topic prediction, and topic refinement, which implies that the system is designed to control the topic attribute explicitly.
    - The results and experiments show that the system outperforms previous state-of-the-art models in terms of topic-related metrics, further supporting the focus on topic control.

3. **Evidence Collection**:
    - The sentence "We propose a joint framework with a topic refinement mechanism to solve the topic-aware multi-turn end-to-end dialogue generation problem" supports the idea that the system controls the topic attribute.
    - The description of the three-pass mechanism (response generation, topic prediction, and topic refinement) in the methodology section provides evidence for how the topic attribute is controlled.
    - The use of specific models like GPT2 and BERT for topic prediction and response generation implies fine-tuning as the control implementation method.

4. **Category Matching**:
    - **Control Type**: Since the system is primarily focused on controlling the topic attribute, the control type is Single-attribute.
    - **Control Attributes**: The primary attribute controlled by the system is the topic.
    - **Control Implementation**: The system fine-tunes pre-trained models (GPT2 and BERT) for topic prediction and response generation, indicating Fine-Tuning as the control implementation.
    - **Control Integration**: The control signal (topic information) is integrated into the system through the input, as the topic prediction and refinement stages are directly influenced by the input dialogue context and previous topics, suggesting Input Addition.

5. **Verification**:
    - The evidence supports that the system is designed to control the topic attribute, making Single-attribute the correct control type.
    - Topic is the primary attribute being controlled.
    - Fine-Tuning is the correct implementation, as the system adapts pre-trained models for its tasks.
    - Input Addition is the correct integration method, as the topic information is used to influence the generation process through the input stages of the system.
```

Figure 2.6: Example of a generated reasoning trace in the training set using LLaMa3.3 70B Instruct.

category labels exactly matching the gold-standard manual annotations for all four fields. Traces failing any check are regenerated. All instances produced valid traces within a maximum of six attempts, indicating that the task formulation and prompt

Think Traces Generation prompt

```
<paper>
{paper_text}
</paper>

<question>
{question}
</question>

Your task is to produce a reasoning to explain how to extract the {question_type} from the paper text.

The expected answer is:
<answer>
{answer}
</answer>

Instructions for reasoning:
1. Initial Analysis: First, identify what specific aspect of the CTG system the question is asking about and what the possible answer categories are.
2. Systematic Paper Review:
    - Start with the abstract and introduction to understand the proposed system
    - Examine the methodology/approach section for technical details
    - Review results/experiments to see how the system works in practice
    - Check any architectural diagrams or system descriptions
3. Evidence Collection:
    - Quote specific sentences or phrases that directly support your answer
    - Note section names/numbers where you found key information
    - If information is implicit, explain your inference process
4. Category Matching:
    - For each possible category, explain why it does or doesn't apply
    - Pay special attention to edge cases and hybrid approaches
    - If the answer involves multiple components (like "Hybrid"), identify all relevant parts
5. Verification:
    - Double-check that your chosen category aligns with all the evidence
    - Consider if there are any contradictory statements in the paper
    - Assess your confidence level and note any ambiguities

Special considerations:
- For Hybrid categories: Clearly identify each component and provide evidence for all parts
- For attribute identification: Look beyond obvious mentions, consider implicit attributes controlled through the methodology
- For technical distinctions: Be precise about implementation details (e.g., distinguishing between fine-tuning vs. prompting)

Encapsulate your complete reasoning process between <think> and </think> tags, following the structure above.
<think>
[Your detailed reasoning here]
</think>
<answer>
[Your final answer here aligned to the answer format and reasoning provided]
</answer>
```

Figure 2.7: Prompt used to generate reasoning traces with LLaMa 3.3 70B Instruct. Placeholders are shown in {braces}.

are well-suited to the generation requirements. An example trace is provided in Figure 2.6.

Dataset statistics. Table 2.2 presents statistics for the final dataset. Paper lengths range from approximately 2,100 to 13,700 tokens (mean ≈ 8,900 tokens for training papers), measured using the Qwen3 8B tokeniser. Reasoning traces average approximately 565 tokens, ranging from 200 to 900 tokens; longer traces correspond to papers describing multiple-attribute control systems with multiple implementation methods, which require more extensive justification across all categorisation fields. Test set papers do not include reasoning traces, as these are generated by the model during inference.

	# samples	Paper text			Reasoning trace		
		Max	Min	Avg	Max	Min	Avg
Train	99	13673	3902	8905.96	900	200	564.8
Validation	10	11118	6785	9151.4	684	337	548.2
Test	29	12141	2134	8116.6	-	-	-

Table 2.2: Dataset statistics for the CTG annotation task. Paper text and multi-field reasoning trace lengths are measured in tokens using the Qwen3 8B tokenizer. Test set reasoning traces are not used (model generates them during inference).

2.5.2.2 Model architecture

Figure 2.8 presents the overall architecture of our automatic annotation system. The diagram shows how the preprocessed papers and synthetic reasoning traces, previously described, flow through our two-phase training procedure to produce the final annotation model used at inference time.

Our automatic annotation system consists of two main training phases: (i) supervised fine-tuning (SFT) to teach the model the annotation task using gold-standard examples, and (ii) reinforcement learning (RL) refinement to further improve performance using reward signals based on annotation correctness and output format compliance.

Within the SFT phase, we explore two variants: SFT without reasoning traces (where the model directly generates structured annotations) and SFT with reasoning traces (where the model first produces step-by-step justification before generating

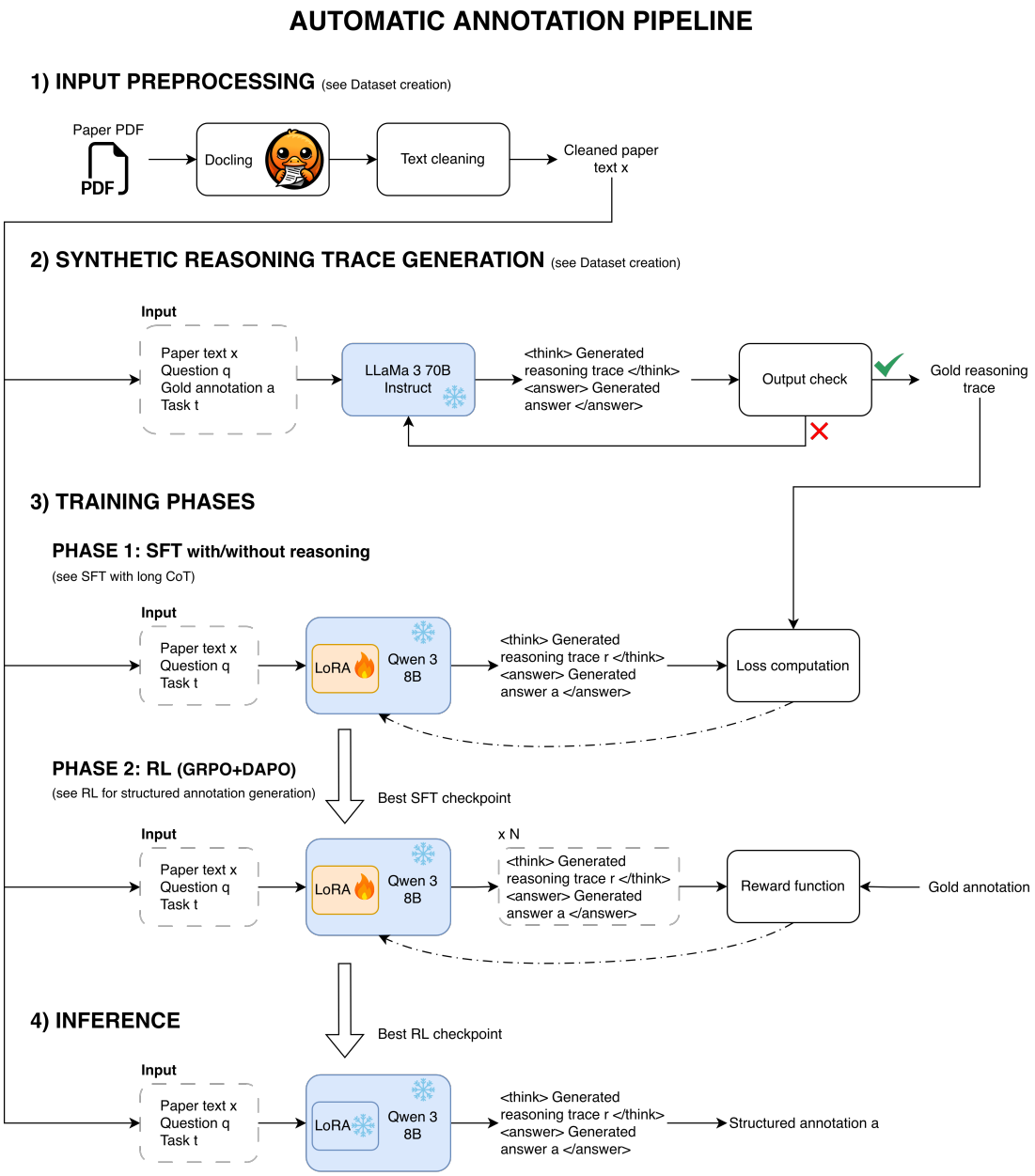


Figure 2.8: Overview of the automatic annotation pipeline. (1) Input preprocessing converts PDFs to cleaned text via Docling and text cleaning (see 2.5.2.1 Dataset Creation). (2) Synthetic reasoning traces are generated using LLaMA 3.3 70B Instruct and validated against gold annotations (see 2.5.2.1 Dataset Creation). (3) Training proceeds in two phases: supervised fine-tuning (SFT) with or without reasoning traces (see 2.5.2.3 SFT with long CoT), followed by reinforcement learning (RL) refinement using GRPO with DAPO loss (see 2.5.2.4 RL for structured annotation generation). The best SFT checkpoint is selected for RL training, and the best RL checkpoint is used for inference. (4) The final model generates reasoning traces and structured annotations for new papers at inference time. All training uses Qwen3 8B with LoRA (r=32, α=64).

annotations). The RL phase builds upon the best-performing SFT checkpoint, using reward signals derived from exact match with gold annotations, format compliance,

and JSON validity to further refine predictions.

Multi-Field Architecture. We adopt a multi-field architecture that predicts all four annotation dimensions jointly within a single model, rather than training separate models for each dimension. This design offers three key advantages that become particularly important for our training approach.

First, joint prediction enables the model to capture inter-dependencies between annotation dimensions. Papers describing prompting-based control (Control Implementation) are more likely to use input-level control integration; such relationships can inform more accurate predictions when dimensions are modelled jointly rather than independently.

Second, joint prediction is computationally efficient, requiring only a single forward pass to annotate all dimensions rather than four separate inference calls. This efficiency matters both during inference and during RL training, where multiple rollouts are generated for reward computation.

Third, the multi-field formulation provides naturally dense reward signals for RL fine-tuning. In single-field settings, correctness rewards are necessarily binary: a prediction is either fully correct or fully incorrect, producing sparse gradient signals that can prevent stable training. With multi-field outputs, however, partially correct predictions receive intermediate rewards. A model that correctly predicts two of four fields (e.g., Control Type and Control Implementation) receives partial credit proportional to the number of correct predictions, even if the remaining fields are incorrect. This reward structure enables more stable and sample-efficient RL training by providing informative feedback even when predictions are imperfect, a critical property when working with limited training data (99 papers).

Input-Output Structure. The model follows the formulation defined in Section 2.5.1: it receives the preprocessed paper text x and a composite query q describing all four annotation dimensions, and generates a reasoning trace r (Figure 2.5) within `<think>...</think>` tags followed by structured annotations a

within `<answer>...</answer>` tags as a JSON object.

2.5.2.3 SFT: Standard and Long CoT

Following dataset creation, we fine-tune pre-trained language models using supervised learning on the reasoning-augmented training samples. We use parameter-efficient fine-tuning via Low-Rank Adaptation (LoRA) (Hu et al., 2022), which substantially reduces the number of trainable parameters while maintaining model performance, making it well-suited to our low-resource setting.

Each training instance is represented as a tuple (x, q, y), where x is the preprocessed paper text, q is the composite categorisation query defining all four annotation dimensions, and y is the target output. The target output comprises the structured annotation a, a JSON object containing gold-standard labels for all four fields, optionally preceded by a synthetic reasoning trace r (Section 2.5.2.1) that provides step-by-step justification for the annotation decisions.

We explore two SFT variants that differ in the structure of the target output y. In **standard SFT**, the model is trained to map the input directly to the structured annotation, without the reasoning trace: $y =$ `<answer>`a`</answer>`. In **long Chain-of-Thought (CoT) SFT**, the target additionally includes the reasoning trace preceding the structured output: $y =$ `<think>`r`</think><answer>`a`</answer>`. Unless otherwise qualified, "SFT" is used throughout the remainder of this thesis to refer to the standard variant; the long CoT variant is explicitly identified as such wherever it is used. Comparing the two variants isolates the contribution of reasoning augmentation to annotation quality (Section 2.5.4).

Given a model $p_\theta(y \mid x, q)$ parametrised by θ, we optimise the standard autoregressive objective:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{t=1}^{T} \log p_\theta(y_t \mid y_{<t}, x, q) \tag{2.1}$$

where T is the total number of tokens in the target output y. This objective trains the model to generate outputs token by token, conditioning on both the input paper x and the categorisation query q, while respecting the sequential dependencies

between tokens in the reasoning trace and structured annotations.

This approach is a form of *long chain-of-thought* (long CoT) SFT (Yeo et al., 2025; Chen et al., 2025), which trains models to generate extended reasoning traces before producing final answers. Unlike standard SFT, which maps inputs directly to outputs, or short CoT approaches, which include only brief explanations, long CoT training encourages systematic multi-step reasoning prior to committing to a final prediction. At inference time, the model first generates a `<think>` block that analyses the paper content and justifies each categorisation decision with reference to specific passages, then outputs the structured JSON annotations within an `<answer>` block. As discussed in Section 2.5.1, this two-stage process supports both annotation accuracy and human verification of model outputs.

The long CoT approach is particularly well suited to scientific paper categorisation, which requires integrating information distributed across multiple sections. Identifying control attributes, for instance, may require synthesising evidence from the introduction (research motivation), methods (implementation details), and experiments (evaluation scenarios). The extended reasoning trace allows the model to gather and connect this distributed evidence before producing final categorisation decisions, rather than relying on local pattern matching within individual sections.

2.5.2.4 RL for structured annotation generation

To further improve annotation quality beyond SFT, we apply RL to optimise the model's ability to produce correct, well-formed outputs. In RL, the model is treated as a *policy* that generates annotations given paper text, and learning is driven by *reward signals* that quantify how good each generated annotation is. The model learns to adjust its generation strategy to maximise expected rewards, encouraging outputs that score highly on our evaluation criteria while discouraging low-quality predictions.

We use Group Relative Policy Optimisation (GRPO) (Shao et al., 2024), a policy gradient algorithm designed for stable RL fine-tuning of language models, with

the DAPO (Direct Alignment through Pairwise Optimization) loss formulation (Yu et al., 2025). This combination enables stable policy optimisation using group-relative rewards, where each generation is evaluated relative to other samples from the same input, without requiring a separately trained value function or reference model.

In details, the model acts as a policy $\pi_\theta(y \mid x, q)$ that generates structured output y given paper x and categorisation query q, maintaining the same `<think>...</think>` and `<answer>...</answer>` structure as in SFT. For each training input (x, q), we sample $G = 8$ candidate completions $\{y_i\}_{i=1}^{G}$ from the current policy and score each using the composite reward function described below.

Rewards are normalised within each group of G completions to obtain relative advantages, reducing variance caused by differences in paper difficulty or query complexity:

$$\hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^{G})}{\text{std}(\{R_j\}_{j=1}^{G}) + \epsilon}. \tag{2.2}$$

where R_i is the raw reward for completion i and ϵ is a small constant for numerical stability. The DAPO loss formulation[2] applies token-level normalisation to handle variable-length sequences:

$$\mathcal{L}_{\text{DAPO}}(\theta) = -\frac{1}{\sum_{i=1}^{G} |y_i|} \sum_{i=1}^{G} \sum_{t=1}^{|y_i|} \Big(p_{i,t}(\theta)\, \hat{A}_{i,t} \Big), \tag{2.3}$$

where y_i is the i-th completion, $\hat{A}_{i,t}$ is the advantage for token t, $p_{i,t}(\theta)$ is the ratio of the current policy probability to a stop-gradient copy of itself, preventing gradient flow through the denominator and stabilising training:

$$p_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|q, y_{i,<t})}{[\pi_\theta(y_{i,t}|q, y_{i,<t}))]_{\text{no grad}}}. \tag{2.4}$$

Reward function. We design a composite reward function evaluating three aspects of generated outputs: annotation correctness, structural validity, and JSON

[2] `https://huggingface.co/docs/trl/main/grpo_trainer`

parsability.

The **correctness reward** measures annotation accuracy by comparing predicted labels against gold-standard annotations across all four fields F:

$$R_{\text{correct}} = \frac{1}{|F|} \sum_{f \in F} \begin{cases} \mathbb{1}\{A_f^{\text{pred}} = A_f^{\text{gold}}\} & \text{(exact match)} \\ \frac{|A_f^{\text{pred}} \cap A_f^{\text{gold}}|}{|A_f^{\text{pred}} \cup A_f^{\text{gold}}|} & \text{(Jaccard)} \end{cases} \tag{2.5}$$

where A_f^{pred} and A_f^{gold} are the predicted and gold label sets for field f. We experiment with both exact match (requiring perfect agreement) and Jaccard similarity (rewarding partial overlap) for all fields.

The **format reward** verifies the presence and correct ordering of structural tags (`<think>`, `</think>`, `<answer>`, `</answer>`):

$$R_{\text{format}} = \frac{n_{\text{correct}}}{n_{\text{total}}} \tag{2.6}$$

where n_{correct} counts correctly placed tags and n_{total} is the total number of expected structural elements, including the four tags and their required ordering.

The **JSON validity reward** checks that the content within `<answer>` tags constitutes valid, parsable JSON with all expected fields:

$$R_{\text{json}} = \frac{n_{\text{valid}}}{n_{\text{expected}}} \tag{2.7}$$

where n_{valid} counts successfully parsed fields matching the expected schema ($n_{\text{expected}} = 4$ in our task).

The final reward combines these components:

$$R(y) = w_{\text{correct}} R_{\text{correct}}(y) + w_{\text{format}} R_{\text{format}}(y) + w_{\text{json}} R_{\text{json}}(y) \tag{2.8}$$

where we set $w_{\text{correct}} = 0.7$, $w_{\text{format}} = 0.05$, and $w_{\text{json}} = 0.25$. This weighting prioritises prediction accuracy while ensuring structural validity. Format compliance receives low weight (0.05) as tag structure is simple and learnt early; higher weights

would over-optimise this constraint at the expense of correctness. JSON validity receives moderate weight (0.25) as it is both critical for usability and more challenging than format compliance. We did not conduct systematic ablation as our focus was on reward formulation rather than weight optimisation.

We experimented with both Jaccard similarity and exact match for the correctness reward across all fields, finding that exact match provides better training signal overall (see Section 2.5.5).

2.5.3 Experimental setup

2.5.3.1 Models

We compare our fine-tuned models against a range of open-source LLMs evaluated in a zero-shot setting. The zero-shot setup is motivated by practical constraints arising from long input lengths: preprocessed papers average approximately 8,900 tokens and can reach up to 14,000 tokens. Each few-shot demonstration would need to include both the full paper text and the complete structured output (reasoning trace plus JSON annotations), consuming an additional 10,000+ tokens per example. With typical context windows of 32,000 tokens for the models we evaluate, incorporating even 2–3 demonstrations would leave insufficient space for the target paper's full text, forcing truncation that could remove critical information about the system being categorised.

Furthermore, zero-shot inference with LLaMA 3.3 70B Instruct already required two A100 GPUs (80 GB each) due to model size and context length. Few-shot prompting would increase memory requirements further while also substantially increasing inference time, evaluating each test paper with 3 demonstrations would effectively quadruple the tokens processed per forward pass. Given that our experimental protocol involves averaging over 5 runs for each configuration to account for sampling variance, few-shot evaluation would have been computationally prohibitive.

We therefore use zero-shot evaluation throughout to ensure models can access

all relevant paper content without truncation while remaining within practical computational constraints.

We evaluate two model families spanning a range of parameter scales: Qwen3 (Team, 2025) (8B, 14B, and 70B parameters) and LLaMA 3.3 (Grattafiori et al., 2024) (8B and 70B parameters), covering small, medium, and large scales. This allows us to assess how zero-shot annotation quality varies with model size and whether fine-tuning a smaller model can match or exceed the performance of larger zero-shot baselines. Closed-source models such as GPT-4o were not included due to the computational cost of running extensive experiments across our full pipeline; LLaMA 3.3 70B Instruct provides a strong open-source comparison at comparable scale, and we discuss this limitation further in Section 2.5.8.

We further distinguish between two types of out-of-the-box models based on their pre-training. *Reasoning models* were pre-trained to produce explicit chain-of-thought reasoning before generating answers. In our experiments, we use Qwen3 models with their native thinking mode enabled, which triggers the model's built-in reasoning capabilities. *Non-reasoning models* were not trained with reasoning capabilities and generate structured outputs directly without intermediate reasoning steps. In our experiments, we use LLaMA 3.3 models in their standard configuration. This distinction allows us to assess the contribution of explicit reasoning to zero-shot annotation quality independently of model fine-tuning. All models are used in their pre-trained, out-of-the-box form without any task-specific adaptation. Prompting templates and hyperparameter configurations for all baselines are provided in Appendix A.1.1.1.

For the models we fine-tune, we use Qwen3 8B as the base architecture for both SFT and RL training, motivated by its strong instruction-following capabilities, its native support for reasoning trace generation, and its computational feasibility within our budget. We investigate two fine-tuning configurations: SFT alone and SFT followed by RL refinement (SFT+RL), allowing us to isolate the contribution of each training phase to final annotation quality. Template and hyperparameter

configurations for all our training setups are provided in Appendix A.1.1.

2.5.3.2 Evaluation setup

We evaluate annotation quality achieved by our systems using metrics appropriate to each field. For **Control Type**, which requires a single binary prediction, we report overall Accuracy and macro-averaged F1. For the three multi-label fields, **Control Implementation**, **Control Integration**, and **Control Attributes**, we report Exact Match (requiring perfect agreement between predicted and gold label sets), Jaccard similarity (intersection-over-union, rewarding partial overlap), and Micro-averaged F1 (measuring precision and recall across all labels). Together, these metrics capture both strict correctness and the degree of partial agreement.

For the open-vocabulary Control Attributes field, predicted labels are extracted strings rather than selections from a fixed taxonomy, and we apply strict string matching without normalisation. This means that even minor lexical variations such as "sentiment" versus "sentiments", or "topic control" versus "topic", are treated as incorrect predictions. While this conservative approach may underestimate true annotation quality by penalising semantically equivalent predictions, we prefer it over more semantic measures such as embedding-based semantic similarity matching, which could introduce alignment artefacts and lead to inflated performance estimates. Strict string matching provides a reliable lower bound on annotation quality.

All results are averaged over five independent runs with different random seeds (7097, 4613, 89723, 9875, 24651) to account for variance.

2.5.3.3 Training setup

All training runs are performed on two NVIDIA A100 GPUs (80 GB each), using Qwen3 8B as the base model fine-tuned with LoRA ($r = 32$, $\alpha = 64$). We explore three training setups: (i) SFT on structured annotation outputs without reasoning traces; (ii) SFT on the reasoning-augmented training set described in Section 2.5.2,

where each instance includes a synthetic chain-of-thought reasoning trace prior to the structured output; and (iii) RL refinement using DAPO (see Section 2.5.2.4 - RL for structured annotation generation), applied either on top of the SFT checkpoint from setup (ii) or directly on the base model as an ablation. Comparing setups (i) and (ii) isolates the contribution of reasoning augmentation; comparing setup (iii) variants isolates the contribution of the SFT initialisation to RL training stability. All setups use a batch size of 1 with gradient accumulation over 8 steps (effective batch size of 8). SFT training runs for up to 60 steps with early stopping based on validation loss, using a learning rate of 2×10^{-4}. RL training runs for up to 3 epochs with a learning rate of 1×10^{-5} and $G = 8$ candidate generations per sample. Full hyperparameter configurations are provided in Appendix A.1.1.

Code and results are available on GitHub.[3]

2.5.4 Main Results

Table 2.3 shows results for all models on the CTG annotation task. Among pre-trained LLMs, larger models generally achieve better performance, with LLaMA 3.3 70B Instruct achieving the strongest zero-shot results overall: 74.5% accuracy on Control Type, 47.6% exact match on Control Integration, and 62.1% F1 on Control Integration, substantially outperforming smaller models and often achieving second-best performance across dimensions, surpassed only by our fine-tuned SFT+RL system. An exception to the size-performance trend is Qwen3 32B, which performs worse than Qwen3 14B on Control Attributes despite being more than twice the size, suggesting that model family and training setup matter beyond parameter count alone. Despite LLaMA 3.3 70B's strong performance, multi-label fields remain challenging for zero-shot approaches: even this largest baseline achieves only 44.9% F1 and 35.2% Jaccard on Control Attributes, reflecting the difficulty of open-vocabulary label extraction without task-specific training.

Fine-tuning Qwen3 8B yields substantial improvements over both the base model

[3] `https://github.com/michelalorandi/sft_dapo_paper_annotation`

Model	R	Control		Attributes			Integration			Implement.		
		Acc	F1	EM	F1	J	EM	F1	J	EM	F1	J
Pre-trained LLMs												
Qwen3 8B	✓	30.3	30.3	8.3	29.7	13.8	16.6	36.5	23.3	15.2	46.7	26.7
Qwen3 14B	✓	62.1	62.1	24.8	49.2	36.6	29.0	51.8	42.4	23.4	63.2	49.4
Qwen3 32B	✓	68.3	68.3	21.4	43.5	31.7	32.4	54.7	45.3	21.4	57.7	44.5
LLaMa3.1 8B Instr	✗	34.5	34.5	10.3	37.6	23.6	25.5	35.6	30.0	11.0	33.8	25.7
LLaMa3.3 70B Instr	✗	*74.5*	*74.5*	20.0	44.9	35.2	*47.6*	*62.1*	*58.3*	23.4	62.9	51.6
Our systems (Qwen3 8B)												
SFT only w/o reason	✗	68.3	68.3	23.4	49.4	35.5	35.9	49.5	42.9	29.7	55.8	46.8
SFT only w reason	✓	71.7	71.7	*32.4*	*53.7*	*41.2*	36.6	46.2	44.0	*40.0*	65.4	*59.3*
RL only	✓	67.6	67.6	24.1	51.8	35.1	40.7	56.0	50.0	37.2	**69.8**	58.7
SFT+RL	✓	**84.1**	**84.1**	**34.5**	**54.5**	**42.4**	**62.8**	**66.7**	**65.9**	**46.2**	*67.1*	**62.0**

Table 2.3: Performance on CTG system annotation across four dimensions: Control Type (binary classification), Control Attributes, Control Integration, and Control Implementation (multi-label). Results are averaged over 5 runs. **Bold** indicates best result, *italics* second-best. R=reasoning trace; EM=Exact Match; J=Jaccard score (intersection-over-union).

and larger zero-shot alternatives. Comparing the two SFT variants, reasoning augmentation consistently improves performance: reasoning-augmented SFT outperforms direct annotation SFT by 3.4 points on Control Type accuracy, 9.0 points on Control Attributes Exact Match, and 10.3 points on Control Implementation Exact Match. These gains are particularly pronounced on multi-label fields (Control Attributes, Control Integration, and Control Implementation), consistent with the hypothesis that explicit reasoning traces provide a stronger training signal for complex structured annotation tasks.

Training with RL alone from the base model achieves performance broadly comparable to SFT without reasoning, though it underperforms reasoning-augmented SFT on Control Attributes. This suggests that reward signals alone are insufficient to substitute for the supervision provided by synthetic reasoning traces, particularly for open-vocabulary extraction.

Our best results are obtained by combining both training stages. SFT followed by RL refinement (SFT+RL) achieves 84.1% accuracy on Control Type, a gain of 12.4% points over reasoning-augmented SFT alone, alongside improvements on all multi-label fields: 54.5% F1 (+0.8 points) on Control Attributes, 66.7% F1 (+20.5 points) on Control Integration, and 62.0% Jaccard (+2.7 points) on Control Implementation.

The gains are particularly notable on Exact Match metrics: SFT+RL achieves 62.8% EM on Control Integration compared to 36.6% for SFT alone, indicating that RL refinement helps the model produce more completely correct multi-label predictions rather than merely partial matches. The sole exception is Control Implementation F1, where RL alone achieves 69.8% compared to 67.1% for SFT+RL, suggesting that RL without SFT initialisation may favour recall on this particular field. Overall, these results demonstrate that reasoning-augmented fine-tuning followed by RL refinement enables effective structured annotation under severely limited training data. SFT+RL substantially outperforms both zero-shot baselines and single-stage training across all except one metric, namely F1 in Control Implementation.

2.5.5 Ablation Studies

We conducted two ablation studies to validate our key design choices: the reward formulation for RL training, and the architectural decision to train a single multi-field model rather than separate single-field models.

2.5.5.1 Reward function ablation

Table 2.4 compares different reward formulations for RL training, examining the effect of the correctness metric (Exact Match versus Jaccard), the inclusion of auxiliary format and JSON validity rewards, and the initialisation strategy (RL from the base model versus RL from the SFT checkpoint).

Starting with SFT before applying RL proves consistently beneficial across all reward configurations: SFT+RL approaches outperform their RL-only counterparts by 14–17 points on Control Type, regardless of which reward components are used. This gap persists across multi-label fields as well, where SFT+RL configurations achieve 10–22 point improvements in exact match scores. Notably, RL-only training produces substantially longer reasoning traces (averaging 879–992 tokens versus 563–675 tokens for SFT+RL), yet achieves lower performance, suggesting the model without prior supervised fine-tuning generates verbose but less effective reasoning

Model	Control		Attr.		Integr.		Implem.		# Avg
	Acc	F1	Acc	J	Acc	J	Acc	J	Reason
Qwen3 8B	30.3	30.3	8.3	13.8	16.6	23.3	15.2	26.7	338.0
+ SFT only	71.7	71.7	32.4	41.2	36.6	44.0	40.0	59.3	574.8
+ RL only EM	69.7	69.7	22.1	33.8	46.9	54.9	37.2	60.2	992.2
+ RL only EM+F	69.7	69.7	22.1	33.9	36.6	49.3	29.0	54.3	975.7
+ RL only EM+F+JSON	67.6	67.6	24.1	35.1	40.7	50.0	37.2	58.7	879.7
+ RL only J	66.9	66.9	24.1	35.6	35.9	49.3	40.7	63.1	979.3
+ RL only J+F	66.9	66.9	24.8	35.7	37.9	46.9	35.2	55.3	942.6
+ RL only J+F+JSON	66.2	66.2	21.4	33.8	28.3	45.6	30.3	53.6	955.8
+ SFT+RL EM	81.4	81.4	34.5	43.7	**63.4**	**66.9**	44.1	61.1	656.4
+ SFT+RL EM+F	**84.1**	**84.1**	**35.9**	**45.4**	57.9	61.0	**46.9**	*63.2*	615.4
+ SFT+RL EM+F+JSON	**84.1**	**84.1**	34.5	42.4	*62.8*	*65.9*	46.2	62.0	563.4
+ SFT+RL J	82.8	82.8	33.1	43.4	60.7	64.5	44.8	62.7	580.1
+ SFT+RL J+F	81.4	81.4	35.2	*44.7*	62.1	*65.9*	45.5	63.1	675.4
+ SFT+RL J+F+JSON	78.6	78.6	**35.9**	44.6	51.0	55.2	**46.9**	**64.4**	651.8

Table 2.4: Ablation study on reward components for RL training. We compare training with RL only (from base model) vs. SFT+RL (from SFT checkpoint), using different reward combinations: Exact Match (EM) vs. Jaccard (J) for correctness, with optional Format (F) and JSON format rewards. # Avg Reason=average reasoning trace length. Results averaged over 5 runs. **Bold** indicates best results; *italic*: second-best.

that fails to capture the systematic patterns needed for accurate annotation. This indicates that supervised examples provide an essential foundation for RL to build upon, rather than RL being able to discover effective annotation strategies from reward signals alone.

Exact Match and Jaccard rewards exhibit a trade-off in the expected direction: EM yields higher Exact Match scores (63.4% versus 60.7% on Control Integration), while Jaccard rewards lead to better partial-credit scores (64.4% versus 63.2% on Control Implementation Jaccard). More informative is the effect of auxiliary rewards: adding format and JSON validity rewards to the SFT+RL configuration reduces average reasoning trace length from 656 to 563 tokens while maintaining competitive performance, indicating that explicitly rewarding structural correctness encourages more concise and focused explanations. However, this benefit depends on starting with SFT: when training with RL only from the base model, combining multiple reward components often degrades rather than improves performance. For instance, RL only with EM achieves 46.9% exact match on Control Integration, but adding format and JSON rewards (EM+F+JSON) reduces this to 40.7%. Similarly,

J+F+JSON achieves only 28.3% exact match on Control Integration compared to 35.9% for J alone, despite the additional reward signals. This suggests that without the foundation provided by supervised examples, the model struggles to balance multiple competing objectives, leading to unstable optimisation that can degrade performance on some dimensions even when overall reward increases.

We adopt EM+F+JSON as our final reward configuration, achieving 84.1% on Control Type with the shortest average reasoning traces (563 tokens) while maintaining strong performance across all multi-label fields. This model is used for production annotation of 2024–2025 papers (Section 2.5.7) and as the primary system in our main results (Table 2.3).

2.5.5.2 Comparing single-field vs multi-field models

To validate our architectural choice of a multi-field model that predicts all four annotation dimensions jointly, Table 2.5 compares our multi-field approach against four separately trained single-field models (one per annotation dimension), alongside inference time per paper. We compare only SFT variants rather than SFT+RL due to computational constraints: training four separate RL models would require substantially more GPU resources, and more importantly, single-field RL training would produce extremely sparse reward signals (binary correct/incorrect per paper) that we expect would lead to unstable training. Since the SFT comparison already demonstrates the multi-field architecture's advantages, capturing inter-field dependencies and improving computational efficiency, we focus the RL refinement exclusively on the multi-field approach.

The multi-field model consistently matches or outperforms single-field models after SFT across all dimensions, with the exception of a disadvantage on Control Implementation Jaccard (59.3% versus 62.4%). After RL refinement, the multi-field model achieves strong improvements across all dimensions. These results, combined with the multi-field advantages observed in SFT (faster inference, better performance through inter-field dependencies), validate our architectural choice.

Model	Control		Attr.		Integr.		Impl.		Time (s) ↓
	Acc	F1	Acc	J	Acc	J	Acc	J	
SFT Single-field	69.7	69.7	29.7	41.8	33.8	45.5	38.6	**62.4**	120.4
SFT Multi-field	71.7	71.7	32.4	41.2	36.6	44.0	40.0	59.3	39.9
SFT+RL Multi-field	**84.1**	**84.1**	**34.5**	**42.4**	**62.8**	**65.9**	**46.2**	62.0	**36.8**

Table 2.5: Comparison of single-field vs. multi-field models on CTG annotation. Single-field trains four models (one per dimension), requiring four sequential inferences. Multi-field predicts all dimensions jointly in a single forward pass. Time=average inference time per paper (lower is better). Results averaged over 5 runs. **Bold** = best result.

Beyond predictive performance, the multi-field architecture offers substantial computational advantages. Single-field models require four separate forward passes per paper, averaging 120.4 seconds total. The multi-field model produces all four annotations in a single pass, requiring only 36.8 seconds, a 3.3× speedup that becomes increasingly significant when annotating large corpora such as the 196 papers published in 2024–2025. The efficiency gains extend to reasoning generation: the multi-field SFT model produces reasoning traces averaging 575 tokens compared to 1,917 tokens combined across the four single-field models, a 3.3× reduction in output length. RL refinement further improves both efficiency and accuracy, compressing reasoning to 563 tokens while achieving the strongest performance across all dimensions. These results confirm that the multi-field architecture is both more accurate and substantially more efficient than the single-field alternative, validating our architectural choice.

2.5.6 Manual Evaluation

To further validate our approach on completely unseen data, we evaluate our best-performing model (SFT+RL with EM+F+JSON rewards) on 30 papers randomly sampled from Round 3 of our paper collection (2024 publications, Figure 2.4c). We required fresh held-out papers beyond the previously annotated test set from Round 2, which had already been used for automatic evaluation (Table 2.3). These 30 papers were excluded from all training and automatic annotation experiments, providing an independent test of generalisation to publications from more than a

year after the training data was collected.

Rather than creating gold annotations from scratch, requiring approximately 22.5–30 hours for 30 papers (45–60 minutes per paper), we adopt a human-in-the-loop validation approach: the model produces structured annotations alongside step-by-step reasoning traces, which the first author then verifies for correctness. This approach both validates model performance and demonstrates the practical efficiency of semi-automated annotation, measuring the time required to review and correct model outputs rather than annotate from scratch.

Table 2.6 presents the results of manual validation across the four annotation dimensions: Control Type (single-attribute vs multiple-attribute), Control Attributes (which properties are controlled), Control Integration (where control signals enter the system), and Control Implementation (how control is achieved). The model demonstrates strong performance across all dimensions. Control Type and Control Integration achieve the highest accuracy, with 25/30 (83.3%) and 27/30 (90.0%) of outputs completely correct, respectively.

For the more challenging multi-label fields, Control Attributes and Control Implementation, the model produces fully correct predictions in 19/30 and 21/30 cases respectively. A further 6/30 and 7/30 cases are partially correct, meaning the model correctly identified some but not all of the relevant labels (e.g., predicting "sentiment" and "formality" while missing "toxicity" for a system controlling all three attributes). The remaining 5/30 and 2/30 cases are completely incorrect, where none of the predicted labels match the gold annotations (e.g., predicting "sentiment" and "topic" when the correct answer is "keywords"). This pattern suggests the model reliably captures the primary characteristics of CTG systems while occasionally missing secondary attributes or, less frequently, misidentifying the control approach entirely.

The model-generated reasoning traces substantially facilitated human verification. Rather than re-reading entire papers, the annotator could follow the reasoning traces' explicit pointers to specific passages, directly verifying whether the cited

	✓	Partial	×
Control Type	25	-	5
Control Attributes	19	6	5
Control Implementation	21	7	2
Control Integration	27	0	3

Table 2.6: Manual evaluation results on 30 unseen papers from 2024 using our best model (SFT+RL with EM+F+JSON). Partial=some labels correct. No partial results for Control Type, as it is binary classification.

evidence supported the generated annotations and efficiently identifying errors in partially correct cases. Validating model-generated annotations in this way required approximately 10 minutes per paper (5 hours total for 30 papers), a 5–6× reduction compared to 45–60 minutes for fully manual annotation from scratch. This demonstrates that the combination of structured predictions and explicit reasoning traces enables efficient human-in-the-loop validation while maintaining annotation quality.

Figure 2.9 compares annotations for the same paper across three approaches: gold annotation (human expert), our best model (Qwen3 8B SFT+RL), and LLaMA3.3 70B Instruct (zero-shot).

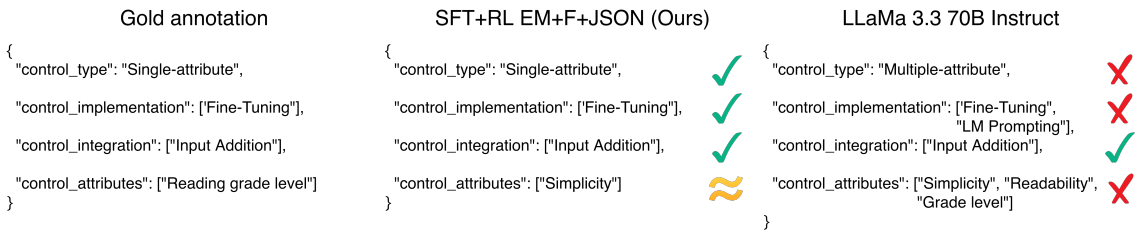


Figure 2.9: Annotation comparison for the same paper. Gold annotation (left), our best multi-field model (middle: Qwen3 8B SFT+RL with EM+F+JSON rewards), and LLaMa3.3 70B Instruct baseline (right).

2.5.7 Automated Annotation of 2024–2025 Papers

Having established that SFT+RL with EM+F+JSON rewards achieves the highest performance on our held-out test set (Section 2.5.4), we apply this configuration to annotate all 178 papers published in 2024-2025 identified through our systematic selection process (Section 2.4.1). This automated annotation differs from the experimental setup in three respects: we retrain on the complete set of 140 manually

annotated papers rather than the 100-paper training subset, making full use of the available supervision; we expand the annotation scope to include four additional metadata fields alongside the four core categorisation dimensions; and we apply a manual verification and correction process to every annotated paper.

We retrain the model once on all 140 manually annotated papers published up to December 2023, including the 110 papers used for training and the 30 papers previously held out for validation and testing in the experimental setup (Section 2.5.3); the 30 papers from 2024 used for manual evaluation of the model (Section 2.5.6) are not included in this retraining, as they are reserved for assessing annotation quality rather than for training. This single retrained model, trained exclusively on pre-2024 data, is then used to annotate both the 2024 and the 2025 papers: no further retraining is performed between annotating the two years. Since this model is not used for model selection or hyperparameter tuning, using all available pre-2024 annotated data is appropriate and increases the supervision available in this low-resource setting. Model performance is instead evaluated through manual verification of all papers from 2024 and 2025, providing direct assessment of annotation quality on new papers. The retraining uses identical hyperparameters to the experimental setup (Section 2.5.3).

To support comprehensive analysis of CTG research beyond control mechanisms alone, we extend annotation to four additional metadata fields that characterise the broader experimental context: performed tasks, evaluation datasets, automatic evaluation metrics, and model architectures. These supplementary fields allow us to analyse evaluation practices and model adoption trends (Section 2.6.6) alongside the core control dimensions. The model generates outputs as JSON objects containing all eight fields: the four core categorisation dimensions (Control Type, Control Attributes, Control Implementation, Control Integration) and four supplementary metadata fields (Tasks, Evaluation Datasets, Evaluation Metrics, Model Overview).

The following is an example JSON output, illustrating how fields are populated for a hypothetical paper:

```
{
  "tasks": "Free text",
  "control_type": "Single-attribute",
  "control_attributes": ["Topic", "Keywords"],
  "control_implementation": "Fine-Tuning",
  "control_integration": "Input Addition",
  "automatic_evaluation_metrics": ["Correctness"],
  "evaluation_datasets": ["AGNews"],
  "model_overview": ["GPT-2"]
}
```

Note that `control_attributes` lists all attributes discussed or explored in the paper, while `control_type` reflects how the implemented system operates. In this example, the paper explores both topic and keyword control, but the resulting system implements single-attribute control, allowing users to manipulate only one attribute at a time.

All 178 papers are preprocessed using the same pipeline described in Section 2.5.2: PDF conversion, removal of non-informative sections, and cleaning of LaTeX artefacts. The retrained model then annotates each paper, generating a reasoning trace followed by a structured JSON output containing all eight annotation fields.

Manual verification was performed by the author. For each paper, the annotator examines the reasoning trace to assess whether it references appropriate evidence, then verifies the structured JSON output against the cited paper content, making corrections where necessary. Details are reported in Section 2.5.7.1.

Annotation quality was consistent with the results reported in Section 2.5.6, with the majority of corrections required for the open-vocabulary Control Attributes field and the newly introduced metadata fields. Comprehensive annotation quality assessment is presented in Section 2.5.7.2. The reasoning traces facilitated efficient verification, reducing annotation time to approximately 10–15 minutes per paper compared to 45–60 minutes for annotation from scratch, a 3–4× reduction. This enabled annotation of all 178 papers in approximately 30–50 hours of verification

effort, compared to an estimated 130–178 hours for fully manual annotation.

The final dataset includes 318 papers annotated according to our multi-dimensional categorisation scheme: 140 papers annotated manually and 178 papers annotated automatically with manual verification. This dataset enables comprehensive analysis of developments and temporal trends in the CTG field, as presented in Section 2.6.

2.5.7.1 Manual Verification Procedure

Manual verification was performed by the author using a structured spreadsheet-based interface. The procedure includes the following steps:

Setup and Data Format. Automatic annotations from the model were exported as CSV with the following structure: a row for each paper, columns for the reasoning trace (`thinking_trace`), predicted values for each field (`pred_*`), and initially empty columns for verified gold-standard annotations (`gold_*`). Each paper included a persistent hyperlink to the original publication, enabling direct access to source material during verification.

Verification Workflow. For each paper, the annotator followed a standardised procedure. First, the full reasoning trace generated by the model was read, which documents the model's interpretation of the paper's content and the rationale for each prediction. Subsequently, for each predicted field (`control_type`, `control_attributes`, `control_implementation`, `control_integration`, `tasks`, `evaluation_datasets`, `evaluation_metrics`, `model_overview`), the corresponding predicted value(s) were examined.

The source paper was then consulted via the provided link to identify evidence supporting or contradicting the prediction. The verified annotation was recorded in the corresponding `gold_*` column using the same format and terminology as the predictions (e.g., lists separated by semicolons for multi-value fields). Where the prediction matched the source material, the predicted value was copied into the gold column. Where the prediction contained errors, omissions, hallucinations,

or misinterpretations, the corrected annotation was recorded based on the paper content.

Correction Tracking and Quality Control. Corrections were systematically tracked by comparing predicted (`pred_*`) and verified (`gold_*`) values for each field. Any discrepancy between prediction and verification was preserved in the dataset, enabling downstream error analysis (Section 2.5.7.2). The spreadsheet format ensured that all corrections were explicit and auditable: for each paper and each field, both the model's prediction and the verified gold annotation were recorded side-by-side, making it possible to identify exactly which predictions required correction and why.

Consistency and Standardisation. To ensure consistent application of the categorisation scheme across all papers, verification followed the definitions and examples provided in Section 2.3 (for control-related fields) and standard naming conventions (for tasks, datasets, metrics, and models).

2.5.7.2 Error analysis

To assess the quality of automatic annotations on the manually verified 2024–2025 papers, we conducted a detailed error analysis across all annotation fields. Table 2.7 shows overall annotation accuracy using two metrics: Exact Match (strict accuracy where all predicted values must match the gold annotation) and F1-score (harmonic mean of precision and recall, capturing partial correctness for multi-value fields).

As can be seen from the table, overall accuracy varies substantially across fields, ranging from 61.8% (Evaluation Metrics) to 90.4% (Control Type) for exact match. Control Type is the only single-value field and achieves 90.4% exact match accuracy, reflecting the relative simplicity of binary classification. Multi-value fields exhibit a wider range of performance (61.8–88.8%). Tasks field performs exceptionally well among multi-value fields at 88.8% exact match, likely because tasks are typically explicit and identifiable from paper abstracts and keywords. Other multi-value fields

show lower exact match rates (61.8–83.1%) because the model must predict multiple interrelated values correctly, but F1-scores reveal that partial correctness is common, ranging from 72.5% to 85.4%, indicating the model frequently captures some but not all correct values.

Field	Exact Match (%)			F1 (%)		
	Overall	2024	2025	Overall	2024	2025
Control Type	90.4	88.6	92.2	—	—	—
Control Attributes	64.6	60.2	68.9	72.5	72.0	72.9
Control Implementation	62.9	63.6	62.2	81.3	81.0	81.5
Control Integration	83.1	78.4	87.8	85.4	81.4	89.3
Tasks	88.8	87.5	90.0	88.8	87.5	90.0
Evaluation Datasets	77.5	71.6	83.3	85.1	82.1	88.1
Evaluation Metrics	61.8	52.3	71.1	84.8	82.2	87.4
Model Overview	68.0	65.9	70.0	78.2	78.0	78.3

Table 2.7: Annotation accuracy on 2024–2025 papers. Exact Match is strict accuracy (100% correct predictions). F1-score applies to multi-value fields only; Control Type is a single-value binary field.

Temporal Stability. Annotation accuracy improved substantially and consistently from 2024 to 2025 across nearly all fields. Exact Match accuracy increased for seven of eight fields, with particularly large gains in Evaluation Metrics (+18.8 points), Evaluation Datasets (+11.7), Control Integration (+9.4), and Control Attributes (+8.7). Only Control Implementation showed a marginal decline (-1.4), while F1-scores similarly improved across multi-value fields, with gains ranging from +0.3 (Model Overview) to +7.9 (Control Integration).

This consistent improvement despite increased dataset size suggests either that 2025 papers are more explicitly structured in their presentation of methods and results, or that the automatic annotation model benefits from the expanded 2024 training data.

Error Type Breakdown. Table 2.8 decomposes errors into three categories: omission errors (model predicts a subset of correct values), hallucination errors (model predicts values not present in the reference annotation), and mixed errors

(containing both omission and hallucination). The distribution reveals distinct error profiles across fields.

Field	Fully Correct	Omission	Hallucination	Mixed	*Total*
Control Attributes	115	12	1	50	178
Control Implementation	112	18	30	18	178
Control Integration	148	3	3	24	178
Tasks	158	0	0	20	178
Evaluation Datasets	138	6	8	26	178
Evaluation Metrics	110	35	7	26	178
Model Overview	121	8	10	39	178

Table 2.8: Error type breakdown for multi-value fields. Omission errors occur when the model predicts a subset of correct values. Hallucination errors occur when the model predicts values not in the reference annotation. Mixed errors contain both.

Omission errors are most common in the Evaluation Metrics field (35 of 68 incorrect cases), suggesting the model struggles to identify all relevant metrics when multiple ones are present. Hallucination errors occur less frequently overall but are more frequent in Control Implementation (30 cases), indicating the model sometimes proposes implementation methods not actually used in papers. Mixed errors are substantial across all fields (18–50 cases), representing cases where the model both misses some values and incorrectly predicts others.

Tasks and Control Integration exhibit the fewest errors among multi-value fields. Tasks achieves near-perfect accuracy with 158 of 178 papers (88.8%) fully correct, zero omission or hallucination errors, and only 20 mixed errors. This performance likely reflects that tasks tend to be explicit components of paper titles and abstracts, making them straightforward to identify. Control Integration similarly performs well: 148 of 178 papers (83.1%) are fully correct, with only 3 omission errors and 3 hallucination errors, suggesting integration strategies are relatively straightforward for the model to identify. Conversely, Control Attributes shows the highest proportion of mixed errors among multi-value fields (50 of 63 incorrect cases, 79.4%), though this warrants careful interpretation which we turn to next.

Several fields, Control Attributes, Tasks, Evaluation Metrics, Evaluation Datasets, and Model Overview, do not have closed sets of predefined categories; instead, these

are identified from paper text using natural language. However, they differ substantially in their susceptibility to string-matching artifacts. Metrics, datasets, and models are formal named entities with relatively standardised naming conventions (e.g., BLEU, ROUGE, YELP, GPT-2), making them easier to extract and match exactly. In contrast, Control Attributes involve more varied terminology: the same underlying control attribute might be described using different phrasings. For instance, the model may predict `emotion` when the reference annotation uses `affect`, or predict `sentiment control` when the annotation uses `sentiment`. These represent genuine semantic equivalence but fail exact string matching. Consequently, a substantial portion of reported errors for Control Attributes likely represent minor terminology variations or subtle differences in naming rather than genuine annotation failures.

As explained in Section 2.5.7.1, manual verification was performed by the author, ensuring consistency with the manual evaluation presented in Section 2.5.6. This consistency strengthens reliability, though the single-annotator approach cannot fully capture the complete range of valid terminology. When interpreting attribute-level findings, readers should recognise that reported error rates reflect strict string matching rather than semantic equivalence. The error rates for Evaluation Metrics (61.8%), Evaluation Datasets (77.5%), and Model Overview (68.0%) are more reliable, as these fields employ established naming conventions with fewer terminology variations.

Most Common Errors. Further analysis identifies systematic error patterns that show the sources of model failures. Hallucinated values (Table 2.9) reveal that the model show strong priors towards common categories within each field. For Control Implementation, the model over-predicts ***Modification of Token Distribution*** (23 hallucinations) and ***Fine-Tuning*** (15 hallucinations), reflecting that these are among the most frequent methods in the training data. For Control Integration, ***Input Addition*** is hallucinated 20 times, consistent with its dominance in the dataset (appearing in approximately 63% of papers). Similarly, for Tasks, `Free`

Field	Value	Count
Control Attributes	`CEFR level`	5
	`Style`	4
	`Detoxification`	3
Control Implementation	`Modification of Token Distribution`	23
	`Fine-Tuning`	15
	`Complete Training`	9
Control Integration	`Input Addition`	20
	`Model Injection`	6
	`System Addition`	2
Evaluation Datasets	`PPLM prompts`	3
	`PPLM`	2
	`REALTOXICITYPROMPTS`	2
Evaluation Metrics	`Toxicity`	4
	`Perplexity`	3
	`BLEU`	3
Model Overview	`LLM`	4
	`GPT-2`	3
	`GPT-4`	3
Tasks	`Free Text`	7
	`Conversational Agent`	2
	`Text Readability Control`	1

Table 2.9: Most commonly hallucinated values (top 3 per field). Hallucinations occur when the model predicts a value not present in the gold annotation.

`Text` is hallucinated 7 times, suggesting the model associates CTG research with open-ended generation tasks. Conversely, rarer categories like ***System Addition*** (2 hallucinations for Control Integration) and task-specific approaches are rarely over-predicted.

Omitted values (Table 2.10) show a complementary pattern: the model frequently fails to identify less common categories. For Control Attributes, `Toxicity` (6 omissions) and `CEFR Level` (5 omissions) are missed despite being important control attributes, likely because they appear in fewer training examples. For Control Implementation, ***LM Prompting*** is omitted 15 times despite being the second-most common method (appearing in approximately 30% of papers), suggesting the model requires substantial exemplars to reliably identify this category. For Control Integration, ***Model Injection*** is omitted 15 times, indicating difficulty distinguishing it from related categories. For Evaluation Metrics, generic metrics (`Accuracy`, `ROUGE`, `BLEU`) are frequently omitted, suggesting the model struggles when papers

Field	Value	Count
Control Attributes	Toxicity	6
	CEFR Level	5
	Length	3
Control Implementation	LM Prompting	15
	Modification of Token Distribution	10
	Fine-Tuning	9
Control Integration	Model Injection	15
	System Addition	7
	Input Addition	5
Evaluation Datasets	GYAFC	2
	TSAR 2025 Shared Task	2
	Guhr et al. (2020) dataset	1
Evaluation Metrics	Accuracy	8
	ROUGE	7
	BLEU	6
Model Overview	GPT-4	5
	GPT-2	4
	GPT-4o	4
Tasks	Text Style Transfer	3
	Free Text	3
	Dialog generation	2

Table 2.10: Most commonly omitted values (top 3 per field). Omissions occur when the model fails to predict a value present in the gold annotation.

use multiple metrics simultaneously.

These error patterns point to two primary sources of annotation failures. First, **data sparsity**: categories appearing in fewer than 5% of training papers are frequently missed, indicating the model requires sufficient exemplars to learn reliable associations. Second, **category confusion and prior bias**: the model shows strong priors towards majority-class items, leading to both hallucination of common categories and difficulty distinguishing semantically similar categories. For instance, ***Modification of Token Distribution*** (hallucinated 23 times) is conceptually related to ***Fine-Tuning*** (hallucinated 15 times, omitted 9 times), suggesting the model conflates these implementation strategies. Similarly, in Control Integration, ***Input Addition*** and ***Model Injection*** appear to be confused, with ***Model Injection*** frequently omitted (15 times) while ***Input Addition*** is over-predicted. Both issues can be improved through targeted data augmentation for rare categories or contrastive fine-tuning to better distinguish similar concepts, but they

reflect inherent challenges in learning from imbalanced, domain-specific datasets.

Implications for Automatic Annotation. Despite the error rates observed in automatic annotation, this analysis demonstrates that automated approaches can effectively support large-scale systematic review when combined with systematic validation. Automatic annotation allowed fast processing of 178 papers across 2024–2025, a task that would have required approximately 130–178 hours if performed entirely manually (based on an estimated 45–60 minutes per paper for full manual annotation from scratch). Manual verification of the 2024–2025 cohort, by contrast, required only 5 hours total for 30 papers (approximately 10 minutes per paper), representing a 5–6× efficiency gain over manual annotation while maintaining verification quality.

Performance varies substantially across annotation dimensions. The exceptional performance on Tasks (88.8%) and Control Integration (83.1%) indicates that certain annotation dimensions are sufficiently straightforward for reliable automation, with these fields benefiting from explicit terminology (task names are typically formalised; integration strategies have consistent naming conventions). Other fields (Control Attributes at 64.6%, Evaluation Metrics at 61.8%) require more careful handling, partly due to terminology variation across papers but also potentially due to alignment difficulties between the model's learned terminology and how authors describe concepts. For instance, Control Integration terms like ***Model Injection*** may be under-represented in the model's training distribution, making them harder to predict reliably. Conversely, the model may overpredict common terms that appear frequently in training data but less frequently in the actual 2024–2025 papers. These terminology misalignment issues suggest that performance could be improved through terminology standardisation or through contrastive training to better distinguish between similar concepts.

2.5.8 Limitations

Our automatic annotation approach has several limitations that should be acknowledged and suggest directions for future research.

We evaluate only open-source models (Qwen3 and LLaMA 3.3) due to the computational cost of running extensive experiments with closed-source LLMs, such as GPT-4, across our full experimental pipeline, including five-run averaging over large input contexts (up to 14,000 tokens). While our open-source baselines, particularly LLaMA 3.3 70B Instruct, provide strong comparison points at comparable scale, closed-source models might achieve different performance levels or exhibit different responses to reasoning-augmented fine-tuning and RL refinement. Future work with larger budgets could explore whether similar improvements generalise to proprietary models.

Our use of LoRA as the PEFT technique reduces computational requirements while maintaining reasonable performance, but other PEFT methods, such as Adapters (Poth et al., 2023) or Prefix-Tuning (Li and Liang, 2021), might yield different performance trade-offs. Additionally, full fine-tuning of all model parameters could potentially achieve better results, but was computationally infeasible within our budget. The relative benefits of different PEFT approaches and full fine-tuning for structured annotation tasks remain open questions for future investigation.

The low-resource setting with 100 training papers reflects realistic annotation budgets in academic research, but performance scaling with larger training sets remains unexplored. It is unclear whether additional training data would yield substantial improvements or whether our approach has already extracted most available signal from the task structure. While we demonstrate generalisation across two distinct annotation tasks within NLP (CTG technique categorisation and evaluation methodology categorisation via QCET; see Appendix A.2), applicability to systematic reviews in other scientific domains (e.g., medicine, social sciences) with different paper structures and terminology has not been tested.

Our synthetic reasoning traces are generated using LLaMA 3.3 70B Instruct and

validated automatically against gold annotations. While this approach produces traces that improve model performance and facilitate human validation, the quality and diversity of reasoning patterns may be limited by the generation model's capabilities. Human-written reasoning traces or traces from more capable models might yield further improvements, though the cost of manual trace authorship would substantially reduce the efficiency benefits of our approach.

Our manual evaluation on 2024 papers and production annotation through 2025 demonstrates reasonable temporal stability. However, we do not systematically evaluate performance degradation over longer time periods as research practices, terminology, and methodological conventions evolve. Continuous model retraining or adaptation strategies may be necessary to maintain annotation quality as the field develops.

Finally, despite using LoRA to reduce resource consumption, our approach still requires two high-end GPUs (A100 80GB) for training, which may limit accessibility for researchers with more constrained computational resources. Exploring smaller base models or more aggressive parameter-efficient techniques could improve accessibility while potentially sacrificing some annotation quality.

These limitations notwithstanding, our approach demonstrates that reasoning-augmented fine-tuning combined with RL refinement enables effective structured annotation under realistic low-resource conditions, achieving substantial time savings (5-6×) compared to fully manual annotation while maintaining quality sufficient for systematic literature analysis.

2.6 Dataset Statistics and Composition

Our final annotated dataset is composed of 318 papers spanning from early CTG research in 2018 through 2025, providing comprehensive coverage of the field's evolution. The dataset includes 110 training papers (pre-September 2023), 30 validation papers (September–December 2023), 88 papers from 2024, and 90 papers from 2025. All statistics and analyses reported in this section are computed using gold labels

obtained through manual verification (Section 2.5.7). The automatic annotations served as starting points and were systematically validated and corrected by the author; consequently, all reported findings reflect verified, high-quality annotations rather than raw automatic predictions.

The statistics we present in this section serve multiple purposes. They provide quantitative evidence of field-wide trends and patterns, inform our analysis of the CTG landscape in Chapter 3, and help identify systematic gaps that represent opportunities for future research. The comprehensive coverage and systematic annotation provided by our categorisation scheme enable both high-level characterisation of the field and fine-grained analysis of specific methodological choices and their relationships. And this in turn is useful for identifying research directions, understanding which control approaches have been thoroughly explored, and spotlighting areas where systematic investigation remains lacking.

We structure our analysis of this dataset along different dimensions. In Section 2.6.1, we first show the temporal distribution of papers and the prevalence of each control type (single-attribute vs multiple-attribute). We then analyse the three core dimensions of our categorisation framework, namely control attributes (Section 2.6.2), implementation methods (Section 2.6.4), and integration strategies (Section 2.6.3), reporting both overall distributions and temporal trends. Subsequently, we present multi-dimensional analyses (Section 2.6.5) that reveal interactions between implementation and integration strategies. Finally, we describe the evaluation practices (metrics and datasets used) and model architectures used (Section 2.6.6). Throughout, we highlight temporal trends to show how the field has evolved.

2.6.1 Temporal Distribution and Control Type

Temporal Distribution. Figure 2.10 shows the temporal distribution of papers in our dataset. The field has experienced substantial growth over the past eight years, with a particular acceleration in recent years. Early work (2018–2020) rep-

resents only 5.9% of our dataset (19 papers), reflecting the nascent stage of CTG research. The field began expanding notably in 2021–2022, with 60 papers (18.9%) published during this period. The most striking trend is the recent acceleration: 2023 saw 61 papers (19.2%), while 2024 and 2025 together account for 56% of our entire dataset (178 papers). This near-exponential growth, with 88 papers in 2024 (27.7%) and 90 papers in 2025 (28.3%), reflects both the growth of the field and broader increased interest in controlled generation following the widespread adoption of large language models.

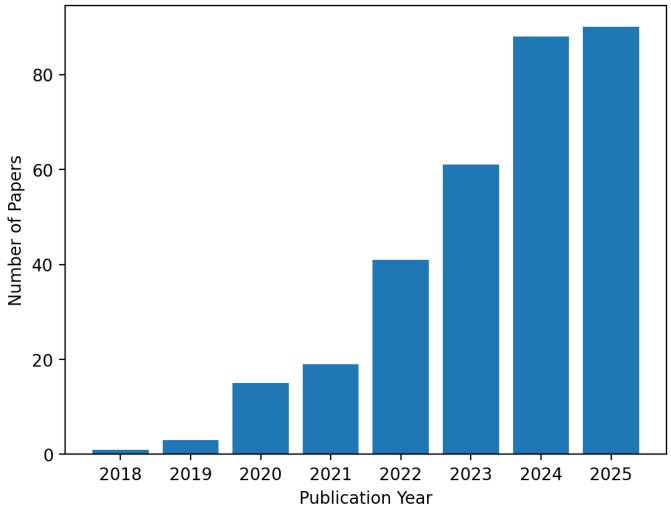
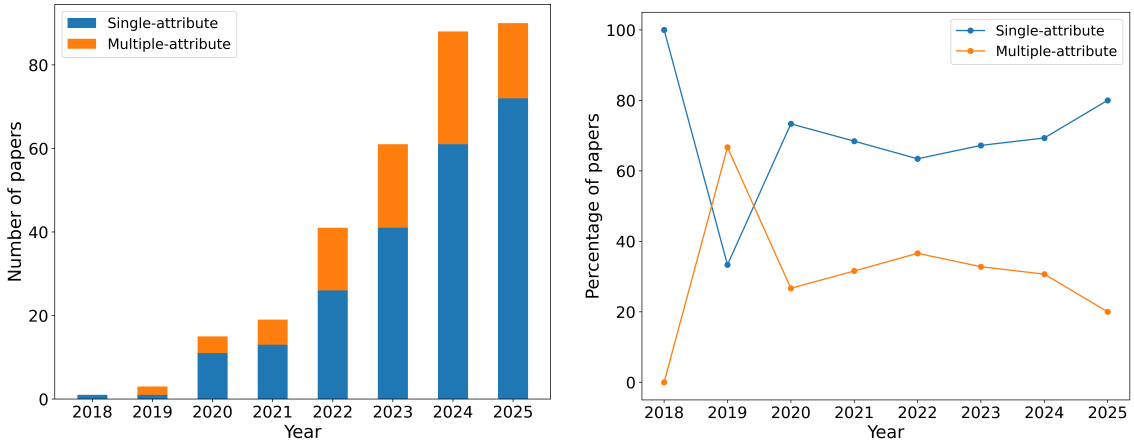


Figure 2.10: Temporal distribution of papers in our dataset (N=318).

Control Types. Figures 2.11a and 2.11b show the distribution of papers addressing single-attribute versus multiple-attribute control. ***Single-attribute control***, i.e. systems that manipulate only one control attribute at a time, dominates the field, accounting for 226 of 318 papers (71.07%) in our dataset. ***Multiple-attribute control***, which coordinates multiple potentially conflicting attributes simultaneously, represents a minority of papers with 92 papers (28.93%).

The temporal trend reveals interesting dynamics in research focus. Early work was exclusively single-attribute (100% in 2018), reflecting the nascent stage of the field where establishing control over even a single attribute presented substantial challenges. Multiple-attribute control emerged in 2019 (67% of papers) as researchers began exploring more complex control scenarios, peaking proportionally



(a) Temporal distribution of control type categories.

(b) Proportional trends in control type categories over time.

Figure 2.11: Distribution of single-attribute versus multiple-attribute control papers.

at 37% in 2022. However, the proportion of multiple-attribute papers has declined in recent years to 20% in 2025, despite the absolute number of multiple-attribute papers continuing to grow (from 15 papers in 2022 to 18 in 2025).

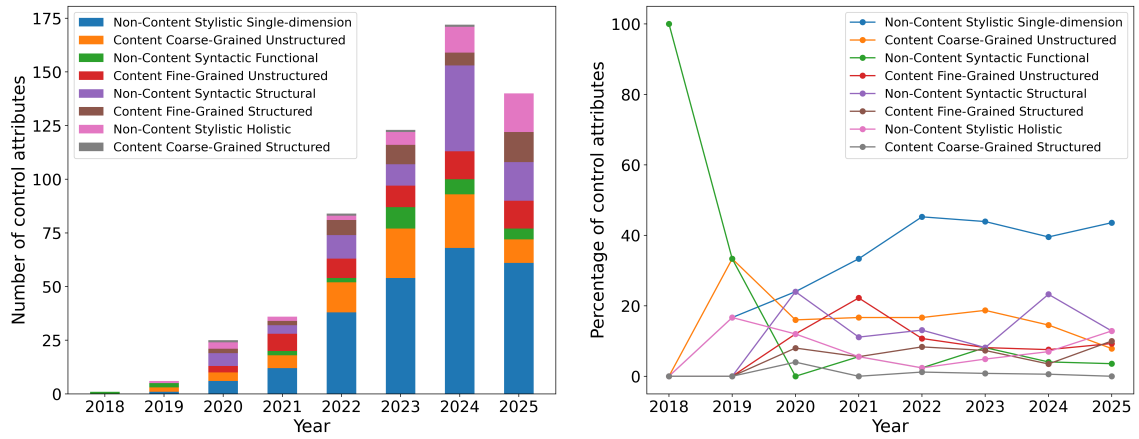
This proportional decline in multiple-attribute work does not necessarily indicate reduced interest, but may just reflect a rapid expansion of single-attribute research, particularly driven by prompting-based approaches for LLMs, which often focus on individual attributes. The persistent dominance of single-attribute control reflects the field's primary research focus on individual control objectives.

2.6.2 Control attributes

Control attributes, the specific properties of generated text being manipulated, represent the most diverse dimension of our categorisation scheme, with researchers exploring control over a wide range of characteristics. We analyse control attributes at two levels of granularity. First, we examine high-level attribute categories using the taxonomy presented in Section 2.3.2, which distinguishes attributes along two primary dimensions: (1) whether they are related to content or stylistic/syntactic properties, and (2) whether they are related to content at a coarse-grained (high-level semantic concepts) or fine-grained (detailed, specific constraints) level. This yields eight primary categories: Non-Content Stylistic Single-dimension (e.g., senti-

ment, formality), Content Coarse-Grained Unstructured (e.g., topic), Non-Content Syntactic Functional (e.g., tense), Non-Content Syntactic Structural (e.g., sentence length), Content Fine-Grained Unstructured (e.g., detailed semantic constraints), Content Fine-Grained Structured (e.g., tables), Non-Content Stylistic Holistic (e.g., persona), and Content Coarse-Grained Structured (rare). Second, we investigate the specific individual attributes researchers have targeted within these categories.

Figures 2.12a and 2.12b show the distribution of control attribute categories across our dataset. Non-Content Stylistic Single-dimension attributes, such as sentiment, formality, or toxicity, dominate the field, account for 68 attribute instances in 2024 (the peak year) and decline to 60 in 2025. This represents approximately 40–43% of all attribute instances in recent years. Content Coarse-Grained Unstructured attributes, which control high-level semantic content such as topic or theme, constitute the second most common category with 24 instances in 2024 and 11 in 2025, appearing in roughly 7–14% of attribute instances in recent years. This category showed a peak in 2019 (33% of the limited attributes explored that year), driven by early work on topic-controlled generation, but has since stabilised at lower proportions as research attention has diversified across attribute types.



(a) Temporal distribution of control attributes categories.

(b) Proportional trends in control attributes categories over time.

Figure 2.12: Distribution of control attribute categories.

Non-Content Syntactic Structural attributes (e.g., sentence length, discourse structure) show interesting dynamics, with 10 instances in 2023, peaking at 38 in

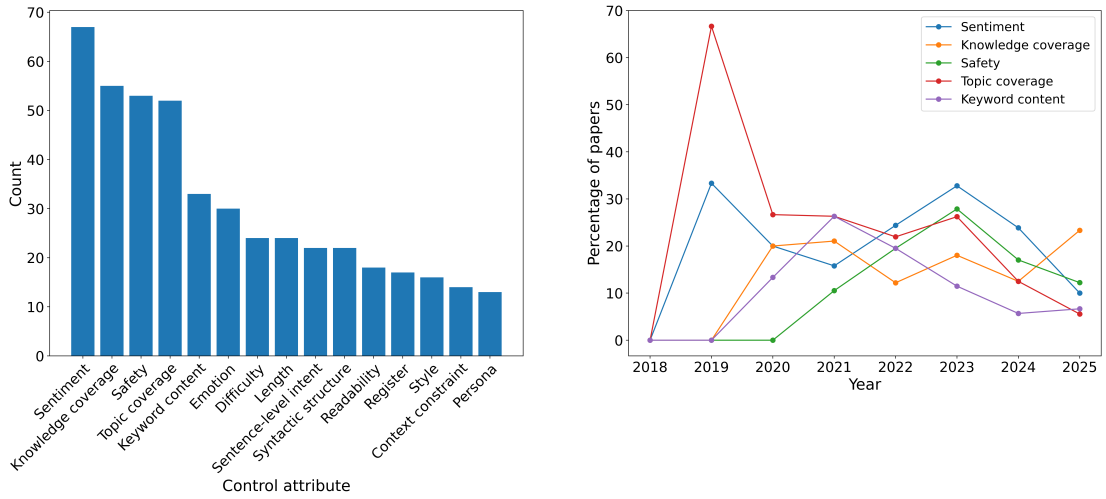
2024, then declining sharply to 18 in 2025. This temporal spike suggests a focused research effort in 2024 on syntactic control, though sustainability remains unclear. Non-Content Syntactic Functional attributes (e.g., tense, verb forms) remain relatively rare at 5–10 instances annually.

Content Fine-Grained Structured and Content Fine-Grained Unstructured attributes, requiring precise control over specific semantic content, appear in 6–14 and 12–13 attribute instances respectively in recent years, representing a smaller proportion of the dataset compared to coarse-grained and stylistic approaches. Fine-grained content control appears in fewer instances compared to coarse-grained and stylistic approaches. Non-Content Stylistic Holistic attributes, which coordinate multiple stylistic dimensions simultaneously (e.g., formality combined with politeness), show gradual growth from 6 instances in 2023 to 18 in 2025, suggesting emerging interest in holistic stylistic control.

The dominance of single-dimension stylistic control, combined with the rarity of fine-grained and holistic content control, reveals a field that has achieved substantial progress on isolated, tractable control problems but continues to face challenges when confronting more complex attribute types or attribute combinations. Notably, 2024 saw the highest number of total attribute instances (168 for Non-Content Stylistic Single-dimension alone, 168 total across all categories), but this declined in 2025, showing that while the absolute number of papers increased, researchers in 2025 explored fewer attributes per paper on average.

Beyond categorical analysis, we examine the specific individual attributes researchers have targeted. Figure 2.13a shows the 15 most frequently controlled attributes across all papers in our dataset. Sentiment emerges as the single most studied attribute with 67 papers (21.1%), followed by Knowledge coverage (55 papers, 17.3%), Safety (52 papers, 16.4%), and Topic coverage (52 papers, 16.4%). This concentration reflects the practical importance of these attributes—sentiment and safety control address critical needs for appropriate text generation—and the availability of annotated datasets and evaluation metrics for these properties.

Keyword content (33 papers, 10.4%), Emotion (30 papers, 9.5%), and Difficulty (24 papers, 7.6%) form a second tier of frequently studied attributes. These attributes represent diverse control objectives: keyword content enables content specification through term constraints, emotion extends beyond binary sentiment to nuanced affective states, and difficulty control addresses readability and accessibility requirements. The long tail of less frequent attributes, including Sentence-level intent, Syntactic structure, Readability, Register, Style, Context constraint, and Persona, illustrates the breadth of control objectives researchers have pursued, though many appear in relatively few papers.



(a) Overall top 15 most frequently used control attributes.

(b) Temporal trends for the five most common control attributes.

Figure 2.13: Control attributes usage across CTG papers.

Figure 2.13b reveals striking temporal dynamics in attribute focus. Topic coverage dominated early research, peaking at 67% in 2019 before declining sharply to 5–13% in recent years. This pattern mirrors the broader trend observed in Content Coarse-Grained Unstructured categories, reflecting the field's evolution from early topic-focused work to more diverse control objectives. Sentiment control has shown more sustained but variable attention, rising from 0% in 2018 to 33% in 2019, maintaining 15–32% presence through 2023, and declining to 10% in 2025. This recent decline likely reflects maturation of sentiment control as a widely-adopted capability.

Safety control represents the emergence of a new research focus, appearing first in 2022 at approximately 27% and remaining at similar levels (13–28%) through 2025, indicating sustained and growing concern about safe text generation in the era of large language models. Knowledge coverage shows a similar rise beginning around 2020 (20%), peaking at 23% in 2023, and declining slightly to 23% in 2025, reflecting increased interest in knowledge-grounded generation.

Emotion control shows more variable patterns, peaking at 20% in 2022 before declining, while Keyword content demonstrates steady presence at 10–27% across 2020–2025. The convergence of the top five attributes to similar proportions (5–23%) by 2025 suggests increasing diversification of research focus, with no single attribute dominating as topic coverage did in 2019 or sentiment did during 2019–2023. This diversification aligns with the field's expansion and the broader range of applications now being pursued.

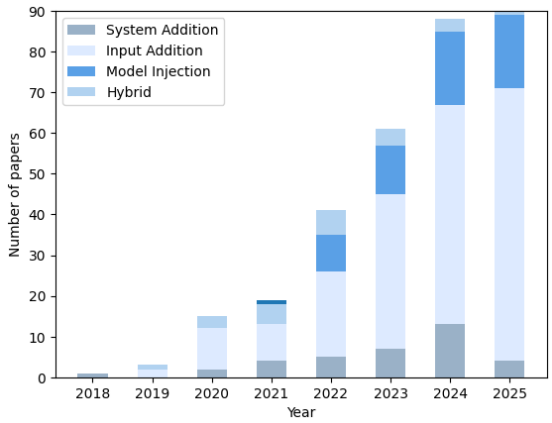
A complete list of all unique control attributes identified across our dataset, including their categorical classification, example control signals, and associated papers, is provided on GitHub.[4]

2.6.3 Control Integration Strategies

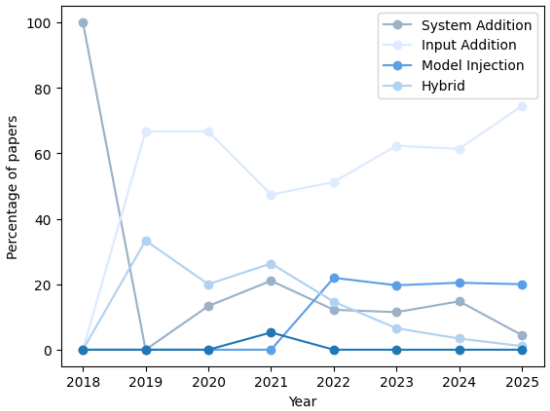
Figures 2.14a and 2.14b show the distribution of control integration strategies across our dataset, showing both absolute counts and proportional trends over time. The temporal analysis reveals a shift in how control signals are integrated into generation systems. ***Input Addition***, where control signals are provided at the input level, has become increasingly dominant, rising from 67% of papers in 2019 to 74% in 2025. This growth reflects the field's convergence towards architectures that accept control specifications as part of the input, a trend strongly reinforced by the emergence of instruction-tuned LLMs that naturally accommodate control directives through prompting.

Conversely, ***System Addition***, which was the sole integration strategy in early

[4] `https://github.com/michelalorandi/survey_of_ctg_techniques`



(a) Temporal distribution of control integration strategies.



(b) Proportional trends in control integration strategies over time.

Figure 2.14: Control integration usage across CTG papers.

work (100% in 2018), has declined to just 4% by 2025. ***Model Injection*** has remained relatively stable at 20–22% in recent years, maintaining a consistent presence for systems that require specialised internal components for different control attributes. This shift from ***System Addition*** to ***Input Addition*** represents a fundamental architectural evolution: early CTG research relied heavily on external modules and post-processing components, while contemporary approaches increasingly embed control capabilities directly into models through input conditioning and fine-tuning.

Hybrid integration strategies, which combine multiple integration approaches within a single system, appear in a relatively small proportion of papers. Figure 2.15 shows the composition of control integration strategies using an UpSet plot, which shows the frequency of individual integration methods (left), their possible combinations (matrix), and the size of each combination group (top). ***Input Addition*** alone dominates with 200 papers, accounting for approximately 63% of all papers in our dataset. ***Model Injection*** appears alone in 57 papers (18%), while ***System Addition*** alone accounts for 36 papers (11%). The remaining papers use hybrid strategies combining two integration approaches.

Hybrid combinations are notably rarer than single-strategy approaches. All hybrid approaches involve combinations of exactly two strategies; no papers use all

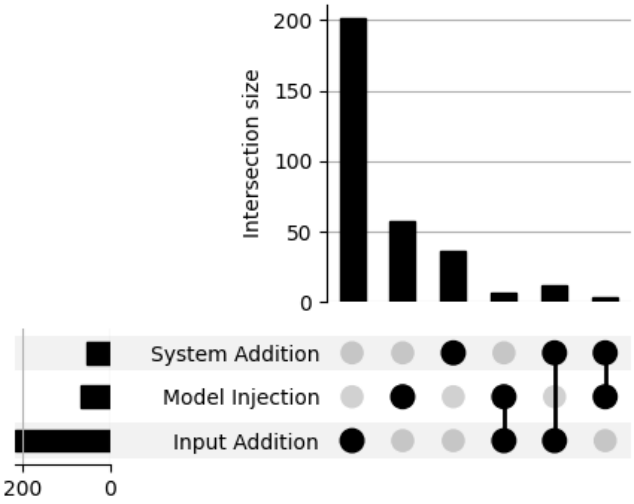


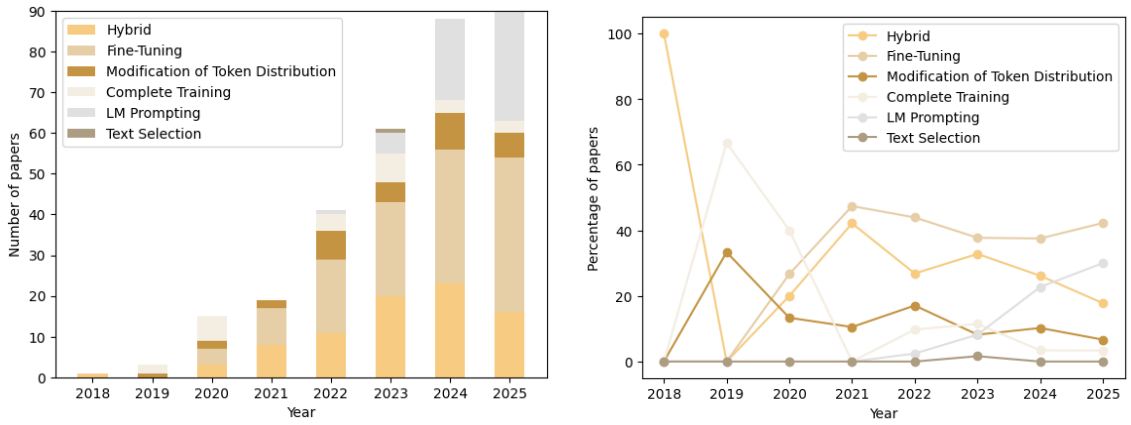
Figure 2.15: Distribution of Control Integration strategies across all papers. The horizontal bar chart (left) shows the frequency of each individual method; the matrix below indicates which methods co-occur; the vertical bar chart (top) shows the total number of papers for each combination.

three integration types simultaneously, suggesting that combining more than two integration approaches introduces excessive architectural complexity without commensurate benefits. The most common hybrid combination pairs ***Input Addition*** with ***System Addition*** (12 papers), representing systems that provide control signals as input while also including external post-processing components for constraint enforcement or output refinement. ***Input Addition*** combined with ***Model Injection*** appears in 7 papers, typically representing systems that accept control signals as input while using specialised model components for different attributes or control dimensions. The combination of ***System Addition*** with ***Model Injection*** is rare, appearing in only 4 papers.

2.6.4 Control Implementation Methods

Figures 2.16a and 2.16b show the distribution of control implementation methods across our dataset, revealing significant methodological shifts over time. The most striking trend is the rise of ***LM Prompting***, which has grown from 0% in 2018 to 30% of papers by 2025. This surge directly parallels the emergence and widespread

adoption of instruction-tuned LLMs, which enable control through natural language prompts rather than requiring architectural modification or training-based interventions. The growth of prompting-based control represents a fundamental shift in the field: from methods requiring substantial computational resources for training to training-free approaches that leverage the inherent instruction-following capabilities of modern LLMs, even though these LLMs themselves require massive computational resources for pre-training and inference.



(a) Temporal distribution of control implementation methods.

(b) Proportional trends in control implementation methods over time.

Figure 2.16: Control implementation usage across CTG papers.

Fine-Tuning has remained a dominant implementation method throughout the field's evolution, though its proportional share has declined from peaks of 67% (2018) and 47% (2021) to approximately 43% in recent years. It is important to note that "***fine-tuning***" encompasses diverse approaches: in early CTG work (2018–2021), this typically referred to full-parameter fine-tuning, where all model parameters are updated. In contrast, recent work increasingly employs parameter-efficient fine-tuning (PEFT) methods such as LoRA, adapters, or prompt tuning, which update only a small subset of parameters. This distinction is relevant because PEFT methods have become increasingly prevalent as model sizes have grown, particularly with the adoption of large language models from 2023 onwards. Our categorisation does not distinguish between full-parameter and PEFT approaches, treating them uniformly as "***Fine-Tuning***", so the observed proportions reflect a heterogeneous

category that may mask underlying shifts in fine-tuning methodology. The relative decline in reported fine-tuning proportions likely reflects both the diversification of available methods as prompting-based approaches have become viable alternatives, and a shift towards more parameter-efficient fine-tuning strategies rather than a reduction in fine-tuning's importance.

Similarly, ***Hybrid*** implementation strategies, which combine multiple techniques within a single system, have decreased from a peak of 44% (2020) to stabilise around 18% in 2024–2025. This suggests that while combined approaches remain valuable for complex control scenarios, simpler single-method approaches suffice for many applications.

Modification of Token Distribution, which manipulates output probabilities during decoding without modifying model parameters, peaked at 33% in 2019, but has declined to approximately 7% by 2025. This decline likely reflects both the computational overhead of decoding-time interventions for large models and the effectiveness of prompting-based alternatives. ***Complete Training*** from scratch has remained consistently rare (0–5%), as pre-trained models have dominated the field. Text Selection, which generates multiple candidates and selects the best according to control criteria, similarly appears in fewer than 2% of papers, suggesting limited adoption of generate-and-filter approaches.

Figure 2.17 reveals the composition of hybrid implementation strategies. ***Fine-Tuning*** alone is most common (125 papers), followed by ***LM Prompting*** alone (52 papers). Among hybrid approaches, the combination of ***Fine-Tuning*** + ***Modification of Token Distribution*** is most prevalent (43 papers), representing systems that fine-tune models for improved controllability and then apply decoding-time interventions for fine-grained output manipulation. The second most common hybrid pairing combines ***Fine-Tuning*** + ***LM Prompting*** (33 papers), typically representing systems that fine-tune LLMs for specific control capabilities and then apply prompting strategies at inference time. ***Modification of Token Distribution*** alone appears in 26 papers, while more complex hybrid approaches involving

three or more implementation methods are less common. Notable patterns include *Fine-Tuning* + ***Modification of Token Distribution*** + *LM Prompting* (12 papers) and *Fine-Tuning* + ***Modification of Token Distribution*** + *Complete Training* (10 papers), suggesting that the additional complexity of combining multiple techniques is warranted only for particularly demanding control scenarios. Text Selection appears both alone and in a few hybrid combinations, though it remains one of the least common implementation methods overall.

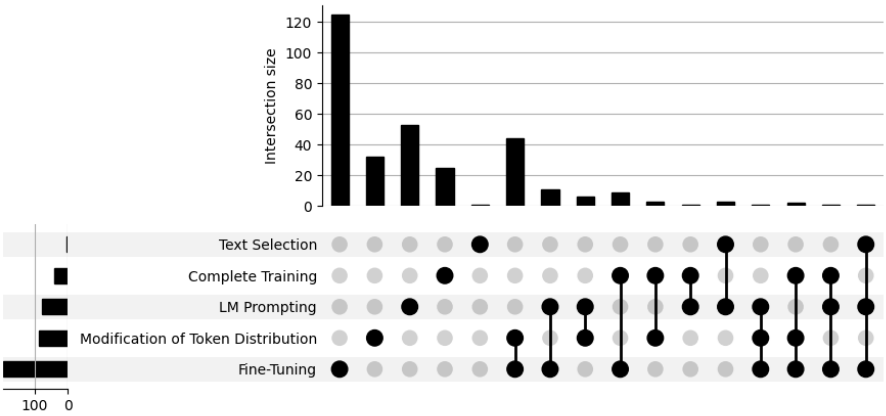
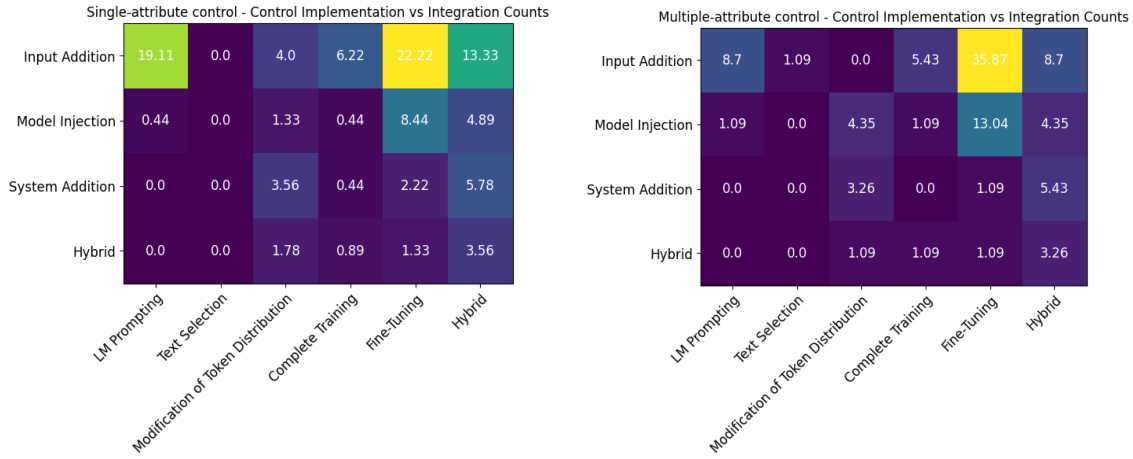


Figure 2.17: Distribution of Control Implementation methods across all papers. The bar chart (left) shows the frequency of each individual method; the matrix (bottom) indicates which methods appear in combination; the bar chart (top) shows the size of each combination.

2.6.5 Multi-dimensional Analysis

Having examined each categorisation dimension independently, we now analyse relationships between dimensions to identify systematic patterns and rarer combinations. Figures 2.18a and 2.18b show heatmaps showing the intersection of Control Implementation and Control Integration strategies, separately for ***single-attribute*** and ***multiple-attribute control*** systems. This decomposition reveals striking differences in architectural choices based on control complexity.

For single-attribute control systems (Figure 2.18a), *Input Addition* + *Fine-Tuning* emerges as the dominant combination, accounting for 22.22% of all single-attribute papers (56 papers). *Input Addition* + *LM Prompting* is the sec-



(a) Single-attribute control techniques (N=226)

(b) Multiple-attribute control techniques (N=92)

Figure 2.18: Control Implementation x Integration patterns for single-attribute versus multiple-attribute control systems. Cell values show percentage of papers within each control type.

ond most common approach at 19.11% (49 papers), reflecting the effectiveness of prompting-based control for single-attribute scenarios. ***Input Addition*** + ***Hybrid*** implementation also appears frequently (13.33%, 34 papers), representing systems that combine multiple implementation techniques while providing control signals as input. ***Model Injection*** + ***Fine-Tuning*** accounts for 8.44% (21 papers), while ***System Addition*** combinations are less common overall. Notably, certain combinations are entirely absent or extremely rare: Text Selection appears with no integration strategy, and several combinations show 0% adoption, suggesting these pairings are either technically incompatible or practically unnecessary for single-attribute control.

Multi-attribute control systems (Figure 2.18b) exhibit markedly different patterns. ***Input Addition*** + ***Fine-Tuning*** is substantially more dominant at 35.87% (23 papers) compared to 22.22% in single-attribute systems, indicating that multiple-attribute coordination benefits particularly from fine-tuning approaches that can learn complex attribute interactions. More strikingly, ***Input Addition*** + ***LM Prompting*** accounts for only 8.70% (6 papers) in multiple-attribute systems compared to 19.11% in single-attribute systems, suggesting that prompting-based approaches are less effective or less commonly adopted for coordinating multiple at-

tributes. Conversely, ***Model Injection*** + ***Fine-Tuning*** appears more frequently in multiple-attribute systems (13.04%, 8 papers) than in single-attribute systems (8.44%, 21 papers in absolute terms but lower percentage), indicating that specialised model components become more valuable when managing multiple control attributes simultaneously. ***Input Addition*** + ***Hybrid*** accounts for 8.70% (6 papers) in multiple-attribute systems versus 13.33% in single-attribute systems.

These patterns reveal systematic architectural differences based on control complexity. Single-attribute control admits simpler solutions: prompting-based approaches (***Input Addition*** + ***LM Prompting***: 19.11%) suffice for many applications, and fine-tuning with input conditioning (22.22%) handles more demanding scenarios. Multi-attribute control, by contrast, requires more sophisticated architectures: heavier reliance on fine-tuning (35.87% for ***Input Addition*** + ***Fine-Tuning***), greater proportional use of ***Model Injection*** to handle different attributes with specialised components (13.04%), and substantially reduced effectiveness of prompting-only approaches (8.70%). This architectural divergence helps explain why multiple-attribute control is less explored despite its practical importance: it involves substantially greater implementation complexity and computational resources than single-attribute control.

2.6.6 Model Architectures, Evaluation Datasets and Evaluation Metrics

Model Architectures. Figure 2.19 shows the distribution of model architectures employed across CTG papers, revealing the overall landscape. The field exhibits strong concentration around major pre-trained models: GPT-2 dominates with 55 papers (17.3%), followed by BART (29 papers, 9.1%) and T5 (20 papers, 6.3%). This concentration reflects the field's overwhelming reliance on pre-trained architectures rather than training models from scratch, consistent with our earlier finding that Complete Training as an implementation method appears in fewer than 5% of papers.

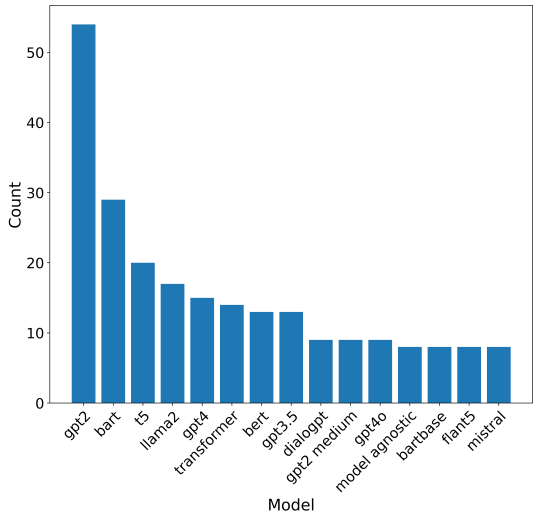


Figure 2.19: Model architecture usage across CTG papers showing the overall top 15 most frequently used models.

The temporal trends reveal distinct periods of model dominance, with clear shifts in which architectures researchers prioritised each year. Table 2.11 provides the most complete picture, showing the top 5 models used in each year.

Year	**Top 5 Models**				
2018	Encoder-Decoder (GRU) with CVAE (1)	—	—	—	—
2019	Transformer (2)	GPT-2 (1)	—	—	—
2020	GPT-2 (3)	DialogPT (2)	BERT (2)	Transformer (2)	XLNet (1)
2021	GPT-2 (10)	BART (5)	Model Agnostic (3)	T5 (1)	VAE with GRUs (1)
2022	GPT-2 (14)	BART (9)	Model Agnostic (5)	T5 (3)	DialogPT (3)
2023	GPT-2 (7)	GPT-2 Medium (7)	BART (6)	BART Base (5)	Transformer (4)
2024	LLaMA2 (14)	GPT-2 (14)	GPT-4 (9)	GPT-3.5 (9)	T5 (6)
2025	GPT-4o (9)	T5 (7)	LLaMA3 (6)	BART (5)	GPT-4 (5)

Table 2.11: Top 5 most frequently used models by year. Numbers in parentheses indicate paper count.

The 2018–2020 period shows early exploration: GPT-2 begins appearing in 2019 and becomes dominant by 2020 (3 papers), but alongside competing approaches

(DialogPT, BERT, Transformer, XLNet). The 2021–2022 period represents clear GPT-2 dominance: reaching 10 papers (52.6% of 2021) and 14 papers (31.8% of 2022), with BART as the secondary choice. Notably, a "Model Agnostic" category appears in both years (3–5 papers), representing researchers prioritising control methods over architecture selection.

A transition point occurs in 2023, where GPT-2 and its variants (GPT-2 Medium) co-exist with equal prevalence (7 papers each), alongside BART variants (11 papers combined). This represents a period of consolidation: while GPT-2 remains common, researchers increasingly experiment with variants and alternatives. The landscape shifts in 2024, where LLaMA2 and GPT-2 tie at 14 papers each (31.8% and 31.8% of papers), while GPT-4 and GPT-3.5 both appear in 9 papers (20.5% each), marking the first year where instruction-tuned models rival or exceed GPT-2. Finally, 2025 shows maximum diversity: no model exceeds 10% of papers, with GPT-4o, T5, LLaMA3, BART, and GPT-4 distributed across the top 5.

Evaluation Metrics. Figure 2.20 shows the overall distribution of evaluation metrics used across CTG papers. Perplexity is most frequently used (68 papers), followed by Distinct-n (53 papers), BLEU (50 papers), Attribute Relevance (35 papers), and ROUGE (26 papers). The long tail of metrics beyond these top five reflects the methodological diversity of CTG evaluation.

Table 2.12 reveals shifts in metric adoption across the survey period. Early years (2018–2019) show limited data, with Perplexity, Distinct-n, and Attribute Relevance appearing in all papers, though sample sizes are small. By 2020, BLEU and Distinct-n emerge as the most frequently used metrics among the top 5 (8 and 7 papers respectively), with Perplexity, Attribute Relevance, and ROUGE also present.

The 2021–2023 period shows Perplexity growing in absolute numbers: from 10 papers in 2021 to 30 papers in 2023. Distinct-n similarly increases (10 to 18 papers), while Attribute Relevance remains relatively stable (6 to declining presence). In 2024, Perplexity reaches its peak at 37 papers, followed by BLEU (24) and ROUGE

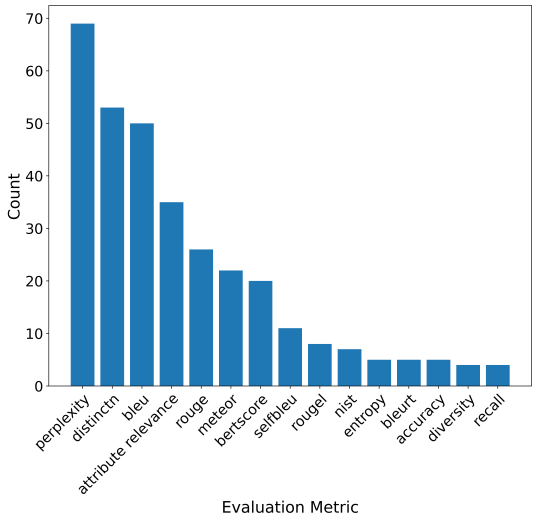


Figure 2.20: Automatic evaluation metrics usage across CTG papers showing the frequency of the top 15 most used evaluation metrics.

Year	**Top 5 Metrics**				
2018	Perplexity (1)	Distinct-n (1)	Attribute relevance (1)	—	—
2019	Perplexity (1)	Distinct-n (1)	Attribute relevance (1)	ROUGE (1)	BLEU (1)
2020	BLEU (8)	Distinct-n (7)	Perplexity (5)	Attribute relevance (5)	ROUGE (5)
2021	Perplexity (10)	Distinct-n (10)	BLEU (9)	Attribute relevance (6)	BERTScore (4)
2022	Perplexity (22)	BLEU (19)	Attribute relevance (18)	Distinct-n (16)	ROUGE (11)
2023	Perplexity (30)	Distinct-n (18)	BLEU (13)	BERTScore (10)	METEOR (9)
2024	Perplexity (37)	BLEU (24)	ROUGE (22)	BERTScore (18)	Distinct-n (14)
2025	BERTScore (24)	BLEU (21)	ROUGE (21)	Perplexity (17)	Attribute relevance (10)

Table 2.12: Top 5 most frequently used automatic evaluation metrics by year. Numbers in parentheses indicate paper count.

(22). Distinct-n drops to 14 papers and does not appear in the top 5 for 2025.

By 2025, the metric landscape has shifted. BERTScore leads with 24 papers, followed by BLEU and ROUGE (21 each), Perplexity (17), and Attribute Relevance (10). This represents a marked change from earlier years: Perplexity declines from its 2024 peak, while BERTScore rises prominently. The emergence and sustained presence of ROUGE and BERTScore in recent years, alongside the reduced

prominence of Distinct-n, suggests a shift in evaluation priorities away from lexical diversity metrics towards reference-based and neural-based alternatives.

Evaluation Datasets. Figure 2.21 shows the overall distribution of evaluation datasets used across CTG papers. RealToxicityPrompts dominates with 19 papers, followed by PPLM Prompts (15 papers), Yelp (11 papers), CommonGen (10 papers), and DailyDialog (7 papers). The fragmentation across numerous distinct datasets reflects the diversity of control objectives and the lack of standardised evaluation benchmarks for controlled text generation.

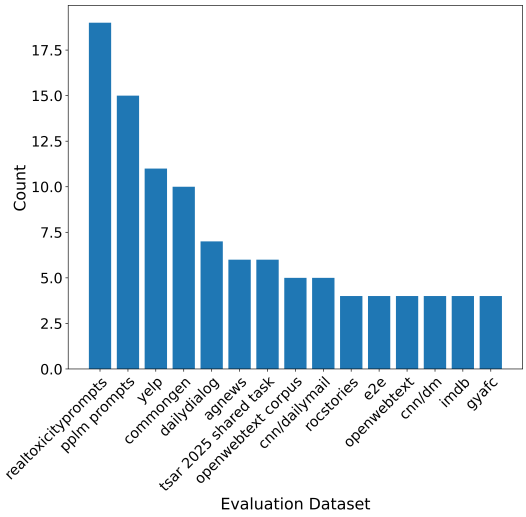


Figure 2.21: Evaluation datasets usage across CTG papers showing the frequency of the top 15 most used evaluation datasets.

Table 2.13 reveals shifts in dataset usage across the survey period. Early years (2018–2020) show high dataset diversity, with each year using different datasets rather than shared benchmarks. PPLM Prompts appears in the top 5 beginning in 2019 (1 paper), increasing to 4 papers in 2022 and 5 papers in 2023, before declining to 3 papers in 2024 and disappearing from the top 5 in 2025.

RealToxicityPrompts emerges in the top 5 in 2022 (2 papers) and grows reaching 6 papers in 2023, 8 papers in 2024, and remaining present with 2 papers in 2025. This sustained presence indicates ongoing research attention to toxicity control. Yelp review datasets show variable adoption across years: appearing with 2 papers

Year	Top 5 Datasets				
2018	Dialogue dataset from weibo.com (1)	—	—	—	—
2019	PPLM prompts (1)	MS Marco 2.1 (1)	NarrativeQA (1)	—	—
2020	Movie plots from Wikipedia (1)	PersonageNLG (1)	EndToEnd challenge dataset (1)	(Adiwardana et al., 2019) prompts (1)	Movie reviews (1)
2021	ROCStory (2)	DailyDialog (2)	CNN/ Dailymail corpus (1)	SemEval Hyperpartisan News (1)	Detection dataset (1)
2022	PPLM prompts (4)	CommonGen (2)	DailyDialog (2)	RealToxicity Prompts (2)	Wizard of Wikipedia (2)
2023	RealToxicity Prompts (6)	PPLM prompts (5)	CommonGen (3)	E2E (2)	Yelp review (2)
2024	Yelp review (9)	RealToxicity Prompts (8)	CommonGen (4)	GYAFC (3)	PPLM prompts (3)
2025	TSAR 2025 shared task (6)	IMDb (3)	MRPC (2)	CNN/ Dailymail (2)	RealToxicity Prompts (2)

Table 2.13: Top 5 most frequently used evaluation datasets by year. Numbers in parentheses indicate paper count.

in 2023, rising to 9 papers in 2024, but absent from the top 5 in 2025. CommonGen similarly appears sporadically (2 papers in 2022, 3 in 2023, 4 in 2024) and does not appear in 2025's top 5.

DailyDialog appears early (2 papers in 2021, 2 in 2022) but is absent from the top 5 in subsequent years. CNN/Dailymail corpus appears in 2020 (1 paper) and 2021 (1 paper), and reappears in 2025 (2 papers). The 2025 cohort introduces new datasets: TSAR 2025 shared task (6 papers), IMDb (3 papers), and MRPC (2 papers), none of which appear in earlier years' top 5.

The absence of a stable, widely-adopted evaluation dataset across multiple years contrasts sharply with general language generation benchmarks that establish clear evaluation standards. This fragmentation reflects the diversity of control objectives researchers pursue, but also indicates a lack of standardised evaluation infrastructure

that would enable systematic comparison across methods and facilitate reproducibility.

2.7 Discussion and Framework Utility

Having developed a multi-dimensional categorisation framework, applied it systematically to 336 papers, and demonstrated its automation potential, we reflect on the framework's utility for CTG research and acknowledge its broader limitations.

The multi-dimensional framework provides a structured lens for understanding the CTG landscape that reveals patterns obscured by single-dimensional categorisation. By treating control attributes, control integration, and control implementation as orthogonal dimensions, the framework enables precise characterisation of any CTG system while facilitating systematic comparison across approaches that differ along multiple axes. Our dataset analysis (Section 2.6) demonstrates this concretely: while certain combinations are well represented, ***Input Addition*** combined with ***Fine-Tuning*** for single-attribute control accounts for 26.1% of papers, other theoretically viable combinations remain under-explored, such as ***System Addition*** with ***LM Prompting*** (fewer than 1% of papers).

The temporal analysis reveals fundamental shifts in research methodology. The rise of ***LM Prompting*** following the emergence of instruction-tuned LLMs, coupled with the shift from ***System Addition*** towards ***Input Addition*** dominance, reflects an architectural evolution from methods relying on external modules towards approaches that embed control capabilities directly within models. The analysis also identifies persistent challenges: despite its practical importance, multiple-attribute control remains a minority focus, and its implementation tends to require more sophisticated architectural choices with greater reliance on fine-tuning.

Beyond organising existing work, the framework provides structured vocabulary for positioning new contributions. Rather than asserting novelty in vague terms, researchers can precisely specify where their approach sits along each dimension. For instance, our approach to ***multiple-attribute control*** (Chapters 5 and 6) combines

Input Addition, control attributes are specified in the input, with ***Fine-Tuning***, we train specialised PEFT modules dedicated to each control attribute.

The annotation dataset also identifies specific research opportunities: multiple-attribute control with diverse integration strategies, training-free multiple-attribute control through prompting, and fine-grained or holistic stylistic control, all of which remain under-represented in the literature. Our analysis of evaluation metrics reveals that while the field is moving towards multi-metric evaluation, establishing robust standards remains an open challenge: the decline of metrics such as distinct-n and BLEU without a clear dominant alternative indicates that fundamental methodological questions remain unresolved.

Several broader limitations should be acknowledged. Our systematic paper selection focuses exclusively on the ACL Anthology, potentially missing relevant work from venues such as ICML, ICLR, or NeurIPS. The framework balances generality against specificity: more fine-grained distinctions would enable more precise characterisation but at the cost of increased complexity and potentially reduced annotation reliability. Some papers span multiple categories or exhibit characteristics not cleanly captured by the framework, and while we developed decision criteria to ensure consistency, these boundaries remain somewhat arbitrary. Finally, our analysis identifies correlations and temporal trends but cannot establish causality: we observe that ***Input Addition*** increased while ***System Addition*** decreased, coinciding with the rise of instruction-tuned LLMs, but cannot definitively attribute one to the other.

By making explicit the dimensions along which CTG systems vary and systematically characterising the research landscape, this framework enables more rigorous and systematic research in controlled text generation. It serves as the foundation for the literature review in Chapter 3 and provides context for understanding how our contributions extend existing work.

2.8 Chapter Summary

This chapter presented a multi-dimensional categorisation framework for controlled text generation systems, addressing the need for systematic organisation of a rapidly growing and diverse research landscape.

We introduced a framework that categorises CTG approaches along three orthogonal dimensions: control attributes (what is being controlled), control integration (where control signals enter the system), and control implementation (how control is achieved). Through systematic paper selection from the ACL Anthology, we developed and validated this framework via manual annotation of 140 papers published through December 2023, demonstrating its stability and generalisability to held-out papers.

We then investigated whether this annotation process could be automated, developing an LLM-based approach combining reasoning-augmented SFT with RL refinement. The resulting system achieved annotation quality sufficient for systematic literature analysis while substantially reducing annotation time compared to fully manual approaches. This enabled comprehensive analysis of 336 papers spanning 2018-2025, revealing key patterns: the dominance of single-attribute control, the rise of ***Input Addition*** and ***LM Prompting*** following instruction-tuned LLMs, and systematic architectural differences between single- and multiple-attribute control systems.

The framework identifies under-explored research opportunities, provides structured vocabulary for positioning new contributions, and serves as the foundation for the literature review in Chapter 3 and for situating our attribute-specialised PEFT approach (Chapters 5 and 6) within the broader CTG landscape.

Chapter 3

Related Research

This chapter surveys the research landscape that informs the contributions of this thesis. We begin by reviewing existing surveys on controlled text generation (Section 3.1), which contextualises the categorisation scheme introduced in Chapter 2. We then examine the main families of Controlled Text Generation (CTG) techniques (Section 3.2), tracing the evolution of the field and grounding our experimental work in Chapters 4 and 6. Finally, we review parameter-efficient fine-tuning (Section 3.3), PEFT module merging (Section 3.4), and PEFT module portability (Section 3.5), which underpin the core contributions of Chapters 5 and 6.

3.1 Surveys on Controlled Text Generation

The rapid growth of the CTG literature has prompted a number of survey efforts, each proposing a way to organise the field. These works differ in scope, architectural focus, and methodological rigour, and collectively reveal the absence of a unified, multi-dimensional framework that is both architecture-agnostic and empirically validated. In this section we review the most relevant surveys and highlight how their limitations motivated the design choices behind the categorisation framework introduced in Chapter 2.

Prabhumoye et al. (2020) analyse CTG by decomposing the language generation pipeline into five components: external input, sequential input, generator opera-

tions, output, and training objective, and categorise control techniques based on which component is modified to achieve control. This component-based schema provides a useful lens for understanding the technical mechanisms underlying different control approaches, with particular attention to autoregressive models. However, the schema is tightly coupled to specific architectural choices and does not generalise cleanly to diverse generation paradigms such as diffusion-based text generation or retrieval-augmented systems. Moreover, the work predates the emergence of large-scale pre-trained language models, and consequently offers limited treatment of prompting-based and instruction-following control methods that have since become central to contemporary CTG research.

Zhang et al. (2023) provide a comprehensive overview of CTG techniques grounded in pre-trained language models, organising control conditions into semantic, structural, and lexical types, and categorising methods into fine-tuning, retraining/ refactoring, and post-processing approaches. Based on how control signals interact with the PLM, they identify seven subtypes of control method, offering detailed coverage of Transformer-based approaches and useful analysis of training paradigms. The scope of the survey is, however, restricted to Transformer-based PLMs, which excludes earlier architectures (e.g., LSTM-based methods) that established foundational CTG concepts, as well as emerging non-Transformer approaches. This architectural focus, while providing depth, limits the generalisability of their categorisation scheme across the broader CTG landscape.

In our own earlier work (Lorandi and Belz, 2023b), we propose a typology based specifically on how control is implemented in neural architectures, identifying five implementation categories: Complete Training, Model Fine-Tuning, Disentanglement, Modification of Token Distribution, and Hybrid methods. This implementation-focused perspective offers useful insights into the architectural and training requirements of different control strategies. However, the analysis is restricted to sentiment-controlled text generation, examining control implementation for a single attribute type. Consequently, the typology does not address how control techniques gener-

alise across different attribute types (e.g., structural, content, or stylistic control), nor does it examine multiple-attribute control scenarios in which multiple potentially conflicting attributes must be managed simultaneously.

Wang et al. (2024a) take a novel perspective by analysing CTG techniques through the lens of causality theory, examining how causal relationships between control attributes and generated text can be modelled and leveraged. Their causal framework provides principled foundations for evaluating control effectiveness and highlights the importance of disentangling correlation from causation in control signal design. This theoretical grounding complements empirical surveys by explaining when and why certain control techniques succeed or fail. Nevertheless, the focus on causal mechanisms means their work offers less comprehensive coverage of the practical implementation landscape, and does not systematically categorise the full range of control techniques, attributes, and integration strategies employed across the CTG literature.

Most recently, Liang et al. (2024b) present a survey focused on CTG with large language models. They divide techniques temporally into training-stage methods (retraining, fine-tuning, reinforcement learning) and inference-stage methods (prompt engineering, latent space manipulation, decoding-time intervention), and organise controlled attributes into content control (structure and vocabulary) and attribute control (safety, sentiment, style, and topic). This temporal division offers a useful perspective on when control is applied in the model lifecycle, and the survey also discusses evaluation metrics employed across CTG research. However, the exclusive focus on Transformer-based LLMs limits coverage to recent large-scale models. Furthermore, while the authors mention a systematic paper selection process, methodological details are not provided, making it difficult to assess potential coverage gaps or selection biases. The survey reports no quantitative statistics on the analysed papers, such as temporal distribution or the prevalence of different implementation strategies, limiting insights into field-wide trends. Finally, no validation process is described to assess whether the proposed categorisation scheme

generalises to papers beyond those used in its development.

Taken together, the surveys reviewed here share several recurring limitations: architectural restrictions that exclude pre-LLM or non-Transformer methods; single-attribute or single-dimension categorisation that obscures relationships between control strategies; and insufficient methodological transparency in paper selection and scheme validation. These shortcomings directly motivated the multi-dimensional categorisation framework we developed in Chapter 2, which is architecturally agnostic, spans a broad range of attribute types and control implementations, and is grounded in a transparent, reproducible annotation methodology applied to a systematically selected corpus of CTG papers. We use that framework to structure the review of CTG techniques in Section 3.2.

3.2 Controlled Text Generation Techniques

In Chapter 2, we introduced a three-dimensional categorisation framework for controlled text generation systems. This section applies that framework to organise and analyse existing work. We structure our review according to the implementation dimension (training-free, training-based, and hybrid approaches), while discussing how each approach handles different attribute types and representations.

Given the size of the corpus underlying our categorisation scheme (Section 2.4), this review does not aim to discuss every paper individually; rather, within each implementation category, we select a subset of papers to show the range of approaches taken. Paper selection was guided by three considerations. First, we prioritised coverage: within each implementation–integration combination, we selected works that collectively span the range of strategies observed in that combination, rather than multiple papers illustrating the same underlying idea. Second, we selected across publication years, so that each subsection reflects how a given approach has evolved over time rather than reflecting only its most recent instantiations. Third, we prioritised papers that introduced techniques with substantial downstream influence, either as foundational methods that later work directly builds upon, or as baselines

widely adopted for comparison across the field, since these are most representative of the direction the field has taken.

3.2.1 Training-Free Implementation Methods

LM Prompting. Prompting-based approaches achieve control by conditioning a pre-trained LM on carefully constructed inputs, requiring no fine-tuning or modification of model weights. In terms of our categorisation framework, these techniques use ***input addition*** as the control integration method, prepending or augmenting the generation prompt with control-relevant context.

Several works apply this strategy to knowledge-grounded and mixed-initiative dialogue generation. Liu et al. (2022) propose a Multi-Stage Dialogue Prompting (MSDP) framework that decomposes generation into a knowledge generation stage and a response generation stage, both using the same pre-trained LM without weight updates. Chen et al. (2023a) extend this line of work to mixed-initiative dialogue, where fine-tuning approaches rely on error-prone intent classifiers as weak supervisors, introducing compounded errors and a trade-off between response quality and control accuracy. Prompting sidesteps both issues by encoding task background and conversation history into general-purpose prompt templates, with human evaluation confirming improvements over fine-tuning on intent accuracy, coherence, and engagingness on two benchmarks. Chen et al. (2023b) propose IKA, a plug-and-play retrieval-augmented framework combining a retrieval module, a diversity-enhancing reranking step, and a prompt construction module, compatible with both open-source and black-box LLMs without parameter updates.

Together, these works show that prompting-based control is effective across a range of dialogue tasks, and that the absence of weight updates is both a practical advantage, enabling deployment with black-box models, and a limitation, as the degree of control achievable is bounded by what can be expressed through input construction alone.

Text Selection. Text selection approaches achieve control not by modifying the generation process itself, but by producing a large pool of candidate outputs and selecting the best response according to explicit evaluation criteria.

Zhou et al. (2023) propose SimOAP (over-sampling and post-evaluation), a two-stage framework for improving coherence and consistency in persona-based dialogue. In the over-sampling stage, large-scale candidate responses are generated efficiently using distilled or compressed dialogue models to manage the additional computational cost of sampling at scale. In the post-evaluation stage, candidates are ranked using TF-IDF for coherence assessment and natural language inference (NLI) for consistency evaluation, with the highest-scoring response selected as the final output.

The relative scarcity of text selection approaches in the CTG literature reflects an inherent tension in the strategy: while decoupling control from generation avoids any modification to the underlying model, its effectiveness is bounded by the quality of the candidate pool and the reliability of the evaluation metrics used for selection. Nevertheless, SimOAP demonstrates that even simple post-hoc selection criteria can yield meaningful improvements in attribute consistency without any training overhead.

Modification of Token Distribution. Methods in this category achieve control by intervening directly in the token probability distribution produced by a pre-trained LM at inference time, without modifying the underlying model weights. The interventions take different forms, gradient-based latent updates, weighted decoding, energy-based sampling, and classifier-guided gating, but share the characteristic that control is exercised dynamically at each generation step.

The most influential early work in this category is Dathathri et al. (2019), who propose Plug and Play Language Models (PPLM), using the term to indicate that no task-specific training of the LM is required and the method can be applied to any model whose internal representations are accessible. At each decoding step, the next-token distribution produced by the LM is fed to external attribute clas-

sifiers; gradients from these classifiers are then used to update the LM's internal latent representations via a backward pass, shifting the distribution towards the desired attribute before the next token is sampled. PPLM supports multiple-attribute control (e.g., sentiment and topic simultaneously) and employs a hybrid control integration combining input addition and system addition. Building on this framework, Madotto et al. (2020) introduce two extensions: the backward pass is repeated n times to modulate the intensity of the attribute signal, and lightweight Residual Adapters (Houlsby et al., 2019) are inserted atop each Transformer layer to steer the output distribution without altering the base model's parameters. A further refinement is proposed by Gu et al. (2022a), who observe that using a controller alone creates a trade-off problem: the controller monopolises the token distribution at the expense of the LM's original fluency. They address this by introducing a regulator module that dynamically amplifies or suppresses the controller's bias signal at each step based on the divergence between the LM's distribution and the target attribute; the regulator can be implemented either as a lightweight module trained on task-specific data or defined heuristically, making the approach broadly applicable without requiring additional supervised training in all settings.

A second group of methods achieves control through weighted decoding and reranking rather than latent gradient updates. Yang and Klein (2021) propose Future Discriminators for Generation (FUDGE), which learns an attribute predictor on partial sequences and uses its predictions to adjust the LM's output logits at each step, requiring only access to the model's output probabilities and no architectural changes. Landsman et al. (2022) take a similar approach at the beam level: diverse beam search, which promotes variety between the k decoded candidates by penalising similarity within the beam, is used to generate candidates, which are then scored by an attribute model and used to reweigh the original token likelihoods, producing a distribution that jointly reflects fluency and attribute characteristics. More recently, Shin et al. (2025) address a limitation common to fixed-weight decoding methods: a constant control strength cannot simultaneously optimise controllability and fluency

across all generation steps. They propose ECO (Entropy-based COntrol) decoding, which dynamically adjusts the weight of the attribute signal at each step according to the entropy of both the LM and the classifier distributions: when the LM is confident (low entropy), fluency is prioritised; when it is uncertain (high entropy), the attribute signal is amplified. This aligns naturally with linguistic intuition, as content words exhibit higher uncertainty than grammatical function words and thus afford greater scope for attribute manipulation without degrading naturalness.

Kumar et al. (2022) take a different approach, combining LM log-likelihoods with arbitrary discriminative constraints in a single energy function, from which samples are drawn in a non-autoregressive manner. This formulation supports multiple-attribute control over sentiment and toxicity and allows constraints to be implemented as classifiers trained from scratch or fine-tuned, under a hybrid input-addition and system-addition control integration strategy.

A final group of methods uses auxiliary classifiers or parsers to gate or rerank the token distribution rather than directly modifying it. Lango and Dusek (2023) propose a critic-driven decoding approach for data-to-text generation in which a text critic classifier evaluates the coherence between the input data and the prefix generated so far, combining its output with the LM's token probabilities at each step without requiring any changes to the underlying model or additional training data and using an input addition control integration strategy. Kim and Cho (2023) observe that existing CTG methods designed to reduce toxicity degrade generation quality on other dimensions, including topic consistency, grammar, and perplexity. They address this by proposing Gated Toxicity Avoidance (GTA), a model-agnostic gating mechanism that applies a CTG method only when a pre-trained binary toxicity classifier detects that the token under consideration is likely to be toxic. By activating the control signal selectively, GTA preserves the LM's default distribution for non-toxic tokens and improves generation speed compared to methods that apply guidance at every step. Finally, Zingale and Kalita (2024) propose a parser-driven decoding scheme for rhetorical structure control, in which an RST (Rhetorical

Structure Theory) tree parser re-ranks the top of the LM's token distribution at each step to favour tokens consistent with the specified discourse relation, requiring no fine-tuning and operating entirely at inference time.

Across all token-distribution intervention methods, a recurring tension emerges between the strength of attribute control and the preservation of fluency. Gradient-based approaches such as PPLM offer strong control but risk overriding the LM's original distribution; weighted decoding methods mitigate this but typically rely on a fixed control strength that may not generalise across generation steps or attribute types. Adaptive methods such as ECO and GTA represent more recent efforts to resolve this tension dynamically, pointing towards a more principled treatment of the control-fluency trade-off.

Hybrid Training-Free Methods. A number of training-free methods combine prompting-based input construction with direct manipulation of the token distribution, exploiting the complementary strengths of both strategies: prompts provide high-level semantic guidance while distribution-level interventions enforce finer-grained attribute control. All methods in this group employ a hybrid control integration combining input addition and system addition.

Pei et al. (2023) propose Prefix-Adaptive Decoding (PREADD), which achieves attribute control without any external classifier or auxiliary model. At each decoding step, the method computes two sets of output logits: one conditioned on a raw prompt and one conditioned on a prefix-prepended version of the same prompt. The difference between these logits is then amplified or negated to adjust control strength, enabling both positive and negative control over any attribute that can be expressed through a natural language prefix. The simplicity of this approach, requiring only two forward passes through the same model, makes it broadly applicable without task-specific training.

Yang et al. (2023) propose the Detailed Outline Control (DOC) framework for long-form story generation, where maintaining plot coherence over thousands of words presents a particular challenge for purely prompt-based methods. DOC de-

composes generation into a planning stage and a drafting stage. In the planning stage, a structured prompting procedure iteratively refines a brief initial outline into a detailed hierarchical structure, with each item associated with characters, settings, and coherence constraints. In the drafting stage, a detailed controller enforces faithfulness to the outline during passage generation by using outline items as control signals that modulate the token distribution, providing sufficient control strength to satisfy the overlapping soft constraints imposed by the outline.

Liang et al. (2024a) propose DATG (Dynamic Attribute Graphs-based controlled text generation), a pluggable framework that introduces an intermediate graph-based representation to mediate between attribute classifiers and token-level control. Candidate text sequences are first generated from attribute-specific prompts and scored by an attribute classifier. These scored sequences are used to construct two dynamic directed weighted graphs, a positive attribute graph, weighted by classifier consistency scores, and a negative attribute graph, weighted by their complements, which together represent the semantic space of the target attribute. A graph ranking algorithm then identifies 'key attribute words' and 'key anti-attribute words', whose occurrence is modulated during regeneration via logit boosting and prefix-prompt strategies.

Across these methods, the hybrid combination of prompting and distribution manipulation offers greater control precision than either strategy alone. PREADD achieves this with minimal overhead by contrasting prompt variants; DOC structures the interaction hierarchically across a long generation process; and DATG introduces an explicit semantic graph as an intermediate representation to make the connection between classifier signals and token-level decisions more principled. Together they illustrate that the boundary between prompting and decoding-time control is increasingly blurred in contemporary training-free CTG.

3.2.2 Training Implementation Methods

Complete Training. Complete training approaches build controllability directly into the model architecture and training objective, rather than adapting a pre-existing language model. This affords greater architectural flexibility but requires training from scratch, making these methods computationally expensive and less readily transferable across tasks or domains.

Early work in this category draws on generative adversarial networks (GANs). Betti et al. (2020) propose a text GAN comprising one generator and two discriminators: a syntax discriminator that distinguishes real from generated sentences, and a semantic discriminator that assesses whether the output expresses the desired attribute. The generator is a Relational Memory network with self-attention (Santoro et al., 2018), and the Gumbel-softmax trick (Jang et al., 2016) is applied to address the non-differentiability of discrete text generation. The approach supports single-attribute control (sentiment or topic) under a hybrid input-addition and system-addition control integration strategy. To extend GAN-based generation to multiple-attribute control, Qiao et al. (2020) propose a Conditional Variational Auto-Encoder trained adversarially for sentiment-controllable topic-to-essay generation. Topic control is achieved via a Topic Graph Attention network that incorporates a knowledge graph into the generation process, while sentiment control is achieved by injecting a sentiment representation into both the encoder and decoder. The model simultaneously governs the topic of the essay and the sentiment of each constituent sentence.

A second group of complete training approaches adopts specialised encoder-decoder architectures designed around structural or psychological control signals. Xie et al. (2022) propose a psychology-guided story generation model that conditions generation on the evolution of a protagonist's psychological state, represented as chains of emotions and needs. Psychology controllers integrate local and global psychological state into the story context representation at each generation step. Yang et al. (2022) propose SI-SCP (Structural Information-augmented Syntax-Controlled

Paraphrasing), a syntax-controlled paraphrase generation model that encodes constituency parse trees using a tree Transformer, capturing parent-child, sibling, and word-node alignment relations. An attention regularisation objective ensures that the decoder accurately selects the syntactic nodes relevant to each generated word, and a Syntactic Template Retriever supports diverse paraphrase generation by retrieving compatible syntactic structures.

Taking a different approach to structural control, Maheshwari et al. (2024) present DICTDIS, a lexically constrained neural machine translation system that addresses the disambiguation problem arising when domain-specific dictionaries provide multiple candidate target-language phrases for a single source word or phrase due to polysemy. Rather than enforcing hard lexical constraints, DICTDIS augments training data with multiple candidate translations and trains the model to select contextually appropriate alternatives in a soft manner, preserving fluency while satisfying translation constraints. Hewitt et al. (2023) propose Backpacks, a novel architecture in which each vocabulary item is represented as a set of non-contextual sense vectors, with the final contextual representation computed as a learned, context-dependent linear combination of these vectors. Sense vectors specialise during training to encode distinct semantic aspects of a word, and can be directly intervened upon to steer model behaviour in predictable ways (model injection). Backpacks thus offer a form of intrinsic interpretability that is absent from standard Transformer architectures.

More recently, diffusion models have emerged as an alternative complete training paradigm for CTG. Zhang et al. (2024) propose Language Rectified Flow, which reformulates text generation as an ordinary differential equation transporting a source noise distribution to a target text distribution, using a VAE to bridge the discrete text space and the continuous flow latent space. Unlike standard diffusion models, which follow curved stochastic trajectories requiring thousands of denoising steps, Language Rectified Flow constrains the transport path to follow straight lines between the two distributions, the theoretically shortest path, and one that can be

simulated precisely without time discretisation. This reduces the number of inference steps required by up to 27 times compared to diffusion-based language models, addressing a key efficiency bottleneck.

Tang et al. (2023) tackle a complementary problem: the training-inference discrepancy that arises in discrete diffusion models because the forward noising process is absent at inference time, and because uniform downsampling of diffusion steps creates trajectory mismatches. They propose two corrective strategies, Distance Penalty, which injects downsampled predicted reverse noise into a post-training stage, and Adaptive Decay Sampling, which adjusts downsampled steps non-uniformly according to the noise magnitude at each stage, to better align training and inference behaviour.

Taken together, complete training approaches offer the greatest architectural freedom for encoding control signals, but at the cost of computational expense and limited transferability. The emergence of diffusion-based methods represents a notable recent diversification of this category, complementing the longer-established GAN and encoder-decoder paradigms. However, the reliance on training from scratch remains a fundamental practical limitation, particularly as the field moves towards adapting large pre-trained models rather than building specialised architectures.

Fine-Tuning. Fine-Tuning can be achieved in many ways. One way is to focus on prefix tuning, i.e. fine-tuning a model to learn continuous attribute-specific vectors that will be prepended to the activations of the pre-trained model to steer text generation. Prefix tuning is itself a form of parameter-efficient fine-tuning (PEFT), a family of methods discussed in detail in Section 3.3, which also underpins the LoRA-based approaches used in Chapters 5 and 6 of this thesis. E.g., Qian et al. (2022) fine-tune GPT-2 (Radford et al., 2019) to obtain prefixes, but they use the contrast between prefixes, for example, positive vs negative sentiment, to encourage the desired attribute and discourage the opposite attribute. In this method, only the prefixes are trained and GPT-2 weights are kept frozen.

Similarly, Gu et al. (2022b) fine-tune BERT (Devlin et al., 2019) to obtain prefixes. The idea is to have an Autoencoder structure, i.e. the encoder-decoder reconstructs the input sentence, to map attribute-relevant sentences to latent representations of attributes. At inference time, the model searches the attribute representation in the attribute space and uses it as a prefix for the decoder. In the case of multiple attributes the intersection of attributes is taken as the prefix, rather than the contrastive approach of Qian et al. (2022), who train multiple prefixes simultaneously by exploiting the opposing relationship between attribute pairs (e.g. positive vs. negative sentiment) as a training signal.

Fang et al. (2022) further explore the usage of Variational Autoencoders to learn a latent representation of control attributes. The idea is to use contrastive learning to separate the latent space into several parts, thus obtaining learnable vectors associated with a control attribute. At inference time, all the vectors associated with the desired attribute are retrieved and combined with a Dirichlet distribution to produce a latent variable, which is fed to the decoder.

All three methods allow the control of multiple attributes (sentiment and topic) at the same time. The last supports control of the tense of the sentences together with the other attributes.

Hybrid Training Methods. Hybrid training methods combine the two training paradigms explained before, i.e. complete training and fine-tuning, to exploit the complementary strengths of each. A common motivation is that purely fine-tuned models may lack the structural inductive biases needed for precise attribute control, while models trained from scratch on small datasets risk poor generalisation. Hybrid approaches address this by using one training stage to establish a useful latent structure and a second to refine controllability.

Several works adopt a two-stage strategy centred on variational autoencoders. Yang et al. (2021) propose Syntactically-informed Unsupervised Paraphrasing (SUP), a framework for syntax-controlled paraphrase generation that avoids the need for parallel paraphrase data. In the first stage, a conditional VAE is trained from scratch

to reconstruct input sentences conditioned on their syntactic parse trees, establishing a latent space organised by syntactic structure. In the second stage, the pre-trained VAE is fine-tuned using syntax-controlling and cycle-reconstruction objectives, with Gumbel-Softmax applied to combine these objectives stably, improving both syntactic controllability and semantic consistency in the final paraphrases. Ding et al. (2023) propose MacLaSa for multi-aspect controllable generation, where the challenge is to reconcile potentially conflicting attribute signals from different domains. A VAE is first trained to map text sequences from diverse sources into a shared compact latent space, mitigating domain discrepancy across attributes. The latent space is then used to formulate joint energy-based models, which define a probability distribution as a product of attribute discriminator scores, allowing multiple attribute constraints to be combined without retraining. Arbitrary attribute discriminators can be plugged into this formulation to specify the desired attributes. Target sequences are produced by drawing latent samples using an ODE-based sampler, where an ordinary differential equation defines a deterministic continuous-time trajectory through the latent space that enables faster sampling than stochastic diffusion approaches. The sampled latent vectors are then decoded through the VAE to produce the final text. A single-layer GAN is additionally fitted on the learned latent space to narrow the gap between the prior and the VAE posterior.

Luo et al. (2022) propose PCG (Prefix-Controlled Generator) for few-shot table-to-text generation, combining a trained content planner with a fine-tuned pre-trained language model. A task-specific prefix is prepended to adapt the PLM's input to tabular structure, and an input-specific prefix generated by the content planner controls factual content and word order in the output. The content planner is trained from scratch while the generator is fine-tuned, with the prefix acting as the interface between the two components and providing a degree of factual control that pure fine-tuning alone cannot guarantee.

Zhu et al. (2024) propose StyleFlow for unsupervised text style transfer, addressing the core difficulty that many words carry both content and style information,

making clean disentanglement challenging. StyleFlow replaces the standard encoder with a normalising flow, a reversible encoder that can both encode input to a latent representation and exactly invert this process, composed of attention-aware coupling layers designed to improve content-style separation.

Across these methods, the two-stage training strategy emerges as the dominant pattern: an initial phase establishes a structured latent space or a base generation capability, and a subsequent fine-tuning phase refines attribute controllability. This separation of concerns mirrors the motivation behind modular PEFT approaches, where attribute-specific adaptation is kept separate from the base model's general capabilities, a parallel that motivates our own work in Chapter 6.

3.2.3 Hybrid Implementation Methods

Hybrid implementation methods combine two or more implementation strategies (LM Prompting, Text Selection, Modification of Token Distribution, Complete Training, Fine-Tuning) within a single system, typically to address limitations that arise when any single strategy is applied in isolation. The most common motivation is that training-based methods provide robust attribute representations but lack flexibility at inference time, while inference-time methods offer adaptability but are constrained by the base model's distribution. By combining the two, hybrid methods aim to achieve stronger and more generalisable control. We organise our review according to the specific combinations identified in our survey.

Modification of Token Distribution and Fine-Tuning. The most prevalent hybrid combination pairs fine-tuning with inference-time modification of the token distribution, using fine-tuned auxiliary models to guide decoding from a frozen base LM. Krause et al. (2021) propose GeDi, which trains class-conditional language models (CC-LMs) as generative discriminators. At each decoding step, Bayes' rule is applied to compute classification probabilities for all candidate next tokens by contrasting distributions conditioned on a desired and an undesired control code prepended

to the input, steering generation towards the target attribute while avoiding the computational cost of standard discriminator-guided decoding. Liu et al. (2021) propose DExperts, which combines a base LM with fine-tuned "expert" and "anti-expert" LMs in a product-of-experts framework: tokens receive high probability only if they are likely under the expert and unlikely under the anti-expert. Because DExperts operates solely on the output distributions of each model, it can steer generation even when the base LM is accessible only via an API. Zhang and Song (2022) propose DisCup, which incorporates attribute discriminator knowledge into the prompt-tuning objective rather than at decoding time. A frozen CLM generates top-k candidate tokens at each step, which are scored by an attribute discriminator to identify desired and undesired tokens; a combined likelihood and unlikelihood objective then optimises control prompts to favour the former and suppress the latter, improving multiple-attribute generalisation and reducing overfitting to corpus features beyond the target attributes. Yu et al. (2024) propose ScoPE (Score-based Progressive Editor), which targets the specific challenge of controlled generation with black-box models. Rather than intervening on the token distribution directly, ScoPE edits previously generated context tokens during generation using a score model trained to maximise a combined attribute, fluency, and task-specific score, guiding subsequent generation to incorporate target attributes naturally without requiring access to model parameters.

LM Prompting and Fine-Tuning. A second group of hybrid methods combines LM prompting with fine-tuning, using prompts either to provide structured control signals at inference time or to synthesise supervised training data. Xu et al. (2020) propose Megatron-CNTRL for controllable story generation, combining a keyword predictor, a knowledge retriever, a contextual knowledge ranker, and a conditional text generator. User-provided keywords are used to retrieve and rank relevant knowledge sentences from an external knowledge base via weak supervision from sentence embeddings; the top-ranked knowledge is then prepended as a prompt to guide the conditional generator, providing interpretable content control without

modifying the generation model's parameters. Han et al. (2024) propose DisenTrans for text style transfer, addressing the scarcity of parallel supervision through a disentanglement-based Chain-of-Thought prompting procedure that uses an LLM to identify and transfer style-bearing components of sentences while preserving content, synthesising parallel training data of sufficient quality. The fine-tuned model is then trained with a contrastive loss that pulls attribute component representations towards the original sentence and a sequence-to-sequence loss that constrains the semantic space. Hong et al. (2024) approach counterspeech generation by targeting conversation-level outcomes rather than purely linguistic attributes, combining instruction prompting, fine-tuning, and RL to optimise for two desired outcomes: reduced incivility in the conversation thread following the counterspeech, and non-hateful re-entry by the original hate speech poster, i.e. whether they subsequently return to the conversation without posting further hate.

Modification of Token Distribution and Complete Training. Two works combine complete training with inference-time distribution modification. Ke et al. (2018) propose a conversation generation model that controls sentence function via a conditional VAE whose latent variable is supervised by a discriminator with the expected function label. A type controller operates at each decoding step to modulate the generation distribution, estimating a distribution over function-related, topic, and ordinary word types, thereby making sentence function control both global, through the latent variable, and local, through the type controller. Ou and Jian (2024) address the compatibility between continuous diffusion models and pre-trained language models for text generation, replacing the standard rounding operation with a pre-trained decoder to convert denoised embedding vectors into natural language. They further propose a linguistic easy-first noise schedule that weights tokens by their importance, aligning the diffusion process with natural language generation patterns and improving output quality.

3.3 Parameter-Efficient Fine-Tuning Techniques

As pre-trained language models have grown in scale, full fine-tuning, which updates all model parameters for each downstream task, has become increasingly impractical. Beyond the computational cost of training, full fine-tuning produces a separate model copy per task, imposing significant storage overhead. Parameter-efficient fine-tuning (PEFT) addresses these limitations by keeping the pre-trained model frozen and introducing a small number of trainable parameters that encode task-specific adaptations. PEFT has been shown to effectively inject task-specific knowledge into pre-trained models (Liu et al., 2024b; Hu et al., 2022; Poth et al., 2023; Whitehouse et al., 2024), including settings where the base model has undergone minimal pre-training and most task-level knowledge must be acquired during fine-tuning (Zhao et al., 2024).

Following the taxonomy of Lialin et al. (2023), PEFT methods can be organised into three broad categories: addition-based, reparametrisation-based, and selection-based (Figure 3.1). We focus on the first two, which are the most widely adopted in the literature and underpin the methods used throughout this thesis. Figure 3.2 (Lialin et al., 2023) shows where each category of PEFT module is attached within the Transformer architecture.

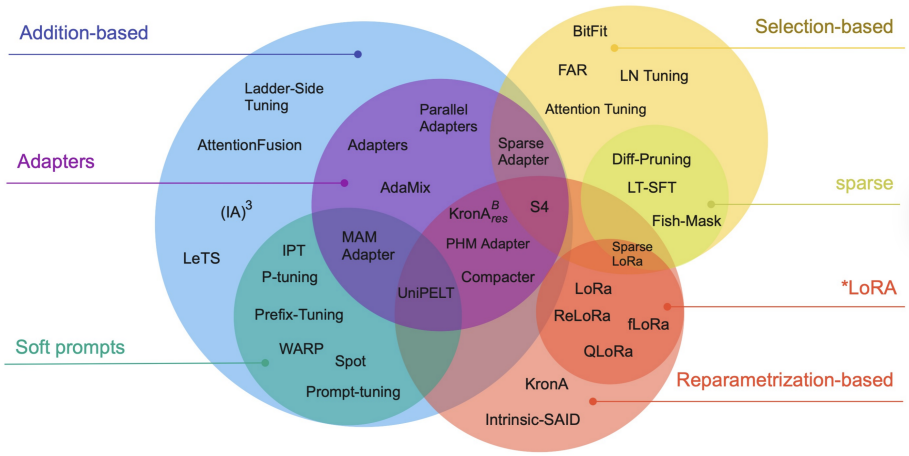


Figure 3.1: Taxonomy of parameter-efficient fine-tuning methods from Lialin et al. (2023), organised into three broad categories: addition-based (left), reparametrisation-based (right), and selection-based (top right). Methods appearing in overlapping regions combine characteristics of multiple categories.

Addition-based methods augment the pre-trained model with newly introduced parameters while keeping the original weights frozen, training only the added components. Two principal families have emerged within this category. The first, **adapter methods**, insert small trainable networks, typically a down-projection followed by a non-linearity and an up-projection, sequentially after specific Transformer sub-layers, as shown in Figure 3.2 (Adapters Attn / FF blocks) (Houlsby et al., 2019). Despite introducing relatively few parameters, adapters achieve performance comparable to full fine-tuning across a range of tasks, and their modular design allows a single frozen base model to serve multiple tasks simultaneously by swapping adapter weights. The second family, **soft prompt methods**, works on the model's input or hidden representations rather than its internal weights. *Prompt Tuning* (Lester et al., 2021) prepends a small set of trainable vectors to the input embeddings (PT block at the embedding layer in Figure 3.2), optimising these via gradient descent while the model parameters remain frozen. *Prefix Tuning* (Li and Liang, 2021) extends this idea by inserting trainable prefix vectors into the key and value hidden states at every model layer rather than only at the input (PF_K and PF_V blocks in Figure 3.2), achieving performance close to full fine-tuning with approximately 0.1% of the trainable parameters, and proving particularly effective on natural language generation tasks and in low-data settings.

Reparametrisation-based methods achieve parameter efficiency by expressing weight updates as low-rank decompositions, motivated by the observation that large neural networks exhibit low intrinsic dimensionality (Aghajanyan et al., 2021). Rather than introducing new components into the model as addition-based methods do, these methods reparametrised the weight update ΔW itself, representing the same transformation through a lower-dimensional parametrisation without adding new operations to the model's forward pass. The most influential method in this family is *Low-Rank Adaptation* (LoRA; Hu et al. 2022), which represents the update ΔW to a frozen weight matrix W as the product of two low-rank matrices: $\Delta W = W_A W_B$, where $W_A \in \mathbb{R}^{m \times r}$, $W_B \in \mathbb{R}^{r \times n}$, and the rank $r \ll \min(m, n)$ is

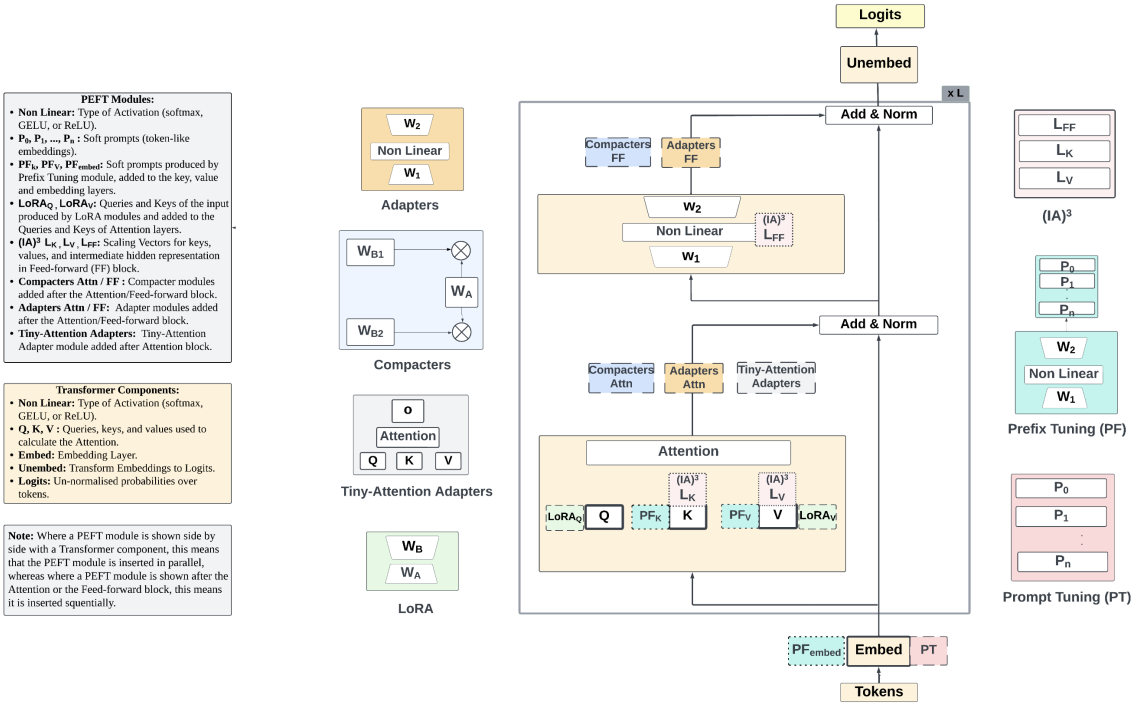


Figure 3.2: Placement of different PEFT modules within the Transformer architecture from Sabry and Belz (2023). Addition-based methods (Adapters, Compacters, Tiny-Attention Adapters, Prefix Tuning, Prompt Tuning) introduce new components either sequentially after Transformer sub-layers or in parallel with attention projections. Reparametrisation-based methods (LoRA) are inserted in parallel with the query and value projection matrices within the attention block.

a hyperparameter controlling the expressivity and parameter count of the adapter. The adapter output is scaled by a factor α/r, where α is a fixed hyperparameter, before being added to the frozen weight matrix, allowing the magnitude of the adaptation to be controlled independently of the rank. As shown in Figure 3.2, LoRA modules (LoRA_Q and LoRA_V) are inserted in parallel with the query and value projection matrices within the attention block, leaving the original weight matrices entirely unchanged during training. After fine-tuning, the low-rank update can be merged back into the base weight matrix ($W \leftarrow W + W_A W_B$), at which point no additional components remain in the model and inference incurs no latency overhead, a key practical advantage over addition-based methods. LoRA has been shown to match full fine-tuning performance while reducing the number of trainable parameters by several orders of magnitude, and its orthogonality to other efficiency techniques such as quantisation has further broadened its applicability.

Building directly on LoRA, Dettmers et al. (2023) propose QLoRA, which combines 4-bit quantisation of the frozen base model weights with LoRA adapters trained in full precision. A key contribution is the 4-bit NormalFloat (NF4) data type, optimised for normally distributed weights, together with double quantisation to reduce the memory footprint of the quantisation constants themselves. These innovations allow models with tens of billions of parameters to be fine-tuned on a single consumer GPU while preserving performance comparable to 16-bit full fine-tuning. QLoRA is the primary PEFT method employed in our experiments in Chapters 5 and 6.

Several variants of LoRA have been proposed to address specific limitations of the original formulation. Kalajdzievski (2023) observe that the α/r scaling factor causes gradient collapse as rank increases, limiting practical use to very low ranks. Their rank-stabilised LoRA (rsLoRA) replaces this with $\alpha/\sqrt{r}$, which is proved theoretically and verified empirically to stabilise gradient flow and enable performance gains at higher ranks. DoRA (Liu et al., 2024a) decomposes weight updates into magnitude and direction components, handling direction via standard LoRA while introducing a separate learnable scalar for magnitude, which enhances performance particularly at lower ranks. OLoRA (Büyükakyüz, 2024) uses QR decomposition, a matrix factorisation that expresses a matrix as the product of an orthogonal matrix and an upper triangular matrix, to initialise LoRA adapters with orthonormal matrices. Empirical evaluations show that this leads to faster convergence and improved performance compared to standard LoRA across a variety of language modelling tasks, while preserving the same parameter count and memory footprint.

While the above variants address complementary aspects of LoRA's behaviour, the core low-rank decomposition principle introduced by Hu et al. (2022) remains the foundation of all of them and of the work presented in this thesis.

3.4 Combining Multiple PEFT Modules

A recurring challenge in multi-task and multiple-attribute learning is how to combine the knowledge encoded in independently trained modules, such as separate LoRA adapters each fine-tuned for a different task, domain, or control attribute, without catastrophic forgetting or destructive parameter interference. Research on PEFT module merging addresses this challenge by developing principled strategies for composing adapter parameters, either through learned combination, weight arithmetic, or routing mechanisms. This section reviews the main approaches, with particular attention to LoRA-based methods that directly motivate our composition experiments in Chapter 6.

A first approach to module composition treats combination as a learned attention problem. Pfeiffer et al. (2021) propose AdapterFusion, a two-stage algorithm that separates knowledge extraction from knowledge composition. Task-specific adapter modules are first trained independently, after which their parameters are frozen and a small set of Key-Value-Query attention matrices is trained to learn a dynamic, context-dependent combination of adapter outputs at each Transformer layer.

A complementary line of work treats the parameter differences between a fine-tuned model and its pre-trained base as *task vectors* that can be combined arithmetically, without access to the original training data. Yang et al. (2024) propose Adaptive Model Merging (AdaMerging), which learns task- or layer-wise merging coefficients automatically using entropy minimisation on unlabelled test samples as a proxy objective. Entropy minimisation is motivated by the observation that lower prediction entropy correlates with lower task loss. Yadav et al. (2023) identify two principal sources of interference in naive parameter averaging, interference due to redundant parameter values and disagreement on the sign of a given parameter's values across models, and propose TIES-Merging (TrIm, Elect Sign & Merge), which addresses these by trimming parameters that changed little during fine-tuning, resolving sign conflicts, and averaging only parameters whose sign agrees with the elected direction. Together, these methods demonstrate that model merging can be

substantially improved by attending to the structure of the parameter space rather than applying uniform averaging.

Several works have developed composition strategies tailored specifically to the low-rank structure of LoRA modules. Asadi et al. (2024) explore LoRA composability for cross-task generalisation, comparing uniform averaging of upstream LoRA parameters with a learned weighted average optimised on a small number of downstream examples. Huang et al. (2024) extend this direction with LoraHub, a framework that automatically assembles LoRA modules trained on diverse tasks for generalisation to unseen tasks using only a handful of examples. Composition is achieved by element-wise weighted summation of LoRA parameters, with weights optimised via gradient-free black-box optimisation and L1 regularisation to prevent degenerate solutions, requiring neither additional model parameters nor gradient computation through the base model. At a finer level of granularity, Zhao et al. (2025) propose LoRA-LEGO, which decomposes LoRA modules into Minimal Semantic Units (MSUs), each corresponding to a single rank component, and recombines them via clustering, assembling a merged LoRA whose rank is determined by the number of clusters.

Rather than merging LoRA parameters into a single module, mixture-of-experts approaches maintain a set of specialised LoRA modules and select or weight them dynamically for each input. Feng et al. (2024) propose MoA (Mixture-of-LoRAs), which trains domain-specific LoRA modules independently and combines them via an explicit routing strategy conditioned on domain labels, preventing cross-domain interference while enabling multi-task deployment within a single model. Dou et al. (2024) propose LoRAMoE with a similar architecture but motivated specifically by the problem of world knowledge forgetting under large-scale instruction tuning: a router network partitions LoRA experts between those that leverage stored world knowledge and those that specialise in downstream task performance, preserving pre-trained knowledge while improving task-specific capabilities.

Taken together, these works reveal a spectrum of composition strategies ranging

from learned attention over frozen adapters, to weight arithmetic, rank-level clustering, and dynamic routing. Most similar to our own work, Asadi et al. (2024) and Huang et al. (2024) combine LoRA modules trained on general language tasks by computing weighted sums of their parameters, with weights either fixed or optimised. However, both approaches require additional learning steps after composition, and weight optimisation becomes inefficient as the number of modules grows with limited supervised data. In Chapter 6, we investigate whether attribute-specialised LoRA modules can be composed for multiple-attribute CTG without any additional training, combining modules directly at inference time.

3.5 Portability of PEFT Modules

PEFT techniques such as adapters, prefix tuning, and LoRA produce structurally modular parameter sets that interact with their host model via dedicated interfaces (Sabry and Belz, 2023). A natural question is whether this structural modularity also implies *functional* modularity, that is, whether the task-specific knowledge encoded in a PEFT module is sufficiently self-contained to be transferred to a different host model without retraining. Such portability would represent an important step towards reusable, plug-and-play neural components, with substantial implications for reducing training costs and increasing the longevity of fine-tuned modules as base models evolve. This section reviews work on PEFT module portability, with particular attention to LoRA-based approaches that directly motivate our portability experiments in Chapter 6.

The foundational empirical study of PEFT module portability is due to Sabry and Belz (2024), who design a large-scale study comprising 1,440 training and testing runs to assess the portability of modules trained by four PEFT techniques across different pre-trained host models, using sentiment analysis as a testbed. Ported modules are compared against equivalent modules trained from scratch and against modules initialised from parameters sampled from the same distribution as the ported module. The results demonstrate that ported modules substantially outper-

form both alternatives, establishing that task-specific knowledge encoded in PEFT modules is highly portable. Performance differences across PEFT techniques and between source and target model pairs are observed, indicating that portability is not uniform but depends on factors such as the structural compatibility of the originating and receiving models. Notably, this study restricts porting to models of the same architectural family (e.g. from Flan-T5 to T5), leaving cross-architecture transfer as an open question.

More recent work has addressed cross-architecture and cross-version LoRA transfer, motivated by the practical challenge that LoRA modules become obsolete when their associated base model is deprecated or updated. Two lines of work have emerged depending on whether any additional training is permitted. Training-free approaches exploit the geometric structure of model weight spaces to transfer LoRA parameters directly. Farhadzadeh et al. (2025) propose Cross-Model Low-Rank Adaptation (LoRA-X), which constrains the adapter to operate within the column-row subspace of the source base model's weights, and transfers it to layers of the target model that exhibit sufficient subspace similarity, measured via a dedicated transferability metric. Xia et al. (2025) propose Cross-LoRA, which performs subspace alignment between source and target models using rank-truncated singular value decomposition and a Frobenius-optimal linear transformation (LoRA-Align), before projecting the aligned source LoRA updates into the target model's parameter space (LoRA-Shift). Both methods are entirely data-free and require no fine-tuning on the target model, and Cross-LoRA demonstrates generalisation across heterogeneous model families such as Qwen, LLaMA, and Gemma.

Where some additional adaptation is acceptable, lightweight retraining approaches offer stronger performance guarantees. Li et al. (2025) propose LoRASuite, which addresses the six principal ways in which LLM updates introduce architectural mismatches (vocabulary size, hidden dimensions, layer depth, attention head count, and attention type) through a modular pipeline: a transfer matrix is computed from the weights of both model versions to handle dimensional mismatches, layer mapping

is performed via Centred Kernel Alignment to maximise representational similarity, and attention heads are matched using the Hungarian method on cosine similarity. A small-scale fine-tuning step follows to correct numerical instabilities introduced by the matrix transformations. Wang et al. (2024b) propose Trans-LoRA, which takes a different approach by using the source LoRA model to generate synthetic training data via the target base model, training a discriminator to filter synthetic samples towards the source training distribution, and then training a new LoRA on the filtered synthetic data. Trans-LoRA supports transfer not only across base model versions but also across PEFT types (e.g. LoRA to DoRA or prompt tuning), making it the most flexible approach reviewed here, at the cost of requiring synthetic data generation.

Taken together, these works establish that LoRA module portability is achievable across a spectrum of scenarios, from same-family model transfers to cross-architecture settings, and from entirely training-free methods to those requiring lightweight adaptation. A common gap, however, is that existing studies evaluate portability on general NLP tasks rather than on attribute-specific CTG modules. In Chapter 6, we investigate whether attribute-specialised LoRA modules trained for CTG retain their control capabilities when ported to a different base model, extending the portability question to the specific setting of this thesis.

Chapter 4

Evaluation Challenges and Framework for Controlled Text Generation

4.1 Background

Evaluation is a central challenge in natural language generation research (Thomson et al., 2024; Schmidtova et al., 2024). Unlike classification tasks where ground truth labels are well defined, the quality of generated text is inherently multidimensional and context-dependent, and there is rarely a single correct output, many different texts may equally well satisfy the requirements of a given generation task. This makes evaluation difficult to conduct reliably and consistently, as any assessment method must somehow account for the legitimate variation in what constitutes a good output. This challenge is compounded in controlled text generation, where evaluation must additionally capture the degree to which the output satisfies the specific control constraints imposed by the task, alongside more general quality criteria such as fluency, coherence, and diversity.

Two distinct but related evaluation challenges motivate the work presented in this chapter. The first concerns the assessment of *semantic consistency* in data-to-

text generation: the degree to which generated text faithfully and completely reflects the information contained in structured inputs. Despite being a fundamental quality criterion for this class of tasks, reliable automatic methods for semantic consistency assessment remain elusive, and the landscape of available metrics is fragmented and poorly understood (Dušek and Kasner, 2020). The second challenge concerns the *comparative evaluation* of controlled text generation systems more broadly: the difficulty of drawing meaningful conclusions about the relative performance of different CTG techniques when existing work evaluates systems on different datasets, using different metrics, and under different experimental conditions.

These two challenges share a common underlying cause: the absence of standardised, principled evaluation frameworks means that reported results are difficult to interpret and compare, and may be systematically misleading. Addressing them requires not only the development of better evaluation methods, but also a more principled approach to how evaluation is designed and conducted. This chapter presents work on both fronts: Section 4.3 addresses the first challenge through a meta-evaluation of semantic consistency assessment methods in data-to-text generation, while Section 4.4 addresses the second through the development and application of a level-playing-field framework for fair comparative evaluation of CTG systems.

4.2 Chapter Overview

This chapter addresses the challenge of evaluation in controlled text generation, focusing on two complementary aspects: the assessment of semantic consistency in data-to-text generation, and the design of a fair comparative evaluation framework for CTG systems. Together, these contributions speak directly to the following research questions:

RQ2 How can we effectively evaluate control in text generation systems?

RQ2.1 What evaluation approaches are appropriate for measuring different

types of control (content, sentiment, topic, keywords)?

RQ2.2 How can we design an evaluation framework that enables fair and reproducible comparison of CTG systems?

Section 4.3 addresses RQ2.1 in the context of content control in data-to-text generation, where the relevant quality criterion is semantic consistency between structured inputs and generated outputs. Through a meta-evaluation of 29 evaluation methods spanning traditional textual similarity metrics, embeddings-based measures, learned regressors, and prompted LLMs, this section establishes which methods best predict human semantic consistency judgements and identifies the current performance ceiling for automatic semantic consistency assessment.

Section 4.4 addresses RQ2.1 and RQ2.2 more broadly, examining how evaluation should be designed to enable fair and informative comparison of CTG systems across a range of control types, including sentiment, topic, keyword, and multiple-attribute control. A level-playing-field (LPF) framework is proposed and applied to the comparative evaluation of 12 CTG systems, demonstrating how principled choices of datasets, evaluation metrics, and experimental protocols affect reported performance and system rankings.

Some of the content reported in this chapter was previously published in the following work, overlapping sections indicated in parentheses:

- Rudali Huidrom, **Michela Lorandi**, Simon Mille, Craig Thomson, and Anya Belz. 2025. **Assessing Semantic Consistency in Data-to-Text Generation: A Meta-Evaluation of Textual, Semantic and Model-Based Metrics.** In Proceedings of the 18th International Natural Language Generation Conference, pages 98–107, Hanoi, Vietnam. Association for Computational Linguistics. (Section 4.3)

- **Michela Lorandi** and Anya Belz. 2026. **A Comparative Study of Controlled Text Generation Systems Using Level-Playing-Field Evaluation Principles.** arXiv preprint arXiv:2605.12395. (Section 4.4)

4.3 Semantic Consistency Assessment in Data-to-Text Generation

The last few years have seen substantial advances in the quality of automatically generated text (Hurst et al., 2024; Grattafiori et al., 2024), in particular with respect to suprasentential fluency and coherence, thanks to pretrained language models with ever larger numbers of parameters trained on ever larger datasets (Hoffmann et al., 2022). However, these advances have come at the price of semantic controllability, with confabulation, replacement, and omission of content all commonly found in state-of-the-art text generation systems (Hao et al., 2025).

In controlled text generation, whether in the form of data-to-text generation or of free text generation with given control attributes, some or all of the input must be matched in specific ways by the output. Ensuring this *semantic consistency* between input and output is a known weakness of otherwise state-of-the-art neural generators, and sufficiently reliable methods are currently lacking both for (i) ensuring semantic consistency as part of the neural generation process, and (ii) evaluating systems in terms of the degree to which they achieve it.

This poses particular problems for tasks like data-to-text generation (González Corbelle et al., 2022), where the information conveyed by the output is intended to be entirely controlled by the information contained in the input. Currently, the only reliable way to perform *semantic consistency assessment* is manual evaluation, with supervision-trained automatic methods correctly predicting up to approximately three quarters of unseen assessments (Dušek and Kasner, 2020; Liu et al., 2023b; Cui et al., 2024). An automatic method that could reliably assess semantic consistency between inputs and outputs directly would be valuable not only for evaluation during and after system development, but also potentially as part of the generation process itself, for instance via reranking of outputs (Harkous et al., 2020) or as a component of a loss function. However, semantic consistency continues to be challenging to assess automatically with a high degree of reliability, and evaluation by

comparison to human-written reference outputs or by human evaluators remains the predominant approach.

This section presents a meta-evaluation of 29 evaluation methods for semantic consistency in data-to-text generation, assessed on two datasets and under two input representation conditions. The main contributions of this work are:

1. A new WebNLG-style data-to-text dataset comprising new triple inputs and corresponding texts for people and city entities sampled from the GREC corpus (Belz et al., 2009), with outputs generated by five different NLG systems.
2. Human semantic consistency annotations for a subset of 100 input-output pairs from the above dataset, following the WebNLG 2017 annotation protocol for the *semantic adequacy* criterion.
3. An empirical meta-evaluation of 29 evaluation methods for semantic consistency on two data-to-text corpora, one established, one newly created, and under two conditions: with input triples textified, and without.
4. A novel LLM-as-judge protocol in which three LLMs rate semantic consistency on a 3-point scale using instructions designed for human annotators, with their scores ensembled into a single metric.
5. An analysis of how model type, input representation, and the underlying definition of semantic consistency affect correlation with human ratings.

4.3.1 Overview and Motivation

Semantic consistency in data-to-text generation refers to the requirement that the content of a generated text faithfully and completely reflects the information contained in the structured input, without adding content absent from the input or omitting content that the input specifies (see also Section 4.3.2 for a formal definition). *Semantic consistency assessment* (SCA) is the task of automatically determining the degree to which a given input-output pair satisfies this requirement.

SCA is considered challenging (Harkous et al., 2020; Liu et al., 2023b; Cui et al., 2024), and reliable automatic methods for it remain elusive.

Existing SCA methods tend not to directly use the structured meaning representations (SMRs) that typically form the input to data-to-text systems. Instead, they first map the SMRs to text, a process known as textification, before applying either a semantic similarity measure (Mille et al., 2023) or performing a natural language inference (NLI) task (Dušek and Kasner, 2020) on the mapped inputs and corresponding textual outputs. These textification-based methods tend not to generalise well beyond the data they were trained on, and the pipelining of multiple processes aggregates errors at each stage.

A further complication is that semantic consistency admits multiple underlying definitions, ranging from textual overlap and semantic similarity to mutual entailment, and different evaluation methods implicitly adopt different definitions. This makes results difficult to compare across methods and contributes to the lack of a clear picture of which approaches are most reliable for the task. Little attention has been paid in existing work to the role these underlying definitions play in shaping evaluation outcomes, which can vary substantially depending on which definition an evaluation method presupposes.

In order to assess how accurately semantic consistency can currently be evaluated, this section presents a systematic meta-evaluation of a broad range of evaluation methods, spanning traditional textual similarity metrics, embeddings-based semantic similarity measures, learned regressors, and prompted LLMs, on two datasets and under two input representation conditions (with and without textification of structured inputs). The aim is to identify which methods best predict human semantic consistency ratings, and to quantify the extent to which metric type, input representation, and the underlying definition of semantic consistency affect correlation with human judgements.

4.3.2 Task Definition

The task addressed in this section is *semantic consistency assessment* (SCA) in data-to-text generation. Given a structured input I, here a set of RDF triples of the form (*subject*, *predicate*, *object*), and a corresponding natural language output text T, the goal is to determine the degree to which T expresses all of the information in I and only that information: that is, without omitting content that I specifies and without adding content that I does not contain.

Semantic consistency is operationalised as a graded notion assessed on a 3-point scale, following the WebNLG 2017 human evaluation protocol (Shimorina et al., 2018): a text may express all, some, or none of the input information correctly. This formulation is more appropriate than binary classification for capturing the range of partial errors, such as the omission of a single triple or the incorrect verbalisation of one attribute value, commonly observed in data-to-text outputs.

The meta-evaluation task is then as follows: given an automatic evaluation method M and a corpus of (input, output) pairs with associated mean human SCA ratings, assess the degree to which the scores produced by M correlate with the human ratings, using Pearson's r as the primary correlation measure.

Each method is evaluated under two input representation conditions. In the *without textification* condition, structured RDF triples are passed to the evaluation method in their raw linearised form. In the *with textification* condition, the triples are first converted to a natural language-like string via a minimal normalisation procedure (Algorithm 3) that replaces special characters with spaces and collapses whitespace. This distinction is particularly relevant for methods designed for text-to-text comparison, which may be sensitive to the surface form of their inputs.

4.3.3 Methodology

This section describes the two datasets used for the meta-evaluation, the algorithm used to construct the newly created dataset, and the 29 evaluation methods assessed. The five NLG systems used to generate outputs for Dataset B are described in

Section 4.3.4.

4.3.3.1 Datasets

Two corpora of RDF-text pairs with human judgements of semantic adequacy are used for the meta-evaluation.

Dataset A (WebNLG 2017). Dataset A contains 100 input–output pairs randomly sampled from the public WebNLG 2017 human-evaluation dataset (Shimorina et al., 2018); the full dataset has 2,230 pairs and the samples come from ten different systems. Each text, in the original human-evaluation, was rated by three annotators on a 3-point scale for semantic adequacy, responding to the question "Does the text correctly represent the input triples?", and we use the mean score for the meta-evaluation.

Dataset B (GREC-derived). Dataset B is a new WebNLG-style benchmark constructed from the People (442) and Cities (243) entities in the GREC 2.0 corpus (Belz et al., 2009). RDF triples are automatically extracted for 100 sampled entities using the procedure described in Algorithms 1 and 2, and verbalisations are generated using five NLG systems previously submitted to WebNLG shared tasks (see Section 4.3.4 for details). We collect three independent human semantic-consistency ratings per input-output pair following the WebNLG 2017 annotation protocol, and ratings are averaged for the meta-evaluation. We have made the dataset publicly available.[1]

Dataset construction. The construction of Dataset B proceeds in two stages, described by Algorithms 1 and 2 respectively. Algorithm 1 builds a property-to-entity mapping by checking, for each entity in the GREC corpus and each ontology property valid for its category, whether at least one corresponding RDF triple exists in DBpedia. This mapping is saved and used as input to Algorithm 2, which extracts,

[1] `https://github.com/mille-s/Build_KGs_entities.git`

Algorithm 1: Property-to-Entity Mapping Construction

Input:
(a) `GREC_NE.json`: list of entities by category
(b) Ontology property definitions: set of valid properties for each category
(c) DBpedia triple data: all RDF triples (subject, property, object)
Output: `dico_entities_for_triple_configuration_GREC_NEs.json`: property-to-entity mapping

```
if dico_entities_for_triple_configuration_GREC_NEs.json does NOT exist then
    for each category in GREC_NE.json do
        for each property valid for the category (from Ontology) do
            for each entity in the category do
                if at least one DBpedia triple exists with entity as subject and
                  property as predicate then
                    Add entity to the property's entity list for that category;
    Save as dico_entities_for_triple_configuration_GREC_NEs.json;
```

filters, and samples triples for each entity. Specifically, for each entity, all available triples are retrieved from DBpedia, filtered to retain only those with valid properties, and randomly sampled down to a target number N of triples per entity (set to $N = 3$ following the WebNLG distribution). The resulting triples are written to an XML file constituting Dataset B.

Algorithm 3 describes the textification procedure applied to the structured triple inputs prior to evaluation in the *with textification* condition. The procedure applies a minimal normalisation that replaces special characters with spaces and collapses consecutive whitespace, converting the raw linearised triple strings into a natural language-like form without any lexical substitution or reordering.

4.3.3.2 Evaluation Methods

A total of 29 evaluation methods are meta-evaluated in terms of their ability to predict human semantic consistency ratings. Many of these methods, including SacreBLEU, ROUGE, METEOR, and the embeddings-based similarity measures, are reference-based metrics originally designed to compare a generated output against a human-written reference text. In this meta-evaluation, we repurpose them in a non-standard way: rather than comparing generated text against a reference

Algorithm 2: Triple Extraction, Filtering, Sampling, and WebNLG-style XML Data Creation

Input:
(a) `GREC_NE.json`
(b) `dico_entities_for_triple_configuration_GREC_NEs.json`
Output: XML file containing the WebNLG-style dataset, grouped by entity

if *`dico_input_contents_DBp_GREC_NEs.pickle` does NOT exist* **then**
  foreach *category, property, entity in property-to-entity mapping* **do**
    if *entity is in target list from `GREC_NE.json`* **then**
      Extract all (subject, property, object) triples for entity from DBpedia;
      Add triples to entity's list in mapping;
  Save as `dico_input_contents_DBp_GREC_NEs.pickle`;
else
  Load `dico_input_contents_DBp_GREC_NEs.pickle`;

Initialise `target_entities` from `GREC_NE.json`;

foreach *entity in `target_entities`* **do**
  Retrieve available triples from pickle;
  Retain only triples with valid properties;
  Optionally remove triples with unsuitable object values;
  Set N per entity following the WebNLG distribution (e.g. $N = 3$);
  if *number of retained triples* $> N$ **then**
    Randomly select N triples;
  else
    Keep all retained triples;

Write grouped triples for each entity to an XML file;

Algorithm 3: Minimal Normalisation of Linearised Triples

Input: List of linearised triple strings L
Output: List of normalised triple strings S

$S \leftarrow$ empty list;
for *each line in* L **do**
  Replace all occurrences of `|`, `_`, and `<br>` with a single space;
  Collapse consecutive whitespace to a single space;
  Trim leading and trailing whitespace;
  Append *line* to S;
return S;

text, we apply them directly to the (textified) source input, treating the input as a proxy for the reference in order to test whether existing metrics, despite not being designed for this purpose, can nonetheless capture semantic consistency between structured inputs and generated outputs. The methods fall into three broad categories depending on the underlying definition of semantic consistency they im-

plicitly adopt: **textual similarity** between inputs and outputs (ROUGE, BLEU, edit-distance, and set-overlap metrics), **semantic similarity** between inputs and outputs (embeddings-based metrics), and **model-based assessment** (learned regressors and LLMs). Both long-established metrics and more recent model-based methods are deliberately included.

Embeddings-based semantic similarity. Seven metrics in this category assess similarity through vector representations of text. BERTScore (Zhang* et al., 2020)[2] operates at the token level, using contextual embeddings to compute cosine similarities between tokens in the two sequences. It produces three scores: Precision (P), defined as the average over each candidate token of its maximum cosine similarity with any reference token; Recall (R), defined analogously from the reference token perspective; and F_1, the harmonic mean of P and R, all in $[0, 1]$. SBERT (Reimers and Gurevych, 2019)[3] operates at the sentence level, encoding each sequence into a dense fixed-length vector via a Siamese BERT network. Four distance or similarity functions are applied to the resulting vector pair: Cosine ($\mathbf{u} \cdot \mathbf{v} / \|\mathbf{u}\| \|\mathbf{v}\|$, in $[-1, 1]$), Dot Product ($\mathbf{u} \cdot \mathbf{v}$, unbounded), Euclidean ($\|\mathbf{u} - \mathbf{v}\|_2 = \sqrt{\sum_i (u_i - v_i)^2}$, in $[0, +\infty)$), and Manhattan ($\|\mathbf{u} - \mathbf{v}\|_1 = \sum_i |u_i - v_i|$, in $[0, +\infty)$).

Overlap-based textual similarity. Eleven metrics in this category quantify similarity through surface-form overlap between sequences. SacreBLEU (Post, 2018)[4] provides a standardised implementation of BLEU that controls for tokenisation and formatting choices, computing n-gram precision with a brevity penalty, in $[0, 1]$. ROUGE (Lin, 2004)[5] captures how much of the reference content appears in the candidate, with three variants differing in the unit of comparison: ROUGE-1 uses unigrams, ROUGE-2 uses bigrams, and ROUGE-L uses the longest common subsequence. Each variant is computed as Precision (matched units divided by candidate length), Recall (matched units divided by reference length), and F_1 (harmonic

[2] https://github.com/Tiiiger/bert_score
[3] https://www.sbert.net
[4] https://github.com/mjpost/sacrebleu
[5] https://pypi.org/project/rouge/

mean), yielding nine scores in total, all in $[0, 1]$. METEOR (Banerjee and Lavie, 2005)[6] extends unigram matching to cover stemmed and synonym-based alignments, combines precision and recall, and adds a fragmentation penalty that penalises non-contiguous matches, in $[0, 1]$.

Edit-distance and set-based textual similarity. Four metrics in this category measure similarity through character-level edit operations or token-set overlap. Levenshtein Distance (Levenshtein et al., 1966)[7] counts the minimum number of single-character insertions, deletions, and substitutions needed to convert one string into the other, in $[0, +\infty)$, and can be normalised to $[0, 1]$. Jaccard Similarity (Jaccard, 1901)[8] treats each sequence as a set of tokens and computes the ratio of shared to total tokens, $J(A, B) = |A \cap B|/|A \cup B|$, in $[0, 1]$; Jaccard Distance (Rogers and Tanimoto, 1960) is the complementary dissimilarity $1 - J(A, B)$, also in $[0, 1]$. The Dice Coefficient (Dice, 1945)[9] is a related set-overlap measure, $D(A, B) = 2|A \cap B|/(|A| + |B|)$, in $[0, 1]$, which gives proportionally greater weight to shared elements than Jaccard.

Assessment by learned regressors. Three metrics in this category use trained models to produce continuous quality estimates. BLEURT (Sellam et al., 2020)[10] is a regression model pre-trained on large amounts of text and then fine-tuned on human quality annotations, enabling it to predict scores that correlate well with human judgement. InferSent (Conneau et al., 2017)[11] derives sentence representations from a model trained on natural language inference data, with similarity between representations computed using cosine or other distance measures. The Universal Sentence Encoder (USE) (Cer et al., 2018)[12] produces fixed-length sentence embeddings from a model trained on a diverse range of tasks, typically compared via cosine

[6] https://huggingface.co/spaces/evaluate-metric/meteor/tree/main
[7] https://pypi.org/project/Levenshtein/
[8] https://www.geeksforgeeks.org/jaccard-similarity/
[9] https://en.wikipedia.org/wiki/Dice-Sorensen_coefficient
[10] https://github.com/google-research/bleurt
[11] https://github.com/facebookresearch/InferSent
[12] https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder

similarity.

Assessment by LLMs (LLM-as-a-judge). Three large language models are used as prompted judges: Command-r-plus,[13] Llama3-70B (AI@Meta, 2024), and Mistral-7B (Jiang et al., 2023). Each model is given the input-output pair and asked to assign a semantic consistency rating on a 3-point scale, using the same instructions as those provided to human annotators. To reduce sensitivity to random variation in generation, each rating is averaged over three random seeds (42, 1234, 1738). A fourth method, the ensemble, is obtained by averaging the scores of all three models, on the basis that aggregating across judges with different training histories reduces the influence of any individual model's biases.

All 29 methods are applied to the input-output pairs in both datasets under two conditions: with the structured RDF triple inputs converted to natural language-like strings using the textification procedure in Algorithm 3, and with the raw linearised triples passed directly to each method without any conversion.

4.3.4 Experimental Setup

The aim is to determine which of the 29 evaluation methods best predicts human semantic consistency judgements. For each method and each input condition, Pearson's r is computed between the method's scores and the mean human ratings, and methods are ranked by correlation accordingly.

The LLM-as-judge methods are evaluated by prompting Command-r-plus, Llama3-70B, and Mistral-7B to rate semantic consistency on a 3-point scale using instructions designed for human annotators. To reduce the influence of random variation, scores are averaged over three random seeds (42, 1234, 1738). An ensemble score is additionally computed by averaging across all three models.

Systems. Dataset B is composed of 100 input-output pairs, with outputs generated by five NLG systems previously submitted to WebNLG shared tasks, selected

[13] `https://huggingface.co/CohereLabs/c4ai-command-r-plus`

to represent a diverse range of generation approaches spanning from rule-based to neural, with each system contributing 20 samples.

FORGe (Mille et al., 2019) is a portable grammar-based system that maps predicate-argument structures to sentences by applying a series of rule-based graph transducers. CycleGT (Guo et al., 2020) is a weakly supervised framework that iteratively bootstraps generation and semantic parsing models by mapping between unaligned text and RDF data to enable joint model improvement. DCU-NLG-Small (Mille et al., 2024a) combines the FORGe rule-based system with a reduced-size language model (T5): the rule-based component converts input triples into semantically correct English text, which is then paraphrased by the language model to improve fluency. DCU-NLG-PBN (Lorandi and Belz, 2024) is a Mistral 7B Instruct model fine-tuned with LoRA to improve task performance while maintaining computational efficiency. DCU-ADAPT-modPB (Osuji et al., 2024) uses a prompt-based approach in which text is generated directly from input triples using Mistral 7B Instruct with five in-context examples.

4.3.5 Results

Table 4.1 reports Pearson's r between each evaluation method's scores and mean human ratings on both datasets, under both input conditions. The results reveal systematic differences in metric performance across datasets and across the two textification conditions.

Textification has a broadly positive effect on automatic metric correlations. On Dataset A, it increases Pearson's r for 20 out of 25 automatic metrics, with an average increase of approximately 0.07 across those 20 metrics; the five exceptions, BLEURT, ROUGE-L Recall, ROUGE-1 Recall, SacreBLEU, and Jaccard Distance, are unaffected or slightly reduced. On Dataset B the positive effect is present but smaller: 16 out of 25 metrics show an increase, averaging approximately 0.04 across those 16; the exceptions here are a different set, comprising the four SBERT metrics, ROUGE-2 Precision, SacreBLEU, Levenshtein Distance, and Jaccard Distance. The

Type of Metrics	Metrics	Dataset A (WebNLG 2017)		Dataset B (new WebNLG-style dataset)	
		r (-Textification)	r (+Textification)	r (-Textification)	r (+Textification)
Automatic Metrics	SBERT (Euclidean)	0.507	**0.582**	**0.655**	**0.655**
	SBERT (Manhattan)	0.502	**0.582**	**0.662**	**0.655**
	SBERT (Cosine)	**0.518**	**0.573**	0.654	0.638
	SBERT (Dot Product)	**0.518**	0.573	0.654	0.638
	BLEURT	**0.530**	0.501	**0.713**	**0.733**
	Universal Sentence Encoder	0.248	0.500	0.271	0.402
	BERTScore R	0.417	0.447	0.431	0.609
	BERTScore F_1	0.421	0.441	0.572	**0.670**
	BERTScore P	0.424	0.433	0.636	0.647
	InferSent	0.372	0.379	-0.148	-0.022
	METEOR	0.206	0.379	0.296	0.306
	ROUGE-L F_1	0.305	0.338	0.223	0.356
	ROUGE-L Precision	0.287	0.337	0.248	0.304
	ROUGE-1 Precision	0.292	0.330	0.268	0.305
	ROUGE-1 F_1	0.309	0.329	0.246	0.387
	ROUGE-L Recall	0.328	0.324	0.158	0.281
	Dice Coefficient	0.290	0.321	-0.021	0.074
	Jaccard Similarity	0.290	0.320	-0.010	0.078
	ROUGE-1 Recall	0.329	0.313	0.182	0.323
	ROUGE-2 Precision	0.203	0.264	0.111	0.078
	ROUGE-2 F_1	0.202	0.263	0.097	0.127
	ROUGE-2 Recall	0.198	0.257	0.051	0.082
	SacreBLEU	0.228	0.198	0.400	0.127
	Levenshtein Distance	-0.044	-0.062	-0.384	-0.400
	Jaccard Distance	-0.290	-0.320	0.010	-0.078
LLM-as-Judge	Command-r-plus + LLaMa + Mistral	**0.635**	**0.367**	**0.716**	**0.720**
	Mistral	**0.531**	**0.362**	0.567	0.488
	LLaMa	**0.569**	**0.284**	**0.644**	**0.689**
	Command-r-plus	0.518	0.214	**0.727**	**0.729**

Table 4.1: Pearson's r for Datasets A and B, sorted (within Automatic Metrics and LLM-as-Judge) by Dataset A's r (+Textification). Top three Pearson's r in each group bolded.

effect of textification on the LLM judges diverges from this pattern: on Dataset B it provides a modest benefit for most judges, with the exception of Mistral, but on Dataset A it causes LLM correlations to collapse across the board, a pattern discussed further in Section 4.3.6.

Among the automatic metrics, the SBERT-based similarity metrics with textification attain the four highest correlations on Dataset A ($r = 0.582$), substantially outperforming BLEURT ($r = 0.530$). The picture is reversed on Dataset B, where BLEURT achieves the highest correlation ($r = 0.733$), followed by BERTScore F_1 ($r = 0.670$) and the SBERT variants ($r = 0.655$). Across both datasets, traditional textual similarity metrics, ROUGE variants, SacreBLEU, METEOR, and the edit- and set-based measures, perform considerably worse, and on Dataset B the metrics from InferSent downwards in the table fail almost entirely, with several producing near-zero or negative correlations.

Among the LLM judges, the three-model ensemble (Command-r-plus + LLaMA

+ Mistral) achieves the highest correlation on Dataset A without textification ($r = 0.635$). On Dataset B, Command-r-plus attains the highest correlation ($r = 0.727$), marginally surpassing both the ensemble ($r = 0.716$) and LLaMA ($r = 0.644$). On Dataset A, the LLM judges without textification and the model-based automatic metrics with textification perform at broadly comparable levels overall. On Dataset B, BLEURT, the LLM ensemble, and Command-r-plus stand out by a considerable margin, and their performance is largely unaffected by textification.

4.3.6 Discussion and Limitations

The most consistent pattern in the results is the uniformly poor performance of traditional textual similarity metrics on the semantic consistency assessment task. InferSent and the Universal Sentence Encoder, the two neural but pre-Transformer metrics, similarly fail to produce meaningful correlations and can be considered unsuitable for this task. While these findings are perhaps unsurprising given that such metrics were designed for text-to-text comparison rather than structured input-output matching, they confirm empirically that their continued use in the data-to-text evaluation community is not warranted for semantic consistency assessment, and establish a clear baseline against which progress with newer methods can be measured.

A second consistent pattern concerns the difference in LLM performance between the two datasets. While traditional textual similarity metrics perform similarly on Dataset A and Dataset B, every LLM-based evaluation method sees a notable increase in correlation from Dataset A to Dataset B. The reason for this is not entirely clear. One plausible explanation is that the outputs in Dataset B are generally of higher quality than those in Dataset A, displaying smaller and less frequent errors, which may make the semantic consistency signal easier for LLMs to detect reliably.

The collapse of LLM-judge correlations under textification on Dataset A, while textification improves almost all other metrics on that dataset, is also noteworthy. Dataset A is derived from the public WebNLG 2017 human evaluation data,

which was almost certainly ingested as part of the training data of the LLMs tested. Textification, by converting the structured triple inputs into a surface form that resembles natural language, has the net effect of obscuring the structural correspondence between input and output, thereby making it harder for the LLM to determine whether the two belong together and reducing the reliability of its semantic consistency judgement.

Several limitations of this study should be noted. The set of LLMs and automatic metrics evaluated, while broad, is not exhaustive, and the findings cannot be straightforwardly generalised beyond the specific models and metrics included. The experiments are furthermore restricted to WebNLG-style English datasets, and whether the observed patterns hold across different domains, input structures, or languages remains an open question. Finally, Dataset B, while constructed to be unseen by the evaluated methods, is relatively small (100 samples), and larger-scale studies would be needed to draw more robust conclusions about the relative performance of the evaluated methods.

4.4 Fair Comparison of Controlled Text Generation Techniques

4.4.1 Overview and Motivation

The controlled text generation literature has grown rapidly in recent years, as demonstrated in Section 2, with a wide range of techniques proposed to steer properties of generated text such as sentiment (Kim et al., 2023), topic (Gu et al., 2022a), lexical content (Carlsson et al., 2022), or toxicity (Gu et al., 2022a), among many others. Despite this proliferation, no clear picture has emerged as to which techniques perform better overall, or even for specific tasks. A key reason for this is that individual papers tend to evaluate their systems using different datasets and different evaluation methods, making direct comparison across works effectively impossible.

To see this problem concretely, consider the task of controlling the binary sentiment of generated text. Krause et al. (2021), Yu et al. (2021), and Zhang and Song (2022) all tackle this task and all use a sentiment classifier to measure the degree of sentiment control achieved. However, the evaluation setups differ substantially: Krause et al. (2021) evaluate on the Bookcorpus test set using RoBERTa fine-tuned on Stanford Sentiment Treebank 2 (SST-2) (Socher et al., 2013) as the classifier; Yu et al. (2021) evaluate on PPLM Prompts using BERT trained on IMDb Movie reviews; and Zhang and Song (2022) evaluate on OpenWebText using DistilBERT fine-tuned on SST-2. Given these discrepancies, it is simply not possible to draw any meaningful conclusion about the relative effectiveness of the three approaches. Compounding this issue, even when a single classifier is used as the sole evaluation metric within a paper, the results may still be misleading, as different classifiers applied to the same test set can yield substantially different scores.

This section addresses these shortcomings by comparing a range of CTG systems under a *level-playing-field* framework designed for fair comparative evaluation. The framework provides principled guidance on the selection of datasets, evaluation metrics, and systems, as well as on how to obtain outputs from each system, with the aim of ensuring that comparisons are equitable across methods and that performance assessments are more faithful.

4.4.2 Comparative Study Design

This section describes the design of a comparative evaluation of CTG systems conducted under *level-playing-field* (LPF) principles, with the aim of ensuring that comparisons are fair and that results are genuinely informative. The principles address potential biases arising at several stages of the evaluation process, from the initial selection of systems and metrics through to the generation and post-processing of outputs.

The first concern is the **systematic selection of CTG methods**. An unguided choice of systems to compare can inadvertently favour certain approaches

or leave important classes of technique unrepresented. The second concern relates to the **choice of evaluation metrics**. As discussed in Section 4.4.1, the metric used to assess control can substantially influence the apparent performance of a system, potentially favouring one approach over another. Third, the **selection of test datasets** is critical: evaluations based on a narrow or system-aligned set of prompts risk producing results that do not generalise and that may unfairly advantage particular systems. Finally, **output generation and post-processing** must be handled consistently across all systems, so that any observed differences in performance reflect genuine differences in control effectiveness rather than artefacts of inconsistent protocols.

Together, these four principles ensure that systems are evaluated on an equal basis, yielding a truer and more informative comparison of CTG techniques. The following subsections describe how each principle was implemented in the present study (Sections 4.4.2.1–4.4.2.4).

4.4.2.1 Selection of CTG techniques

A critical concern in comparative evaluation is the introduction of bias during the selection of systems to be included in the study. Such bias can arise when the choice of systems is driven by subjective preferences, inconsistent inclusion criteria, or inadequate representation of the diversity of approaches present in the literature. To prevent this, the selection of CTG systems for this study follows a systematic, multi-step process, illustrated in Figure 4.1, designed to be transparent, replicable, and guided by objective criteria throughout.

Comprehensive search. The literature search follows the same methodology adopted in Chapter 2, with the pool of papers considered in this study forming a subset of the broader collection assembled there. Specifically, a search of the ACL Anthology was conducted covering papers published up to August 2023, using the keywords "controllable text generation", "controlled text generation", and "controlling text generation" to identify candidate CTG techniques, whereas the

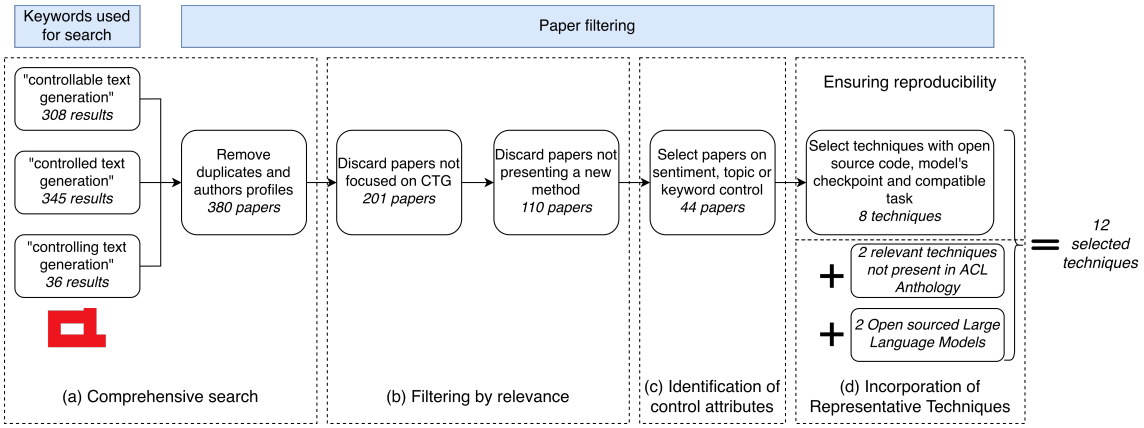


Figure 4.1: CTG techniques selection process from ACL Anthology to which we added 2 open-source LLMs and 2 relevant techniques not in the ACL Anthology.

categorisation scheme in Chapter 2 was subsequently extended to include papers published through to the end of 2025. The full selection process is summarised in Figure 4.1.

Filtering by relevance. Papers not directly addressing CTG or lacking sufficient implementation details were excluded (Figure 4.1b). After removing duplicates, author profile pages, and non-paper resources, 380 papers remained. A further filtering step removed papers that merely referenced CTG methods without directly addressing them, leaving 201 papers. Of these, works that did not present a new model or control method, such as papers introducing new datasets only, were excluded, narrowing the pool to 110 papers.

Identification of control attributes. The most frequently targeted control attributes among the remaining 110 papers were identified, in order to enable the selection of systems that are genuinely comparable within the LPF evaluation (Figure 4.1c). Sentiment, topic, and keyword control emerged as the most prevalent attributes, making them the most suitable focus for this investigation. Accordingly, 44 papers implementing techniques for control over (i) **binary sentiment**, (ii) **topic**, and (iii) **included keywords** in the generated output were selected for further consideration (Section 4.4.3).

Filtering by code and model availability. To ensure consistency and reproducibility, the selection was further refined to include only systems for which open-source code and model checkpoints were publicly available, yielding 10 systems. For topic control specifically, two systems were discarded because their set of topics did not map directly onto those used in the remaining eight papers, which would have compromised comparability.

CTG Technique	Model	Task	C	Attr. S	Attr. T	Attr. K	C Intr	Evaluation Datasets
Complete Training								
CTRL (Keskar et al., 2019)	Transf	FT	Si	-	✓	-		-
Model Fine-Tuning								
C BART (He, 2021)	BART	FT	Si	-	-	✓		One-Billion-Word dataset (Chelba et al., 2013), Yelp dataset (Zhang et al., 2015)
Multi CTG (Gu et al., 2022b)	BERT	FT	M	✓	-	✓	-	PPLM Prompts (Dathathri et al., 2019)
Prior CTG (Gu et al., 2023)	GPT-2 M	FT	M	✓	✓	-		PPLM Prompts
Modification of Token Distribution								
CAT-PAW (Gu et al., 2022a)	GPT-2 M	FT	Si	✓	✓	-		PPLM Prompts
PPLM (Dathathri et al., 2019)	GPT-2 M	FT	M	✓	✓	-		PPLM Prompts
Prompting								
Falcon 40B Instruct (Almazrouei et al., 2023)	Falcon	FT	M	✓	✓	✓	I Add	-
LLaMa2 70B chat (Touvron et al., 2023)	LLaMa2	FT	M	✓	✓	✓	I Add	-
Hybrid								
GeDi (Krause et al., 2021)	GPT-2 XL	FT	Si	✓	✓	-		Bookcorpus (S) (Zhu et al., 2015), AGnews (T) (Zhang et al., 2015)
DisCup (Zhang and Song, 2022)	GPT-2 L	FT	Si	✓	-	-		OpenWebText corpus
DExperts (Liu et al., 2021)	GPT-2 L	FT	Si	✓	-	-		OpenWebText corpus
BOLT (Liu et al., 2023a)	GPT-2 L	FT	Si	✓	-	✓		PPLM Prompts (S, K), CommonGen (K) (Lin et al., 2020)

Table 4.2: Overview of the selected CTG techniques. FT=Free Text Generation, C=Control, Si=Single-attribute control, M=Multiple-attribute control, S=Sentiment, T=Topic, K=Keywords, C Intr=Control Integration, I Add=Input Addition, S Add=System Addition, M Inj=Model Injection.

Incorporation of representative techniques. Two LLM-prompting baselines were added to the final set of systems, namely Falcon 40B Instruct (Almazrouei et al., 2023) and LLaMA-2 70B Chat (Touvron et al., 2023). Although these models were not designed explicitly for CTG and their technical reports do not appear in the ACL Anthology, they were included to provide a point of comparison between

general-purpose instruction-tuned LLMs and models purpose-built for controlled generation. Two additional papers meeting the selection criteria but sourced from outside the ACL Anthology were also incorporated, as both systems are widely referenced and used as baselines across the majority of the selected papers.

This process results in a final set of 12 CTG systems (Figure 4.1d) that together constitute a diverse and representative sample of the state of the art, while minimising the risk of selection bias. Table 4.2 provides an overview of all 12 selected techniques, indicating for each the type of pretrained language model incorporated, the generation task, the control attributes targeted, and the datasets used for evaluation in the original papers. In some cases, no specific evaluation dataset is identified in the original work: this applies to the general-purpose LLMs, which were not originally developed for CTG tasks, and to one system whose paper does not disclose the evaluation dataset used. For the purposes of this study, all systems were evaluated on a common set of datasets.

4.4.2.2 Evaluation methods

A further source of bias in comparative evaluation lies in the choice of evaluation metrics and, crucially, in how those metrics are implemented. Different sets of metrics will almost certainly produce different system rankings, and even the same metric implemented differently can yield substantially different results. A particularly clear illustration of this is the use of sentiment classifiers to assess sentiment control: as discussed in Section 4.4.1, different classifiers applied to the same system outputs can produce markedly different scores and, consequently, different rankings across systems. When a classifier is moreover trained on a single dataset, this bias is further amplified. This is the case, for example, for the classifiers used by Krause et al. (2021), Yu et al. (2021), and Zhang and Song (2022), which are trained on SST-2, IMDb Movie reviews, and SST-5 respectively, each a single dataset with its own distributional characteristics. These issues can lead to misleading conclusions about system performance, highlighting the need for a standardised and principled

evaluation framework.

The LPF approach adopted here combines standardisation of metric implementation with diversity in the selection of metrics. Standardisation ensures that arbitrary differences in what should be identical metrics do not distort results, while diversity provides a more balanced picture of quality across multiple relevant criteria. Concretely, output preprocessing (Section 4.4.2.4 Obtaining system outputs), evaluation metrics, and datasets (Section 4.4.2.3 Datasets) are standardised so that all systems are assessed under identical conditions. At the same time, potential biases introduced by the way evaluation metrics are implemented or by how the underlying models were trained are explicitly addressed: rather than relying on a single classifier or model, multiple classifiers, models, or approaches are used to compute each metric, thereby reducing the risk of bias arising from over-specialisation or training on a specific dataset.

The resulting evaluation framework is multi-faceted and designed to provide a comprehensive assessment of system performance across three key dimensions, *Diversity*, *Fluency*, and *Control Effectiveness* (CE), each described in turn below.

General evaluation aspects. All sets of system outputs are evaluated using (i) Distinct-n (Li et al., 2015) ($n = 1, 2, 3$), which computes the proportion of unique n-grams in the generated texts, and (ii) Syntactic Log-Odds Ratio (SLOR) (Kann et al., 2018), which measures the log-probability of a sentence normalised by unigram log-probability and length (see Section 4.4.5.3 for further details). System-level Distinct-n is calculated as the average of control-level Distinct-n scores, while system-level SLOR is calculated as the average score obtained using GPT-2 XL (Radford et al., 2019) and BLOOM 1B7 (Scao et al., 2022).

Evaluation of Control Effectiveness (CE). CE measures the percentage of generated texts identified as displaying the desired control attribute according to a given measure. To avoid the biases that arise from relying on a single classifier, three different classifiers are used and their scores are averaged to produce a final

overall CE score; Section 4.4.5.2 provides a comprehensive analysis of average and majority-vote variants of CE assessment. Classifiers are selected according to two main criteria. First, preference is given to classifiers already used in the selected CTG techniques themselves: since these systems were originally evaluated using those classifiers, including them in the LPF evaluation allows a direct and meaningful comparison between the LPF results and the originally reported results, as discussed further in Section 4.4.6.6. Second, additional classifiers are identified by searching the HuggingFace ecosystem using the *Text Classification* filter, sorted by number of downloads, to complement the technique-specific classifiers with widely adopted alternatives. Where possible, classifiers based on different model architectures are chosen to avoid architecture-specific bias. No retraining or fine-tuning of the selected classifiers is performed, in order to avoid introducing new biases into CE evaluation.

Sentiment control evaluation. Three sentiment classifiers are applied to the system outputs, and the proportion of outputs for which the classifier prediction matches the intended sentiment is computed for each classifier; these proportions are then averaged to yield an overall Sentiment CE score. The three classifiers used are DistilBERT[14] and T5[15] fine-tuned on SST-2, and DeBERTa fine-tuned on Yelp (Gu et al., 2023).

Topic control evaluation. Topic CE is computed analogously using three topic classifiers: DistilBERT,[16] BERT,[17] and DeBERTa (Gu et al., 2023), all fine-tuned on AG-News. The DeBERTa model is structured as a set of binary classifiers, one per topic, and appears to have been used in prior work (Gu et al., 2023) by selecting the topic in advance and applying only the corresponding classifier with a threshold.[18] Since the target topic is not known in advance in the present evaluation setting, a different approach is adopted: the probability assigned by each binary classifier to its respective topic is computed, and the topic with the highest probability is

[14] https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english

[15] https://huggingface.co/michelecafagna26/t5-base-finetuned-sst2-sentiment

[16] https://huggingface.co/textattack/distilbert-base-uncased-ag-news

[17] https://huggingface.co/fabriceyhc/bert-base-uncased-ag_news

[18] https://github.com/HappyGu0524/MultiControl/blob/main/classify/eval.py

selected as the predicted class.

Keyword control evaluation. Keyword CE is computed as the average of four Word Inclusion Coverage scores, combining two exact-matching approaches with two lemma-based approaches. Following Liu et al. (2023a), Exact Match considers direct string matches between the desired keywords and the words in the generated text, and is calculated in two variants: *Any* (at least one keyword is present) and *All* (all keywords are present). Lemmatised Match, derived from Non-Residual Prompting (Carlsson et al., 2022) using the CommonGen evaluation code (Lin et al., 2020), considers matches between lemmatised forms of the keywords and the generated text, again in two variants: *Cov*, computed using spaCy, and *ExtCov*, which extends the set of possible lemmas using the Lemmingflect library.

Multiple-attribute control evaluation. In prior work, multiple-attribute CE is typically computed as the average of the individual single-attribute CE scores. However, this approach does not constitute a true assessment of multiple-attribute control, as it does not require that all control attributes be satisfied *simultaneously*. To address this, multiple-attribute CE is here calculated as the proportion of outputs in which all desired attributes are correct at the same time, that is, both sentiment and topic must be correct for an output to be counted as a success. Single-attribute classifiers are used to obtain a prediction for each attribute via majority vote across the three classifiers, and multiple-attribute CE is then computed as the proportion of outputs satisfying all desired attributes simultaneously. For transparency, both the average CE (*Avg* in tables) and the proportion of outputs satisfying all attributes at once (*Both*) are reported, to make explicit how much results can vary depending on the aggregation strategy adopted.

4.4.2.3 Datasets

To mitigate dataset bias, diversity is prioritised in dataset selection. By drawing on a varied set of datasets that reflect different text generation scenarios, the aim is to reduce the influence of any single dataset on the overall evaluation and to ensure a

more balanced assessment across systems.

Four datasets are used to evaluate the selected CTG techniques. Two are commonly used free-text generation datasets: the prompts from the evaluation of PPLM (Dathathri et al., 2019), referred to hereafter as PPLM Prompts, and the OpenWebText neutral sentiment prompts (Gokaslan and Cohen, 2019). The remaining two are drawn from the Story Generation domain: the Cloze Winter 2018 test set (Sharma et al., 2018) and the STS benchmark test set (Cer et al., 2017). For the latter, only the *main captions* subset is used, as these are sufficiently general to serve as prompts for texts targeting all of the control attributes considered in this study. For each dataset item, one text is generated per control attribute value, for example, one positive and one negative text for sentiment control, as described in the following section.

Table 4.3 reports the number of samples contained in each dataset.

Dataset	**# Samples**
PPLM Prompts	35
OWT neutral prompts	5000
Cloze 2018 test	1571
STS benchmark test	625

Table 4.3: Number of samples for each dataset used for the execution of the experiments.

4.4.2.4 Obtaining system outputs

Consistency in the process of running code, generating outputs, and post-processing results is essential to ensure that all techniques are evaluated under genuinely comparable conditions. Each of the 12 selected CTG techniques is executed using the same datasets and random seeds to guarantee comparability and reproducibility. Each technique's codebase is treated as a black box, with no modifications to the internal generation logic. Rather than relying on scores reported in the source papers, all 12 techniques are re-executed from their original codebases on all four datasets used in this study, a substantial undertaking given the differing software dependen-

cies, hardware requirements, and configuration conventions of each, and one that ensures every result reported below reflects a genuine re-execution under controlled conditions rather than a figure obtained under different data, hyperparameters, or evaluation protocols.

Across all techniques, the underlying generation task is text continuation: each system is given the text of a dataset sample as input and is required to generate a continuation that completes it, with control exercised over properties of the completion, such as sentiment, topic, or keyword content, rather than through arbitrary rewriting or infilling of the input text. As in the attribute-specialised modules trained in Chapter 6, several techniques additionally prepend a textual prefix to the input to signal the desired control attribute value, for example a control code or an encoded attribute token; unlike in Chapter 6, however, the specific form this prefix takes, if used at all, is dictated by each technique's own published specification rather than by a setup we design ourselves.

Input formatting necessarily varies across techniques, as different systems impose different requirements on how the input text and control attribute values are provided. To handle this while maintaining fairness, the input text is minimally adapted to respect the formatting requirements of each technique, following the specifications in the respective reference papers. Some techniques require only a raw input text, while others require the input text together with an explicit control attribute value such as a target sentiment, a set of keywords, or a topic label. In prompt-based systems, both the input text and control attribute values are embedded directly within a natural language prompt, whereas other techniques use structured input fields or special tokens. Table 4.4 provides a detailed overview of the input format used for each CTG technique.

The hyperparameters and library versions specified in the original papers are used throughout; full details are provided in Appendix B.2.

As described before, four control attributes are considered: Sentiment, Topic, Keywords, and Multiple-attribute. For each sample in each dataset, one text is

Technique	Input format
CTRL	{control_attribute_value} {text}
C BART	{text} {keywords separated by white space}
Multi CTG	{text}
Prior CTG	{text}
CAT-PAW	50256 {encoded text}
PPLM	{control_attribute_value}:{text}
Falcon 40B Instruct	{prompt} containing {text} and {control_attribute_value}
LLaMa2 70B chat	{prompt} containing {text} and {control_attribute_value}
GeDi	{text}
DisCup	{text}
DExperts	{text}
BOLT	{text}

Table 4.4: Input format for each CTG technique.

generated per control attribute value associated with the specified attribute. The control values used for each attribute are listed in Table 4.5; keyword sets follow those used by Liu et al. (2023a).

Attribute	Control Values
Sentiment	Positive, Negative
Topic	World, Sports, Business, Science/Technology
Keywords	[router, Linux, keyboard, server], [plea, subpoena, transcript, bankrupt], [torpedo, headquarters, infantry, battlefield], [court, culture, communism, capitalism], [Bible, church, priest, saint], [microscope, mass, mineral, scientist], [meteor, planet, satellite, astronaut]
Multiple	[Positive World], [Positive Sports], [Positive Business], [Positive Science/Technology], [Negative World], [Negative Sports], [Negative Business], [Negative Science/Technology]

Table 4.5: Control attributes and respective values used in the experiments.

A further challenge arises in topic control, where different systems use different sets of topic labels, making direct comparison difficult. To address this, the topic values are standardised by identifying a set of topics shared across the majority of the selected techniques, and a mapping is then constructed from each label in the standardised set to the corresponding label used by each individual system (Table 4.6). While this ensures that all systems are evaluated on a consistent set of

control values, the mapping introduces an important caveat: topic control results should be interpreted with care, as the mapping may disadvantage systems whose original topic labels align less closely with the standardised set. For instance, as shown in Table 4.6, some systems map a single standardised topic to multiple original labels (e.g. Business → Finance, Legal for CTRL), or have no available mapping for certain topics entirely (indicated by —), meaning those systems are evaluated on a reduced or differently structured set of control values. This is an inherent limitation of cross-system topic comparison and should be taken into account when interpreting the topic control results in Section 4.4.6.2.

Finally, all system outputs are post-processed to extract the generated text, discarding non-text content such as control codes included in the input. The post-processing steps applied to each technique's outputs are detailed in Table 4.7.

4.4.2.5 Summary of Study Design Principles

The LPF framework introduced in this section has four core principles, each targeting a specific source of bias that undermines fair comparative evaluation of CTG systems:

Systematic system selection (Section 4.4.2.1): systems are selected through a transparent, replicable process with objective inclusion criteria, ensuring broad and representative coverage of the CTG landscape rather than a convenience sample.

Standardised and diverse evaluation metrics (Section 4.4.2.2): rather than relying on a single classifier or metric, multiple classifiers and metrics are used to assess each quality dimension, reducing the risk that results are driven by the idiosyncrasies of any one evaluation method.

Diverse and balanced datasets (Section 4.4.2.3): multiple datasets are used to reduce the influence of dataset-specific biases and improve the generalisability of the findings beyond any single evaluation context.

CTG Technique	Topic Mapping
CTRL	Sports → Fitness; Business → Finance, Legal; Science/Technology → Computing, Science, Technology; World → —
Prior CTG	Sports → Sports; Business → Business; Science/Technology → Science/Technology; World → World
CAT-PAW	Sports → —; Business → —; Science/Technology → Science, Computers, Space; World → —
PPLM	Sports → —; Business → —; Science/Technology → Science, Computers, Space; World → —
Falcon 40B Instruct	Sports → Sports; Business → Business; Science/Technology → Science/Technology; World → World
LLaMa2 70B chat	Sports → Sports; Business → Business; Science/Technology → Science/Technology; World → World
GeDi	Sports → Sports; Business → Business; Science/Technology → Science/Technology; World → World

Table 4.6: Mapping to translate the used topic values into topic values available in each CTG technique. Our Topic value → Technique Topic value. A dash (—) indicates that the technique does not support the corresponding topic and was therefore not evaluated on that control value.

Standardised output generation and post-processing (Section 4.4.2.4): all systems are run under identical conditions using the same random seeds, hyperparameters, and post-processing protocols, ensuring that observed differences in performance reflect genuine differences in control effectiveness.

Two key take-home messages follow from this design, which will be illustrated empirically in the sections that follow. First, the choice of evaluation metric mat-

CTG Technique	Postprocessing
CTRL	remove initial control attribute value
C BART	use text as it is
Multi CTG	use text as it is
Prior CTG	use text as it is
CAT-PAW	remove initial |<endoftext>|token
PPLM	remove initial |<endoftext>|token and control attribute value
Falcon 40B Instruct	remove input prompt and use text between first occurrence of "Falcon:" and "User:"
LLaMa2 70B chat	remove input prompt
GeDi	keep text up to and excluding |<endoftext>|token
DisCup	use text as it is
DExperts	use text as it is
BOLT	use text as it is

Table 4.7: Postprocessing for each CTG technique.

ters: as will be shown in Section 4.4.5, different classifiers can produce substantially different CE scores and system rankings for the same outputs, and perplexity is an unreliable fluency measure for LLM-generated text. Second, the choice of dataset matters: as will be shown in Section 4.4.6.6, LPF scores can differ substantially from those reported in original papers, with some systems appearing considerably stronger or weaker depending on which prompts and classifiers are used.

4.4.3 Evaluated Controlled Text Generation Techniques

This section characterises how control is implemented in each of the 12 techniques selected for this study, drawing on the categorisation scheme proposed in Chapter 2. Figure 4.2 provides a diagrammatic overview of where in the model architecture control is injected at inference time for each technique (control integration category), and Table 4.2 summarises the control implementation category assigned to each system. The techniques are described below, grouped according to their control implementation category.

Complete Training Techniques. CTRL (Keskar et al., 2019) is a conditional transformer LM trained to learn control codes which are prepended to the input

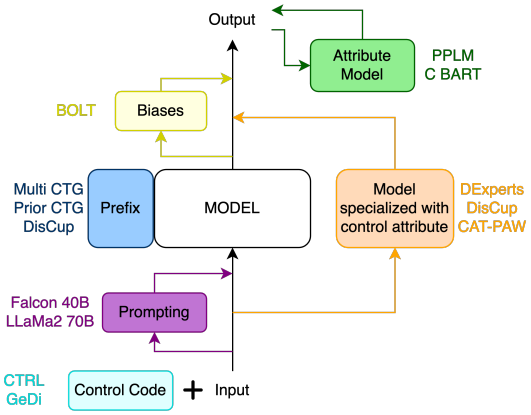


Figure 4.2: Diagrammatic view of where in the model architecture the control is injected at inference time, for each of our 12 CTG techniques evaluated.

to steer text generation towards the desired attribute. It provides control of topic, language and task, e.g. translation and question. The desired control attribute is injected at inference time by adding a control code to the input.

Fine-Tuning Techniques. Controllable BART (C BART) (He, 2021) analyses the input with a token-level classifier to extract information used by the decoder to revise the text by inserting and replacing tokens with the aim of making sure the sentence contains the desired keywords, while remaining fluent.

Multi CTG (Gu et al., 2022b) fine-tunes GPT-2 to learn distinct vector representations for each control attribute. When multiple attributes are desired, the model combines these attribute vectors into a single unified representation, which is then used as a prefix to guide the text generation process toward satisfying all the desired attributes simultaneously.

Applying a similar concept, Prior CTG (Gu et al., 2023) uses prefix-tuning to learn latent representations and applies Normalizing Flow to learn an invertible function to map the latent space into a prior space that is easier to negotiate. Operations on the control attributes are performed in the prior space, which is then reverted back into latent space and used as prefix for the decoder at inference time.

Modification of Token Distribution Techniques. PPLM (Dathathri et al., 2019) guides text generation by using pre-trained attribute classifiers without need-

ing to fine-tune the language model itself. During generation, it computes gradients from these external classifiers to determine how each potential next token would affect the desired attributes. These gradients are then used to adjust the language model's predictions, shifting them toward tokens that better align with the desired attributes while maintaining fluency.

ControllAble Text generation with Position-Aware Weighted decoding (CAT-PAW) (Gu et al., 2022a) introduces a lightweight adjustment mechanism that modifies the language model's output probabilities. This mechanism can either strengthen or weaken the influence of attribute-specific distributions (like sentiment or topic distributions) on the final text generation, by comparing and adjusting the difference between the pre-trained language model's token predictions and the desired attribute-specific token distributions.

LM Prompting Techniques. Falcon 40B Instruct (Almazrouei et al., 2023) is a causal decoder-only LM trained on the RefinedWeb dataset (Penedo et al., 2023). It adapts the GPT-3 architecture with three main differences: (i) rotational positional embeddings; (ii) multi-query attention, i.e. a single key and value shared across all attention heads instead of using separate query, key and value for each head; and (iii) parallel attention decoder blocks, allowing the model to extract different feature efficiently.

LLaMa2 70B chat (Touvron et al., 2023) is a transformer-based autoregressive causal LM built on the LLaMa2 model (trained only on publicly available datasets). It is fine-tuned in two steps: (i) supervised fine-tuning using publicly available instruction tuning data, and (ii) Reinforcement Learning with Human Feedback (RLHF), applying two separately optimised reward models.

For both LLMs, we evaluate two prompting strategies: *Zero Shot*, which includes only a simple description of the task, and *Few Shot*, which builds on the task description by adding two example inputs and outputs illustrating the desired control behaviour. Full prompts text used are provided in Appendix B.1.

Hybrid Techniques. This section's techniques combine multiple control implementation types, e.g. Modification of Token Distribution and Model Fine-Tuning. For example, Generative Discriminator (GeDi) (Krause et al., 2021) fine-tunes an LM with control codes that represent different attributes. During generation, GeDi computes the probability distribution for both the desired control code and its opposite, then uses these contrastive signals to guide text generation. By promoting tokens that align with the desired attribute while penalizing those associated with the opposite attribute, GeDi steers the generation toward the target characteristic.

Similarly, Decoding-time Experts (DExperts) (Liu et al., 2021) guides text generation by using two specialized language models: an expert model fine-tuned on text with desired attributes (e.g., positive sentiment) and an anti-expert model fine-tuned on text with undesired attributes (e.g., negative sentiment). During generation, DExperts combines the predictions from both models with the base language model by boosting token probabilities favored by the expert while suppressing those favored by the anti-expert, allowing for more nuanced control over generated text.

Discriminator Cooperative Unlikelihood Prompt-tuning (DisCup) (Zhang and Song, 2022) fine-tunes an LM incorporating a control attribute discriminator, which evaluates whether generated text exhibits the desired attribute, to optimise control prompts. The optimised prompts are used as prefixes to steer a fixed conditional LM to generate attribute-specific texts. The final next token distribution is encouraged to generate tokens with higher probability as scored by the discriminator assessing the desired attribute and discouraged to generate tokens with lower probability.

Biases Over LogiTs (BOLT) (Liu et al., 2023a) steers text generation by adding learnable bias vectors to the language model's logits. These biases are optimized through an energy function that measures both attribute relevance and language fluency: it rewards token choices that align with the desired attribute while penalizing those that harm the text's fluency. During generation, BOLT applies these optimized biases to adjust the model's token probabilities, making attribute-aligned tokens more likely to be selected.

4.4.4 Experimental Setup

Experiments are carried out for each of the four control attributes considered in this study: Sentiment, Topic, Keywords, and Multiple-attribute, where the last of these refers to the simultaneous control of both Sentiment and Topic. All models are run with the default hyperparameters reported in their respective reference papers; the complete set of hyperparameters for each model is provided in Appendix B.2.

The experimental grid is constructed by combining the 12 CTG techniques described in Section 4.4.3 with each of the four control attributes, four datasets, and three random seeds. For the two LLMs, both Zero-Shot and Few-Shot prompting strategies are additionally evaluated, as described in Section 4.4.3. One exception to the full grid applies to BOLT under keyword control: due to the prohibitive execution time that a full run would require (in excess of ten days), BOLT for keyword control is evaluated only on the PPLM Prompts dataset, which accounts for the gaps visible in the results charts.

The experimental grid is constructed by combining systems, datasets, and seeds for each control attribute as follows:

- **LLMs**: {LLaMa2 70B chat, Falcon 40B Instruct} × {zero-shot minimal instruction, few-shot in context} × {Sentiment, Topic, Keywords, Multiple} × {PPLM Prompts, OWT neutral prompts, Cloze dataset, STS benchmark} × {3 seeds (789, 3443, 9817)}
- **Sentiment**: {DExperts, GeDi, CAT-PAW, Prior CTG, BOLT, PPLM, DisCup} × {PPLM Prompts, OWT neutral prompts, Cloze dataset, STS benchmark} × {3 seeds (789, 3443, 9817)}
- **Topic**: {CTRL, GeDi, CAT-PAW, Prior CTG, PPLM} × {PPLM Prompts, OWT neutral prompts, Cloze dataset, STS benchmark} × {3 seeds (789, 3443, 9817)}
- **Keywords**: {C BART, BOLT} × {PPLM Prompts, OWT neutral prompts, Cloze dataset, STS benchmark} × {3 seeds (789, 3443, 9817)}

- **Multiple**: {Multi CTG, Prior CTG, PPLM} × {PPLM Prompts, OWT neutral prompts, Cloze dataset, STS benchmark} × {3 seeds (789, 3443, 9817)}

All the scripts and results can be found on GitHub.[19]

4.4.5 Evaluation Metrics Analysis

This section analyses Control Effectiveness and SLOR evaluation metrics introduced in Section 4.4.2 (Evaluation methods) with the aim of assessing their reliability, robustness, and potential biases in the context of CTG evaluation. Specifically, two aspects of the metrics are examined: the relative merits of average versus majority-vote aggregation for Control Effectiveness (Section 4.4.5.2 Average control effectiveness vs. majority voting control effectiveness), and the suitability of SLOR versus perplexity as a fluency measure (Section 4.4.5.3 Perplexity vs. SLOR).

4.4.5.1 Classifier testing

As described in Section 4.4.2.2 Evaluation methods, Control Effectiveness is assessed using three classifiers per control attribute, whose scores are averaged to produce a final CE score. Before using these classifiers for evaluation, each is benchmarked on a standard dataset for its respective classification task in order to verify that all three classifiers are sufficiently reliable and to identify any that may introduce systematic bias into the CE assessment. Sentiment classifiers are tested on the Yelp polarity dataset (Zhang et al., 2015), while topic classifiers are tested on the AGNews dataset (Zhang et al., 2015). Ideally, all three classifiers for a given attribute should achieve strong and broadly comparable performance, confirming that they constitute a balanced ensemble; large discrepancies in performance, particularly in precision-recall balance, may indicate that a classifier has a tendency to favour one class over another, which could bias the CE scores. The results are reported in Table 4.8.

For sentiment classification, most classifiers achieve strong performance overall, though with notable differences. T5 fine-tuned on SST-2 obtains the highest accu-

[19] `https://github.com/michelalorandi/comparative_study_of_ctg`

Dataset	Classifier	Acc	F1	Prec	Rec
	Sentiment control				
Yelp	DistilBERT fine-tuned on SST-2	90.49	90.3	92.18	88.49
	T5 fine-tuned on SST-2	95.17	95.26	93.63	96.95
	DeBERTa fine-tuned on Yelp	85.7	83.59	98.02	72.86
	Topic control				
AGNews	DistilBERT fine-tuned on AGNews	94.79	94.79	94.84	94.79
	BERT fine-tuned on AGNews	93.75	93.73	93.72	93.75
	DeBERTa fine-tuned on AGNews	94.55	94.56	94.62	94.55

Table 4.8: Accuracy, F1, Precision and Recall of different sentiment and topic classifiers on the Yelp polarity and AGnews datasets.

racy and F1 score (95.17% and 95.26, respectively), while DistilBERT fine-tuned on SST-2 performs competitively with a more balanced precision-recall trade-off. DeBERTa fine-tuned on Yelp, by contrast, exhibits markedly lower recall (72.86%) despite a very high precision of 98.02%, suggesting a tendency to favour one class over the other rather than classifying both with equal reliability. DeBERTa is nonetheless retained: it is architecturally distinct from the other two classifiers, preserving ensemble diversity, and is widely used as a sentiment classifier in the CTG literature. Section 4.4.5.2 discusses how this imbalance informs the choice of average CE over majority voting.

For topic classification, all three classifiers achieve consistently high performance, with accuracy and F1 scores above 93% across the board. DistilBERT obtains the highest accuracy (94.79%), closely followed by DeBERTa (94.55%) and BERT (93.75%). The variation across classifiers is modest, and the precision–recall balance is stable for all three, suggesting that all are reliable tools for topic classification in this evaluation setting.

4.4.5.2 Average control effectiveness vs. majority voting control effectiveness

This section examines the differences between Control Effectiveness (CE) computed as the average of the individual classifier scores and CE computed via majority

voting across classifiers. Tables 4.9 and 4.10 report results for Sentiment and Topic control, respectively.

CTG Technique		Control Effectiveness ↑ Average		Majority Vote	
Fine-Tuning					
Prior CTG		**92.63** (0.5)	[1]	92.39 (0.5)	[3]
Modification of Token Distribution					
CAT PAW		61.08 (1.6)	[12]	60.87 (1.8)	[12]
PPLM		73.79 (1.2)	[8]	73.16 (1.1)	[8]
LM Prompting					
Falcon 40B	Zero-Shot	79.77 (1.9)	[7]	77.04 (1.8)	[7]
	Few-Shot	65.79 (10.2)	[10]	66.35 (10.0)	[10]
	Overall	72.78 (10.1)	[9]	71.69 (9.0)	[9]
LLaMa2 70B	Zero-Shot	90.74 (2.9)	[4]	89.1 (0.7)	[4]
	Few-Shot	82.13 (8.7)	[6]	81.9 (7.3)	[6]
	Overall	86.43 (7.8)	[5]	85.5 (6.3)	[5]
Hybrid					
GeDi		91.22 (7.3)	[3]	**97.4** (0.4)	[1]
DisCup		91.52 (2.1)	[2]	92.9 (1.1)	[2]
DExperts		50.41 (0.8)	[13]	50.34 (1.0)	[13]
BOLT		63.61 (1.8)	[11]	62.9 (1.0)	[11]

Table 4.9: Comparison between Control Effectiveness computed as the Average of the used classifiers and Majority Voting between the classifiers for Sentiment (Pearson's r = 0.99 with Spearman's ρ 0.98). Techniques are grouped by control implementation category. Standard deviation in round brackets. System rank in square brackets.

The results indicate that the choice of aggregation method can have a non-trivial impact on individual CE scores, though its effect on system rankings is more limited. For sentiment control (Table 4.9), averaging leads to higher scores than majority voting in 8 out of 13 cases, while majority voting produces higher scores in the remaining 5. The direction of the difference is technique-dependent: the most pronounced increases under majority voting are observed for GeDi (+6.18 points), while the largest drops occur for Falcon Zero-Shot (−2.73 points). Despite these score-level differences, the impact on system rankings is minimal: the only rank change is a swap between GeDi and Prior CTG. The high correlation between the

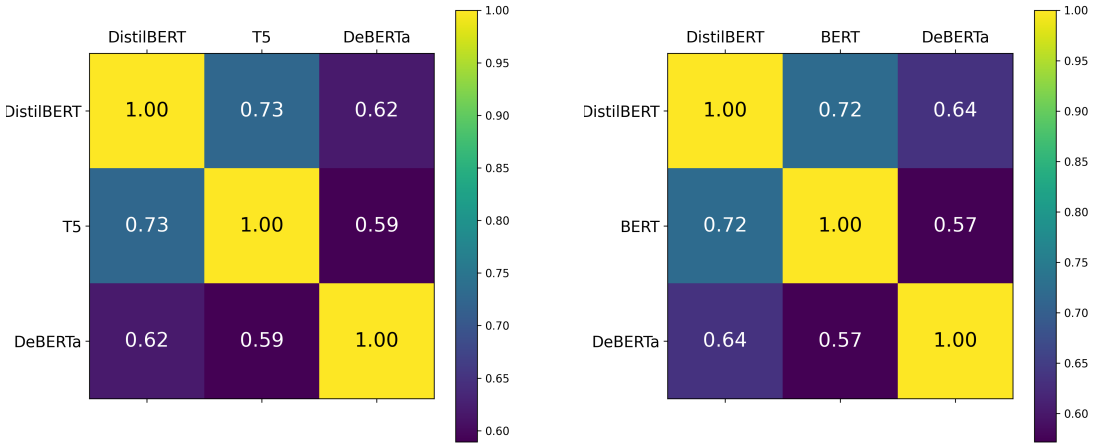
CTG Technique		Control Effectiveness ↑ Average		Control Effectiveness ↑ Majority Vote	
Complete Training					
CTRL		40.57 (3.1)	[10]	37.68 (2.7)	[11]
Fine-Tuning					
Prior CTG		**84.28** (3.1)	[1]	82.69 (0.6)	[2]
Modification of Token Distribution					
CAT PAW		62.02 (8.7)	[4]	55.22 (5.4)	[5]
PPLM		72.14 (9.7)	[3]	65.25 (8.5)	[3]
LM Prompting					
Falcon 40B	Zero-Shot	45.87 (5.3)	[8]	46.84 (1.5)	[8]
	Few-Shot	38.98 (3.7)	[11]	39.4 (2.0)	[10]
	Overall	42.43 (5.7)	[9]	43.12 (4.1)	[9]
LLaMa2 70B	Zero-Shot	55.61 (6.3)	[5]	55.97 (2.6)	[4]
	Few-Shot	51.12 (10.6)	[7]	50.7 (10.4)	[7]
	Overall	53.36 (9.0)	[6]	53.33 (8.0)	[6]
Hybrid					
GeDi		84.0 (2.9)	[2]	**84.66** (1.8)	[1]

Table 4.10: Comparison between Control Effectiveness computed as Average of the used classifiers and Majority Voting between the classifiers for Topic (Pearson's r = 0.98 with Spearman's ρ 0.97). Standard deviation in round brackets. System rank in square brackets.

two scoring methods confirms this stability, with Pearson's $r = 0.99$ and Spearman's $\rho = 0.98$.

The picture is somewhat less consistent for topic control (Table 4.10), where only 5 out of 11 ranks are identical across the two variants. The discrepancies arise primarily from rank inversions between pairs of neighbouring techniques: CTRL and Falcon Few-Shot swap ranks, as do CAT-PAW and Falcon Zero-Shot, and Prior CTG and GeDi once again. Nevertheless, the overall correlation between the two methods remains strong, with Pearson's $r = 0.98$ and Spearman's $\rho = 0.97$.

To further investigate the source of these differences and identify any outlier classifiers, pairwise correlations between classifiers are computed for both control attributes. As shown in Figures 4.3a and 4.3b, the DeBERTa classifier exhibits the lowest correlation with the other two classifiers in both cases. This means



(a) Sentiment control.

(b) Topic control.

Figure 4.3: Correlation matrix between classifiers used in sentiment and topic control. Darker color means less correlation.

that under majority voting, the outcome will predominantly reflect the agreement of the remaining two classifiers, effectively reducing the diversity of the ensemble. For this reason, and because average CE produces higher scores more frequently than majority voting (8 out of 13 cases for sentiment, and 5 out of 11 for topic), average CE rather than majority voting is adopted as the primary evaluation metric throughout this study.

4.4.5.3 Perplexity vs SLOR

This section examines the relative suitability of SLOR and the more commonly used perplexity measure for evaluating fluency in CTG systems. **Perplexity** (Jelinek et al., 1977) computes the exponential cross-entropy of a sentence S, capturing how likely a pretrained language model is to generate the given text sequence. Assuming that the language model captures linguistic knowledge well, perplexity reflects text quality to some extent (Wang et al., 2022) and is frequently used as a proxy for fluency. It is defined as:

$$\mathrm{PPL}(S) = \exp(-\mathrm{NCE}(S)) \tag{4.1}$$

where NCE is the Negative Cross-Entropy, which computes the log-probability of a sentence given a language model, normalised by sentence length:

$$\text{NCE}(S) = \frac{1}{|S|} \ln(p_M(S)) \tag{4.2}$$

and $p_M(S)$ is the probability assigned by language model M to sentence S.

SLOR (Kann et al., 2018) computes the log-probability of a sentence normalised by both its unigram log-probability and its length:

$$\text{SLOR}(S) = \frac{1}{|S|}(\ln(p_M(S)) - \ln(p_u(S))) \tag{4.3}$$

where $p_M(S)$ is the same as before, and $p_u(S)$ is the unigram probability of the sentence:

$$p_u(S) = \prod_{t \in S} p(t) \tag{4.4}$$

and $p(t)$ is the probability of token t in isolation, without context. SLOR can therefore be interpreted as NCE normalised by unigram probability and was specifically designed to address a known limitation of perplexity: because perplexity measures the likelihood of a language model generating a given sequence, it may assign high scores to fluent but less common sequences that the model considers unlikely, thereby failing to reliably capture text quality.

To compare the two metrics empirically, Tables 4.11, 4.12, 4.13, and 4.14 present side-by-side perplexity and SLOR scores for sentiment, topic, keyword, and multiple-attribute control respectively, computed as weighted averages over all datasets and seeds following the same procedure used in Section 4.4.6. A striking observation across all four control attributes is that the two LLMs, Falcon 40B Instruct and LLaMA-2 70B Chat, exhibit very high perplexity scores despite achieving SLOR scores broadly comparable to those of the other techniques. This discrepancy is likely attributable to the fact that the smaller language model used to compute perplexity, GPT-2 XL and BLOOM 1B7, as described in Section 4.4.2.2, fails to

CTG Technique		Perplexity ↓		SLOR ↑	
Fine-Tuning					
Prior CTG		74.55 (4.8)	[5]	12.01 (0.1)	[8]
Modification of Token Distribution					
CAT PAW		128.75 (15.0)	[8]	10.69 (0.1)	[13]
PPLM		29.91 (3.1)	[2]	12.9 (0.1)	[3]
LM Prompting					
Falcon 40B	Zero-Shot	1910.86 (688.4)	[11]	12.66 (0.1)	[5]
	Few-Shot	4326.4 (641.4)	[13]	11.65 (0.6)	[10]
	Overall	3118.63 (1378.9)	[12]	12.15 (0.7)	[7]
LLaMa2 70B	Zero-Shot	46.82 (27.2)	[3]	12.93 (1.0)	[2]
	Few-Shot	1481.46 (819.5)	[10]	12.63 (0.3)	[6]
	Overall	764.14 (922.4)	[9]	12.78 (0.7)	[4]
Hybrid					
GeDi		98.91 (8.9)	[6]	11.55 (0.1)	[11]
DisCup		117.52 (17.3)	[7]	11.43 (0.2)	[12]
DExperts		64.91 (10.5)	[4]	11.92 (0.2)	[9]
BOLT		**17.0** (1.4)	[1]	**13.23** (0.1)	[1]

Table 4.11: Comparison between Perplexity and SLOR for Sentiment control. Standard deviation in round brackets. System rank in square brackets.

assign high probability to the kind of text generated by these much larger models, rather than reflecting any genuine deficit in output quality. The choice of GPT-2 XL and BLOOM 1B7 reflects the pool of openly available pre-trained language models suitable for this purpose at the time these experiments were conducted; more capable open-weight LLMs have since become available, and using one of these as the reference model for perplexity could reduce the gap in assigned probability between smaller and larger generation systems, potentially making perplexity a less unreliable fluency measure for LLM-generated text than observed here. SLOR mitigates this by accounting for the baseline unigram probability of the tokens in the sequence.

More broadly, perplexity and SLOR produce markedly different system rankings across all control attributes. For sentiment and topic control, only one rank is identical between the two metrics, while for keyword and multiple-attribute control two ranks coincide. Notably, however, both metrics agree on the identity of the best-performing system in each case. The divergence is further reflected in the

CTG Technique		Perplexity ↓		SLOR ↑	
Complete Training					
CTRL		**17.87** (6.6)	[1]	**14.98** (0.3)	[1]
Fine-Tuning					
Prior CTG		152.47 (12.1)	[7]	11.15 (0.1)	[11]
Modification of Token Distribution					
CAT PAW		19.53 (1.4)	[2]	12.96 (0.1)	[4]
PPLM		28.04 (3.9)	[4]	13.07 (0.1)	[3]
LM Prompting					
Falcon 40B	Zero-Shot	1677.81 (487.3)	[9]	12.82 (0.1)	[6]
	Few-Shot	4878.04 (909.6)	[11]	11.58 (0.7)	[9]
	Overall	3277.93 (1758.6)	[10]	12.2 (0.8)	[8]
LLaMa2 70B	Zero-Shot	24.76 (11.3)	[3]	13.4 (0.0)	[2]
	Few-Shot	216.04 (273.9)	[8]	12.51 (0.4)	[7]
	Overall	120.4 (216.1)	[6]	12.96 (0.5)	[4]
Hybrid					
GeDi		112.39 (8.5)	[5]	11.51 (0.1)	[10]

Table 4.12: Comparison between Perplexity and SLOR for Topic control. Standard deviation in round brackets. System rank in square brackets.

correlation statistics: for sentiment control, the overall Pearson's r between the two metrics is -0.07 (Spearman's $\rho = 0.39$), but this low correlation is largely driven by the anomalous behaviour of the LLMs. When LLMs are excluded, Pearson's r rises to -0.98 (Spearman's $\rho = 0.97$), where the negative sign reflects the opposing directionality of the two metrics (lower perplexity is better, higher SLOR is better). These sentiment correlations are computed over 13 systems (11 once LLMs are excluded), a small enough sample that the coefficients, particularly the near-perfect value obtained after exclusion, should be read as indicative rather than statistically robust.

A similar pattern holds for topic, keyword, and multiple-attribute control, computed over 11, 8, and 9 systems respectively (9, 6, and 7 once LLMs are excluded), where the same caution applies. When all systems are included, the Pearson's r values are -0.38, 0.17, and -0.65 respectively, with corresponding Spearman's ρ values of 0.69, 0.09, and 0.70. Once LLMs are excluded, the correlations strengthen

CTG Technique		Perplexity ↓		SLOR ↑	
Fine-Tuning					
CTG BART		694.23 (127.5)	[2]	9.75 (0.1)	[7]
LM Prompting					
Falcon 40B	Zero-Shot	7486.81 (614.9)	[8]	11.6 (0.0)	[5]
	Few-Shot	3180.63 (769.6)	[5]	11.82 (0.6)	[3]
	Overall	5333.72 (2263.0)	[7]	11.71 (0.4)	[4]
LLaMa2 70B	Zero-Shot	**455.21** (433.4)	[1]	**13.17** (0.1)	[1]
	Few-Shot	3242.02 (4012.5)	[6]	11.29 (1.5)	[6]
	Overall	1848.61 (3175.8)	[4]	12.23 (1.4)	[2]
Hybrid					
BOLT		1298.33 (76.9)	[3]	9.07 (0.1)	[8]

Table 4.13: Comparison between Perplexity and SLOR for Keywords control. Standard deviation in round brackets. System rank in square brackets.

substantially (Pearson's r: -0.86, -1.00, -0.87; Spearman's ρ: 0.89, 1.00, 0.99), confirming that the LLMs represent a qualitatively different class of system with respect to fluency scoring, and that perplexity is a particularly unreliable metric for evaluating them. These findings motivate the adoption of SLOR as the fluency metric throughout this study.

CTG Technique		Perplexity ↓		SLOR ↑	
Fine-Tuning					
Multi CTG		95.73 (11.4)	[4]	11.44 (0.3)	[8]
Prior CTG		46.65 (1.6)	[3]	11.88 (0.6)	[5]
Modification of Token Distribution					
PPLM		35.58 (3.2)	[2]	12.66 (0.1)	[3]
LM Prompting					
Falcon 40B	Zero-Shot	12328.26 (853.0)	[9]	11.25 (0.1)	[9]
	Few-Shot	3418.53 (468.3)	[7]	11.82 (0.6)	[6]
	Overall	7873.4 (4507.7)	[8]	11.54 (0.5)	[7]
LLaMa2 70B	Zero-Shot	**33.58** (23.0)	[1]	**13.11** (0.6)	[1]
	Few-Shot	492.76 (139.6)	[6]	12.28 (0.8)	[4]
	Overall	263.17 (250.4)	[5]	12.7 (0.8)	[2]

Table 4.14: Comparison between Perplexity and SLOR for Multiple-attribute control. Standard deviation in round brackets. System rank in square brackets.

4.4.6 Results

This section reports the results of all experiments conducted as part of the LPF comparative evaluation: the 12 CTG systems selected through the systematic process described in Section 4.4.2.1, assessed using the standardised evaluation metrics described in Section 4.4.2.2, across the four datasets described in Section 4.4.2.3, following the experimental grid in Section 4.4.4. In contrast to the metric analysis in Section 4.4.5, which examined the properties of the evaluation methods themselves, the results presented here reflect the performance of the CTG systems under the full LPF evaluation conditions. Tables 4.15, 4.16, 4.17, and 4.18 present per-system Distinct-n (diversity), SLOR (fluency), and CE (control effectiveness) scores for Sentiment, Topic, Keywords, and Multiple-attribute control, respectively. Scores are reported as weighted averages over all datasets and seeds, using dataset size as the weighting factor. Figures 4.4, 4.5, 4.6, and 4.7 provide a more detailed view of control effectiveness, disaggregating results by control attribute value and dataset for each of the four control types in turn. Both sets of results are discussed in the subsections that follow, organised by control type.

4.4.6.1 Sentiment control

Table 4.15 reports the overall sentiment control results. In terms of diversity, the hybrid techniques (bottom four rows) outperform all other categories, with DExperts achieving the highest Distinct-1 score and GeDi the highest Distinct-2 and Distinct-3 scores. In terms of fluency, BOLT achieves the highest SLOR score, though the remaining techniques have broadly similar scores.

The Control Effectiveness (CE) results reveal considerable variation across individual classifiers, showing the unreliability of relying on a single classifier as the sole evaluation metric. For instance, GeDi outperforms Prior CTG according to both the DistilBERT and T5 classifiers, while DeBERTa ranks them in the opposite order. Prior CTG is the most consistently high-performing technique across all three classifiers and achieves the highest average CE overall. Regarding the two

CTG Technique		Distinct-n ↑ dist1	dist2	dist3	SLOR ↑	Control Effectiveness ↑ DistilBERT	T5	DeBERTa	Avg
Fine-Tuning									
Prior CTG		7.68 (3.3)	38.77 (6.2)	69.14 (4.8)	12.01 (0.1)	92.77 (0.4)	92.82 (0.4)	**92.3** (0.4)	**92.63** (0.5)
Modification of Token Distribution									
CAT PAW		5.6 (1.7)	33.8 (3.1)	65.3 (2.6)	10.7 (0.1)	61.2 (1.5)	60.8 (1.9)	61.2 (1.1)	61.1 (1.6)
PPLM		5.7 (2.1)	32.7 (3.5)	62.3 (2.5)	12.9 (0.1)	74.8 (0.8)	73.3 (1)	73.3 (1.0)	73.8 (1.2)
LM Prompting									
Falcon	ZS	5.1 (2.7)	33.2 (5)	63.3 (4.1)	12.7 (0.1)	79.8 (1.4)	79.7 (1.8)	79.9 (2.2)	79.8 (1.9)
	FS	8.3 (2.8)	42.1 (3.7)	67.9 (3.4)	11.6 (0.6)	66.3 (10.7)	67.2 (10.1)	63.8 (9.4)	65.8 (10.2)
	Ov	6.7 (3.2)	37.7 (6.3)	65.6 (4.4)	12.1 (0.7)	73 (10.2)	73.4 (9.6)	71.9 (10.6)	72.8 (10.1)
LLaMa2	ZS	5.2 (2.7)	32.3 (5.0)	61.1 (3.9)	12.9 (1.0)	90.6 (1.1)	92.3 (0.5)	89.3 (4.3)	90.7 (2.9)
	FS	8.7 (4.6)	41.8 (6.4)	68 (3.1)	12.6 (0.3)	82.4 (7.8)	84.6 (7.7)	79.3 (9.5)	82.1 (8.7)
	Ov	6.9 (4.2)	37 (7.5)	64.6 (4.9)	12.8 (0.7)	86.5 (6.9)	88.4 (6.7)	84.3 (8.9)	86.4 (7.8)
Hybrid									
GeDi		10.1 (5.1)	**62.8** (4.7)	**88.1** (1.2)	11.5 (0.1)	**95** (0.4)	**97.6** (0.7)	81 (1.1)	91.2 (7.3)
DisCup		9.5 (3.3)	52.8 (4.0)	82.5 (0.7)	11.4 (0.2)	92.9 (0.7)	92.5 (1.7)	89.1 (1.1)	91.5 (2.1)
DExperts		**13.2** (3.8)	56.1 (3.0)	80.8 (0.8)	11.9 (0.2)	50.5 (0.9)	50.3 (0.9)	50.4 (0.5)	50.4 (0.8)
BOLT		7.9 (3.1)	43 (3.9)	72.8 (2.0)	**13.2** (0.1)	65 (1.7)	62.6 (1.4)	63.2 (1.5)	63.6 (1.8)

Table 4.15: Evaluation results for Sentiment control. Each row contains the weighted average over all datasets and 3 seeds. The Control Effectiveness columns show the accuracy of each classifier separately and the average between them. Best score per column in bold. Standard deviation in round brackets. ZS=Zero-Shot, FS=Few-Shot, Ov=Overall results averaged across ZS and FS.

LLMs, Few-Shot prompting yields lower CE scores than Zero-Shot for both Falcon and LLaMA-2, suggesting that the addition of in-context examples does not benefit sentiment control for these models.

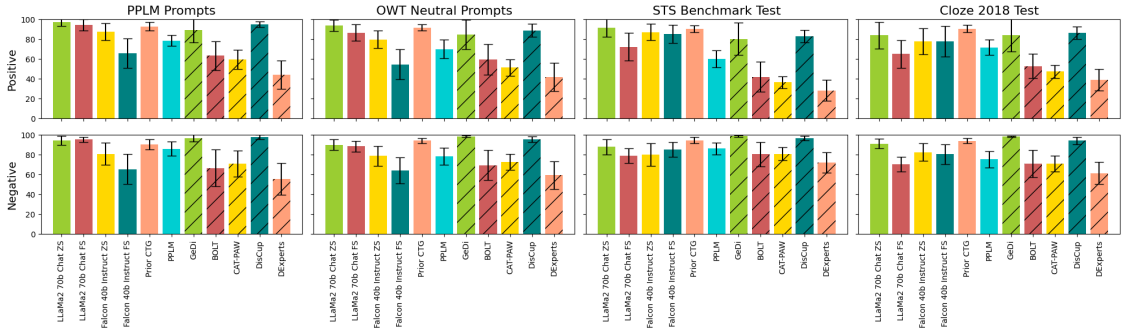


Figure 4.4: Evaluation results for Sentiment control showing the average system-level Average Control Effectiveness for each combination of dataset and control attribute values (positive/negative). Standard deviation depicted as error bars.

A more granular view is provided by Figure 4.4, which disaggregates CE by control attribute value and dataset. Prior CTG maintains consistently strong performance across all combinations. GeDi achieves near-perfect CE for Negative sentiment across all datasets, but its performance is considerably lower and more variable

for Positive sentiment, as reflected in its large standard deviation. DisCup similarly is strong on Negative sentiment, but with a smaller performance drop and lower variance on Positive. Across all systems, a consistent dataset effect is visible: performance is generally higher when evaluated on PPLM Prompts compared to the other datasets.

4.4.6.2 Topic control

Table 4.16 shows the results for Topic control. In terms of diversity, CTRL displays unusually low Distinct-n scores compared to all other techniques, indicating limited lexical variety that is likely attributable to its strong reliance on control codes. While this behaviour is expected given CTRL's design, it stands in notable contrast to its SLOR score, which is the highest across all techniques by a considerable margin, suggesting that despite low diversity its outputs remain fluent and well-formed. LLaMA-2 also performs comparatively better on fluency for topic control than was observed for sentiment control.

CTG Technique		Distinct-n ↑ dist1	dist2	dist3	SLOR ↑	Control Effectiveness ↑ DistilBERT	BERT	DeBERTa	Avg
Complete Training									
CTRL		0.6 (0.4)	4.2 (0.6)	8.5 (0.9)	**15** (0.3)	42.2 (4.0)	38.9 (2.2)	40.6 (1.7)	40.6 (3.1)
Fine-Tuning									
Prior CTG		**9.6** (4.4)	47.9 (6.1)	76.5 (3.2)	11.1 (0.1)	**87** (0.7)	**85.6** (1.0)	80.2 (1.3)	**84.3** (3.1)
Modification of Token Distribution									
CAT PAW		6.8 (2.6)	38.9 (3.6)	70.3 (1.6)	13 (0.1)	58.3 (5.8)	55.2 (4.2)	72.6 (2.2)	62 (8.7)
PPLM		6.0 (2.4)	36.2 (3.4)	66.9 (2.0)	13.1 (0.1)	68.6 (8.7)	65.9 (7.3)	**81.9** (2.9)	72.1 (9.7)
LM Prompting									
Falcon 40B	ZS	5.1 (2.6)	33.3 (5.5)	64.3 (4.6)	12.8 (0.1)	50.3 (1.5)	48.5 (1.4)	38.8 (2.0)	45.9 (5.3)
	FS	8.3 (2.7)	42.8 (3.8)	69.4 (2.8)	11.6 (0.7)	41 (2.6)	40.8 (3.2)	35.1 (1.0)	39 (3.7)
	Ov	6.7 (3.1)	38.1 (6.7)	66.8 (4.6)	12.2 (0.8)	45.7 (5.1)	44.6 (4.6)	37 (2.4)	42.4 (5.7)
LLaMa2 70B	ZS	5.2 (2.8)	31.5 (5.6)	59.4 (4.8)	13.4 (0)	59.2 (2.5)	59.9 (2.3)	47.6 (3.7)	55.6 (6.3)
	FS	9.2 (5.0)	41.3 (6.8)	66.3 (3.3)	12.5 (0.4)	53.7 (10.6)	53.5 (11.5)	46.2 (7.7)	51.1 (10.6)
	Ov	7.2 (4.5)	36.4 (7.9)	62.8 (5.4)	13 (0.5)	56.5 (8.2)	56.7 (8.9)	46.9 (6.1)	53.4 (9.0)
Hybrid									
GeDi		8.7 (5.1)	**61.9** (6.0)	**89.1** (1.7)	11.5 (0.1)	85.8 (1.7)	85.4 (2.0)	80.7 (1.3)	84 (2.9)

Table 4.16: Results for Topic control. Control Effectiveness shows the accuracy of the 3 classifiers and the average between them. Numbers are weighted averages over all datasets and 3 seeds. Best score per column in bold. Standard deviation in round brackets. ZS=Zero-Shot, FS=Few-Shot, Ov=Overall results averaged across ZS and FS.

Overall, Control Effectiveness (CE) scores are lower for topic control than for sentiment control across the same set of techniques. Prior CTG and GeDi lead by a substantial margin in terms of average CE. While PPLM outperforms Prior CTG according to the DeBERTa classifier, Prior CTG achieves the highest scores according to both DistilBERT and BERT and the highest average CE overall. As observed for sentiment control, there are meaningful differences in system rankings depending on which classifier is used, reinforcing the case for employing multiple classifiers rather than relying on a single one. GeDi achieves performance close to Prior CTG on CE while outperforming all other techniques on Distinct-2 and Distinct-3. As with sentiment control, LLaMA-2 obtains higher CE under Zero-Shot prompting than Few-Shot.

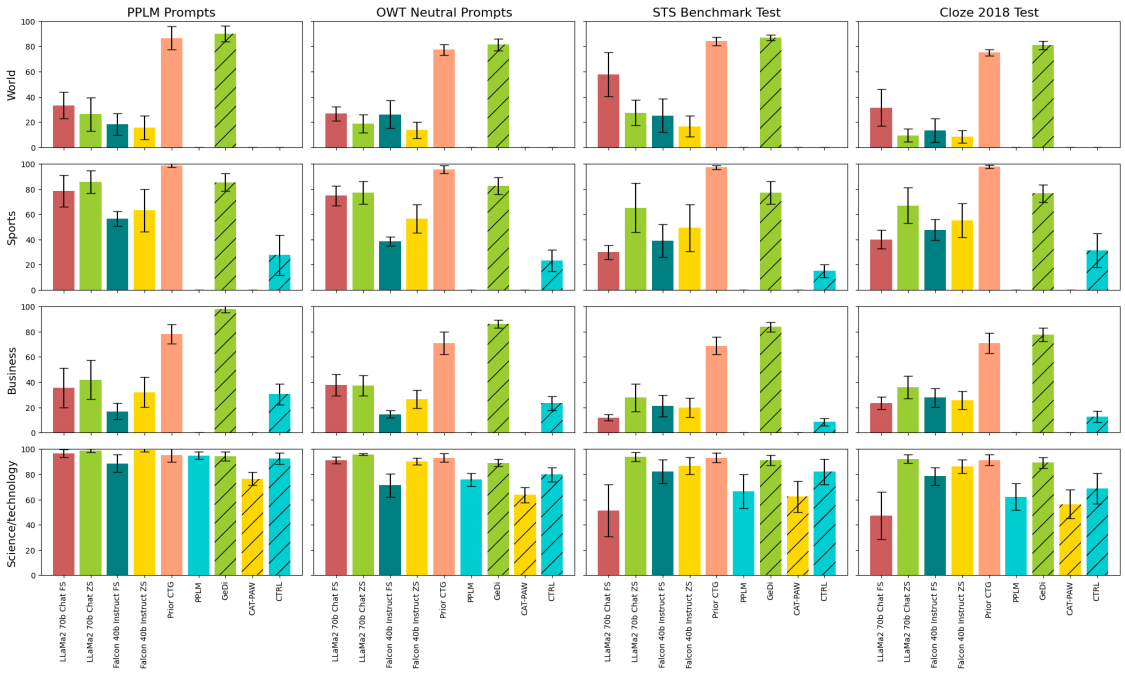


Figure 4.5: Evaluation results for Topic control showing the average Control Effectiveness across the 4 used datasets and control attribute values. Standard deviation depicted as error bars.

The per-attribute breakdown in Figure 4.5 reveals that Science/Technology consistently yields higher CE scores across all systems than the other topic values, while Business and World are the most challenging, with the lowest scores across most techniques. A plausible explanation for this pattern is that Science/Technology-related vocabulary is relatively distinctive and domain-specific, making it easier for

both generation systems and classifiers to recognise and produce topic-consistent text. Business and World topics, by contrast, involve more general and overlapping vocabulary that is less uniquely associated with a single topic, making it harder for systems to generate text that is unambiguously classified as belonging to those categories. Prior CTG and GeDi are the notable exceptions, maintaining strong performance across all four topic values and all datasets, suggesting that their control mechanisms are more robust to the inherent ambiguity of these topic categories.

4.4.6.3 Keyword control

Table 4.17 shows the overall results for Keyword control. In terms of diversity, the prompting techniques (Falcon and LLaMA-2) exhibit comparatively low Distinct-n scores, indicating limited lexical variety relative to the fine-tuning and hybrid approaches. BOLT, by contrast, outperforms all other techniques by a substantial margin across all three Distinct-n values. Regarding fluency, SLOR scores for keyword control are generally lower across all techniques than those observed for sentiment and topic control, with LLaMA-2 achieving the highest score, though still below the best SLOR recorded for topic control.

In terms of CE, C BART achieves near-perfect performance, confirming its suitability for keyword-constrained generation. All techniques perform well at including at least one exact keyword in the generated text (*Any*), but performance drops sharply when all keywords must be present simultaneously (*All*). For the two LLMs, the effect of the prompting strategy differs: Falcon performs better with Few-Shot prompting across most metrics with the exception of Exact Match *All*, while LLaMA-2 continues to perform better under Zero-Shot prompting across all metrics except Distinct-n.

The per-dataset and per-keyword-set breakdown in Figure 4.6 shows further differences. For the STS benchmark and Cloze 2018 test sets, LLaMA-2's CE collapses almost entirely under Few-Shot prompting, while performance is substantially better preserved under Zero-Shot. For the remaining techniques, performance is broadly

CTG Technique		Distinct-n ↑			SLOR ↑	Control Effectiveness ↑ Exact Match		Lemmatization		Avg
		dist1	dist2	dist3		Any	All	ExtCov	Cov	
Fine-Tuning										
C BART		10.1 (1)	33.1 (1.5)	45 (1.5)	9.7 (0.1)	**100** (0)	**100** (0)	**100** (0.0)	**98.9** (0.2)	**99.7** (0.5)
LM Prompting										
Falcon	ZS	5.2 (2.7)	30.8 (5.9)	58.7 (5.5)	11.6 (0)	63.6 (2.7)	28.8 (4.8)	48.4 (4.4)	46.1 (4.3)	46.7 (13.0)
	FS	6.3 (2.3)	34.2 (3.6)	61.8 (3.1)	11.8 (0.6)	77.1 (5.5)	27.0 (2.9)	55.6 (4.1)	53.3 (4.4)	53.2 (18.3)
	Ov	5.7 (2.6)	32.5 (5.2)	60.2 (4.7)	11.7 (0.4)	70.3 (8)	27.9 (4.1)	52 (5.6)	49.7 (5.6)	50 (16.2)
LLaMa2	ZS	4.4 (2.6)	25.8 (6.2)	49.4 (7.2)	**13.2** (0.1)	89.8 (6.7)	27.6 (11.0)	60.2 (11.5)	57.9 (11.2)	58.9 (24.3)
	FS	10.9 (7.2)	37.5 (10.3)	54.8 (3.7)	11.3 (1.5)	62.5 (39.7)	17.7 (11.7)	40.9 (26.5)	39.6 (25.7)	40.1 (31.9)
	Ov	7.7 (6.3)	31.6 (10.3)	52.1 (6.3)	12.2 (1.4)	76.2 (31.6)	22.7 (12.4)	50.5 (22.6)	48.8 (21.9)	49.5 (29.9)
Hybrid										
BOLT		**40.5** (0.4)	**82.8** (0.4)	**93.9** (0.3)	9.1 (0.1)	99 (0.8)	26.7 (7.8)	65.9 (2.6)	64.7 (2.9)	64.1 (26)

Table 4.17: Evaluation results for Keywords control. Control Effectiveness shows the Word Inclusion Coverage considering exact match and lemma match. Each row contains the weighted average over all datasets and 3 seeds. Best score per column in bold. Standard deviation in round brackets. Cov=Word Inclusion Coverage with lemmas extracted with Spacy, ExtCov=Word Inclusion Coverage with lemmas extracted with Spacy and Lemmingflect, ZS=Zero-Shot, FS=Few-Shot, Ov=Overall results averaged across ZS and FS.

stable across different keyword sets. C BART stands out across virtually all settings, successfully including all required keywords in nearly every case. In the few instances where its average CE score falls short of perfect, the cause appears to lie with the evaluation metric rather than the system itself: given C BART's perfect Exact Match scores in Table 4.17, any remaining discrepancies in the lemmatised coverage scores are likely attributable to errors introduced by the lemmatiser (spaCy and Lemmingflect, as described in Section 4.4.2.2).

4.4.6.4 Multiple-attribute control

Table 4.18 reports results for Multiple-attribute control. As discussed in Section 4.4.2.2 (Classifier testing), the *Both* column provides the most meaningful assessment of control effectiveness in this setting, as it measures the proportion of outputs in which both control attributes are satisfied *simultaneously*. The separate Sentiment (S) and Topic (T) CE columns and their average (Avg) are included for reference and to facilitate comparison with prior work, which typically reports multiple-attribute CE as an average of single-attribute scores rather than requiring simultaneous sat-



Figure 4.6: Evaluation results for Keywords control showing the average Control Effectiveness across the 4 used datasets and control attribute values. Standard deviation depicted as error bars.

isfaction.

In terms of diversity, all prompting techniques achieve lower Distinct-n scores than the other approaches, while PPLM attains the highest scores across all three values, reflecting greater output variability. Regarding fluency, LLaMA-2 achieves the highest SLOR scores, indicating strong fluency despite its comparatively lower diversity, while Falcon Zero-Shot records the lowest SLOR score.

As would be expected, overall performance drops considerably for multiple-attribute control relative to the single-attribute control types, since techniques must

CTG Technique		Distinct-n ↑ dist1	dist2	dist3	SLOR ↑	Control Effectiveness ↑ Both	S	T	Avg(!)
Fine-Tuning									
Prior CTG		3.5 (2.1)	17.9 (4.8)	36.3 (5.1)	12.4 (0)	**53.9** (1.5)	85.5 (0.5)	64.9 (1.5)	**75.2** (0.9)
Multi CTG		0.5 (1.3)	6.3 (2.6)	13.8 (3.3)	11.4 (0.3)	12.5 (0.1)	50.1 (0.3)	24.9 (0.1)	37.5 (0.2)
Modification of Token Distribution									
PPLM		**5.7** (2)	**32.5** (3.7)	**61.5** (3.1)	12.7 (0.1)	51.9 (1.3)	59.7 (1.5)	**86.6** (2.5)	73.2 (1)
LM Prompting									
Falcon 40B	ZS	2.3 (1.9)	24.2 (5)	54.7 (4.7)	11.2 (0.1)	29.3 (1.3)	67.9 (0.7)	40.7 (2.1)	54.3 (0.9)
	FS	3.1 (2)	27.8 (3.8)	55.4 (3.2)	11.8 (0.6)	24.6 (3.9)	63.7 (7.9)	37.9 (1.4)	50.8 (4.6)
	Ov	2.8 (2)	26 (4.8)	55 (4)	11.5 (0.5)	26.9 (3.7)	65.8 (6)	39.3 (2.2)	52.5 (3.7)
LLaMa2 70B	ZS	1.6 (1.7)	18.3 (4.9)	43.3 (5.5)	**13.1** (0.6)	48.4 (4)	**87.6** (1.4)	55.7 (3.9)	71.6 (2.5)
	FS	3.7 (3.1)	24.2 (4.7)	46.3 (4.2)	12.3 (0.8)	39.3 (14.1)	77.5 (11.9)	48.7 (12.9)	63.1 (12.4)
	Ov	2.6 (2.7)	21.2 (5.6)	44.8 (5.1)	12.7 (0.8)	43.9 (11.3)	82.6 (9.8)	52.2 (10.1)	67.4 (9.9)

Table 4.18: Results for Multiple-attribute control. Control Effectiveness is calculated via majority voting between three classifiers. The 'Both' column shows the proportion of outputs where both control attributes are correct. The **Avg CE**(!) column shows the average of the single-attribute CE scores and is included for reference only. Each row contains the weighted average over all datasets and 3 seeds. Best score per column in bold. Standard deviation in round brackets. S=Sentiment, T=Topic, ZS=Zero-Shot, FS=Few-Shot, Ov=Overall results averaged across ZS and FS.

now satisfy two constraints simultaneously. In terms of CE, both LLMs perform better under Zero-Shot prompting than Few-Shot, consistent with the pattern observed for sentiment and topic control. Prior CTG achieves the highest *Both* score and the highest average CE, making it the strongest system overall for multiple-attribute control, despite this representing a marked drop relative to its near-ceiling single-attribute performance (Tables 4.15 and 4.16), suggesting that even the strongest-performing prefix-based technique does not fully preserve its single-attribute control strength when both constraints must be satisfied simultaneously. PPLM, by contrast, attains the highest T score and the highest diversity scores across all three Distinct-n values, but does not lead on *Both* or *Avg*. Multi CTG, which was specifically designed for multiple-attribute control, performs among the lowest observed, indicating that its ability to combine attribute representations does not translate into effective simultaneous control in practice.

The per-dataset and per-attribute-value breakdown in Figure 4.7 reveals that the combination of Business topic with either positive or negative sentiment is the most challenging setting across all models and datasets, consistent with the difficulty

observed for Business in the topic control results. Conversely, the highest CE scores are achieved for positive sentiment combined with Sports or Science/Technology, again in line with the topic control findings. These patterns may be related, at least in part, to the topic label standardisation described in Section 4.4.2.1: as shown in Table 4.6, some systems map a single standardised topic label to multiple of their own internal labels (e.g. Science/Technology maps to Science, Computers, and Space for CAT-PAW and PPLM), while others have no available mapping for certain topics such as Business or World. Since the standardised label is used to determine whether an output is classified as correct, systems whose internal topic representations align less closely with the standardised label may be systematically disadvantaged for those topics, which could contribute to the lower CE scores observed for Business and World and the comparatively higher scores for Science/Technology.

4.4.6.5 Efficiency assessment

This section analyses the evaluated CTG techniques in terms of memory requirements and inference execution time. Efficiency is an important dimension of evaluation for two reasons: first, performance must always be considered relative to computational cost; and second, the scalability and practicality of deploying a technique in real-world applications depends directly on its efficiency, particularly in resource-constrained environments or large-scale systems.

Table 4.19 reports the memory usage (in GB) of each technique during inference. It is worth noting that some CTG techniques are model-agnostic and can, in principle, be applied to any pretrained language model, meaning that their memory footprint can vary considerably depending on the size of the underlying model. Among the techniques evaluated, the LLM-based prompting approaches require by far the largest amount of memory, with LLaMA-2 70B Chat consuming approximately 137.9 GB and Falcon 40B Instruct approximately 83.6 GB. In contrast, techniques based on Model Fine-Tuning, Modification of Token Distribution, and Hybrid approaches occupy substantially less memory, while also achieving the strongest performance



Figure 4.7: Evaluation results for Multiple-attribute control showing the average Control Effectiveness across the 4 used datasets and control attribute values. Standard deviation reported with error bars.

on sentiment, topic, and keyword control.

Table 4.20 shows the average inference time per sample for each technique across all control attributes. These figures are computed by recording the start and end time of each generation and averaging the elapsed time across all samples, datasets, and seeds. Cells marked with '–' indicate that the technique does not support the corresponding control attribute and was therefore not evaluated in that setting.

Modification of Token Distribution techniques (CAT PAW and PPLM) have by

CTG Technique	Memory used (GB) Details	Total
	Complete Training	
CTRL	CTRL model (∼14.4)	∼14.4
	Fine-Tuning	
C BART	GPT-2 (∼0.5) + C BART large (∼1.6)	∼2.1
Multi CTG	BERT base (∼0.4) + GPT-2 medium (∼1.5) + Multi model (∼4.6)	∼6.5
Prior CTG	BERT base (∼0.4) + GPT-2 medium (∼1.5) + Prior model (∼4.7)	∼6.6
	Modification of Token Distribution	
CAT PAW	2* GPT-2 medium (∼1.5)	∼3
PPLM	GPT-2 medium (∼1.5)	∼1.5
	LM Prompting	
Falcon 40B Instruct	Falcon model (∼83.6)	∼83.6
LLaMa2 70B Chat	LLaMa model (∼137.9)	∼137.9
	Hybrid	
GeDi	GPT-2 xl (∼6.4) + Sentiment model (∼1.5) + Topic model (∼1.5)	∼9.4
DisCup	GPT-2 large (∼3.2) + Positive Sentiment (∼0.9) + Negative Sentiment (∼0.9)	∼5
DExperts	GPT-2 large (∼3.2) + Positive Sentiment (∼3) + Negative Sentiment (∼3)	∼9.2
BOLT	GPT-2 large (∼3.2) + Sentiment model (∼0.5)	∼3.7

Table 4.19: Memory used to store each CTG technique in GB.

far the longest inference times. This is because these methods iteratively adjust the token distribution at each generation step to steer outputs towards the desired attribute, introducing substantial computational overhead. PPLM is particularly slow for sentiment control (37.42 seconds per sample), and BOLT requires an exceptional 288.69 seconds per sample for keyword control, which directly motivated the decision to evaluate it only on PPLM Prompts for that control attribute (see Section 4.4.4).

In contrast, Complete Training, Model Fine-Tuning, and LM Prompting techniques generally achieve low inference times, since they rely on a single forward pass through the model without any iterative external intervention. Few-Shot prompting

CTG Technique		Sentiment	Topic	Keywords	Multiple
Complete Training					
CTRL		–	01.93 ±0.04	–	–
Fine-Tuning					
C BART		–	–	**00.05** ±0.0	–
Multi CTG		–	–	–	07.92 ±0.01
Prior CTG		00.35 ±0.0	**00.31** ±0.0	–	03.46 ±0.03
Modification of Token Distribution					
CAT PAW		14.69 ±1.5	12.09 ±0.35	–	–
PPLM		37.42 ±0.13	11.71 ±0.06	–	14.58 ±0.07
LM Prompting					
Falcon 40B Instr	ZS	01.37 ±0.03	01.35 ±0.01	01.46 ±0.0	**01.38** ±0.0
	FS	03.20 ±0.25	02.91 ±0.02	03.13 ±0.02	02.96 ±0.0
LLaMa2 70B Chat	ZS	01.91 ±0.0	01.92 ±0.0	02.01 ±0.0	01.93 ±0.01
	FS	02.41 ±0.0	02.41 ±0.0	02.38 ±0.01	02.37 ±0.0
Hybrid					
GeDi		02.84 ±0.04	03.15 ±0.01	–	–
DisCup		00.5 ±0.01	–	–	–
DExperts		**00.19** ±0.0	–	–	–
BOLT		22.80 ±0.14	–	288.69 ±0.0	–

Table 4.20: Inference execution time in seconds per one sample for each CTG technique using the different control attributes (sentiment, topic, keywords, and multiple-attribute). Each row contains the average over all datasets and 3 seeds with the standard deviation. In bold the fastest technique per column. Cells marked with '–' indicate that the technique does not support the corresponding control attribute and was therefore not executed for that setting.

introduces modest additional overhead compared to Zero-Shot, but remains competitive overall. The behaviour of Hybrid techniques is more variable, as their inference time depends heavily on the specific implementation and on which control attributes are being targeted.

Taken together, these findings highlight the importance of considering the efficiency–performance trade-off when selecting a CTG technique for a given application. Techniques that achieve strong control effectiveness, such as Prior CTG and GeDi, do so at a fraction of the memory and computational cost of the LLM-based approaches, making them considerably more practical for deployment in resource-constrained

settings.

4.4.6.6 Comparison with original results

To provide a concrete idea of the difference that LPF principles can make in practice, Table 4.21 contrasts CE results from the LPF evaluation presented above (averaged over the same three classifiers for all systems, as described in Section 4.4.2.2) with those reported in the papers in which each system was first presented, where a different single classifier was used in each case.

Note that only systems for which original results are available on a dataset included in the LPF evaluation set can be included in this comparison. Access to the original results is limited for two main reasons: first, some datasets used in the original papers are not publicly available, or their test splits have not been released; and second, the details of dataset usage are insufficiently specified in some papers to allow direct comparison. These constraints reduce the number of systems that can be directly compared to 6 out of the 12 evaluated, though this subset is nonetheless sufficient to illustrate the impact of standardised and consistent evaluation conditions on CE assessment.

Control	Dataset	CTG Technique	LPF	Original
Sentiment	PPLM Prompts	Prior CTG	91.59	99.7
		CAT-PAW	65.24	52.08
		BOLT	64.92	80.12
		PPLM	82.22	78.8
	OWT neutral prompts	DExperts	50.53	95.4
		DisCup	92.18	94.31
Topic	PPLM Prompts	Prior CTG	89.68	97.8
		CAT-PAW	76.51	49.09

Table 4.21: Side by side comparison of Control Effectiveness scores for sentiment control and topic control systems as assessed in original work (Original) and in LPF where it is the average of three classifier scores (DistilBERT, BERT, DeBERTa for topic; DistilBERT, T5, DeBERTa for sentiment). All scores express the proportion of outputs that are classified as having the property they are intended to have.

All scores in Table 4.21 are classification accuracies representing the proportion of outputs identified as exhibiting the intended control attribute, with the target

control value (e.g. sentiment = positive) serving as the gold label. For instance, the first row contrasts the three-classifier LPF CE score of 91.59 with the single-classifier accuracy of 99.7 reported in the original paper for Prior CTG.

While most paired scores are broadly aligned, some differences are striking. In 5 out of 8 cases, the LPF evaluation yields lower scores than the original paper. The most pronounced discrepancy is observed for DExperts on OWT Neutral Prompts (50.53 vs. 95.40), a difference of 44.87 points, followed by CAT-PAW on PPLM Prompts for sentiment control (65.24 vs. 52.08, a difference of 27.43 percentage points in the opposite direction) and BOLT on PPLM Prompts (64.92 vs. 80.12). These large drops suggest that the classifiers and datasets used in the original evaluations were particularly favourable to those systems, and that the LPF conditions reveal a more sobering picture of their actual control effectiveness. In the remaining cases, however, the LPF scores are substantially *higher* than those originally reported: CAT-PAW performs considerably better under LPF conditions for both sentiment (65.24 vs. 52.08) and topic control (76.51 vs. 49.09), as does PPLM for sentiment (82.22 vs. 78.80). One plausible contributing factor for the higher LPF scores observed for CAT-PAW and PPLM is that the LPF evaluation uses a much larger and more varied set of prompts than the original papers: PPLM Prompts contains only 35 samples, while the LPF evaluation aggregates results across four datasets. With a larger and more diverse set of prompts, the average CE score may better reflect the system's general capability rather than its performance on a narrow set of inputs that may or may not be particularly suited to it. The overall correlation between LPF and original scores is weak and not statistically significant, with Pearson's $r = 0.31$ and Spearman's $\rho = 0.37$, underlining how substantially evaluation outcomes can depend on the choice of classifier and dataset, and reinforcing the importance of the LPF approach for fair comparative assessment.

4.4.7 Discussion and Limitations

The results presented in this study highlight the advantages of employing an LPF framework for the evaluation of CTG systems. Beyond the specific findings related to CTG, the LPF approach carries broader implications for evaluation practices in NLP more generally.

Generalisation beyond CTG. While this study focuses on CTG systems, the principles underlying LPF evaluation, such as standardisation, fairness, and diversity in evaluation metrics, are applicable to other NLP tasks, for example, summarisation and dialogue generation. These tasks show the same issues of inconsistent datasets, metric implementation biases, and ad hoc experimental setups that motivated the LPF approach in the first place, at least in their current state of development. Extending these principles to other domains could promote more reliable and reproducible comparisons, enabling a clearer understanding of model strengths and weaknesses across varied contexts.

Relation to benchmarks. Benchmarks play a central role in NLP evaluation, but they are not without limitations (Bowman and Dahl, 2021). While standardised benchmarks provide a shared foundation for comparison, they often fail to address biases introduced by metric implementation choices or dataset selection. Furthermore, benchmarks tend to favour static evaluation setups that may not account for variability in real-world applications or the evolving nature of NLP tasks. Complementing benchmarks with LPF principles, such as the use of multiple metrics and diverse datasets, can address some of these shortcomings and contribute to maintaining clarity and comparability across systems.

Towards a balanced evaluation framework. Standardisation and diversity are not opposing goals in evaluation design: the LPF framework demonstrates that it is possible to apply consistent protocols and datasets across all systems while simultaneously using a diverse set of metrics and classifiers to capture multiple dimensions of

performance. Fixing the experimental conditions ensures that observed differences between systems are genuine, while diversifying the evaluation methods ensures that no single metric or classifier unduly determines the outcome. By combining these two principles, this study takes a step towards fairer and more informative comparative evaluation of CTG systems. Future work should explore the scalability of this approach, particularly in tasks involving more complex evaluation setups or a broader range of control attributes.

Challenges of reporting results for multiple metrics. Reporting results across a large number of metrics can provide a multi-faceted view of system performance, but it also introduces challenges. Different metrics applied to the same outputs may prioritise different aspects of performance and yield conflicting indications of quality, which can obscure rather than clarify the overall picture. A related risk, prevalent in multi-task benchmarking, is the practice of averaging accuracy scores across tasks or metrics that, while all carrying the label "accuracy", actually assess fundamentally different things (Ethayarajh and Jurafsky, 2020). Collapsing such scores into a single aggregate figure creates a misleading impression of overall performance, hiding the variation across individual dimensions that is often where the most informative signal lies. Reporting too many metrics also risks diluting the significance of genuinely informative measures and introducing noise into the evaluation. Careful consideration of which metrics best capture the key dimensions of performance is therefore important for maintaining clarity and enabling meaningful comparisons across systems.

Limitations. Several limitations of the present study should also be acknowledged. The evaluation focuses on techniques implementing only three control attributes: sentiment, topic, and keywords. While these are among the most widely studied attributes in the CTG literature and are broadly representative of the field, they do not cover the full range of control attributes that have been explored, such as style, syntactic constraints, or knowledge grounding. Further work is needed to

extend the LPF analysis to techniques targeting additional attributes. Similarly, the set of values considered for each attribute (e.g. binary sentiment or a fixed set of four topics) was limited; expanding these sets could yield further insights into the generalisability of the evaluated techniques.

A second limitation concerns the scope of the literature search, which was restricted to techniques published in the ACL Anthology. While this ensures the inclusion of peer-reviewed, state-of-the-art techniques with publicly available implementations, it necessarily excludes work published at other venues or systems without accessible code or model checkpoints. Broadening the scope in future work could provide a more comprehensive picture of the CTG landscape.

Finally, the evaluation relies exclusively on automatic metrics. While automatic metrics are essential for large-scale comparisons and for reproducibility, they may not fully capture aspects of text quality such as coherence, creativity, or contextual appropriateness, which are more reliably assessed through human judgement. Incorporating human evaluation in future work would provide complementary insights and strengthen the validity of the comparative findings.

4.5 Chapter Summary

This chapter has examined evaluation in controlled text generation from two angles, producing both empirical findings about the reliability of existing evaluation methods and a methodological framework for fairer comparative assessment of CTG systems.

Section 4.3 presented a meta-evaluation of 29 automatic evaluation methods for semantic consistency assessment in data-to-text generation. The study revealed that traditional textual similarity metrics and pre-Transformer neural metrics are uniformly unsuitable for this task, while LLM-based methods and embeddings-based metrics such as BLEURT and SBERT achieve the strongest correlations with human judgements, though best results still fall below $r = 0.73$. The effect of input textification was found to be broadly positive for automatic metrics but detrimental

for LLM judges on the established WebNLG 2017 dataset, a pattern attributed to training data contamination. These findings provide a comprehensive picture of the current state of semantic consistency evaluation and establish clear baselines for future work in this area.

Section 4.4 introduced a novel level-playing-field (LPF) framework for fair and informative comparative evaluation of CTG systems, and applied it to a comparative study of 12 CTG techniques. The framework addresses four key sources of evaluation bias, system selection, metric selection, dataset selection, and output generation protocols, and applies principled, standardised choices at each stage of evaluation. The study covered four control attributes (sentiment, topic, keywords, and multiple-attribute control) across four datasets and demonstrated that evaluation outcomes, including both individual scores and system rankings, can vary substantially depending on methodological choices. An analysis of evaluation metrics confirmed the unreliability of single-classifier assessment and the unsuitability of perplexity as a fluency measure for LLM-generated text, motivating the use of multiple classifiers and SLOR, respectively. A direct comparison with results reported in original papers illustrated concretely the extent to which non-standardised evaluation conditions can distort performance assessments, with some systems appearing substantially stronger or weaker under LPF conditions than in their original evaluations.

Taken together, the two studies in this chapter underscore the importance of principled evaluation design in CTG research, and provide a reusable framework and empirical baselines that can inform evaluation practice in future work. The findings also motivate the evaluation choices made in the experimental chapters that follow, where the metrics and protocols established here are applied to the assessment of the attribute-specialised PEFT modules developed in Chapter 6.

Chapter 5

Enable Content Control with Data-to-Text Generation

5.1 Background

Data-to-text generation is the task of converting structured data representations, such as knowledge graphs, database records, or structured semantic representations, to fluent and coherent natural language text (Gatt and Krahmer, 2018). Data-to-text systems can take various structured input formats. As illustrated in Figure 5.1, the systems in this work take as input a set of structured triples (subject-predicate-object relations) and produce fluent textual descriptions that verbalise the information contained in these triples. This task represents a fundamental form of CTG, where the input data explicitly constrains the semantic content that must be expressed in the output.

Data-to-text generation represents a natural starting point for investigating content control in CTG systems for several reasons. First, the input data explicitly specifies what information must be conveyed, making the ground truth content requirements clear and measurable against the output, a property that is essential for studying content control. Second, the task provides a controlled experimental setting where the content to be generated is predetermined and unambiguous, isolating

(a) Set of RDF triples

```
<entry category="Company" eid="Id21" shape="(X (X) (X) (X) (X))" shape_type="sibling" size="4">
    <modifiedtripleset>
        <mtriple>Trane | foundingDate | 1913-01-01</mtriple>
        <mtriple>Trane | location | Ireland</mtriple>
        <mtriple>Trane | foundationPlace | La_Crosse,_Wisconsin</mtriple>
        <mtriple>Trane | numberOfEmployees | 29000</mtriple>
    </modifiedtripleset>
</entry>
```

(b) English text

Trane, which was founded on January 1st 1913 in La Crosse, Wisconsin, is based in Ireland. It has 29,000 employees.

Figure 5.1: Example in English from WebNLG 2020 dataset showing input structured triples and corresponding reference output. The system must verbalise all information from the structured input while producing grammatically correct and coherent text.

content control from other generation challenges.

Despite this apparently straightforward formulation, data-to-text generation presents several technical challenges. The primary challenge lies in generating text that is simultaneously semantically faithful to the structured input and linguistically natural. Systems must learn to linearise structured information into coherent narratives, aggregate related facts into single sentences where appropriate, introduce appropriate discourse connectives and referring expressions, and maintain consistency in entity references throughout the text (Gatt and Krahmer, 2018). Additionally, the structured input normally lacks explicit information about sentence planning, aggregation strategies, or linguistic realisation choices, requiring the generation system to make these decisions while ensuring the output remains both accurate and readable. The tension between semantic coverage (ensuring all input information is expressed) and linguistic quality (producing natural, fluent text) represents a fundamental trade-off in data-to-text systems.

These characteristics make data-to-text generation an ideal testbed for evaluating control implementation strategies. The well-defined input-output relationship allows us to isolate the effectiveness of different control mechanisms (prompting, PEFT) in enabling models to respond to explicit content specifications. Success in this task requires both strong content control, accurately reflecting the structured input, and linguistic competence, producing grammatical, coherent text. By exam-

ining how different control implementation methods handle these requirements, we gain insights into their broader applicability for CTG tasks.

5.2 Chapter Overview

This chapter investigates control implementation strategies for data-to-text generation, comparing training-free and training-based approaches across multiple experimental settings. We focus on two primary research questions:

- **RQ3.1:** How effective are different prompting strategies with out-of-the-box LLMs for content-controlled generation tasks?
- **RQ3.2:** How does parameter-efficient fine-tuning (PEFT) compare to prompting-based approaches for enabling content control?

Our investigation proceeds through two complementary experimental studies, each investigating a different approach to control implementation. Section 5.3 examines training-free prompting with out-of-the-box large language models on the WebNLG 2023 shared task across multiple languages. Section 5.4 investigates parameter-efficient fine-tuning as an alternative, applying PEFT methods to Mistral 7B for English data-to-text generation. By comparing these two system types on the same fundamental task, converting structured data to natural language while controlling both what is said (content) and how it is said (form), we isolate the impact of the training regime (adaptation vs. frozen) on control effectiveness.

Characterising the system types using the categorisation framework. Figure 5.2 situates these two system types within the three-dimensional CTG framework introduced in Chapter 2. Both system types are characterised as follows:

- **Dimension 1 (Control Attributes):** Both use ***multiple-attribute control***, coordinating ***content control*** (specifying which facts to convey via structured RDF triples) with ***non-content control*** (specifying the target language via language labels).

- **Dimension 2 (Control Integration):** Both use *input addition*, where control signals are incorporated directly into the model's input (prompts or structured input).

- **Dimension 3 (Control Implementation):** The system types differ here. Section 5.3 uses *training-free LM prompting* (no parameter updates), while Section 5.4 uses *training-based PEFT fine-tuning* (gradient-based task specialisation).

This systematic variation, holding content and form attributes constant, holding control integration constant, while varying only the implementation method, enables us to isolate the impact of task specialisation on control effectiveness.

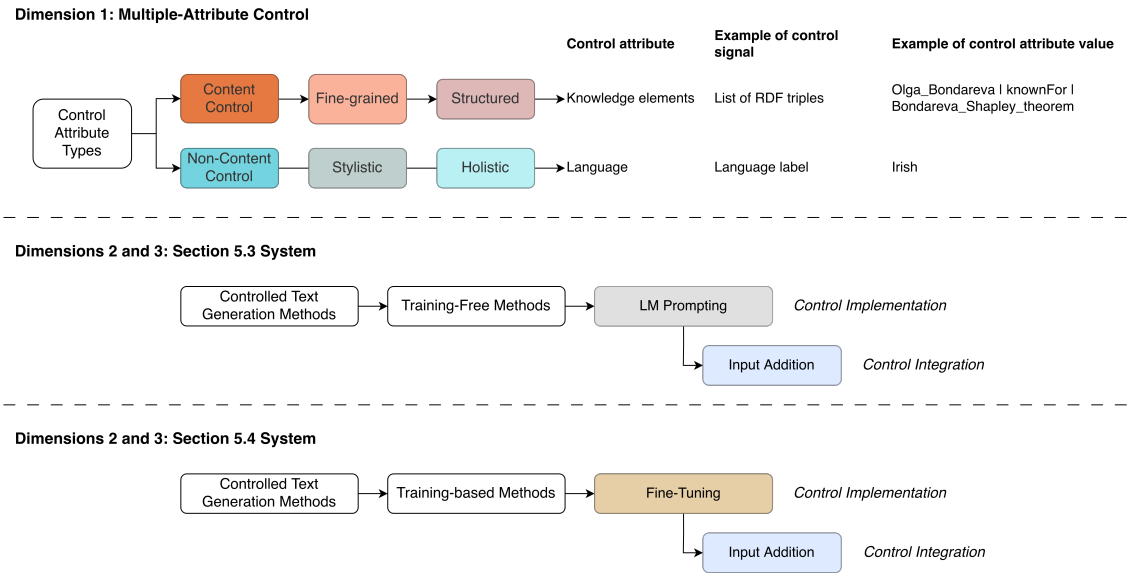


Figure 5.2: Categorisation of the two system types investigated in Chapter 5 according to the three-dimensional framework established in Chapter 2. **Dimension 1 (top):** Both systems use ***multiple-attribute control***, coordinating *content control* (via structured RDF triples specifying knowledge elements) with *non-content control* (via language labels). **Dimensions 2 and 3 (middle and bottom):** The systems share the same control integration strategy (*input addition*) but differ in control implementation, with Section 5.3 using *training-free LM prompting* and Section 5.4 implementing *training-based PEFT fine-tuning*.

Section 5.3: Training-free prompting with out-of-the-box LLMs. We first examine the out-of-the-box effectiveness of pre-trained large language models for

data-to-text generation using various prompting strategies. This approach uses *input addition* for control integration (Dimension 2, Section 2.3.3), where both control signals, the structured RDF triples specifying knowledge elements and the language label specifying the target language, are incorporated directly into textual prompts provided to the frozen model. For control implementation (Dimension 3, Section 2.3.4), this system uses *training-free LM prompting*, requiring no parameter updates or task-specific adaptation. Specifically, we evaluate closed-source and open-source LLMs on the WebNLG 2023 shared task across multiple languages (English, Irish, Welsh, Maltese, and Breton), comparing zero-shot, few-shot, and chain-of-thought prompting approaches. This study addresses RQ3.1 by systematically varying prompt design while holding the model constant, isolating the impact of prompting strategy on the model's ability to coordinate control over both *semantic content* (which facts to verbalise) and *linguistic form* (which language to use). The multilingual setting additionally enables assessment of whether this training-free, input-level control strategy generalises across languages with different resource availability and linguistic characteristics.

Section 5.4: Training-based PEFT for task specialisation. We then investigate parameter-efficient fine-tuning as an alternative control implementation strategy. Like Section 5.3, this system also uses *input addition* for control integration: the RDF triples specifying knowledge elements are provided at the input level, while the language control is handled through post-generation machine translation. However, the systems differ fundamentally in their control implementation (Dimension 3). Rather than relying on frozen model capabilities activated through prompting, Section 5.4 uses *training-based fine-tuning* via PEFT methods, specifically LoRA, prefix-tuning, and prompt tuning, applied to Mistral 7B for English data-to-text generation. Crucially, the learned PEFT modules encapsulate the general task mapping from structured data to natural language text, not specific control attribute values. The modules represent "how to perform data-to-text generation" rather than "generate with property X", meaning the actual control values (which

specific triples to verbalise) still enter through input addition, while the PEFT adaptation provides task-specific competence. This study addresses RQ3.2 by directly comparing ***training-free prompting*** (no adaptation) with ***training-based PEFT*** (gradient-based task specialisation), both using input addition integration.

Together, these studies provide a comprehensive evaluation of control implementation strategies for content control in data-to-text generation. By holding the control attributes (***content*** and ***form***) and control integration strategy (***input addition***) constant while varying the implementation method (***training-free prompting*** vs. ***training-based PEFT***), we isolate the impact of task specialisation on control effectiveness. The investigation spans different model scales (7B to 175B+ parameters), training paradigms (frozen versus adapted models), and linguistic contexts (direct multilingual generation versus generate-then-translate approaches), enabling robust assessment of when training-based adaptation improves upon out-of-the-box capabilities. These findings inform the design of attribute-specialised modules in Chapter 6, where we extend the PEFT-based approach to enable flexible, composable control over multiple attributes.

The remainder of this chapter is organised as follows. Section 5.3 presents our investigation of prompting strategies with closed-source and open-source LLMs on WebNLG 2023. Section 5.4 describes our PEFT-based approach for the GEM 2024 shared task. Section 5.5 synthesises our findings and discusses their implications for CTG system design.

Some of the content reported in this chapter was previously reported in the following publications, with overlapping sections indicated in parentheses:

1. **Michela Lorandi** and Anya Belz. 2023. **Data-to-text Generation for Severely Under-Resourced Languages with GPT-3.5: A Bit of Help Needed from Google Translate (WebNLG 2023)**. In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 80–86, Prague, Czech Republic. Association for Computational Linguistics. (Section 5.3)

2. Liam Cripwell, Anya Belz, Claire Gardent, Albert Gatt, Claudia Borg, Marthese Borg, John Judge, **Michela Lorandi**, Anna Nikiforovskaya, William Soto-Martinez, and Craig Thomson. 2023. **The 2023 WebNLG Shared Task on Low Resource Languages. Overview and Evaluation Results (WebNLG 2023)**. In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 55–66, Prague, Czech Republic. Association for Computational Linguistics.(Section 5.3)

3. **Michela Lorandi** and Anya Belz. 2024. **High-quality Data-to-Text Generation for Severely Under-Resourced Languages with Out-of-the-box Large Language Models**. *In Findings of the Association for Computational Linguistics: EACL 2024*, pages 1451–1461, St. Julian's, Malta. Association for Computational Linguistics. (Section 5.3)

4. **Michela Lorandi** and Anya Belz. 2024. **DCU-NLG-PBN at the GEM'24 Data-to-Text Task: Open-Source LLM PEFT-Tuning for Effective Data-to-Text Generation**. In *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 76–83, Tokyo, Japan. Association for Computational Linguistics. (Section 5.4)

5.3 Out-of-the-Box Pre-trained LLMs for Data-to-Text Generation

In this section, we explore whether Pre-trained Large Language Models (LLMs) can achieve reliable content control in a data-to-text setting without task-specific fine-tuning. Specifically, we explore data-to-text generation in the context of the WebNLG 2023 shared task, which presents the challenge of verbalising RDF triples into fluent natural language text across multiple under-resourced languages (Irish, Welsh, Maltese, and Breton).

We examine two fundamental questions. First, can state-of-the-art LLMs handle structured data-to-text generation out-of-the-box through prompting alone? Specifically, we investigate this question for (a) English, a well-resourced language, and (b) highly under-resourced languages (Irish, Welsh, Maltese, and Breton), to assess whether out-of-the-box prompting generalises across diverse linguistic contexts. Second, how do different prompting strategies, zero-shot minimal instruction versus few-shot in-context learning, affect the model's ability to maintain semantic consistency with the input data? To address these questions, we evaluate both closed-source models (GPT-3.5) and open-source alternatives (BLOOM, LLaMa2-chat, Falcon-chat) across two generation paradigms: direct generation into target languages and a generate-then-translate pipeline via English. Our findings provide insights into the capabilities and limitations of out-of-the-box LLMs for content-controlled generation, establishing a baseline that motivates our investigation of Parameter-Efficient Fine-Tuning (PEFT) approaches in Section 5.4.

5.3.1 WebNLG Shared Tasks Overview

The WebNLG shared tasks focus on developing methods to convert a given knowledge graph into fluent text that verbalises the graph's content. The objective is to generate texts that verbalise all and only the information contained in the input knowledge graph, which is expressed as RDF triples as shown in the example from the WebNLG 2023 website[1] in Figure 5.1.

The WebNLG+ 2020 challenge (Ferreira et al., 2020) required participants to generate English text from sets of DBpedia RDF triples, with Russian generation as an additional optional track. While this iteration advanced the state-of-the-art in data-to-text generation, participating systems remained dominated by approaches developed for high-resource languages with abundant training data. However, recent years have witnessed encouraging developments in generation for under-resourced and under-represented languages, paralleled by growing interest in multilingual mod-

[1] `https://synalp.gitlabpages.inria.fr/webnlg-challenge/docs`

els and applications across NLP more broadly.

Building on this, we use **WebNLG 2023** (Cripwell et al., 2023). This shared task shifts focus to four severely under-resourced European languages: Breton, Irish, Maltese, and Welsh, alongside Russian. The shared task provides professionally translated reference texts for evaluation. Additionally, silver-standard training samples are provided where texts were automatically translated from English.

Table 5.1 shows all the samples in training, development, and test sets in all available languages.

Language	# Train samples	# Dev samples	# Test samples
WebNLG 2020+			
English (en)	13,211	1,665	1,778
Russian (ru)	5,573	790	1,101
WebNLG 2023			
Breton (br)	13,211	1,399	1,778
Welsh (cy)	13,211	1,665	1,778
Maltese (mt)	13,211	1,665	1,778
Irish (ga)	13,211	1,665	1,778

Table 5.1: Number of training, development, and test samples per language in the WebNLG datasets. English and Russian from WebNLG 2020+ (Ferreira et al., 2020); Breton, Welsh, Irish, and Maltese from WebNLG 2023 (Cripwell et al., 2023).

5.3.2 System Description

Our system uses state-of-the-art LLMs in their pretrained, out-of-the-box configuration without any task-specific fine-tuning. We investigate two generation strategies (Figure 5.3): (i) **direct generation**, where the model verbalises structured data directly in the target language, and (ii) **generate-then-translate**, where the model first produces English text from the input data, which is subsequently translated to the target language. Generate-then-translate scenario is investigated in all languages except English.

According to the categorisation framework introduced in Chapter 2 (see Figure 5.2), this system employs *multiple-attribute control* (Dimension 1), coordinating

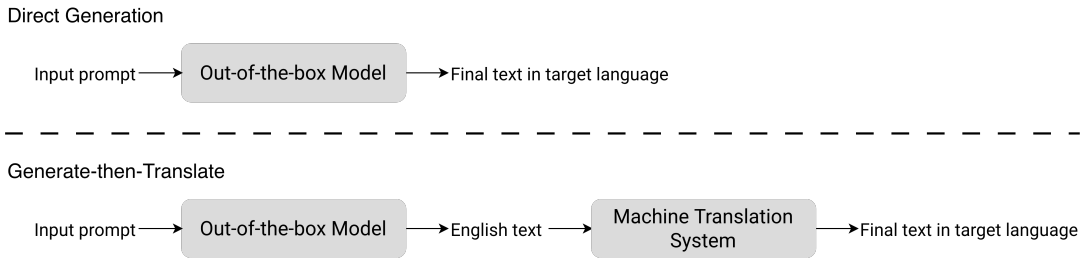


Figure 5.3: Two generation strategies for data-to-text generation in under-resourced languages. (**top**) *Direct Generation*: The model generates text directly in the target language from the input prompt. (**bottom**) *Generate-then-Translate*: The model first generates English text from the input prompt, which is then translated to the target language using a machine translation system.

content control through structured RDF triples that specify knowledge elements with *non-content control* through language labels that specify the target language. For control integration (Dimension 2, Section 2.3.3), the system uses *input addition*, where both control signals, the RDF triples and the language label, are incorporated directly into textual prompts provided to the frozen model. For control implementation (Dimension 3, Section 2.3.4), the system implements *training-free LM prompting*, requiring no parameter updates or task-specific adaptation. This positions our approach as a lightweight, zero-training method for coordinated multiple-attribute control in data-to-text generation.

5.3.2.1 Prompt types used in the final systems

Prompt testing (as described above) had revealed zero-shot and few-shot prompts with triple quotes generating into English plus MT as clearly the most promising. Figure 5.4 shows examples of these two prompt variants in terms of the prompt template, few-shot examples (where applicable), and system completion. For more example prompts, see Appendix C.1.2. **Zero-shot minimal instruction** prompts are simple requests to perform a task without details or examples, in our case consisting of a brief sentence describing the task and the input to the task. **Few-shot in-context learning** prompts are composed of a brief description of the task and a list of examples providing both inputs and target outputs, ending in just an input, for which the system response is taken to be the output.

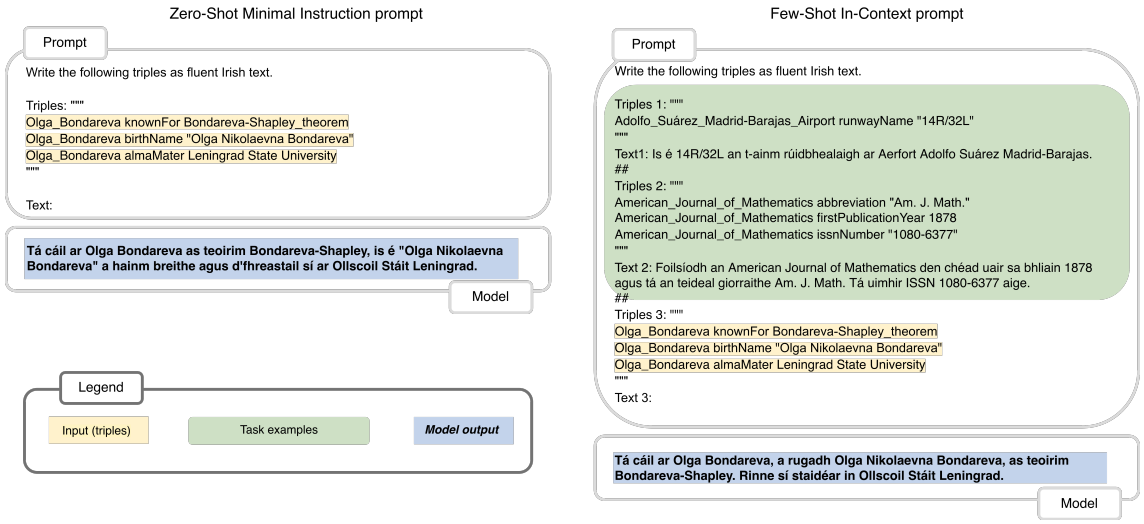


Figure 5.4: The two prompt types used in the final systems. Input triples are highlighted in yellow, model output in blue. The *zero-shot minimal instruction prompt* (**left**) consists of a brief task description followed by the input triples. The *few-shot in-context learning prompt* (**right**) additionally incorporates three example input-output pairs (highlighted in green) before the test input.

The final prompt formats follow best practice for prompt engineering provided by OpenAI;[2] input triples are recast as *subject predicate object*.

5.3.2.2 Prompt testing and selection

We conducted three prompt testing phases to identify the most promising prompt type/format for our final systems. Each phase evaluated a fresh random sample of 20 data/text pairs from the development set, stratified by triple set size, with outputs generated in all target languages. These small subsets of the dev set represent all the data we used during system development. We did not use the rest of the development set, or the test set at all until we generated outputs for it for the final system evaluations.

Phase 1. In the first prompt testing phase, we tested seven prompt variants on **text-davinci-003**, a 175B-parameter instruction-tuned model from OpenAI's GPT-3.5 family trained on diverse text and code data with reinforcement learning from human feedback (RLHF), on all four target languages. We additionally tested a sub-

[2] `https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api`

set of four prompts on **GPT-4**, a more advanced proprietary model from OpenAI, on Irish and Breton. The prompts were of three types (detailed in Section 5.3.2.1 with full prompt text in Appendix C.1.2): **zero-shot minimal instruction**, **few-shot in-context learning**, and **chain-of-thought** Wei et al. (2022).

On **text-davinci-003** (GPT-3.5), we tested each prompt in two scenarios: (i) generating the texts directly into the target languages, (ii) generating all texts into English and then translating them into each target language. The translation was performed using Google Translate for Irish, Maltese, and Welsh (Breton is not available).

In all prompts, the input triples were surrounded by triple quotes (`"""`) as per OpenAI's best-practice recommendations for prompt engineering[3]; we tested removing these quotes for the zero-shot prompt variant. On GPT-4, we tested zero-shot and few-shot with and without translation, and chain-of-thought (CoT) without translation.

The clearest results immediately were that (i) generation into English plus translation worked better than direct generation into the target languages, and (ii) the CoT prompts performed far worse than the others, a difference that was exacerbated when using GPT-4. Zero-shot looked better for Irish, but few-shot had the edge for the other three languages.

GPT-4 with zero-shot prompting achieved marginally higher BLEU, METEOR, and ChrF++ scores, but substantially worse TER scores[4]. Moreover, performance patterns across languages were less consistent with GPT-4. We tested both quoted and unquoted input for the zero-shot prompt on one language. The results showed that input triples with triple quotes `"""` performed better than without them (though this was only tested on one language).

[3] `https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api`

[4] Metric implementations available at: `https://github.com/WebNLG/2023-Challenge/tree/main/evaluation/automatic/scripts`

Phase 2. As we were testing on a very small sample, the risk was poor representativeness of the data as a whole. Therefore, in the second prompt testing phase, we used a fresh sample of 20 pairs to further test the most promising zero-shot and few-shot prompts in both scenarios: directly generating into the target language, and, for Irish, Maltese, and Welsh, generating in English followed by the translation in the target language. Only the above two prompts were tested in this phase, and they were fed only to text-davinci-003. We dropped the chain-of-thought prompts and GPT-4, as the Phase 1 results had been less promising.

Results confirmed better performance for few-shot prompting, this time in all scenarios with all evaluation metrics (although for a small number of results, scores were virtually the same). Again, having quotes worked better than leaving them out.

At this point, we tested a post-processing method on most of the previous experiments. This revealed clear improvements in almost all cases. The method replaces HTML with ASCII characters (e.g. &quot is replaced with "). Furthermore, we remove underscores and quotes that wrap the entire text.

Phase 3. Based on Phase 2 results, we tested only the zero-shot and few-shot prompts (the highest-performing configurations) in English-to-Irish and English-to-Welsh translation on fresh samples. We focused on Irish and Welsh because Phase 2 showed these languages exhibited the smallest performance differences across prompt types, making them ideal for resolving whether this convergence was robust or dataset-dependent. Even with this focused testing, zero-shot and few-shot prompting with machine translation again produced comparable results for both languages, confirming the pattern observed in Phase 2.

5.3.3 Experimental Setup

We test four different pretrained LLMs (paid-for GPT-3.5, and open-source Bloom, LLaMa2-chat, and Falcon-chat), each in two modes: (i) direct generation into the

target language, and (ii) generation into English followed by translation into the target language with one of three machine translation (MT) engines (Google Translate, Alibaba Translate, and No Language Left Behind system (Costa-jussà et al., 2022)) which enable pivoting via English to generate texts in our target languages.

Code and results available on GitHub.[5]

5.3.3.1 Models

GPT-3.5 (text-davinci-003, approximately 175B parameters) is OpenAI's InstructGPT model (Ouyang et al., 2022), an instruction-tuned variant of GPT-3. The model undergoes supervised fine-tuning on human-written instruction-response pairs, followed by reward model training and Reinforcement Learning from Human Feedback (RLHF) to align outputs with human preferences. GPT-3.5 is accessed via OpenAI's API and represents the commercial state-of-the-art for instruction-following at the time of our experiments. The training data composition is proprietary, though it is known to include substantial web text, books, and code.

BLOOM (176B parameters) (Scao et al., 2022) is an open-access multilingual language model trained by the BigScience consortium. Unlike the other models evaluated, BLOOM is a base pretrained model without instruction-tuning. It is trained on the ROOTS corpus (Laurençon et al., 2022), a collection of 498 datasets from HuggingFace spanning 46 natural languages and 13 programming languages (1.6TB of text). BLOOM's multilingual training makes it particularly relevant for our under-resourced language evaluation, though its lack of instruction-tuning may limit zero-shot task performance.

LLaMa2-chat (70B parameters) (Touvron et al., 2023) is Meta's instruction-tuned variant of LLaMa2, an open-source model trained exclusively on publicly available data (2 trillion tokens). The chat variant undergoes a two-stage fine-tuning process similar to GPT-3.5: supervised fine-tuning on instruction data followed by RLHF. However, LLaMa2-chat employs two separately optimised reward models,

[5] `https://github.com/michelalorandi/D2T-Gen-for-Under-Res-Lang-w-LLMs`

one for helpfulness and one for safety, rather than a single combined model, potentially providing finer-grained control over alignment objectives. We use the 70B parameter variant, the largest publicly available version at the time.

Falcon-chat (180B parameters) (Almazrouei et al., 2023) is the Technology Innovation Institute's instruction-tuned model, built upon Falcon-base. Falcon-base is pretrained on RefinedWeb (Penedo et al., 2023), a filtered and deduplicated web corpus (5 trillion tokens), making it one of the largest training datasets among open-source models. The chat variant is then fine-tuned on a mixture of large-scale conversational and instruction-following datasets. With 180B parameters, Falcon-chat is comparable in scale to GPT-3.5 and BLOOM, allowing for direct assessment of how training data quality and instruction-tuning methodologies affect performance at similar model sizes.

Hyperparameters and API access are provided in Section C.1.1 in the appendix.

5.3.3.2 Machine Translation systems

For the generate-then-translate approach, we use three machine translation systems: **Google Translate**[6] and **Alibaba Translate**[7] (both commercial/paid APIs) and the open-source **No Language Left Behind (NLLB)** system (Costa-jussà et al., 2022)[8]. Note that Google Translate and NLLB do not support Breton, limiting some experimental configurations for this language.

Google Translate is a commercial neural machine translation system accessed via API. The underlying architecture and training data are proprietary, though Google has indicated the system uses multilingual neural models with continual updates (Johnson et al., 2017). Google Translate supports Irish, Welsh, and Maltese but not Breton, limiting its use in our experiments. As one of the most widely deployed MT systems globally, it represents a practical baseline for real-world application scenarios.

[6] https://docs.cloud.google.com/translate/docs/reference/rest
[7] https://www.alibabacloud.com/en/product/machine-translation?_p_lc=1
[8] https://github.com/facebookresearch/fairseq/tree/nllb

Alibaba Translate is Alibaba Cloud's commercial neural machine translation service, accessed via API. Like Google Translate, the specific architecture and training data are not publicly disclosed. Alibaba Translate supports all four target languages, including Breton, making it the only commercial option available for the full language set.

No Language Left Behind (NLLB) (Costa-jussà et al., 2022) is Meta's open-source multilingual neural machine translation model supporting 200 languages. NLLB is trained on parallel corpora mined from the web using the LASER encoder, with particular focus on low-resource languages often neglected by commercial systems. The model architecture is a Transformer-based encoder-decoder with 54.5B parameters (we use the dense distilled 3.3B parameter variant for computational efficiency). NLLB represents the open-source state-of-the-art for low-resource translation and provides a reproducible alternative to commercial APIs. Like Google Translate, NLLB does not support Breton at the time of our experiments.

5.3.3.3 Experimental grid

We tested all combinations of our four LLMs, two translation engines, two prompts, and five languages, i.e. the basic experimental grid looks as follows: {GPT-3.5, Bloom, Llama2, Falcon} × {Google Translate, Alibaba Translate, NLLB system} × {zero-shot minimal instruction, few-shot in context} × {Irish, Breton, Maltese, Welsh, English}.

Having tested the postprocessing method with GPT-3.5 (Section 5.3.2.2), we conducted a subsequent testing phase with BLOOM, LLaMa2-chat, and Falcon-chat to identify whether model-specific postprocessing was required. Testing on these three models revealed that additional, model-specific postprocessing steps were necessary: we remove all Python code, occurrences of triple quotes `"""`, and output start markers (e.g., `Falcon:`) from the output of BLOOM, LLaMa2-Chat, and Falcon-Chat. These artifacts were not present in GPT-3.5 outputs and did not require equivalent treatment.

5.3.3.4 Evaluation setup

Automatic evaluation. Following the WebNLG 2023 evaluation protocol (Cripwell et al., 2023), we compute four automatic metrics using the standard WebNLG implementation.[9] **BLEU** (Papineni et al., 2002) measures n-gram overlap (0–100, higher is better), **ChrF++** (Popović, 2017) uses character n-gram F-scores, providing better correlation with human judgement for morphologically rich languages such as Irish and Welsh (0–1, higher is better). **TER** (Snover et al., 2006) counts edits required to match the reference, normalised by reference length (lower is better). Complementing these surface-form metrics, COMET (Rei et al., 2020) is a neural metric trained on human quality assessments, demonstrating superior correlation with human judgement (0–1, higher is better). Although COMET was originally developed for machine translation evaluation, we include it here to assess similarity between generated text and gold references more broadly: pretrained COMET models are available for a wide range of languages. Results for all four metrics are presented in Tables 5.3 and 5.4.

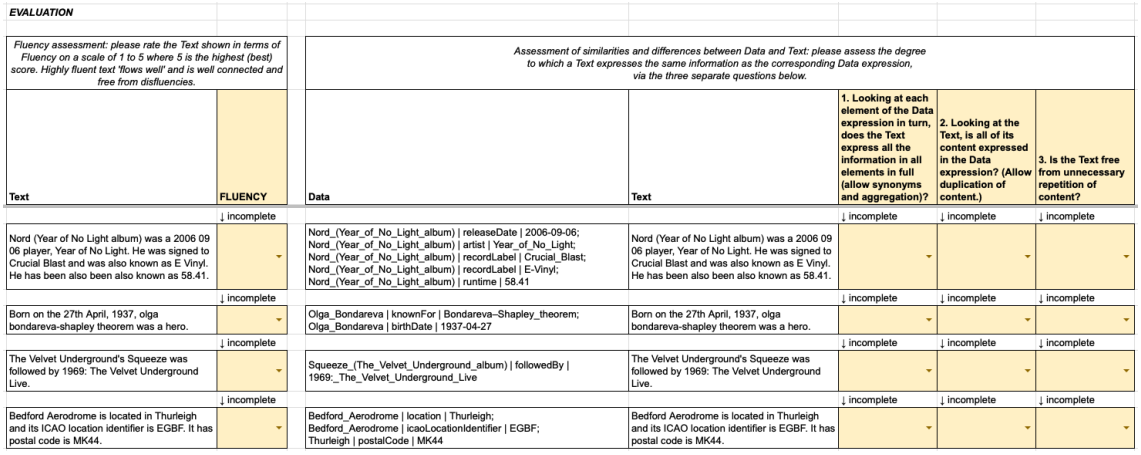


Figure 5.5: Screenshot of the human evaluation interface.

Human evaluation. We conducted our own human evaluation of four English systems using the WebNLG 2023 methodology (Cripwell et al., 2023), evaluating 100 randomly sampled outputs per system with four annotators from our re-

[9] https://github.com/WebNLG/2023-Challenge/tree/main/evaluation/automatic/scripts

search group. The evaluated systems includes GPT-3.5 with zero-shot and few-shot prompting (without translation), the best-performing WebNLG 2020 system (Amazon AI), and human-authored reference texts. We assessed four dimensions following Cripwell et al. (2023): **Fluency** (5-point Likert scale), **Absence of Additions** (Yes/No), **Absence of Omissions** (Yes/No). All annotators followed the annotation guidelines from Cripwell et al. (2023). Figure 5.5 shows the evaluation interface.

To contextualise our results, we also include comparative data from the WebNLG 2023 shared task for GPT-3.5 with Google Translate and few-shot prompting across Irish, Maltese, and Welsh, as well as the second-best performing system from that task.

5.3.4 Results

5.3.4.1 Automatic Evaluation English Results

Table 5.2 presents results for English, allowing direct comparison with WebNLG 2020 systems. GPT-3.5 achieves the highest ChrF++ (0.691) and COMET (0.831) scores among all systems including WebNLG 2020 participants, and competitive BLEU (52.74) and TER (0.42) scores, falling just short of the winning Amazon AI system (BLEU: 54, TER: 0.406). BERTScore similarly reflects GPT-3.5's strong performance (F1: 95.8). This demonstrates that out-of-the-box instruction-following LLMs can match or exceed task-specific fine-tuned systems from previous shared tasks.

Among the models tested, GPT-3.5 substantially outperforms open-source alternatives across all metrics. LLaMa2-chat (70B) and Falcon-chat (180B) achieve comparable performance to each other across surface-form metrics (BLEU: 46.86 and 46.38 respectively) and semantic similarity measures (BERTScore F1: 95.23 and 95.31; COMET: 0.821 and 0.823), while BLOOM (176B) lags considerably behind on all metrics (BLEU: 32.14, BERTScore F1: 93.68, COMET: 0.780), despite having a similar parameter count to GPT-3.5. This performance gap is largely attributable

to instruction-tuning. GPT-3.5, LLaMa2-chat, and Falcon-chat are all instruction-tuned variants of their base models, trained with supervised fine-tuning and reinforcement learning from human feedback to follow instructions and complete tasks. BLOOM, by contrast, is a base pretrained language model without instruction-tuning, explaining its substantially weaker performance on this instruction-following task. The comparison underscores that raw parameter count alone does not determine task performance, alignment training plays a critical role in enabling models to effectively interpret and execute structured generation tasks.

The impact of prompting strategy varies substantially by model alignment. Instruction-tuned and aligned models like GPT-3.5 show minimal sensitivity to prompting strategy: in English, few-shot and zero-shot performance differs only marginally (BLEU: 52.74 vs 49.66; COMET: 0.831 vs 0.826), suggesting strong inherent task understanding. In contrast, non-aligned open-source models rely critically on prompting strategy for task comprehension. BLOOM, which lacks explicit instruction-tuning and alignment training, demonstrates particularly large gains from few-shot prompting: +18.69 BLEU and +0.061 COMET in English, reflecting its dependence on in-context demonstrations. Similarly, LLaMa2-chat and Falcon-chat show consistent 6 BLEU point and 0.02–0.06 COMET improvements with few-shot prompting across both English and under-resourced languages (Irish, Welsh, Maltese, Breton). This distinction between aligned and non-aligned models is fundamental to understanding prompting effectiveness.

The consistency of these patterns across surface-form metrics (BLEU, ChrF++, TER) and learned semantic metrics (COMET, BERTScore) validates that instruction-tuning produces genuine quality improvements rather than merely optimising for specific evaluation measures.

5.3.4.2 Automatic Evaluation Under-Resourced Languages Results

Tables 5.3 and 5.4 present results for the four target languages. Across all under-resourced languages and models, the generate-then-translate approach via English

Model	Prompt	BLEU ↑	ChrF++ ↑	TER ↓	COMET ↑	BERTScore ↑		
						P	R	F1
Our systems								
GPT-3.5	ZS MI	49.66	0.689	0.45	0.826	95.55	95.68	95.55
(175B)	FS IC	*52.74*	***0.691***	*0.42*	***0.831***	***95.88***	***95.82***	***95.8***
BLOOM	ZS MI	13.45	0.457	0.705	0.719	90.92	92.34	91.51
(176B)	FS IC	*32.14*	*0.582*	*0.588*	*0.7799*	*93.8*	*93.7*	*93.68*
LLaMa2-chat	ZS MI	40.47	0.6421	0.575	0.796	94.49	94.65	94.49
(70B)	FS IC	*46.86*	*0.67*	*0.485*	*0.821*	*95.22*	*95.35*	*95.23*
Falcon-chat	ZS MI	31.35	0.592	0.654	0.765	92.76	93.79	93.19
(180B)	FS IC	*46.38*	*0.668*	*0.489*	*0.823*	*95.32*	*95.43*	*95.31*
WebNLG 2020 systems								
Baseline FORGE2020		40.6	0.621	0.517	-	-	-	-
Amazon AI (Shanghai)		**54**	0.69	**0.406**	-	-	-	-
OSU Neural NLG		53.5	0.688	0.416	-	-	-	-

Table 5.2: Automatic evaluation results for **English**. Best score per column in bold, best score per model in italics. Number of model parameters in brackets. COMET scores are between 0 and 1. ZS MI=Zero-Shot Minimal Instruction, FS IC=Few-Shot In Context.

substantially outperforms direct generation into the target language across all metrics. For GPT-3.5 with few-shot prompting, improvements from adding MT range from +4.76 BLEU points (Maltese) to +6.16 BLEU points (Welsh), with similar patterns across ChrF++, TER, and COMET, demonstrating the value of pivoting through a high-resource language even when models have been trained on multilingual data.

Translation engine comparison. We compare three machine translation systems for the generate-then-translate approach: Google Translate and Alibaba Translate (commercial APIs) and the open-source NLLB system. For GPT-3.5 with few-shot prompting, Google Translate yields the best results across all languages: Irish (20.40 BLEU, 0.510 ChrF++), Welsh (25.11 BLEU, 0.548 ChrF++), and Maltese (21.27 BLEU, 0.525 ChrF++). Alibaba Translate performs comparably, typically 1-3 BLEU points lower, while the open-source NLLB system generally produces weaker results, particularly for Irish (17.14 BLEU, 0.450 ChrF++) and Welsh (19.11 BLEU, 0.472 ChrF++).

M	Prompt	Irish BLEU↑	Irish ChrF++↑	Irish TER↓	Irish COM↑	Welsh BLEU↑	Welsh ChrF++↑	Welsh TER↓	Welsh COM↑
Our systems									
GPT-3.5 (175B)	ZS MI	12.99	0.412	0.93	0.661	15.87	0.462	0.822	0.730
	FS IC	15.35	0.43	0.845	0.699	18.95	0.474	0.719	0.752
	ZS MI + GT	***20.52***	***0.515***	0.712	0.739	24.71	***0.55***	0.67	0.792
	FS IC + GT	20.4	0.51	***0.689***	*0.743*	***25.11***	0.548	***0.643***	***0.794***
	ZS MI + AT	18.38	0.498	0.718	0.720	23.48	0.541	0.672	0.778
	FS IC + AT	18.34	0.495	0.699	0.728	23.89	0.541	0.649	0.78
	ZS MI + NLLB	17.50	0.455	0.736	0.715	19.29	0.476	0.695	0.751
	FS IC + NLLB	17.14	0.45	0.714	0.715	19.11	0.472	0.678	0.754
BLOOM (176B)	ZS MI	2.61	0.212	2.88	0.452	1.86	0.204	3.04	0.415
	FS IC	4.98	0.253	1.503	0.452	6.56	0.27	1.182	0.484
	ZS MI + GT	6.63	0.367	2.204	0.657	7.46	0.388	2.158	0.701
	FS IC + GT	*14.81*	*0.452*	*0.907*	*0.706*	*15.45*	*0.468*	*0.97*	*0.751*
	ZS MI + AT	6.22	0.36	2.145	0.646	7.312	0.387	2.13	0.686
	FS IC + AT	12.25	0.43	1.018	0.688	14.84	0.462	0.989	0.739
	ZS MI + NLLB	4.98	0.256	1.496	0.633	5.62	0.259	1.507	0.669
	FS IC + NLLB	7.69	0.271	1.013	0.681	8.57	0.27	1.004	0.719
LLaMa2-chat (70B)	ZS MI	6.44	0.235	1.271	0.476	6.64	0.253	1.102	0.477
	FS IC	10.41	0.364	1.068	0.596	8.19	0.334	1.361	0.541
	ZS MI + GT	16.7	0.487	0.837	0.72	19.84	0.521	0.805	0.766
	FS IC + GT	*19.34*	*0.503*	*0.738*	*0.738*	*23.64*	*0.541*	*0.697*	*0.786*
	ZS MI + AT	16.03	0.477	0.839	0.7005	19.3	0.514	0.812	0.755
	FS IC + AT	17.92	0.491	0.746	0.718	22.51	0.532	0.706	0.772
	ZS MI + NLLB	15.19	0.419	0.826	0.688	16.83	0.443	0.799	0.73
	FS IC + NLLB	16.57	0.442	0.755	0.709	18.36	0.463	0.721	0.749
Falcon-chat (180B)	ZS MI	6.32	0.27	1.324	0.539	6.05	0.268	1.425	0.544
	FS IC	11.23	0.366	0.99	0.618	13.07	0.361	0.882	0.657
	ZS MI + GT	13.49	0.458	1.177	0.706	15.41	0.486	1.172	0.749
	FS IC + GT	*19.61*	*0.503*	*0.745*	***0.746***	*23.17*	*0.539*	*0.712*	*0.792*
	ZS MI + AT	12.59	0.45	1.176	0.687	14.728	0.48	1.174	0.737
	FS IC + AT	17.48	0.492	0.754	0.726	22.51	0.533	0.715	0.782
	ZS MI + NLLB	12.93	0.401	1.102	0.681	13.87	0.425	1.08	0.719
	FS IC + NLLB	16.2	0.438	0.757	0.715	18.56	0.463	0.724	0.755
WebNLG 2023 systems									
FORGe		16.66	0.44	0.75	-	-	-	-	-
IREL		-	-	-	-	20.97	0.49	0.67	-
Baseline		11.63	0.36	0.74	-	10.70	0.36	0.77	-

Table 5.3: Automatic evaluation results for **Irish** and **Welsh**. Highest score in each column for each language in **bold**, highest score for each model in *italics*. Number of parameters in brackets in column 1. COMET scores are between 0 and 1. M=Model, ZS MI=Zero-Shot Minimal Instruction, FS IC=Few-Shot In Context, GT=Google Translate, AT=Alibaba Translate, NLLB=No Language Left Behind system, COM=COMET.

M	Prompt	Maltese BLEU↑	Maltese ChrF++↑	Maltese TER↓	Maltese COM↑	Breton BLEU↑	Breton ChrF++↑	Breton TER↓	Breton COM↑
Our systems									
GPT-3.5 (175B)	ZS MI	13.03	0.445	0.85	0.638	16.42	0.43	0.781	0.677
	FS IC	15.43	0.454	0.76	0.642	***18.59***	***0.447***	***0.722***	***0.696***
	ZS MI + GT	20.35	***0.526***	0.67	0.676	-	-	-	-
	FS IC + GT	***21.27***	0.525	***0.646***	***0.674***	-	-	-	-
	ZS MI + AT	16.83	0.49	0.72	0.658	10.54	0.356	0.795	0.5698
	FS IC + AT	17.57	0.487	0.693	0.656	10.24	0.35	0.786	0.571
	ZS MI + NLLB	16.46	0.481	0.726	0.658	-	-	-	-
	FS IC + NLLB	17.13	0.479	0.701	0.658	-	-	-	-
BLOOM (176B)	ZS MI	2.73	0.23	2.919	0.443	1.12	0.161	1.88	0.428
	FS IC	9.46	0.307	*0.959*	0.54	5.61	0.258	0.992	0.524
	ZS MI + GT	66.37	0.37	2.072	0.612	-	-	-	-
	FS IC + GT	*12.77*	*0.45*	0.968	*0.643*	-	-	-	-
	ZS MI + AT	5.63	0.355	2.12	0.602	4.5	0.281	1.194	0.522
	FS IC + AT	10.76	0.423	1.012	0.627	*8.25*	*0.32*	*0.877*	*0.56*
	ZS MI + NLLB	4.9	0.261	1.232	0.603	-	-	-	-
	FS IC + NLLB	6.48	0.27	0.917	0.629	-	-	-	-
LLaMa2-chat (70B)	ZS MI	10.3	0.32	0.896	0.54	4.01	0.215	0.873	0.416
	FS IC	12.59	0.39	0.827	0.592	*10.23*	0.329	0.809	0.487
	ZS MI + GT	16.73	0.503	0.786	0.655	-	-	-	-
	FS IC + GT	*19.71*	*0.519*	*0.69*	*0.669*	-	-	-	-
	ZS MI + AT	13.79	0.471	0.835	0.639	9.56	0.344	0.845	0.558
	FS IC + AT	15.62	0.479	0.748	0.649	10.01	*0.349*	*0.801*	*0.57*
	ZS MI + NLLB	14.56	0.454	0.811	0.637	-	-	-	-
	FS IC + NLLB	15.77	0.465	0.739	0.648	-	-	-	-
Falcon-chat (180B)	ZS MI	6.79	0.276	1.301	0.533	7.97	0.264	0.92	0.485
	FS IC	12.21	0.366	0.872	0.599	9.75	0.322	0.808	0.56
	ZS MI + GT	12.91	0.467	1.101	0.636	-	-	-	-
	FS IC + GT	*19.59*	*0.516*	*0.691*	*0.671*	-	-	-	-
	ZS MI + AT	10.62	0.438	1.157	0.623	8.52	0.334	0.898	0.552
	FS IC + AT	15.9	0.479	0.749	0.653	*10.28*	*0.35*	*0.801*	*0.573*
	ZS MI + NLLB	11.27	0.425	1.074	0.622	-	-	-	-
	FS IC + NLLB	15.4	0.462	0.74	0.6498	-	-	-	-
WebNLG 2023 systems									
IREL		16.49	0.47	0.7	-	-	-	-	-
CUNI-Wue		-	-	-	-	10.09	0.33	0.80	-
Baseline		15.60	0.42	0.67	-	9.92	0.33	0.76	-

Table 5.4: Automatic evaluation results for **Maltese** and **Breton**. Highest score in each column for each language in bold, highest score for each model in italics. Number of parameters in brackets in column 1. M=Model, ZS MI=Zero-Shot Minimal Instruction, FS IC=Few-Shot In Context, NLLB=No Language Left Behind system, COM=COMET.

Breton presents a notable exception: Google Translate does not support this language, forcing reliance on Alibaba Translate. Unexpectedly, for GPT-3.5, direct generation (18.59 BLEU, 0.447 ChrF++, 0.696 COMET) outperforms Alibaba Translate (10.24 BLEU, 0.350 ChrF++, 0.571 COMET) by substantial margins across all metrics. This contrasts with the other three languages, where Alibaba consistently improves over direct generation. The consistent degradation across surface-form (BLEU, ChrF++) and neural (COMET) metrics suggests genuine translation quality issues rather than metric-specific artefacts, indicating either poor training data for English-Breton translation in Alibaba's system, or that GPT-3.5's training included sufficient Breton data for reasonable direct generation.

Model comparison across languages. We now compare the performance of different models (GPT-3.5, BLOOM, LLaMa2-chat, and Falcon-chat) across target languages, evaluating both direct generation and generate-then-translate approaches. GPT-3.5 achieves the highest scores across all languages and configurations. Open-source models demonstrate substantially weaker performance, particularly BLOOM, which struggles with direct generation (BLEU scores of 1.12-9.46) but improves considerably when combined with MT (up to 15.45 BLEU, 0.468 ChrF++ for Welsh). LLaMa2-chat and Falcon-chat perform comparably to each other, achieving respectable scores with MT, for Welsh, 23.64 BLEU (LLaMa2-chat) and 23.17 BLEU (Falcon-chat), though still trailing GPT-3.5 by approximately 2 BLEU points. COMET scores follow similar patterns, with GPT-3.5 + Google Translate achieving 0.674-0.794 compared to 0.669-0.792 for instruction-tuned open-source models with MT, and substantially lower scores (0.415-0.540) for BLOOM's direct generation.

The few-shot versus zero-shot gap observed in English persists across languages. For open-source models, few-shot prompting remains essential, typically improving performance by 3-7 BLEU points. GPT-3.5 shows minimal differences between prompting strategies when using MT, with zero-shot occasionally matching or exceeding few-shot (e.g., Irish: 20.52 vs 20.40 BLEU). TER scores corroborate these findings, with GPT-3.5 + Google Translate achieving the lowest edit distances

(0.643-0.701) across all target languages.

Comparison with WebNLG 2023 systems. Our best GPT-3.5 systems exceed the WebNLG 2023 baseline across all languages and substantially outperform the second-best participating systems in most cases. For Irish, GPT-3.5 + Google Translate (20.52 BLEU) exceeds both the FORGe system (16.66 BLEU) and baseline (11.63 BLEU). For Welsh, our system (25.11 BLEU) outperforms IREL (20.97 BLEU) and baseline (10.70 BLEU). Similar patterns hold for Maltese and Breton, where GPT-3.5 outperforms participating systems despite being used out-of-the-box without task-specific tuning.

5.3.4.3 Human Evaluation

English results. Table 5.5 presents human evaluation results for English. Following standard practice, statistical significance is assessed using Tukey's Honestly Significant Difference (HSD) test at $\alpha = .05$, with systems assigned to one or more letter-labelled subsets (A, B, ...) per criterion; systems sharing a letter are not significantly different from each other, and a system spanning multiple letters (e.g. A B) is not significantly different from either adjacent subset. GPT-3.5 with few-shot prompting achieves the highest mean scores for Fluency (4.50) and Absence of Omissions (0.93), matching zero-shot GPT-3.5 for omissions. Zero-shot prompting yields the highest score for Absence of Additions (0.91). However, statistical testing reveals that all four systems fall within the same homogeneous subset for Fluency and Additions, indicating no significant differences.

For Omissions, GPT-3.5 (both variants) and human references form subset A (0.92-0.93), significantly outperforming Amazon AI (subset B, 0.82). This demonstrates that out-of-the-box GPT-3.5 not only matches the best WebNLG 2020 system but actually surpasses it in semantic completeness while maintaining comparable fluency and factual accuracy.

Notably, both GPT-3.5 variants achieve fluency scores (4.33-4.50) matching or exceeding human references (4.28), suggesting that the generated texts are as fluent

System	Fluency		Addition		Omission		
GPT-3.5 FS IC (ours)	**4.50**	A	0.88	A	**0.93**	A	
Amazon AI	4.33	A	0.90	A	0.82		B
GPT-3.5 ZS MI (ours)	4.33	A	**0.91**	A	**0.93**	A	
Human reference	4.28	A	0.83	A	0.92	A	B

Table 5.5: Human evaluation results for **English** for human-authored references, GPT-3.5 zero-shot, GPT-3.5 few-shot), and best WebNLG20 system (Amazon AI). Means and homogeneous subsets from Tukey HSD (alpha = .05); a system spanning multiple letters does not differ significantly from either subset.

as the texts written by human annotators. The high Absence of Addition scores (0.88-0.91) across all systems indicate that hallucination is minimal when generating from structured data.

Under-Resourced Languages Results. Table 5.6 presents the official WebNLG 2023 Shared Task human evaluation results for Welsh, Maltese, and Irish. Our GPT-3.5 + Google Translate system (DCU-NLG-PBN) performs remarkably close to human-authored references across all three languages and evaluation dimensions.

Lang	System	Fluency			Addition			Omission		
Welsh	Human ref	**3.28**	A		**0.9**	A		**0.84**	A	
	DCU-NLG-PBN (ours)	3.25	A		0.86	A		0.77	A	
	IREL	2.67		B	0.6		B	0.47		B
Maltese	Human ref	**4.27**	A		0.89	A		0.85	A	
	DCU-NLG-PBN (ours)	4.06	A	B	**0.91**	A		**0.86**	A	
	IREL	3.74		B	0.69		B	0.56		B
Irish	Human ref	**4.07**	A		0.81	A		0.82	A	
	DCU-NLG-PBN (ours)	3.83	A	B	0.83	A		**0.85**	A	
	DCU/TCD-FORGe	3.35		C	**0.84**	A		0.81	A	

Table 5.6: Mean **WebNLG 2023** human scores for **Welsh**, **Maltese** and **Irish**, per system for Fluency, Addition and Omission. Means and homogeneous subsets from Tukey HSD (alpha = .05); a system spanning multiple letters does not differ significantly from either subset.

For **Welsh**, DCU-NLG-PBN achieves a Fluency score of 3.25, statistically indistinguishable from human references (3.28, both subset A). Similarly, Additions (0.86 vs 0.90) and Omissions (0.77 vs 0.84) show no significant differences from

human performance. Both the human references and DCU-NLG-PBN significantly outperform the second-best system (IREL) across all criteria.

For **Maltese**, DCU-NLG-PBN again matches human references statistically: Fluency (4.06 vs 4.27, overlapping subsets A-B), Additions (0.91 vs 0.89), and Omissions (0.86 vs 0.85). Remarkably, DCU-NLG-PBN achieves the highest scores for Additions and Omissions, though differences from human references are not significant. Both substantially exceed the second-best system (IREL, subset B for all criteria).

For **Irish**, DCU-NLG-PBN (Fluency: 3.83, subset A-B) performs comparably to human references (4.07, subset A), with no significant difference for Additions (0.83 vs 0.81) or Omissions (0.85 vs 0.82). The second-best system (DCU/TCD-FORGe) achieves comparable Addition and Omission scores but significantly lower Fluency (3.35, subset C).

Across all three languages, DCU-NLG-PBN shares the same homogeneous subset with human references in 7 out of 9 criteria. Where differences do occur (2 cases), they span at most one subset tier, suggesting minimal performance gaps.

Across all three languages, DCU-NLG-PBN shares homogeneous subsets with human references for the majority of criteria (7 out of 9 comparisons), with differences never exceeding one subset level where they exist. The human evaluators found no statistically significant differences between generate-then-translate outputs from GPT-3.5 and professionally authored texts across most criteria.

Comparison with WebNLG 2020. Our English evaluation (Table 5.5) shows that GPT-3.5+MT achieves higher scores than both the best-performing WebNLG 2020 systems (Amazon AI: 90.29 Fluency, 95.20 Additions, 94.39 Omissions) and the WebNLG 2020 human references (89.85 Fluency, 94.39 Additions, 95.44 Omissions). Notably, GPT-3.5+MT exceeds the WebNLG 2020 human references across all three criteria, with particularly strong performance on Omissions. Taken together, these results indicate that the generate-then-translate approach with GPT-3.5 produces outputs that substantially outperform previous state-of-the-art systems.

System	Fluency		Addition		Omission	
Amazon AI	**90.286**	A	**95.196**	A	94.393	A
OSU Neural NLG	90.066	A	94.615	A	95.123	A
Human reference	89.846	A	94.392	A	**95.442**	A

Table 5.7: Human evaluation results of **English** from **WebNLG 2020**. Means and homogeneous subsets from Tukey HSD (alpha = .05).

5.3.5 Discussion

Synthesis of Automatic and Human Evaluation. Automatic metrics consistently show GPT-3.5 + Google Translate achieving the highest scores across all languages (Tables 5.2–5.4). Human evaluation confirms superior performance relative to previous state-of-the-art systems (Tables 5.5 and 5.6).

The human evaluation particularly highlights GPT-3.5's strength in semantic completeness (Absence of Omissions), where it significantly outperforms the best WebNLG 2020 system. This suggests that instruction-tuned LLMs effectively preserve input information in their outputs, a crucial requirement for content control in data-to-text generation.

The Effectiveness of Generate-Then-Translate. The generate-then-translate strategy proves remarkably effective for severely under-resourced languages, consistently outperforming direct generation by 4-7 BLEU points across Irish, Welsh, and Maltese. This success reflects the compound advantage of: (1) superior English generation quality from models predominantly trained on English data, and (2) commercial MT systems optimised for English-to-X translation.

Interestingly, this strategy succeeds even though it introduces an additional error-propagation step. Errors in English generation could be preserved or amplified during translation, yet the higher quality of English generation more than compensates for this risk. The human evaluation confirms that the pipeline produces outputs comparable to human references.

The Breton results provide a cautionary note: when MT quality is poor (Alibaba for English-Breton), direct generation may be preferable. This suggests the

strategy's effectiveness depends on a quality threshold for the MT component, a threshold met by Google Translate and partially by Alibaba, but not for Breton.

For languages with minimal training data in LLM pre-training and no Google Translate support, practitioners face a choice: invest in improving the English-to-target MT system, or attempt to improve direct generation through fine-tuning.

Implications for Severely Under-Resourced Languages. These results have practical implications for NLG in low-resource settings. Traditional approaches require substantial parallel data for training task-specific models, data rarely available for languages like Breton or Irish. Our results demonstrate that combining instruction-tuned LLMs with commercial MT systems can achieve human-level performance without any language-specific training data.

However, this approach has limitations. First, it requires access to commercial APIs (GPT-3.5, Google Translate), creating cost barriers and dependency on third-party services. Second, it inherits biases and errors from both the LLM and MT system. Third, it comes with more limited interpretability: understanding *why* a system fails is difficult when errors could originate from either component.

Temporal Limitations. Our evaluation of out-of-the-box prompting approaches used models from 2022–2023: GPT-3.5 (text-davinci-003), BLOOM, LLaMA 2 70B chat, and Falcon 40B chat. While these were state-of-the-art at the time of experimentation, the rapid evolution of the LLM landscape means our findings should be interpreted within this temporal context. Models released in 2024–2025 (such as LLaMA 3, Qwen 3) have demonstrated substantial improvements in instruction-following, multilingual reasoning, and compositional understanding. By 2026, these capabilities have likely advanced further. The performance gaps reported here, particularly between GPT-3.5 and open-source alternatives, and between languages (especially the under-resourced languages in Table 5.6), may narrow considerably with newer models. Improved instruction-following may enhance prompting effectiveness without fine-tuning; better multilingual models may reduce language-specific per-

formance disparities; and stronger compositional reasoning may enable more robust prompt-based attribute control. However, the core findings regarding the effectiveness of systematic prompting strategies (Section 5.3.2.1), the generate-then-translate approach, and the trade-offs between quality and accessibility are methodologically sound and likely to remain relevant across model generations. Future work should validate these prompting approaches with contemporary 2025–2026 LLMs to assess whether the relative rankings and performance gaps have shifted.

5.3.6 Limitations and Motivation for PEFT

The out-of-the-box approach, while effective for content control, has several limitations that motivate our investigation of parameter-efficient fine-tuning which we turn to next (Section 5.4).

First, while prompting enables some degree of control, deeper adaptation to domain-specific requirements or organisational constraints typically requires parameter-efficient fine-tuning or full model retraining. A user requiring formal medical language or informal conversational style has no mechanism for customisation beyond prompt engineering.

Second, the substantial performance gap between GPT-3.5 and open-source alternatives (14–20 BLEU points) creates a difficult trade-off: users requiring on-premise deployment or wishing to avoid commercial dependencies must accept significantly lower quality, while those prioritising quality must accept ongoing API costs, internet connectivity requirements, and uncertainty about model stability and availability.

Third, and most critically for our research trajectory, while models like GPT-3.5 may achieve multi-attribute control through carefully crafted prompts, this approach relies entirely on prompt engineering, a brittle and difficult-to-compose method that lacks the systematic, modular control our research aims to establish. Users cannot independently train, combine, or reuse attribute-specific controllers; instead, they must iteratively refine prompts for each new combination of desired attributes. This

approach offers no guarantee that performance will remain consistent or generalise compositionally across new attribute combinations.

Fourth, the generate-then-translate approach used in our experiments has additional limitations. It requires access to commercial APIs (GPT-3.5, Google Translate), creating cost barriers and dependency on third-party services. Furthermore, this hybrid approach inherits biases and errors from both the LLM and machine translation system, potentially compounding quality issues. Finally, it offers limited interpretability: understanding *why* a system fails is difficult when errors could originate from either the generation or translation component, complicating diagnosis and improvement.

These limitations motivate Section 5.4, where we investigate whether parameter-efficient fine-tuning can: (1) improve open-source model performance to match GPT-3.5, (2) reduce reliance on commercial systems, and (3) lay the groundwork for the attribute-specialised modules explored in Chapter 6, which provide systematic, compositional multi-attribute control through independently trainable and reusable components.

5.4 Data-to-Text Generation using PEFT Techniques

Section 5.3 demonstrated that out-of-the-box instruction-tuned models like GPT-3.5, distinguished by their substantial scale, intensive instruction tuning with reinforcement learning from human feedback (RLHF), and proprietary training regimes, can achieve human-level content control in data-to-text generation without task-specific training. However, this approach revealed important limitations: open-source alternatives (BLOOM, LLaMA 2, Falcon, Mistral) lag significantly behind GPT-3.5, creating a difficult trade-off between performance and practical considerations such as cost, deployment flexibility, and commercial dependencies. Moreover, the out-of-the-box approach constrains customisation to prompt engineering alone.

While prompting enables some degree of adaptation to domain-specific requirements and stylistic preferences, it offers no mechanism for deeper customisation to organisational constraints or systematic, reusable domain specialisation without full model retraining.

Parameter-efficient fine-tuning (PEFT) techniques offer a potential solution to these limitations. PEFT methods enable task-specific adaptation of LLMs while updating only a small fraction of parameters, making fine-tuning computationally feasible even for models with billions of parameters. Unlike full fine-tuning, which requires substantial computational resources and risks catastrophic forgetting of the base model's capabilities (Luo et al., 2025), PEFT approaches such as Low-Rank Adaptation (LoRA) (Hu et al., 2022) and its quantised variant QLoRA (Dettmers et al., 2023) achieve competitive performance while training only a small percentage of model parameters.

This section addresses RQ3.2 (How does parameter-efficient fine-tuning compare to prompting-based approaches?) using triple-to-text datasets from GEM 2024 (Mille et al., 2024b). Specifically, we investigate: (i) whether PEFT-adapted open-source models can match GPT-3.5's out-of-the-box performance, (ii) how PEFT-adapted models compare to the best prompting strategies from Section 5.3, and (iii) which PEFT methods are most effective for data-to-text generation. We evaluate these approaches on WebNLG across English and under-resourced languages, aiming to improve open-source model performance and enable domain-specific customisation.

Beyond addressing immediate performance gaps, this investigation lays the groundwork for Chapter 6, where we explore whether PEFT modules can be designed to control individual textual attributes independently, enabling compositional multi-attribute control through module combination.

5.4.1 Data

We evaluate our PEFT approaches using datasets from the GEM 2024 shared task (Mille et al., 2024b), in which we participated. The evaluation comprises two subtasks with distinct data sources and characteristics. The first subtask (D2T-1) uses the official WebNLG+ 2020 test set (Ferreira et al., 2020), which contains 1,779 triple sets with DBpedia predicates across 15 categories. While this dataset has been publicly available since 2020 and thus may have been seen by recent LLMs during training, it serves as an in-domain well-established point of reference for comparison. The second subtask (D2T-2) uses newly collected triple sets from Wikidata, queried specifically for this shared task using the methodology described by Axelsson and Skantze (2023). This Wikidata-based dataset comprises 1,800 triple sets (containing between 2 and 7 triples) featuring 74 different predicates, none of which overlap with WebNLG's 220 DBpedia predicates. Furthermore, the entities in the Wikidata dataset are predominantly distinct from those in WebNLG, making D2T-2 an out-of-domain challenge.

A key innovation of GEM 2024 is the creation of three parallel test sets for each subtask, designed to assess system robustness across different input scenarios. The **Factual** (FA) dataset contains triples with factually correct information as found in DBpedia or Wikidata. For D2T-1-FA, this corresponds to the original WebNLG+ 2020 test set, for which reference texts are available. The **Counterfactual** (CFA) dataset systematically replaces entities with other real entities of the same ontological class (e.g., substituting one person entity with another person, one date with another date), while maintaining the same predicates and triple structures, creating hypothetical scenarios that are structurally valid but potentially factually incorrect. The **Fictional** (FI) dataset replaces real entities with fictitious ones generated via LLM prompting, creating entirely invented scenarios while preserving the predicate vocabulary and data structure. Notably, the same fictional entity may appear across multiple examples in different contexts, though these do not form coherent narratives. The specific LLM used to generate these fictional entities is not disclosed in

the GEM 2024 shared task report (Mille et al., 2024b). This three-way design allows for systematic evaluation of how systems handle factually correct information, plausible-but-incorrect scenarios, and entirely novel fictional contexts.

Unlike WebNLG 2023 (Cripwell et al., 2023), which focused on severely under-resourced languages (Breton, Irish, Maltese, Welsh), GEM 2024 prioritises data novelty and contamination mitigation over multilingual evaluation. The task provides no training or development data, and no reference texts were initially provided with the test sets, creating a zero-shot evaluation setting that differs fundamentally from traditional shared tasks, where training data with input-output pairs is standard. The task organisers later compiled small sets of reference texts (180 samples per test set) in English and Swahili by recruiting annotators to manually write texts for each input: bilingual Swahili-English speakers were recruited in person from universities in Kenya to write both languages, while English texts were also collected via Amazon Mechanical Turk and Prolific. These reference texts were created specifically for evaluation purposes rather than as training material. This design shifts the evaluation focus away from pure performance optimisation. Instead, it assesses: (i) model generalisation without task-specific fine-tuning data; (ii) robustness to entities and scenarios not seen during pretraining; (iii) ability to handle factually correct versus hypothetical or fictional content; and (iv) in-domain versus out-of-domain performance. The task thus directly addresses growing concerns about LLM data contamination and encourages the development of systems that can generalise beyond memorised training examples.

5.4.2 System Description

We consider two settings to create our systems using pretrained LLMs (Figure 5.6): (i) generate text in English using out-of-the-box LLMs with prompting, (ii) generate text in English using a fine-tuned LLM. In the first setting, we employ pretrained LLMs without additional training and use various prompting strategies to guide the model in generating text based on the input RDF triples. In the second setting,

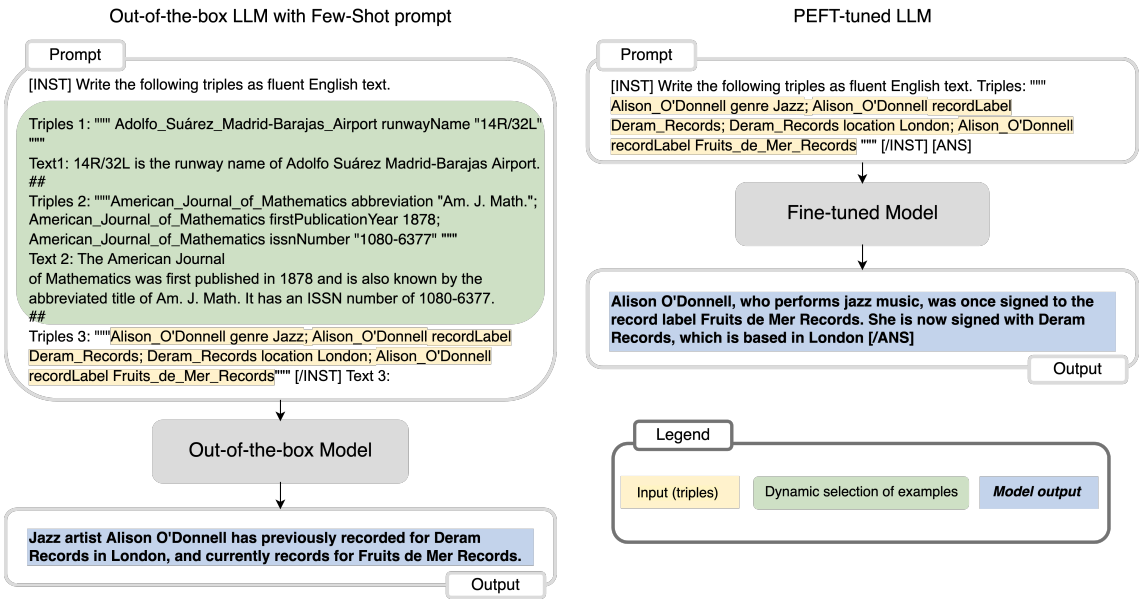


Figure 5.6: The two systems used in the final evaluation with input and output structure examples. Given Input (triples) highlighted in yellow, model output in blue. The few-shot in-context prompt also incorporates examples (highlighted green).

we fine-tune pretrained LLMs using Low-Rank Adaptation (LoRA). Regardless of the generation method, the generated English text is then translated into Chinese, German, Russian, Spanish, Korean, Hindi, Swahili, and Arabic using a Machine Translation system.

According to the categorisation framework introduced in Chapter 2 (Figures 2.1 and 2.2), this system, like those in Section 5.3.2, employs ***multiple-attribute control*** (Dimension 1; Figure 5.2), coordinating *content control* through structured RDF triples that specify knowledge elements with *non-content control* through language, handled via post-generation machine translation.

For control integration (Dimension 2; Figure 5.2), the system uses *input addition*: the RDF triples specifying knowledge elements are provided at the input level.

However, the systems differ fundamentally from Section 5.3.2 in their control implementation (Dimension 3; Figure 5.2). Rather than relying on frozen model capabilities activated through *prompting*, this section implements *training-based fine-tuning* via PEFT methods (specifically LoRA, prefix-tuning, and prompt tuning) applied to Mistral 7B. Crucially, the learned PEFT modules encapsulate the

general task mapping from structured data to natural language text, not specific control attribute values. The modules represent "how to perform data-to-text generation" rather than "generate with property X", meaning the actual control values (which specific triples to verbalise) still enter through input addition, while the PEFT adaptation provides task-specific competence. This represents a shift from the zero-training prompting approach to parameter-efficient task specialisation (Figure 5.2), trading initial training cost for potentially improved control effectiveness and deployment efficiency.

5.4.2.1 Task Construal

In both settings, the task is to generate a fully verbalised, fluent and coherent text based on the input triples. The input is composed of the task description, task examples in the case of LLM prompting, and the input triples for which we want to obtain the text. The output is expected to be the verbalisation of the input triples. The generated text must contain all and only the information contained in the input triples.

For fine-tuning, we construe the task as an instruction-based task where special tokens delimit the task description, input, and output. The special tokens are designed to train the model to accurately identify the answer, ensuring it includes all and only the information contained in the input, thereby reducing hallucinations and omissions.

For fine-tuning, we construe the task as an instruction-based task where special tokens delimit the task description, input, and output. The special tokens are designed to train the model to accurately identify the answer, ensuring it includes all and only the information contained in the input, thereby reducing hallucinations and omissions. The instruction template itself is kept deliberately minimal and generic (Table C.3), stating only the general task ("write the following triples as fluent English text") rather than attribute- or dataset-specific guidance: since task competence is intended to be encoded in the PEFT module's learned parame-

ters rather than in the prompt, richer prompt engineering of the kind used for the training-free systems in Section 5.3 would confound the comparison between control implementation strategies.

5.4.2.2 Parameter-Efficient Fine-Tuning techniques

LLMs typically contain billions of parameters, making their initial training computationally expensive and time-consuming. While these pretrained models demonstrate strong general capabilities, adapting them to specific tasks through traditional fine-tuning requires updating all parameters, which demands substantial computational resources and risks catastrophic forgetting of the original knowledge (Luo et al., 2025).

Parameter-efficient fine-tuning (PEFT) addresses these challenges by updating only a small subset of parameters and keeping the pretrained model frozen. This approach substantially reduces computational costs and memory requirements while maintaining, or even improving, task-specific performance. PEFT methods achieve efficiency by introducing small trainable components (adapters) while preserving the knowledge encoded in the pretrained model's parameters.

We investigate three PEFT techniques for data-to-text generation: Low-Rank Adaptation (LoRA), prefix-tuning, and prompt tuning.

Low-Rank Adaptation (LoRA). LoRA (Hu et al., 2022) introduces trainable low-rank decomposition matrices into the model's attention layers while keeping the original pretrained weights frozen. Rather than updating the full weight matrices during fine-tuning, LoRA decomposes the weight updates into two smaller low-rank matrices. These matrices are significantly smaller than the original weights, drastically reducing the number of trainable parameters. During inference, the learned low-rank matrices are combined with the frozen pretrained weights to produce task-specific outputs. This approach leverages the insight that the weight updates required for task adaptation often have low intrinsic dimensionality, enabling efficient fine-tuning without sacrificing performance.

Prefix-Tuning. Prefix-tuning (Li and Liang, 2021) prepends a series of trainable continuous vectors, called prefixes, to the input of each transformer layer while maintaining all model parameters frozen. These prefix vectors serve as task-specific conditioning signals that guide the model's behaviour towards the target task. Unlike discrete prompts, prefix-tuning optimises continuous vectors in the model's representation space, allowing for more flexible task adaptation. The prefix parameters are substantially fewer than the full model parameters, but achieving comparable performance.

Prompt Tuning Prompt tuning (Lester et al., 2021) adapts pretrained models by learning a small set of trainable vectors, called soft prompts or virtual tokens, that are prepended to the model's input embeddings. Unlike prefix-tuning, which inserts trainable vectors at every layer, prompt tuning modifies only the input representation and keeping all model parameters frozen. These soft prompts function as continuous task-specific signals that condition the frozen model to produce appropriate outputs. During training, only the soft prompt embeddings are updated through gradient-based optimisation, making this approach extremely parameter-efficient. Prompt tuning represents the most lightweight PEFT method, as it introduces the fewest additional parameters among the techniques considered.

Selection of PEFT Method. Model selection in our preliminary testing phase (described below) is conducted using LoRA only, given its demonstrated effectiveness across various NLG tasks and its balance between parameter efficiency and adaptation capability: LoRA's mechanism of modifying attention layers directly addresses the components most crucial for generation quality, while its low-rank constraint ensures computational efficiency. Having selected Mistral 7B base as our best-performing model on this basis, we then compare all three PEFT techniques, LoRA, prefix-tuning, and prompt tuning, applied to this model in our final experiments, to assess how the choice of PEFT method itself affects data-to-text generation quality (Section 5.4.4).

For more details about PEFT techniques, see Section 3.3.

5.4.2.3 Prompt types

In our experiments, we adopted the prompting strategies proposed by Lorandi and Belz (2023a) and described in Section 5.3.2.1: zero-shot minimal instruction and few-shot in-context learning. The **zero-shot minimal instruction** approach provides only a concise task description alongside the input data, without including any detailed guidance or examples. In contrast, the **few-shot in-context learning** strategy retains the same brief task description while incorporating a series of demonstration examples that show both the input format and expected output structure.

Building on the hypothesis that selecting demonstration examples similar to the target input would enhance generation quality, we investigated four variants of the few-shot in-context learning approach, distinguished by their example selection mechanisms:

1. *Fixed examples*: A predetermined set of examples remains fixed across all inputs in the dataset.

2. *Length-based dynamic selection*: Examples are sampled randomly from candidates whose triple set cardinality matches that of the input.

3. *Predicate-based dynamic selection*: Examples are sampled randomly from those sharing at least one predicate with the input triples; when no such matches exist, selection defaults to random sampling from the entire example pool.

4. *Combined length and predicates dynamic selection*: The selection process first filters candidates by matching triple set cardinality to retain only those that have the same number of input triples, then randomly samples from those sharing at least one predicate with the input; if this yields no candidates, random selection proceeds from the cardinality-matched subset.

5.4.3 Experimental Setup

We structure our experimental investigation in two complementary stages. First, we conduct a preliminary testing phase (Section 5.4.3.3) using the WebNLG+ 2020 validation set to evaluate multiple model and prompting configurations, allowing us to identify the most effective systems for comprehensive evaluation. Based on these preliminary findings, we then describe our main experimental configurations (Section 5.4.3.4) that form the basis for our full results. Both stages use the same set of models (Section 5.4.3.1), and both use identical post-processing procedures (Section 5.4.3.2), implementation details (Section 5.4.3.5), and evaluation methodology (Section 5.4.3.6). We describe the shared models first, then detail the preliminary testing phase and its outcomes, then present the main experimental setup informed by these results.

5.4.3.1 Models

We selected two prominent open-source language models for our experiments: Mistral 7B (Jiang et al., 2023) and Falcon 40B (Almazrouei et al., 2023). Both models represented state-of-the-art open-source performance at the time of experimentation (March 2024) and offered distinct characteristics for investigating PEFT effectiveness in Data-to-Text generation.

Mistral 7B[10] (Jiang et al., 2023) is a transformer-based decoder model with 7 billion parameters, trained on a diverse corpus of publicly available data. The model uses grouped-query attention and sliding window attention mechanisms to achieve efficient inference while maintaining strong performance across various natural language tasks. At the time of our experiments, Mistral 7B demonstrated performance competitive with substantially larger models, making it an ideal candidate for investigating whether PEFT techniques could further enhance its task-specific capabilities. In contrast, **Falcon 40B**[11] (Almazrouei et al., 2023) is a causal decoder model trained primarily on the RefinedWeb corpus (Penedo et al., 2023), a

[10] `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2`
[11] `https://huggingface.co/tiiuae/falcon-40b-instruct`

large-scale, high-quality dataset filtered from Common Crawl. With approximately 5.7 times more parameters than Mistral 7B, Falcon 40B allows us to investigate whether model scale influences PEFT performance for data-to-text generation.

For each model, we evaluate both the base pretrained version and its instruction-tuned variant. This design allows us to assess not only the impact of model scale but also how prior instruction tuning affects the parameter-efficient fine-tuning process and subsequent task performance.

The selection of these two model families serves multiple experimental purposes. The substantial difference in model size allows us to investigate whether PEFT techniques scale effectively across model capacities, addressing questions about the relationship between model scale and adaptation efficiency. By evaluating both base and instruction-tuned variants, we gain insights into how different pretraining objectives interact with task-specific fine-tuning. Furthermore, both models were among the leading open-source alternatives available at the time of experimentation, representing realistic choices for researchers with limited computational resources and providing complementary perspectives through their distinct architectural choices and training data compositions.

We acknowledge that larger models such as LLaMA 2 70B (Touvron et al., 2023) were available; however, the computational requirements for fine-tuning such models, even with PEFT techniques, exceeded our available resources. Moreover, our focus on smaller models reflects an important research consideration: investigating whether task-specific adaptation via PEFT can compensate for model scale, rather than defaulting to the largest available option, provides valuable insights about the trade-off between model size and compute efficiency. Our selection thus balances model capability with practical feasibility while deliberately investigating model scale effects, providing sufficient diversity across four model configurations (two sizes, two training paradigms) to draw meaningful conclusions about PEFT performance. This pragmatic approach reflects both the resource constraints typical in academic research settings and principled experimentation design, while main-

taining experimental rigour through systematic comparison across model scales and instruction-tuning variants.

5.4.3.2 Post-processing

We performed post-processing based on the validation set results in both settings, as follows. We removed special tokens for the start of the sentence, end of the sentence, and padding. The answer was considered to be the text between the special answer tokens in the case of fine-tuning, and the text up to the first occurrence of the character sequence *Triples* or *(Note:* in the case of prompting. We further removed `[` and `]` characters and replaced \n with a space.

5.4.3.3 Testing Phase and Model Selection

To decide which model to use as our final pretrained LLM, we conducted a testing phase using the entire English validation set of WebNLG 2020 to evaluate our settings and identify which settings to use in our final experiments. We tested two instructed-tuned LLMs for prompting and four LLMs for fine-tuning, resulting in the following experimental grids:

1. {Mistral 7B Instruct, Falcon 40B Instruct} x {Zero Shot, Few Shot, Few Shot with dynamic examples based on triple set length, Few Shot with dynamic examples based on predicates, Few Shot with dynamic examples based on triple set length and predicates} x {English}

2. {Mistral 7B Base, Mistral 7B Instruct, Falcon 40B Base, Falcon 40B Instruct} x {WebNLG 2020 (English)} x {LoRA} x {English}

Regarding prompting, we tested all prompt configurations described in Section 5.4.2.3: zero-shot minimal instruction and four few-shot variants with different example selection mechanisms (fixed, length-based, predicate-based, and combined). For the dynamic selection of examples, we created a pool of possible examples from the train set and translated them into all languages using No Language Left Behind

(NLLB) (Costa-jussà et al., 2022). This translated example pool is used only for the direct generation condition, where the few-shot examples shown to the model must be in the target language, and is not needed for the generate-then-translate condition, where prompting and generation remain in English throughout and translation is applied only to the final output. All prompts were tested on the complete validation set of WebNLG 2020. The full text of the used prompts is shown in Appendix C.2.1.

Regarding model fine-tuning, we PEFT-tuned all four LLMs (Mistral 7B Base and Instruct, Falcon 40B Base and Instruct) described in Section 5.4.3.1. We used LoRA (Hu et al., 2022) as the PEFT technique to fine-tune the selected models using the training and validation sets of WebNLG 2020.

Both systems use the post-processing steps described in Section 5.4.3.2.

5.4.3.4 System testing experimental design

Based on the testing phase results, we configured our final systems to comprehensively evaluate both prompting and PEFT approaches for data-to-text generation across multiple languages. Our final experimental design comprises three complementary system configurations that together address our research questions building on top of the preliminary findings.

We evaluated the following experimental configurations:

1. {Mistral 7B Instruct} x {Few Shot with dynamic examples based on triple set length and predicates} x {Google Translate} x {English, Chinese, German, Russian, Spanish, Korean, Hindi, Swahili, Arabic}

2. {Mistral 7B} x {WebNLG 2020 (English)} x {LoRA, Prompt tuning, Prefix tuning} x {English}

3. {Mistral 7B} x {WebNLG 2020 (English)} x {LoRA} x {Google Translate} x {Chinese, German, Russian, Spanish, Korean, Hindi, Swahili, Arabic}

We use Google Translate to translate English-generated texts into Chinese, German, Russian, Spanish, Korean, Hindi, Swahili, and Arabic.

5.4.3.5 Implementation overview

We executed our experiments using the transformer library[12] of HuggingFace and the paid-for Google Translate API[13] in late March/early April 2024. The systems are tested using the six datasets described in Section 5.4.1. All systems are executed on a Nvidia A100 GPU with 80GB RAM.

For prompting, we set mistralai/Mistral-7B-Instruct-v0.2 parameters to maximum sequence length 512. Regarding model fine-tuning, we use the PEFT library[14] of HuggingFace to create and load PEFT modules. We set mistralai/Mistral-7B-v0.1 parameters to max steps 10000, learning rate 2e-4, LoRA alpha 16, LoRA dropout 0.1, LoRA r 64, and maximum sequence length 512. For LoRA fine-tuning, we use the checkpoint at step 6000 at inference time as it has the lowest loss based on the validation set. WebNLG 2020 train set is used for the model fine-tuning.

The complete set of hyperparameters used can be found in Appendix C.2.2.

Code and results available on GitHub.[15]

5.4.3.6 Evaluation setup

Table 5.8 provides an overview of all evaluation configurations performed in this chapter. We conducted five separate evaluations across three stages: preliminary testing (row 1), internal final evaluation (row 2), and shared task evaluation (rows 3–5). These assessments progress from initial model selection to comprehensive system comparison, using both automatic metrics and human judgement across varying dataset scopes and languages. We describe each evaluation configuration below.

[12] https://huggingface.co/docs/transformers/index
[13] https://cloud.google.com/translate
[14] https://huggingface.co/docs/peft/index
[15] https://github.com/michelalorandi/DCU-NLG-PBN-GEM24

Evaluation Type	Dataset	Languages	Metrics/Criteria	N
Auto (testing)	WebNLG val	EN	BLEU, ChrF++, TER	1,667
Auto (internal)	WebNLG test	EN	BLEU, ChrF++, METEOR, BERT	1,779
Auto (shared task)	All 6 datasets	EN	BLEU, ChrF++, METEOR, BERT	180
Human (shared task)	All 6 datasets	EN, ES, SW	No Omissions, No Additions, Grammaticality, Fluency	180
LLM-as-judge (shared task)	All 6 datasets	EN, ES, SW	No Omissions, No Additions, Grammaticality, Fluency	180

Table 5.8: Overview of evaluation configurations performed across development (WebNLG validation), internal testing (WebNLG 2020 test), and shared task (GEM 2024 datasets). Human and LLM-as-judge evaluations used the same 180-sample subset with four quality criteria rated on a 7-point scale.

Automatic evaluation. Our automatic evaluation is composed of three complementary assessments at different stages and scopes. For the preliminary testing phase (Section 5.4.3.4) (Table 5.8, row 1), we compute BLEU (Papineni et al., 2002), ChrF++ (Popović, 2017), and TER (Snover et al., 2006) on the complete WebNLG 2020 English validation set. BLEU measures n-gram overlap between generated and reference texts (0-100, higher is better), ChrF++ uses character n-gram F-scores (0-1, higher is better), and TER counts the edits required to match the reference, normalised by reference length (lower is better).

For our internal final evaluation (Table 5.8, row 2), we assess system performance on the complete WebNLG 2020 English test set using: BLEU, ChrF++, METEOR (Banerjee and Lavie, 2005), and BERTScore (Zhang* et al., 2020). METEOR scores machine translations using unigram alignment with exact, stemmed, and synonym matching, combining recall-weighted precision with a word-order fragmentation penalty. BERTScore measures semantic similarity using contextual embeddings from pretrained language models, offering complementary evidence to surface-form metrics (0-1, higher is better). We exclude TER from final evaluation as preliminary analysis showed it provided limited additional insight beyond BLEU and ChrF++,

and we exclude COMET, used in our earlier experiments (Section 5.3.3.4), as it is optimised for translation quality assessment rather than data-to-text generation.

The GEM 2024 shared task organisers conducted an additional automatic evaluation (Table 5.8, row 3) on a sample of 180 inputs from the English test sets across all six datasets (WebNLG-based and Wikidata-based, each with Factual, Counterfactual, and Fictional variants), using a fixed metric set (BLEU, ChrF++, METEOR, BERTScore) that, unlike our internal evaluation, was not our choice: the organisers ran this evaluation directly on non-public test sets. This evaluation differs from our internal assessment in two key respects: it includes datasets beyond WebNLG, and it uses a single AI-generated reference text per sample rather than the multiple human-authored references available in the original WebNLG test set. As with other shared-task automatic evaluations, no statistical significance testing was applied to these scores; the comparison across participating systems reported in Section 5.4.4 is therefore based on raw metric scores rather than significance-tested differences. The shared task evaluation enables comparison with other participating systems across the full range of dataset variants, while our internal evaluation on the complete WebNLG test set with multiple references provides more robust assessment for this specific dataset.

Human evaluation. The shared task organisers conducted human evaluation (Table 5.8, rows 4) on the same 180-sample subset used for their automatic evaluation, covering all six datasets in three languages: English, Spanish, and Swahili. Human annotators assessed system outputs according to four criteria: (i) *no omissions*, whether the text includes all information from the input triples, (ii) *no additions*, whether the text avoids introducing information not present in the input, (iii) *grammaticality*, whether the text is free of grammatical and spelling errors, and (iv) *fluency*, whether the text flows well and is easy to read, with parts connected in a natural way. Each criterion was evaluated on a 7-point Likert scale ranging from "Strongly Disagree" to "Strongly Agree".

In addition to human judgements, the organisers performed an LLM-as-judge

evaluation (Table 5.8, rows 5), using LLMs to assess the same four criteria on the same samples. This dual evaluation strategy, human and LLM-based, provides complementary perspectives on system performance: human evaluation offers ground-truth quality assessment, while LLM-as-judge enables scalable evaluation and explores the viability of automated quality assessment for data-to-text generation. Results from both evaluation modalities are reported in Section 5.4.4.4.

5.4.4 Results

This section presents the results progressing from preliminary testing through final system evaluation. We structure our findings according to the evaluation stages outlined in Table 5.8: preliminary testing on the validation set, comprehensive evaluation on the English test set, automatic evaluation results from the GEM 2024 shared task, and human evaluation and LLM-as-judge findings from the GEM 2024 shared task.

5.4.4.1 Preliminary testing results

Table 5.9 presents the automatic evaluation results from our testing phase on the WebNLG 2020 English validation set, comparing LoRA fine-tuning and prompting approaches across two model families (Mistral 7B and Falcon 40B, each in base and instruction-tuned variants). The table reports three automatic metrics: BLEU (n-gram overlap), ChrF++ (character n-gram F-score), and TER (translation edit rate).

In terms of BLEU, ChrF++, and TER, LoRA fine-tuning substantially outperforms all prompting approaches across both model families. Mistral 7B base fine-tuned with LoRA achieves the highest overall scores (62.878 BLEU, 0.75 ChrF++, 0.33 TER), outperforming the best prompting configuration (Mistral 7B Instruct with dynamic few-shot selection based on length and predicates) by 22.7 BLEU points, 0.08 ChrF++, and 0.22 TER. For Falcon 40B, LoRA fine-tuning similarly dominates: both Falcon 40B base (46.0399 BLEU) and Falcon 40B Instruct (46.0189

BLEU) fine-tuned with LoRA substantially exceed the best prompting configuration (22.2892 BLEU), a gap of approximately 23 BLEU points.

Fine-tuning performance differs between the two model families with respect to instruction tuning. For Mistral 7B, fine-tuning the base model (62.878 BLEU) substantially outperforms fine-tuning the instruction-tuned variant (55.1306 BLEU), a difference of 7.7 BLEU points. For Falcon 40B, by contrast, fine-tuning performance is nearly identical between base (46.0399 BLEU) and instruction-tuned (46.0189 BLEU) variants. These divergent patterns suggest that instruction tuning's effects on fine-tuning performance are model-dependent: for Mistral, instruction tuning may introduce biases that complicate task-specific adaptation, or alternatively, the base model's more neutral starting point may provide greater flexibility for learning the data-to-text mapping; for Falcon, instruction tuning neither substantially aids nor hinders fine-tuning adaptation.

Among prompting approaches, few-shot strategies consistently outperform zero-shot across both models. For Mistral 7B Instruct, zero-shot achieves 23.2855 BLEU, while few-shot with fixed examples reaches 36.8946 BLEU, and few-shot with dynamic selection based on length and predicates achieves the best prompting performance at 40.1638 BLEU. Falcon 40B Instruct follows a similar but weaker pattern: zero-shot achieves 22.0014 BLEU, fixed few-shot reaches 25.9916 BLEU, though notably the dynamic selection variants (18.5744–22.2892 BLEU) do not consistently improve over fixed few-shot. These results validate that for Mistral, selecting contextually relevant examples enhances in-context learning for data-to-text generation; however, the Falcon results suggest this benefit is not universal across model architectures.

Model scale does not guarantee superior performance in our experiments. Despite having 5.7 times more parameters, Falcon 40B achieves lower scores than Mistral 7B across all configurations. Falcon 40B's best fine-tuned result falls well below Mistral 7B's, while its best prompting result similarly lags behind Mistral 7B Instruct. Surprisingly, Falcon 40B's dynamic example selection strategies perform

Model	Setting	BLEU↑	ChrF++↑	TER↓
Mistral 7B	Fine-tuning (LoRA)	**62.878**	**0.75**	**0.33**
Mistral 7B Instruct	Fine-tuning (LoRA)	55.1306	0.71	0.45
	Zero-shot	23.2855	0.58	0.82
	Few-shot fixed	36.8946	0.65	0.61
	Few-shot dynamic, length	36.3098	0.65	0.61
	Few-shot dynamic, predicates	38.8017	0.66	0.57
	Few-shot dynamic, length and predicates	*40.1638*	*0.67*	*0.55*
Falcon 40B	Fine-tuning (LoRA)	46.0399	0.5	0.55
Falcon 40B Instruct	Fine-tuning (LoRA)	46.0189	0.68	0.48
	Zero-shot	22.0014	0.24	0.82
	Few-shot fixed	25.9916	0.42	0.75
	Few-shot dynamic, length	18.5744	0.21	0.84
	Few-shot dynamic, predicates	16.4993	0.17	0.89
	Few-shot dynamic, length and predicates	22.2892	0.22	0.81

Table 5.9: Preliminary automatic evaluation results of our testing phase on the validation set of WebNLG 2020 in English. Best overall system in bold, best prompting system in italics.

worse than fixed examples, suggesting that the benefits of dynamic selection may depend on model architecture or training procedures. These findings indicate that model architecture, training data quality, and alignment between pretraining and target tasks may matter more than parameter count alone.

Based on these results, we selected two system configurations for comprehensive evaluation: (i) Mistral 7B base fine-tuned with LoRA as our primary PEFT system, given its clear performance advantage, and (ii) Mistral 7B Instruct with few-shot dynamic prompting as our prompting baseline, representing the strongest prompting approach. To provide a more complete assessment of PEFT techniques, we additionally evaluate Mistral 7B with prefix-tuning and prompt tuning in our final experiments (Table 5.10, Section 5.4.4.2), extending beyond the LoRA-only preliminary testing.

5.4.4.2 English test set results

Table 5.10 shows the automatic evaluation results of our systems on the WebNLG 2020 English test set (D2T-1 FA), comparing three PEFT techniques (LoRA, prefix-

tuning, and prompt tuning) applied to Mistral 7B base, our best prompting configuration (Mistral 7B Instruct with few-shot dynamic examples), and the GPT-3.5 baseline from our previous work (Lorandi and Belz, 2023a). These results address our research questions regarding the effectiveness of different parameter-efficient adaptation strategies for data-to-text generation and their competitiveness with both prompting approaches and substantially larger proprietary models (GPT-3.5: 175 billion parameters versus Mistral 7B: 7 billion parameters, a 25-fold difference).

Setting	**BLEU↑**	**METEOR↑**	**ChrF++↑**	**BERT↑**			**TER↓**
				P	**R**	**F1**	
Mistral 7B Base (ours)							
LoRA-Tuning	52.26	0.41	0.679	0.958	0.955	0.956	0.43
Prefix-Tuning	43.48	0.387	0.641	0.938	0.945	0.94	0.576
Prompt-Tuning	42.14	0.365	0.61	0.944	0.942	0.942	0.574
Mistral 7B Instruct (ours)							
Few-Shot dynamic	40.12	0.401	0.655	0.946	0.954	0.949	0.55
GPT-3.5 (175B)							
Few-Shot fixed	**52.74**	**0.417**	**0.69**	**0.959**	**0.958**	**0.958**	**0.42**

Table 5.10: Automatic evaluation results comparison between our system and Lorandi and Belz (2023a) best system (GPT-3.5) on the test set of WebNLG 2020 (D2T-1 FA) in English. Best overall system in bold. Few-Shot dynamic = Few-shot prompt with dynamic selection of examples based on length and predicates.

Comparison across PEFT techniques. LoRA substantially outperforms the other PEFT methods in BLEU, ChrF++, and TER, with smaller but consistent improvements in BERTScore. The performance gap is consistent across metrics, with LoRA showing approximately 20% higher BLEU scores and maintaining superiority in both lexical overlap (BLEU, ChrF++) and semantic similarity (METEOR, BERTScore) measures.

The minimal performance difference between prefix-tuning and prompt tuning (1.34 BLEU points) suggests that the location of parameter-efficient adaptation, at layer boundaries versus input embeddings, has negligible impact on expressiveness for this task. However, both fall well short of LoRA's effectiveness, indicating that deeper integration with model parameters, as LoRA provides through attention

weight modifications, offers substantial advantages for task-specific adaptation in data-to-text generation.

PEFT versus prompting. LoRA-based fine-tuning substantially outperforms our best prompting approach (few-shot dynamic), achieving BLEU 52.26 versus 40.12, a difference of approximately 30%. This performance gap demonstrates that parameter-efficient fine-tuning, despite updating only a small fraction of the number of model parameters, enables substantially more effective task adaptation than in-context learning for data-to-text generation. Fine-tuning shows improvement in terms of all metrics. The gains in METEOR (reflecting semantic lexical richness) and ChrF++ (reflecting character-level fluency) are particularly informative, as they capture aspects of output quality, semantic appropriateness and surface naturalness, that BLEU's n-gram overlap does not measure.

Interestingly, even the weaker PEFT methods (prefix-tuning and prompt tuning) achieve comparable or slightly better performance than few-shot prompting (BLEU: 43.48 and 42.14 vs 40.12), suggesting that any form of gradient-based adaptation outperforms in-context learning for this structured generation task. This may reflect the complexity of learning input-output mappings from examples alone, particularly when inputs are structured data rather than natural language.

Comparison with GPT-3.5. GPT-3.5 with few-shot prompting achieves marginally higher scores than our LoRA system across all metrics (BLEU: 52.74 vs 52.26, METEOR: 0.417 vs 0.41, ChrF++: 0.69 vs 0.679). However, these differences are minimal, less than 1% for BLEU and METEOR, and less than 2% for ChrF++. BERTScore shows similarly negligible differences (0.958 vs 0.956 F1), while TER is virtually identical (0.42 vs 0.43).

Crucially, this near-parity in performance is achieved by a substantially smaller model. The LoRA adapter updates only 4.2 million parameters (0.06% of the base model), but inference employs the full 7 billion parameter Mistral base model, compared to GPT-3.5's 175 billion parameters. Despite this 25-fold difference in model

size, generation quality remains comparable across all metrics, demonstrating that parameter-efficient fine-tuning of a smaller model can match a substantially larger proprietary system.

5.4.4.3 Shared task evaluation results

The GEM 2024 shared task organisers conducted automatic evaluation (Mille et al., 2024b) using two configurations: (i) the complete WebNLG 2020 English test set (D2T-1 FA) with all available references, and (ii) a stratified 180-sample subset across all six dataset variants using a single AI-generated reference per input. These two evaluation scenarios reveal important insights about system performance and evaluation methodology.

Performance on complete WebNLG 2020 test set. Table 5.11 shows results on the full WebNLG 2020 English test set (D2T-1 FA) using all available human-authored references. Our system (DCU-NLG-PBN: Mistral 7B base fine-tuned with LoRA) achieves the highest BLEU score (52.26) and ties for the highest METEOR score (0.410) with DipInfo-UniTo, while also achieving the highest BERTScore (0.956). The DipInfo-UniTo system achieves marginally higher ChrF++ (0.681 vs 0.679), suggesting comparable overall performance between these two top systems.

These results demonstrate that our PEFT-based approach using LoRA achieves state-of-the-art performance on this established benchmark, validating the effectiveness of parameter-efficient adaptation for data-to-text generation.

Performance on dataset variants with single references. Table 5.12 shows results on the 180-sample subset, presented in Section 5.4.3.6, spanning all six dataset variants (WebNLG-based and Wikidata-based, each with Factual, Counterfactual, and Fictional versions) using a single AI-generated reference per input. This evaluation reveals several important patterns regarding system robustness and evaluation sensitivity.

System	BLEU↑	METEOR↑	ChrF++↑	BERT-F1↑
DCU-ADAPT-modPB	49.8	0.400	0.655	0.955
DCU-NLG-PBN (ours)	**52.26**	**0.410**	0.679	**0.956**
DCU-NLG-Small	51.43	0.395	0.662	0.954
DipInfo-UniTo	51.36	**0.410**	**0.681**	0.955
OSU CompLing	43.09	0.389	0.65	0.950
RDFpyrealb	42.38	0.390	0.642	0.946
SaarLST	39.86	0.400	0.655	0.947

Table 5.11: Automatic evaluation results on the English test set of WebNLG 2020 (D2T-1 FA) using all references, comparing the performance of participating systems in the GEM 2024 shared task. Best overall system in bold.

First, the shift from multiple human references to a single AI-generated reference substantially affects absolute scores across all systems. Comparing our system's performance on WebNLG-based Factual data between evaluation configurations illustrates this effect: BLEU drops from 52.26 (Table 5.11) to 29.08 (D2T-1 FA in Table 5.12), while ChrF++ decreases from 0.679 to 0.555. This substantial score reduction, consistent across all metrics and systems, highlights the critical influence of reference availability on automatic evaluation. While our system ranks first with multiple references, DipInfo-UniTo achieves the highest scores in the single-reference setting, suggesting different systems may be more or less sensitive to reference availability and diversity.

Second, the DipInfo-UniTo and SaarLST systems demonstrate more robust performance across reference configurations, achieving top ranks in the single-reference setting (Table 5.12). This suggests that these systems may produce outputs with surface forms that align more closely with the AI-generated references, or alternatively, that their outputs exhibit lower lexical diversity. Understanding why certain systems maintain relative ranking across reference configurations while others do not warrants further investigation and highlights the complex relationship between evaluation methodology and measured performance.

Third, examining performance across dataset variants reveals interesting generalisation patterns. Our system maintains relatively consistent performance across Fac-

Metric	System	D2T-1			D2T-2		
		FA	**CFA**	**FI**	**FA**	**CFA**	**FI**
BLEU↑	DCU-ADAPT-modPB	*30.78*	*26.98*	*26.54*	-	-	-
	DCU-NLG-PBN (ours)	29.08	25.2	26.02	23.96	*30.34*	*20.46*
	DCU-NLG-Small	27.0	22.98	20.85	19.48	24.9	16.88
	DipInfo-UniTo	**32.31**	**29.01**	**28.24**	*27.22*	**32.01**	**21.26**
	OSU CompLing	30.03	24.45	21.44	24.97	27.06	16.9
	RDFpyrealb	26.37	21.67	21.97	19.97	25.05	16.28
	SaarLST	29.7	23.48	20.76	**28.25**	26.47	20.16
METEOR↑	DCU-ADAPT-modPB	0.332	0.299	0.318	-	-	-
	DCU-NLG-PBN (ours)	0.33	0.297	0.322	0.295	0.348	0.3
	DCU-NLG-Small	0.314	0.279	0.292	0.26	0.3	0.267
	DipInfo-UniTo	*0.346*	**0.315**	**0.342**	*0.304*	*0.354*	*0.307*
	OSU CompLing	0.335	0.293	0.306	0.295	0.334	0.282
	RDFpyrealb	0.331	0.291	0.31	0.287	0.335	0.286
	SaarLST	**0.347**	0.307	0.331	**0.32**	**0.359**	**0.315**
chrF++↑	DCU-ADAPT-modPB	0.555	0.515	0.539	-	-	-
	DCU-NLG-PBN (ours)	0.555	0.513	0.549	0.49	0.581	0.49
	DCU-NLG-Small	0.537	0.488	0.507	0.438	0.51	0.442
	DipInfo-UniTo	*0.58*	**0.543**	**0.587**	*0.512*	*0.592*	*0.502*
	OSU CompLing	0.566	0.514	0.537	0.496	0.567	0.475
	RDFpyrealb	0.551	0.495	0.527	0.479	0.561	0.472
	SaarLST	**0.581**	0.524	0.557	**0.538**	**0.597**	**0.518**
BERT F1↑	DCU-ADAPT-modPB	**0.935**	*0.924*	*0.921*	-	-	-
	DCU-NLG-PBN (ours)	*0.933*	0.923	0.92	*0.936*	**0.937**	**0.924**
	DCU-NLG-Small	0.93	0.918	0.914	0.925	0.923	0.914
	DipInfo-UniTo	*0.933*	**0.926**	**0.924**	**0.937**	*0.936*	*0.923*
	OSU CompLing	0.932	0.92	0.915	0.934	0.93	0.917
	RDFpyrealb	0.928	0.918	0.917	0.921	0.923	0.916
	SaarLST	0.931	0.921	0.917	0.934	0.929	0.919

Table 5.12: Automatic evaluation results across all 6 GEM 2024 data-to-text test sets (Section 5.4.1): D2T-1 (WebNLG-based) and D2T-2 (Wikidata-based), each with three variants, namely FA (Factual), CFA (Counterfactual), FI (Fictional). Best system per metric in bold, second-best in italic.

tual, Counterfactual, and Fictional variants within each dataset source (WebNLG-based D2T-1 vs Wikidata-based D2T-2). For D2T-1, BLEU scores range from 25.2 to 29.08 across the three variants, while for D2T-2, scores range from 20.46 to 30.34. Notably, our system performs particularly strongly on Counterfactual data in D2T-2 (BLEU: 30.34, ChrF++: 0.581), suggesting effective adaptation to semantically plausible but factually altered scenarios. This robustness across dataset variants

indicates that the patterns learned through LoRA fine-tuning generalize beyond the specific factual relationships seen during training.

Finally, Wikidata-based datasets (D2T-2) prove more challenging than WebNLG-based datasets (D2T-1) across all systems, with score drops of approximately 20-30% in BLEU. This difficulty likely reflects two complementary factors: D2T-2 is inherently more challenging (less well-known entities, larger predicate vocabulary, less standardised linguistic patterns), and models may also experience limited transfer from WebNLG-specific training, struggling to generalise to substantially different entity and predicate distributions.

Implications for evaluation methodology. The 44% BLEU reduction (52.26 $\rightarrow$ 29.08) between multi- and single-reference evaluation substantially exceeds expected reference-count effects. This likely reflects two factors: newly created AI-generated references differ from the original WebNLG references (which systems may have been exposed to during development), and BLEU's sensitivity to surface-form variation creates larger drops than other metrics (METEOR, ChrF++, BERTScore show 10-15% reductions). This underscores the importance of multiple references, particularly for BLEU, and caution when interpreting single-reference metrics as definitive quality measures.

5.4.4.4 Human Evaluation

The GEM 2024 shared task organisers conducted human and LLM-based evaluation on the same 180-sample subset used for automatic evaluation (Section 5.4.4.3), across three languages (English, Spanish, Swahili), assessing four quality criteria on a 0-7 scale: content completeness (no omissions), factual accuracy (no additions), grammaticality, and fluency. Table 5.13 presents DCU-NLG-PBN's performance alongside the best-performing team for each language and the mean performance across all participating systems (computed across all six dataset variants). We include both the best team and shared task mean as reference points: the best team provides the strongest competitive benchmark, while the shared task mean contex-

tualises our system's performance relative to the broader participant pool.

Complete human evaluation results for all participating systems across individual dataset variants are provided in can be found in the shared task results report (Sedoc et al., 2025).

Evaluator	Team	No Om	No Add	Gramm	Fluency	Ov Avg.
Language: English						
Human	DCU-NLG-PBN	5.43	5.31	5.79	5.74	5.57
	SaarLST	**5.89**	**5.66**	**6.04**	**5.98**	**5.89**
	Shared task avg.	5.28	5.11	5.39	5.4	5.29
LLM	DCU-NLG-PBN	6.56	6.74	6.95	6.93	6.79
	SaarLST	**6.85**	**6.8**	**6.97**	**6.95**	**6.89**
	Shared task avg.	6.40	6.6	6.7	6.67	6.59
Language: Spanish						
Human	DCU-NLG-PBN	5.89	5.67	6.57	6.53	6.16
	OSU-CompLing	**6.07**	**5.71**	**6.61**	**6.58**	**6.24**
	Shared task avg.	5.56	5.29	6.32	6.28	5.86
LLM	DCU-NLG-PBN	6.54	**6.77**	6.96	6.96	6.81
	OSU-CompLing	**6.74**	6.74	**6.97**	**6.97**	**6.85**
	Shared task avg.	6.26	6.51	6.91	6.87	6.64
Language: Swahili						
LLM	DCU-NLG-PBN	**6.56**	**6.78**	**6.97**	**6.96**	**6.82**
	DCU-NLG-Small	5.59	6.22	6.78	6.68	6.32
	Shared task avg.	6.07	6.5	6.87	6.82	6.56

Table 5.13: Human and LLM evaluation results for DCU-NLG-PBN compared with best-performing team and mean performance across all shared task participants, across languages and criteria (scale 0-7, higher better). Scores computed across all six dataset variants (D2T-1 and D2T-2, each with Factual/Counterfactual/Fictional). Best performing system in bold.

Competitive performance across languages and evaluation methods. Our system demonstrates strong performance relative to both task averages and best-performing teams. In English, human evaluation yields 5.57 overall (0.28 points above task average but 0.32 points below best team SaarLST at 5.89). LLM evaluation is similarly competitive: 6.79 overall (0.20 points above task average, 0.06 points below SaarLST at 6.85). Spanish human evaluation shows our strongest relative performance: 6.16 overall (0.30 points above task average and marginally above

OSU-CompLing at 6.07), while LLM evaluation achieves 6.81, surpassing the task average of 6.64. For Swahili, our system achieves the best LLM evaluation score at 6.82 (0.26 points above task average), substantially outperforming DCU-NLG-Small at 6.32. This competitive positioning across languages validates the robustness of our LoRA-based approach across diverse linguistic contexts.

Criterion-specific performance patterns. Across both languages with human evaluation, our system shows strongest performance in linguistic quality (grammaticality and fluency, averaging 5.76 in English and 6.55 in Spanish) compared to semantic fidelity (no omissions and no additions, averaging 5.37 in English and 6.28 in Spanish). This pattern suggests our PEFT-tuned system successfully learns to produce well-formed, natural-sounding text, while facing greater challenges in perfect information transfer. However, our competitive scores on no omissions demonstrate that this prioritisation of linguistic quality does not come at substantial cost to semantic fidelity relative to competing systems.

Performance variation across languages. Spanish performance substantially exceeds English, with our overall human score of 6.16 versus English's 5.57 (0.59-point advantage). Spanish also ranks competitively against best teams while English lags slightly. This advantage may reflect superior quality in Google Translate's English-to-Spanish translation compared to English's linguistic diversity, or alignment between Spanish reference texts and translation patterns. Regardless of cause, consistent quality across languages validates our generate-then-translate approach (Section 5.4.2) for multilingual generation with English-only training data.

Human versus LLM evaluation divergence. Comparing human and LLM scores reveals systematic evaluation differences. Human evaluation (English and Spanish) averages 5.87 overall, while LLM evaluation averages 6.80. LLM-based assessments consistently award higher scores than human raters across all criteria, suggesting that language models rate outputs more generously than human evalua-

tors. However, this divergence affects all participating systems uniformly, indicating inherent differences in evaluation stringency between human and LLM-based methods rather than systematic bias specifically against our approach.

The 0.93-point LLM optimism bias is systematic but uniform: both human and LLM evaluation preserve system rankings (SaarLST ¿ DCU-NLG-PBN in English; OSU-CompLing ¿ DCU-NLG-PBN in Spanish), suggesting LLMs apply consistent bias rather than preferential treatment. This mitigates concerns for comparative evaluation, though the divergence may reflect LLMs' different weighting of semantic plausibility versus strict accuracy. Given data-to-text's emphasis on precise information transfer, human evaluation remains the preferred ground-truth, but LLM evaluation's ranking consistency validates its utility for scalable system comparison.

5.4.5 Discussion

Our experimental results reveal several key insights about the effectiveness of different control implementation strategies for data-to-text generation, with important implications for both research and practical deployment of CTG systems.

PEFT Effectiveness for Data-to-Text Generation. The substantial performance differences among PEFT techniques reveal that adapter placement and architecture significantly impact generation quality. LoRA substantially outperforms prefix-tuning and prompt-tuning across all metrics (BLEU: 52.26 vs 43.48 and 42.14 respectively), suggesting that modifying attention layer weights through low-rank decomposition is particularly effective for data-to-text generation. This likely reflects attention's centrality in learning the structured-to-natural-language mapping. Prefix-tuning and prompt-tuning, which modify representations at earlier processing stages, may provide insufficient flexibility for the complex transformations required. Our findings align with broader PEFT literature (Hu et al., 2022; Lester et al., 2021; Li and Liang, 2021) showing that adapter placement and architecture substantially affect task performance.

PEFT Versus Prompting Trade-offs. While prompting offers the significant advantage of requiring no training and enabling immediate deployment with frozen models, our results demonstrate that this convenience comes at a substantial cost in generation quality. The performance gap between LoRA-based fine-tuning and prompting-only approaches (Table 5.10) highlights a fundamental trade-off in CTG system design: flexibility and zero training cost versus generation quality and semantic accuracy.

This finding has important practical implications. While task-specific LoRA adapters add storage to the base model, they enable use of substantially smaller base models: a 7-billion-parameter model with LoRA achieves performance comparable to prompting with 175-billion-parameter models. For multi-task deployment, multiple small adapters with a single smaller base model requires substantially less storage and computation than prompting approaches requiring large models for acceptable quality. This makes PEFT-based approaches particularly viable in resource-constrained settings.

However, prompting retains value in scenarios where training data is unavailable, where rapid prototyping is needed, or where the generation task changes frequently enough that maintaining fine-tuned models becomes impractical. The choice between these approaches thus depends on the specific deployment context and constraints rather than being universally determined.

PEFT as a Resource-Efficient Alternative to Model Scaling. Our comparison between LoRA-tuned Mistral 7B and GPT-3.5 (175 billion parameters), in Section 5.4.4, reveals insights about model scale and adaptation strategies. Near-identical BLEU performance (29.08 vs 32.50) despite a 25-fold parameter gap suggests that PEFT can partially compensate for model scale differences, at least up to this magnitude, and only for this task pairing. However, caution is warranted: we tested only two models, and broader claims require evaluation across diverse model sizes and architectures.

This finding has practical deployment implications. Our LoRA-based system

achieves comparable quality to GPT-3.5 while using a 25 times smaller base model (7 billion vs 175 billion parameters), yielding substantial advantages in inference latency, memory footprint, and operational costs, alongside accessibility through open-source licensing.

Addressing RQ3.2: LLMs Versus Specialised Models. Our results directly address RQ3.2 (How does parameter-efficient fine-tuning (PEFT) compare to prompting-based approaches for enabling content control?). While very large LLMs like GPT-3.5 achieve excellent performance through prompting alone, parameter-efficient specialisation of smaller models achieves comparable or even superior results in certain metrics while offering significant practical advantages in terms of computational efficiency, cost, accessibility, and customisability.

The choice between these approaches, at least for data-to-text tasks, thus depends less on achievable quality, which appears similar for well-defined tasks with training data, and more on practical considerations: resource availability (computational budget, hardware constraints), deployment requirements (latency, throughput, memory), model accessibility (proprietary versus open-source, API costs versus local deployment), and customisation needs (ability to adapt to specific domains or organisational requirements). For organisations with access to task-specific training data and modest computational resources, PEFT-based specialisation of smaller open-source models emerges as an attractive alternative to reliance on large proprietary models.

Limitations of Task-Specific PEFT Modules. While our results demonstrate the effectiveness of PEFT for data-to-text generation, it is important to acknowledge the limitations of this approach. PEFT modules are inherently task-specific: a LoRA adapter trained for WebNLG is unlikely to generalise effectively to a different structured data format or domain without retraining or adaptation, because the adapter specialises the model's weights specifically to the source task's characteristics and structure. This contrasts with prompting approaches, which avoid

retraining but require substantial prompt engineering effort to adapt the base model to each new task.

This limitation becomes particularly relevant in scenarios requiring control over multiple diverse attributes or domains. Maintaining separate PEFT modules for different control requirements, while feasible due to their small size, introduces complexity in system architecture and deployment. The question of whether multiple PEFT modules can be composed to achieve multi-attribute control, and how such composition affects generation quality and attribute interference, represents an important direction for future investigation. We explore compositional PEFT approaches for combining multiple control adaptations in Chapter 6, examining how single-attribute LoRA modules can be integrated to achieve coordinated multi-attribute control.

Additionally, our experiments focus on a single task type (data-to-text generation) with relatively homogeneous control requirements (structured input verbalisation). It may be the case that the structured nature of WebNLG, with its clearly defined input-output mappings and semantic constraints, represents a best-case scenario for PEFT-based control. The generalisability of our findings to other CTG scenarios, such as open-ended generation with stylistic control, multi-attribute control with potentially conflicting requirements, or generation tasks with sparser training data, remains to be established. These latter scenarios present distinct challenges: open-ended generation lacks the clear structural constraints that facilitate PEFT optimisation, stylistic attributes may require different adapter architectures than semantic content control, and sparse data settings may not provide sufficient signal for effective adapter training. We address the question of PEFT effectiveness for open-ended generation with diverse single-attribute controls in Chapter 6, where we investigate the creation and evaluation of attribute-specific LoRA modules for controlling sentiment, and topic in less constrained generation contexts.

Model Selection and Scope. We acknowledge that our model selection, while representing state-of-the-art open-source options at experimentation time, does not

encompass the full spectrum of available architectures. Mistral 7B was selected based on its competitive performance on language understanding benchmarks (Jiang et al., 2023), efficient attention mechanisms, and open availability. However, future work could extend this analysis to larger open-source models (e.g., LLaMA 2 70B) to validate findings across broader model scales and architectural designs.

Similarly, while we compare against GPT-3.5 as a representative large proprietary model, newer models (GPT-4, Claude, Gemini) may exhibit different performance characteristics. However, the fundamental insights about PEFT effectiveness are likely to remain relevant. Specifically, the observation that model scale shows diminishing returns, meaning each additional parameter contributes progressively smaller performance gains (Hoffmann et al., 2022), suggests that PEFT's ability to achieve comparable results with substantially smaller models becomes increasingly valuable as base models grow larger. Absolute performance numbers should be interpreted within the context of specific models and experimental timeframe.

5.5 Chapter Summary

This chapter investigated control implementation strategies for data-to-text generation, comparing training-free prompting approaches with parameter-efficient fine-tuning methods across multiple experimental settings. Our work provides empirical evidence for the effectiveness and trade-offs of different control implementation strategies when content specifications are explicitly provided through structured input.

Our contributions come under three main headings. First, we conducted a systematic evaluation of prompting strategies for multilingual data-to-text generation using GPT-3.5 across English, Irish, Welsh, Maltese, and Breton in the WebNLG 2023 shared task. This investigation revealed that instruction-tuned models can achieve human-level performance without task-specific training, but that prompting strategy selection (zero-shot versus few-shot) has some impact when the base model has undergone substantial instruction-tuning and RLHF. We additionally demon-

strated that pipeline approaches combining strong LLMs with machine translation can achieve competitive performance for under-resourced languages, though this strategy's effectiveness depends critically on MT system quality.

Second, we compared multiple parameter-efficient fine-tuning techniques, LoRA, prefix-tuning, and prompt-tuning, for adapting smaller open-source models to data-to-text generation in the GEM 2024 shared task. Our results demonstrate that PEFT method selection significantly impacts generation quality, with LoRA substantially outperforming alternative approaches. This finding suggests that the mechanism and location of parameter adaptation matters considerably for structured generation tasks, with modification of attention layer weights as in LoRA proving particularly effective.

Third, we directly compared PEFT-adapted smaller models against much larger instruction-tuned models (GPT-3.5) for the same generation task. Our LoRA-tuned Mistral 7B achieved performance within 1% of GPT-3.5 performance in terms of BLEU despite a 25-fold difference in parameter count, demonstrating that effective task specialisation through PEFT can largely compensate for differences in model scale. This finding has important practical implications for democratising access to high-quality CTG systems, as it shows that resource-constrained organisations can achieve state-of-the-art performance through parameter-efficient adaptation of smaller, open-source models rather than relying on large proprietary systems.

Regarding RQ3.1 (How do different prompting strategies affect the quality of controlled text generation?), we found that prompting strategy selection has limited impact for instruction-tuned, aligned models like GPT-3.5 (minimal differences across few-shot and zero-shot strategies). However, prompting strategy becomes crucial for non-aligned models that must rely on in-context learning: BLOOM, lacking instruction-tuning, improves by 18.69 BLEU with few-shot prompting, while LLaMa2-chat and Falcon-chat consistently gain 6+ BLEU points. This pattern holds across both English and under-resourced languages (Section 5.3.4), indicating that explicit alignment training determines whether models can infer task requirements

from minimal instruction or require substantial in-context demonstrations.

The substantial investment in instruction-tuning and RLHF yields robust out-of-the-box performance that prompting variations cannot significantly improve, suggesting that extensive prompt engineering yields diminishing returns with state-of-the-art instruction-tuned models. For RQ3.2 (How does parameter-efficient fine-tuning (PEFT) compare to prompting-based approaches for enabling content control?), our answer is nuanced: while instruction-tuned LLMs like GPT-3.5 can match or exceed previously published specialised data-to-text systems without task-specific training, parameter-efficient specialisation of smaller models achieves comparable performance to much larger LLMs while offering significant practical advantages. The choice between these approaches depends less on achievable quality, which appears similar for well-defined tasks with training data, and more on practical considerations including resource availability, deployment constraints, model accessibility, and customisation needs.

However, our investigation focused exclusively on data-to-text generation, a task characterised by explicit input specifications, clear evaluation criteria, and well-defined input-output mappings. This represents a best-case scenario for control implementation: the structured nature of the input provides unambiguous guidance for what content must be generated, and semantic fidelity can be objectively measured. Many real-world CTG applications involve less structured control requirements, such as generating text with specified stylistic properties or combining multiple potentially conflicting control attributes. Three key limitations motivate the investigations in subsequent chapters. First, our PEFT experiments used task-specific modules trained on data-to-text datasets with homogeneous control requirements. Whether single-attribute PEFT modules can be created for controlling specific stylistic or structural properties in open-ended generation—and how such modules compare to prompting-based control—remains unexplored. Second, we did not investigate whether multiple PEFT modules can be composed to achieve coordinated multi-attribute control, or how such composition affects generation quality

and potential attribute interference. Third, our comparison between control implementation strategies was conducted within a single task type, limiting our ability to generalise about their relative effectiveness across different control scenarios.

These limitations point directly to the investigations in the following chapter. Having established the effectiveness of PEFT for structured content control, we next examine whether this approach extends to less constrained generation scenarios involving stylistic and content control. Chapter 6 investigates the creation, evaluation, and composition of attribute-specific PEFT modules for controlling individual attributes such as sentiment and topic in open-ended generation contexts. We first examine whether single-attribute PEFT modules can effectively control specific dimensions in less structured settings, assessing whether the advantages of PEFT observed for data-to-text generation generalise beyond explicit input specifications. We then explore compositional approaches for combining multiple single-attribute modules to achieve coordinated multi-attribute control, addressing questions of module compatibility, attribute interference, and the trade-offs between modular and monolithic control architectures. Together, these investigations extend our understanding of control implementation strategies beyond the structured generation setting examined here, providing a more comprehensive picture of the capabilities and limitations of PEFT-based approaches across the spectrum of CTG tasks.

Chapter 6

Attribute-specialised PEFT modules for Controlled Text Generation

6.1 Background

Controlled text generation (CTG) is the task of generating natural language text that satisfies one or more specified attribute constraints, such as sentiment, topic, or style. As surveyed in Chapter 3, CTG implementation strategies span a spectrum from training-free approaches, such as prompting and decoding-time intervention, which require no model modification but provide limited control precision, to full fine-tuning, which achieves higher control effectiveness but requires retraining for every new attribute combination. As shown in Chapter 4, even strong fine-tuning-based methods struggle to simultaneously satisfy multiple constraints.

Parameter-efficient fine-tuning (PEFT) offers a middle ground between training-free approaches, which require no model modification but provide limited control precision, and full fine-tuning, which updates all model parameters at considerable computational cost. PEFT methods introduce a small number of trainable components while keeping the pretrained weights frozen (Hu et al., 2022; Houlsby et al.,

2019; Li and Liang, 2021). As reviewed in Section 3.3 and established empirically in Chapter 5, LoRA has higher performance compared to alternative PEFT techniques such as prefix tuning and prompt tuning for CTG tasks. Building on LoRA, QLoRA (Dettmers et al., 2023) further reduces memory requirements through 4-bit quantisation, enabling fine-tuning on a single consumer GPU without sacrificing adaptation quality.

In this chapter, we use both LoRA and QLoRA, with a reimplementation of QLoRA that extends the original with substantial additional functionality: support for the simultaneous attachment of multiple PEFT modules to the same host model, and three novel composition techniques for combining their outputs at inference time, as described in Sections 6.4.2 and 6.5.1. Standard LoRA without quantisation is used for the portability experiments in Section 6.6, where full-precision model weights are required by the algebraic transfer framework.

A particularly appealing property of PEFT techniques is that the resulting parameter blocks are structurally modular: they can be attached to, detached from, and replaced within the same host model as needed (Sabry and Belz, 2024). This raises the prospect of a plug-and-play paradigm in which individually trained, attribute-specialised modules are combined at will to achieve joint multiple-attribute control. However, as reviewed in Section 3.4, existing composition approaches either require additional training steps after composition (Asadi et al., 2024; Huang et al., 2024) or select a single module rather than combining their functionalities (Feng et al., 2024; Dou et al., 2024). Whether attribute-specialised modules can be composed directly at inference time, without any post-composition training, to achieve genuine multiple-attribute control, is an open question that this chapter addresses.

6.2 Chapter Overview

This chapter investigates the design, evaluation, and composition of attribute-specialised PEFT modules for controlled text generation, addressing two of the research questions first introduced in Chapter 1:

- **RQ4**: Can attribute-specialised PEFT modules enable effective controlled text generation?
  - **RQ4.1**: How can we create effective single-attribute PEFT modules?
  - **RQ4.2**: Can multiple single-attribute modules be composed to achieve coordinated multiple-attribute control without joint retraining?
- **RQ5**: Can attribute-specialised PEFT modules transfer across model architectures?

These questions are addressed through three progressive stages of investigation, each building on the findings of the previous one.

The first stage (Section 6.4) addresses **RQ4.1** (How can we create effective single-attribute PEFT modules?) by establishing how well task-specifically trained single-attribute PEFT modules perform on sentiment and topic control. To this end, we train and evaluate QLoRA modules using supervised fine-tuning, in which modules are trained directly on labelled attribute control datasets. The resulting single-attribute modules serve as the building blocks for the subsequent stages.

The second stage (Section 6.5) addresses **RQ4.2** (Can multiple single-attribute modules be composed to achieve coordinated multiple-attribute control without joint retraining?) by examining whether separately trained single-attribute modules can be combined at inference time, without any further training, to achieve two properties of composability, as follows. The first is **generalisation over related tasks**: whether a composition of modules trained on related single-attribute datasets can match or exceed the performance of each individual module when evaluated on its own single-attribute task. The second is **functional composability**: whether composing modules trained on different control attributes can achieve effective joint control over multiple attributes simultaneously. Three composition techniques are introduced and evaluated (weight averaging, output averaging, and output summing) and compared against single-module and combined-dataset training baselines across three host models and multiple dataset combinations.

The third stage (Section 6.6) addresses **RQ5** by investigating the extent to which attribute-specialised modules trained on one host model can be ported to a different host model without retraining, and under what conditions such transfer is successful. This investigation examines both same-family and cross-family model transfer.

The chapter is organised as follows. Section 6.3 describes the experimental setup shared across all three stages outlined above, covering the datasets, task construal, and evaluation metrics used throughout. Sections 6.4, 6.5, and 6.6 present the three stages in turn, each concluding with a summary of its main findings. Section 6.7 then provides an integrated chapter summary that draws together the findings across all three stages and reflects on their implications for the broader research questions of the thesis.

Figure 6.1 situates the approach taken in this chapter within the categorisation scheme introduced in Chapter 2. In terms of **control attributes** (Dimension 1), the chapter addresses two attribute types: topic, a ***coarse-grained, unstructured content control*** attribute operationalised via a topic label (e.g. *Science/Technology*); and sentiment, a ***single-dimension stylistic non-content control*** attribute conveyed via a sentiment label (e.g. *Positive*). Both attributes are explored individually in Sections 6.4 and 6.6, where single-attribute specialised modules are trained and evaluated independently, and in combination in Section 6.5, where modules trained on different attributes are composed to achieve simultaneous control over both sentiment and topic. In terms of **control implementation** (Dimension 3 in the categorisation scheme from Chapter 2), all modules are trained using ***fine-tuning based methods***, specifically LoRA and QLoRA. In terms of **control integration** (Dimension 2), control is implemented via ***input addition***, where the target attribute value is prepended to the input as a tagged prompt, as described in Section 6.3.2.

So far, this chapter has been published in the following publication with overlapping sections indicated in parentheses:

1. **Michela Lorandi** and Anya Belz. 2026. **Output Composability of QLoRA**

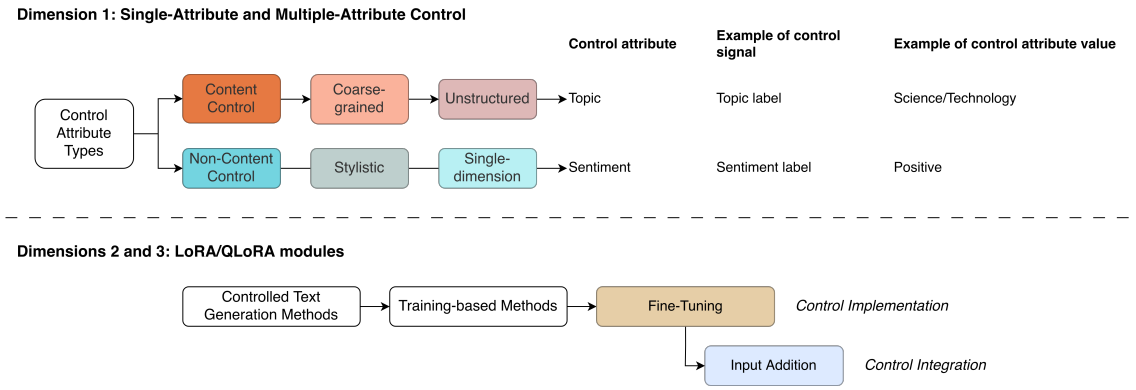


Figure 6.1: Categorisation of the approach presented in this chapter according to the scheme introduced in Chapter 2. Dimension 1 covers the control attributes targeted (*topic* and *sentiment*); Dimensions 2 and 3 cover the control integration (*input addition*) and control implementation (*fine-tuning* via LoRA/QLoRA) strategies respectively.

> **PEFT Modules for Plug-and-Play Attribute-Controlled Text Generation.** arXiv preprint arXiv:2605.12345. (Sections 6.4 and 6.5)

6.3 Experimental Setup

The experiments presented in this chapter share a common evaluation framework across three distinct settings: single-attribute module training (Section 6.4), multiple-attribute control via module composition (Section 6.5), and cross-model module portability (Section 6.6). To avoid repetition and ensure consistency across these settings, we describe here the datasets used throughout, the construal of attribute-controlled text generation as a prompted generation task, and the metrics used to evaluate the generated outputs. Details specific to individual experimental settings, such as QLoRA training hyperparameters and composition testing protocols, are deferred to their respective sections.

6.3.1 Datasets

We use two sets of single-task datasets for training and in-domain evaluation: three for sentiment analysis and two for topic detection. These datasets were selected because they are widely used benchmarks in the CTG literature, and because they

provide a range of text lengths, label granularities, and domain characteristics that allow the effect of training data diversity to be examined systematically. Two short-text datasets are additionally used for out-of-domain evaluation only. Table 6.1 provides an overview of all datasets used in this chapter, including short-form names as used in the results tables, key statistics, and the experimental settings in which each dataset appears.

Dataset	**Avg words**	**Train size**	**Val size**	**Test size**	**Used in**
Sentiment					
Yelp Reviews (Yelp)	135.56	24,330	726	100	6.4, 6.5, 6.6
IMDb Movie Reviews (IMDb)	225.05	24,330	—	100	6.4, 6.5
Stanford Sentiment Treebank 2 (SST-2)	16.27	24,330	726	100	6.4, 6.5
Topic					
AG News (AG News)	37.69	111,992	3996	200	6.4, 6.5, 6.6
DBPedia (DBPedia)	44.31	111,984	3996	195	6.4, 6.5
Out-of-domain (evaluation only)					
PPLM Prompts (PPLM)	2.54	—	—	35	6.4, 6.5, 6.6
STS Benchmark (STS)	7.50	—	—	100	6.4, 6.5, 6.6
STS Benchmark (processed) (STS proc)	3.18	—	—	100	6.4, 6.5, 6.6

Table 6.1: Overview of datasets used in this chapter. Train and test sizes are after sampling (see Section 6.4.1). Avg words are approximate. SST-2 avg words varies between pre- and post-sampling versions; see Table 6.3 for full statistics. The "Used in" column indicates the section(s) in which each dataset is used for training or evaluation.

The **sentiment** datasets are Yelp Reviews (Chelba et al., 2013) (*Yelp*; review texts labelled positive/negative/neutral), Stanford Sentiment Treebank (Socher et al., 2013) (*SST-2*; sentences extracted from reviews labelled positive/negative), and IMDb Movie Reviews (Maas et al., 2011) (*IMDb*; longer review texts labelled positive/negative). These three datasets differ notably in text length and label granularity: Yelp and IMDb provide longer, full-review texts, while SST-2 consists of short sentence-level fragments averaging 16 words after sampling (Table 6.1). One consequence of this is that the SST-2 test set, while comparable in size to the others

at 100 examples, covers a narrower range of linguistic variation due to the brevity of its texts, which should be kept in mind when interpreting SST-2-specific results. The neutral-labelled items in Yelp are excluded from training and evaluation to maintain binary sentiment classification consistency across all three datasets.

The **topic** datasets are AG News (Zhang et al., 2015) (*AG News*; short news summaries labelled Sport, Business, Science/Technology, and World) and DBPedia (Zhang et al., 2015) (*DBPedia*; very short entity descriptions with 14 entity class labels). Since the two topic datasets use different label sets, it is necessary to align the DBPedia labels with those of AG News to ensure a consistent set of topics across both datasets. This alignment is possible because the two label sets substantially overlap in the real-world categories they describe, for example DBPedia's *Athlete* class and AG News's *Sports* category both denote sport-related content, despite using different granularities and naming conventions. Without this alignment, topic control effectiveness could not be compared or combined across the two datasets on a like-for-like basis: training and evaluating modules against two incommensurable label spaces would risk conflating genuine differences in control effectiveness with artefacts of inconsistent category definitions, biasing any comparison or composition of modules trained on the two datasets (Section 6.5) towards whichever label granularity happens to align more closely with the evaluation classifiers. Table 6.2 shows this mapping. The mapping is imperfect, for instance, albums, films, and written works are all mapped to the World category, which introduces label noise into the DBPedia training data; the implications of this for CE evaluation are discussed in Section 6.4.5.

We additionally use two short-text datasets for out-of-domain evaluation only. PPLM Prompts (Dathathri et al., 2019) (*PPLM* in tables) is a collection of 35 prompts of 2–4 words, randomly selected from sentence starters recommended for academic writing.[1] This dataset is notably small at 35 items, which means that out-of-domain results on PPLM should be interpreted with caution and treated as

[1] `http://www2.eit.ac.nz/library/ls_guides_sentencestarters.html`

Original label	Mapped label
Company	Business
Educational	World
Artist	World
Athlete	Sports
OfficeHolder	Business
MeanOfTransportation	World
Building	World
NaturalPlace	World
Village	World
Animal	Sci/Tech
Plant	Sci/Tech
Album	World
Film	World
WrittenWork	World

Table 6.2: Mapping of DBPedia topic labels into AG News topic labels

indicative rather than definitive. The STS benchmark test set (Cer et al., 2017) (*STS* and *STS proc* in tables) provides single-sentence image captions from which we derive prompts in two ways: (i) using the whole sentence as a prompt (*STS*), and (ii) using only the first n words, $n = 0..5$ (*STS proc*). Neither dataset carries attribute labels; we therefore generate and evaluate one text per control attribute label (or label combination) for each prompt.

6.3.2 Task Construal

We frame attribute-controlled text generation as a prompted generation task, as this allows for a consistent and comparable evaluation across all three host models without requiring model-specific input formats or instruction-tuning. Each input consists of a leading text fragment tagged with the target attribute value(s), and the expected output is the text generated by the model in response, up to the first end-of-output tag or a specified maximum length, whichever comes first.

Leading fragments are constructed by taking the first n words of a given text from the in-domain datasets, where n ranges from 0 to 5 (and from 2 to 4 for PPLM Prompts). Input/output pairs are then formed by splitting the tagged text after n words. For example, a training instance for sentiment-controlled generation takes

the following form:

```
[SENTIMENT] Positive [\SENTIMENT] [ANS] The sushi is
great!  Not expensive & good quality.  My favorite rolles are the
Vegas, Dragon, and the baked scallops :)  [\ANS]
```

where the attribute tag and its value precede the answer tag `[ANS]`, which marks the beginning of the generated text. For multiple-attribute control, multiple attribute tags are included in the input, one per controlled attribute. The out-of-domain datasets (PPLM Prompts and STS) are used exclusively at test time.

6.3.3 Evaluation Metrics

The evaluation framework used in this chapter follows the setup established in Section 4.4.2.2, to which we refer the reader for a full discussion of metric selection, classifier choice, and the implications of each metric for the interpretation of CTG evaluation results. Here we summarise the three metrics used throughout and note any adaptations specific to this chapter.

Diversity. Diversity is measured with Distinct-n (Li et al., 2015), defined as the proportion of unique n-grams in the generated texts. The aggregate Distinct-n score is the mean of item-level scores. We report Distinct-1, Distinct-2, and Distinct-3 throughout.

Fluency. Fluency is assessed using the Syntactic Log-Odds Ratio (SLOR) (Kann et al., 2018), computed using GPT-2 XL (Radford et al., 2019) and BLOOM 1B7 (Scao et al., 2022). SLOR is used here as an automatic proxy for fluency rather than a direct measure of it: it calculates the log-probability of a sentence normalised by its unigram log-probability and length, capturing the degree to which a text is syntactically well-formed relative to unigram expectations. It is preferred over perplexity as the latter may not effectively capture fluency for low-frequency items, as discussed in Section 4.4.5.3. The final SLOR score reported is the average of the scores obtained from both language models.

Control Effectiveness. Control Effectiveness (CE) measures the mean percentage of generated texts identified by a set of classifiers as possessing the target attribute value. For each classifier, we compute the percentage of cases in which the detected attribute value matches the input control value; the final CE score is the average of these individual percentages across all classifiers. The classifiers used here are the same as those selected in Section 4.4.2.2, where the classifier selection criteria are discussed in detail. For **sentiment control**, we use DistilBERT and T5 fine-tuned on SST-2 (Socher et al., 2013), and DeBERTa fine-tuned on Yelp (Chelba et al., 2013). For **topic control**, we use DistilBERT, BERT, and DeBERTa fine-tuned on AG News (Zhang et al., 2015). As discussed in Section 4.4.2.2, classifiers were selected based on two criteria: preference for classifiers used in prior CTG work, and architectural diversity to mitigate architecture-specific bias. An important implication of this selection for the topic control results reported in this chapter is that all three topic classifiers are trained on AG News, which may introduce a mild evaluation bias in favour of systems trained on AG News data. The extent of this bias is assessed in Section 6.5.5, where we conduct an additional evaluation using an instruction-tuned LLM as an independent classifier and confirm that the overall trends are not an artefact of classifier training data.

In the multiple-attribute control setting, CE is extended to require that both the sentiment and topic labels are correctly predicted simultaneously. Majority voting across the respective classifier sets is first applied to obtain a single sentiment label and a single topic label for each generated text; CE is then computed as the percentage of texts for which both predicted labels match the input control values.

6.4 Training Single-Attribute Specialised Modules

This section presents the first of the three experimental settings outlined in Section 6.2: the design and evaluation of single-attribute specialised PEFT modules using supervised fine-tuning (SFT) with QLoRA (Dettmers et al., 2023). The core question addressed in this setting is how well a PEFT module trained specifically

on a single attribute control task performs on that task, both within and beyond its training distribution. The modules trained here form the foundation for the second and third settings, module composition (Section 6.5) and cross-model portability (Section 6.6) respectively, where they are combined and transferred across model architectures.

As established in Section 5.4, LoRA consistently outperforms other PEFT techniques such as prompt tuning and prefix tuning across a range of controlled text generation tasks, motivating its use as the adaptation method of choice in this chapter. We build on QLoRA (Dettmers et al., 2023), which extends LoRA with 4-bit quantisation to reduce memory requirements while preserving task-specific adaptation quality, enabling training on a single NVIDIA A100 GPU without sacrificing performance. However, the original QLoRA implementation supports only a single adapter module per host model and provides no mechanism for attaching, detaching, or combining multiple modules at inference time, which are all requirements of the plug-and-play framework investigated in this chapter. We therefore develop our own extended reimplementation of QLoRA (Section 6.4.2), which adds support for the simultaneous attachment of multiple PEFT modules to the same frozen host model, and for their composition using three novel techniques at inference time. Each module produced by this framework is trained on a single-attribute control dataset and designed to steer the host model towards generating texts that exhibit the target attribute, without modifying the frozen pretrained weights. Crucially, our reimplementation treats each trained module as a lightweight, self-contained parameter block that can be attached to and detached from the host model independently of any other module, a property not present in the original QLoRA implementation and one that is essential for the composition and portability experiments in Sections 6.5 and 6.6.

The models and datasets used throughout this section are described in Section 6.3.1, the task construal in Section 6.3.2, and the evaluation metrics in Section 6.3.3. The remainder of this section covers the elements specific to the SFT

setting: the construction of the training datasets (Section 6.4.1), the plug-and-play QLoRA implementation and its mathematical formulation (Section 6.4.2), the training hyperparameters (Section 6.4.4), and the results on single-attribute control (Section 6.4.5).

6.4.1 Composite dataset construction

Each QLoRA module is trained on a single dataset from Section 6.3.1. In addition, we construct two composite datasets, one for sentiment control and one for topic control, by combining the individual single-task datasets. These composite datasets serve two purposes: they are used to train combined-dataset baseline modules in this section, and they are reused as the source of training data for the composition experiments in Section 6.5.

The composite datasets are constructed via the following three-step process:

1. **Filtering**: Dataset items with fewer than 10 words are removed, as are any neutral-labelled items in the Yelp dataset.

2. **Smallest dataset balancing**: The smallest dataset is identified; if its label distribution is imbalanced, it is randomly downsampled to obtain a balanced distribution across labels.

3. **Stratified sampling**: All remaining datasets are downsampled to match the size of the smallest dataset, ensuring that each dataset is balanced across both labels and text length.

The three sampled sentiment datasets are combined to form the *Combined Sentiment* dataset, and the two topic datasets to form the *Combined Topic* dataset. Sampling is performed only on the training and validation splits; for single-dataset test sets, up to 50 examples per label are included. Table 6.3 reports the number of samples, average number of words, and average number of characters per text for each dataset split before and after sampling. The stratified sampling procedure effectively produces balanced datasets while preserving the diversity of input lengths across splits.

Dataset	Split	Before sampling			After sampling		
		# samples	Avg words	Avg char	# samples	Avg words	Avg char
IMDb	train	25000	231.49	1302.62	24330	231.50	1302.54
	test	25000	226.25	1243.82	100	218.61	1271.64
SST-2	train	67349	9.41	97.21	24330	17.86	53.03
	validation	872	19.55	116.24	726	21.62	105.05
	test	1821	19.23	49.86	100	9.33	103.15
Yelp	train	470000	134.03	728.80	24330	135.85	718.88
	validation	50000	133.87	746.85	726	138.81	717.61
	test	40000	133.87	712.02	100	132.03	718.08
Combined Sentiment	train	-	-	-	72990	128.41	709.54
	validation	-	-	-	1452	80.22	431.55
AG News	train	112000	37.85	236.11	111992	37.86	236.10
	validation	8000	37.78	235.75	3996	37.78	235.67
	test	7600	37.72	231.37	200	37.42	234.90
DBpedia	train	546000	46.13	275.05	111984	44.91	280.56
	validation	14000	46.16	274.94	3996	44.90	280.49
	test	70000	46.14	265.66	195	43.12	280.57
Combined Topic	train	-	-	-	223976	41.38	255.58
	validation	-	-	-	7992	41.34	255.35
PPLM	test	35	2.54	13.31	35	2.54	13.31
STS	test	625	8.02	39.36	100	7.50	36.83
STS proc	test	-	-	-	100	3.18	13.71

Table 6.3: Statistics of the datasets used for each data split (train, validation, test), showing the number of samples, average number of words per text, and average number of characters per text, both before and after dataset sampling.

6.4.2 Plug-and-play QLoRA

To support the plug-and-play framework investigated in this chapter, we developed an extended reimplementation of QLoRA (Dettmers et al., 2023), which we refer to as **Plug-and-play QLoRA**.[2] The original QLoRA implementation supports only a single adapter module per host model, with no mechanism for attaching or detaching modules independently, and no support for combining multiple modules at inference time. Plug-and-play QLoRA addresses these limitations by introducing two layers of additional functionality: (i) the ability to attach any number of independently trained PEFT modules to the same frozen host model simultaneously, with each

[2]Available from `https://github.com/michelalorandi/plug_and_play_qlora`

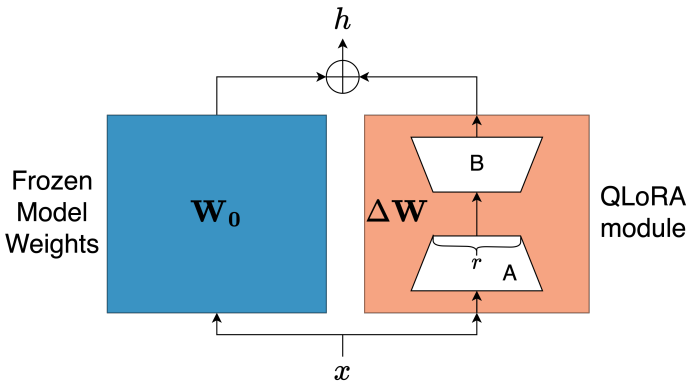


Figure 6.2: A single QLoRA block (orange) shown attached to its corresponding model weights.

module remaining self-contained and detachable without affecting the others or the host model weights; and (ii) three novel techniques for composing the outputs or weights of attached modules at inference time, described in Section 6.5. This section describes the single-module foundation of Plug-and-play QLoRA; the composition extensions are introduced in Section 6.5 where they are used.

In vanilla single-module QLoRA, a PEFT block consisting of a down-projection and an up-projection is attached in parallel to every query, key, value, and feed-forward layer in every transformer block, as illustrated in Figure 6.2.

Adopting the notation of Asadi et al. (2024), QLoRA learns a far smaller set of task-specific parameters $\Delta\Theta$ in conjunction with the frozen pretrained model Θ_0. The QLoRA module $\Delta\Theta$ consists of trainable low-rank decomposition matrices added as adapter modules to the query, key, value, and feed-forward weight matrices of Θ_0. Concretely, QLoRA introduces task-specific parameters $\Delta\mathbf{W}$ alongside the frozen weight matrices $\mathbf{W}_0 \in \mathbb{R}^{d\times c}$, yielding the adapted weight matrix:

$$\hat{\mathbf{W}} = \mathbf{W_0} + \frac{\alpha}{r}\mathbf{\Delta W} \tag{6.1}$$

where $\Delta\mathbf{W} = \mathbf{A}\mathbf{B}^\top$ is the QLoRA block, with $\mathbf{A} \in \mathbb{R}^{d\times r}$ the down-projection matrix, $\mathbf{B}^\top \in \mathbb{R}^{r\times d}$ the up-projection matrix, α a scaling factor, and r the rank. $\mathbf{A}$ is initialised using the Kaiming uniform distribution (He et al., 2015), while $\mathbf{B}$ is initialised to all zeros, ensuring that $\Delta\mathbf{W} = \mathbf{0}$ at the start of training and the adapter

therefore introduces no perturbation to the pretrained model at initialisation.

$\Delta\Theta$ is optimised using a standard conditional language modelling objective with cross-entropy loss computed over $\Theta_0 + \Delta\Theta$, i.e. the frozen model with the QLoRA blocks attached. For a given input $\mathbf{x}$, the output h of the adapted layer is:

$$h = \mathbf{W_0}\, x + \frac{\alpha}{r}\mathbf{\Delta W} x \tag{6.2}$$

The frozen parameters $\mathbf{W}_0$ thus contribute the base language modelling behaviour, while the QLoRA block $\Delta\mathbf{W}$ contributes the task-specific adaptation. Since only $\Delta\Theta$ is updated during training, the number of trainable parameters is a small fraction of the total model size, and the frozen weights remain unchanged and reusable across different modules.

In the plug-and-play extension of this framework, multiple QLoRA blocks can be attached at each layer location simultaneously. This enables the composition of separately trained modules at inference time, without any further training, as will be described in Section 6.5.

6.4.3 Models

We consider three base pretrained LLMs as host models: LLaMA 3 8B (AI@Meta, 2024), LLaMA 3.1 8B, and Mistral 7B (Jiang et al., 2023). By *base* we mean models that have undergone pretraining on large text corpora but have not been instruction-tuned, chat-tuned, or otherwise aligned through reinforcement learning from human feedback (RLHF). This is an important distinction for the experiments in this chapter: since the modules are trained to inject attribute-specific control signal via the tagged prompt format described in Section 6.3.2, using instruction-tuned models would conflate the effect of the adapter with the effect of instruction following, making it difficult to isolate the contribution of the PEFT module itself.

These three models were selected for three reasons. First, they represent state-of-the-art open-source LLMs at the time of experimentation, providing a competitive baseline for the controlled generation tasks. Second, they span two distinct model

families (LLaMA and Mistral), enabling assessment of whether findings generalise across architectural choices within a similar parameter range. Third, their comparable parameter counts (7–8 billion) allow the effect of model family to be examined while keeping model scale approximately constant.

LLaMA 3 8B is a pretrained autoregressive transformer model with 8 billion parameters. It has a decoder-only architecture optimised with grouped-query attention for faster inference and the SwiGLU activation function for improved training efficiency. It additionally incorporates Rotary Positional Embeddings (RoPE) to handle longer context lengths, and was trained on 15 trillion tokens collected from publicly available, highly curated sources.[3] LLaMA 3.1 8B is an incremental update to LLaMA 3 8B, retaining the same 8 billion parameter transformer architecture while incorporating improvements in training stability, dataset quality, and tokenisation. Although specific architectural details are not publicly disclosed, this version is expected to refine pretraining efficiency, dataset diversity, and representation quality, leading to better perplexity scores and more coherent text generation. Mistral 7B is a pretrained transformer model with 7 billion parameters, designed to be highly efficient while maintaining strong language modelling performance. It incorporates grouped-query attention together with sliding window attention, enabling efficient handling of longer context lengths without increasing computational complexity.

6.4.4 Experimental Setup

We train one QLoRA module per dataset, covering the five individual datasets (IMDb, Yelp, SST-2, AG News, and DBPedia) and the two composite datasets (Combined Sentiment and Combined Topic) described in Section 6.4.1. Each module is trained independently on its respective dataset, with the host model weights kept frozen throughout.

Hyperparameters are selected via a small grid search for each model, balancing computational cost with task performance. Following the recommendations of

[3] `https://ai.meta.com/blog/meta-llama-3/`

Dettmers et al. (2023), we set the rank $r = 64$, scaling factor $\alpha = 16$, and dropout to 0.1. The down-projection matrix $\mathbf{A}$ is initialised using the Kaiming uniform distribution with $a = \sqrt{5}$, and the up-projection matrix $\mathbf{B}$ is initialised to all zeros, as described in Section 6.4.2. QLoRA blocks are attached to all linear layers of the pretrained model, specifically the query, key, value, output, gate, up, down, and language model head projections. All modules are trained in half-precision (bf16) on a single NVIDIA A100 80GB GPU for up to 3 epochs using the Paged AdamW optimiser (Loshchilov, 2017). The remaining hyperparameters are set as follows: batch size of 4, gradient accumulation steps of 4, maximum gradient norm of 1.0, warmup ratio of 0.1, weight decay of 0.5, and maximum sequence length of 1024 tokens. Regarding model quantisation, we use 4-bit NormalFloat (NF4) quantisation with bfloat16 as the compute dtype.

A checkpoint is saved at the end of each epoch, and the checkpoint achieving the highest Control Effectiveness on the validation set is selected as the final module for each run. The grid search explores two learning rates (5×10^{-6} and 2×10^{-4}) and two learning rate schedulers (cosine and constant). The best-performing hyperparameter configurations identified for each model and seed are reported in Appendix D.1.1.

All experiments are repeated three times with different random seeds, and results are reported as averages across the three runs.

6.4.5 Results

We evaluate the single-attribute QLoRA modules trained as described above on both in-domain and out-of-domain datasets, following the evaluation framework introduced in Section 6.3.3. Results are reported separately for sentiment control (Tables 6.4 and 6.5) and topic control (Tables 6.6 and 6.7), with diversity and fluency reported separately from control effectiveness given the different nature of these metrics. For each setting, we compare the following configurations:

- the raw pretrained model with no QLoRA module attached, serving as a lower-bound baseline;

- individual modules trained on each single dataset (IMDb, Yelp, and SST-2 for sentiment; AG News and DBPedia for topic);
- a single module trained on the corresponding composite dataset (either Combined Sentiment or Combined Topic).

CTG Technique	**Distinct-n↑**			**SLOR↑**
	dist-1	**dist-2**	**dist-3**	
Llama 3 8B *(no module)*	0.03	0.09	0.12	9.44
+ QLoRA Yelp	0.09	0.24	0.34	9.87
+ QLoRA IMDB	0.04	0.13	0.19	**10.86**
+ QLoRA SST-2	**0.34**	**0.64**	**0.71**	8.24
+ QLoRA Combined Sentiment dataset	0.11	0.27	0.35	10.07
Llama 3.1 8B *(no module)*	0.04	0.10	0.13	9.66
+ QLoRA Yelp	0.09	0.26	0.36	9.94
+ QLoRA IMDB	0.04	0.14	0.21	**10.81**
+ QLoRA SST-2	**0.36**	**0.69**	**0.76**	8.24
+ QLoRA Combined Sentiment dataset	0.12	0.27	0.35	10.00
Mistral 7B *(no module)*	0.04	0.09	0.12	7.45
+ QLoRA Yelp	0.03	0.07	0.10	10.44
+ QLoRA IMDB	0.02	0.04	0.06	**11.25**
+ QLoRA SST-2	**0.27**	**0.51**	**0.58**	7.65
+ QLoRA Combined Sentiment dataset	0.03	0.06	0.09	10.83

Table 6.4: **Sentiment Control:** Diversity (Distinct-n) and Fluency (SLOR) for the model + QLoRA module. All values are averages over 3 runs. Bold = highest score in column/section.

6.4.5.1 Sentiment control

Tables 6.4 and 6.5 show the diversity (Distinct-n), fluency (SLOR), and control effectiveness results for sentiment control across the three host models. Regarding **diversity**, training on SST-2 (+ QLoRA SST-2 in Table 6.4) consistently produces the most diverse outputs by a substantial margin across all three models, achieving Distinct-3 scores of 0.71, 0.76, and 0.58 for LLaMA 3 8B, LLaMA 3.1 8B, and Mistral 7B respectively. This is likely attributable to the short, sentence-level nature of SST-2 texts: because SST-2 training instances are brief sentence fragments rather

CTG Technique	Control Effectiveness↑								
	Avg All	Avg	Yelp	IMDB	SST-2	Out-Of-Domain			
						Avg	PPLM S	STS S	STS proc S
Llama 3 8B *(no module)*	59.93	63.70	68.00	57.56	65.56	57.16	64.60	52.83	54.06
+ QLoRA Yelp	**85.11**	91.44	**92.78**	90.67	90.89	79.49	**88.41**	**67.78**	**82.28**
+ QLoRA IMDB	83.07	**92.07**	92.11	89.67	**94.44**	73.61	80.95	62.00	77.89
+ QLoRA SST-2	77.80	85.96	84.11	82.44	91.33	70.73	82.86	57.39	71.94
+ QLoRA Combined data	82.99	91.07	92.22	**92.00**	89.00	**75.88**	87.46	63.50	76.67
Llama 3.1 8B *(no module)*	58.32	61.52	66.22	54.22	64.11	56.41	64.29	51.56	53.39
+ QLoRA Yelp	**86.68**	91.52	91.00	89.00	94.56	**82.93**	**91.75**	**71.28**	**85.78**
+ QLoRA IMDB	83.46	92.26	89.67	**91.44**	**95.67**	75.04	85.24	63.33	76.56
+ QLoRA SST-2	78.29	85.48	82.33	82.89	91.22	72.49	84.76	58.17	74.56
+ QLoRA Combined data	84.75	**92.67**	**94.89**	**91.44**	91.67	77.86	89.52	62.28	81.78
Mistral 7B *(no module)*	57.54	60.00	59.33	55.00	65.67	56.23	62.86	52.50	53.33
+ QLoRA Yelp	**84.33**	**92.30**	**94.78**	**90.56**	91.56	**76.42**	**84.60**	**61.33**	**83.33**
+ QLoRA IMDB	80.24	90.63	89.67	88.56	**93.67**	70.38	82.70	58.78	69.67
+ QLoRA SST-2	78.17	85.67	83.11	81.67	92.22	71.87	83.65	55.94	76.00
+ QLoRA Combined data	78.28	86.30	87.11	85.67	86.11	71.32	83.17	58.22	72.56

Table 6.5: **Sentiment Control:** Control Effectiveness for the model + QLoRA module. All values are averages over 3 runs. Bold = highest score in column/section; underline = train/test on same dataset.

than full reviews, the model is exposed to a wider variety of syntactic structures and lexical contexts during fine-tuning, which may lead to less repetitive and more lexically varied generation compared to training on the longer, more formulaic review texts in Yelp and IMDb, where the training signal is dominated by a narrower range of review-style sentence patterns. Training on IMDb consistently yields the highest fluency scores across all models, with SLOR scores of 10.86, 10.81, and 11.25 respectively, albeit by smaller margins than the diversity differences observed for SST-2. The combined dataset modules achieve intermediate diversity and fluency scores in most cases, broadly reflecting the mixture of text styles present in the training data.

Regarding **control effectiveness**, all three raw models (the *no module* rows in each section of Table 6.5) perform at a similarly low level of around 58–60% average CE, only marginally above the 50% chance baseline for binary sentiment classification, confirming that the pretrained models have limited inherent ability to follow attribute control prompts without task-specific fine-tuning. QLoRA fine-tuning consistently improves CE across all models and datasets, with average CE scores ranging from 77.80% to 86.68% depending on the training dataset and host

model (the *+ QLoRA* rows in each section of Table 6.5). A clear trend emerges whereby Mistral 7B responds slightly less strongly to QLoRA fine-tuning than the two LLaMA models across most settings, with the best Mistral 7B result (84.33%, QLoRA Yelp) falling below the best LLaMA 3.1 8B result (86.68%, QLoRA Yelp).

Across all three models, the Yelp-trained module achieves the highest average CE and the best out-of-domain performance by a considerable margin, with the overall best results obtained by LLaMA 3.1 8B + QLoRA Yelp (86.68% average CE). This advantage on out-of-domain data is notable, as it suggests that the longer, more stylistically varied Yelp reviews provide a richer training signal for generalisation beyond the training distribution. In contrast, the SST-2-trained module consistently achieves the lowest CE scores despite its high diversity, suggesting a trade-off between output variety and attribute control precision.

Strikingly, it is rarely the case that a module trained on a given dataset achieves the best CE score when tested on that same dataset. In Table 6.5, the underlined score in each row indicates the CE obtained when the module is tested on the dataset it was trained on: for example, the underlined score in the *+ QLoRA Yelp* row is the CE on the Yelp test set, the underlined score in the *+ QLoRA IMDb* row is the CE on the IMDb test set, and so on. Inspecting these scores, the train/test same-dataset result is the highest in its column in only a minority of cases, suggesting that modules trained on a given dataset do not necessarily specialise exclusively to that dataset's distribution. For the LLaMA models, training on the combined dataset improves or maintains CE compared to training on the corresponding single dataset in all cases but SST-2. The exception for SST-2 may reflect the fact that SST-2 training texts are very short sentence fragments (average 9–18 words after sampling, see Table 6.3), which differ considerably in style and length from the other sentiment datasets, making it harder for the combined dataset module to match the specialisation of a module trained exclusively on SST-2-style inputs. For Mistral 7B, however, the combined dataset module performs considerably worse than the best single-dataset modules across all three in-domain test sets, indicating that this

model is less able to benefit from the increased training data diversity in the SFT setting.

6.4.5.2 Topic control

Tables 6.6 and 6.7 present the diversity (Distinct-n), fluency (SLOR), and control effectiveness results for topic control across the three host models.

CTG Technique	Distinct-n↑			SLOR↑
	dist-1	dist-2	dist-3	
Llama 3 8B *(no module)*	0.07	0.17	0.22	8.45
+ QLoRA AG News	0.25	0.53	0.62	**9.39**
+ QLoRA DBPedia	**0.32**	**0.60**	0.68	8.96
+ QLoRA Combined Topic dataset	0.30	**0.60**	**0.70**	8.92
Llama 3.1 8B *(no module)*	0.05	0.12	0.17	**9.45**
+ QLoRA AG News	0.26	0.55	0.65	9.40
+ QLoRA DBPedia	**0.31**	0.59	0.68	8.74
+ QLoRA Combined Topic dataset	**0.31**	**0.62**	**0.73**	9.29
Mistral 7B *(no module)*	0.07	0.13	0.17	7.98
+ QLoRA AG News	0.18	0.37	0.44	**9.48**
+ QLoRA DBPedia	**0.26**	**0.48**	**0.56**	8.81
+ QLoRA Combined Topic dataset	0.21	0.43	0.52	9.09

Table 6.6: Diversity (Distinct-n) and Fluency (SLOR) for **Topic Control**, training on *single* and *combined* datasets. All values are averages over 3 runs. Bold = highest score in column and section.

Regarding **diversity**, the DBPedia-trained module achieves the highest Distinct-1 scores across all three models (bold scores in the dist-1 column of Table 6.6). For Distinct-2 and Distinct-3, the picture is more varied: the combined dataset module achieves the highest scores for LLaMA 3 8B and LLaMA 3.1 8B, while DBPedia achieves the highest Distinct-2 and Distinct-3 scores for Mistral 7B. There is therefore no single configuration that dominates diversity across all three models beyond Distinct-1. **Fluency** scores are broadly similar across all configurations, with no notable trends emerging, though the AG News-trained module tends to achieve marginally higher SLOR scores than the other configurations. Notably, for LLaMA 3.1 8B the raw model achieves the highest fluency score overall (9.45), suggesting

CTG Technique	Control Effectiveness↑							
	Avg All	Avg	AG News	DBPedia	Out-Of-Domain			
					Avg	PPLM T	STS T	STS proc T
Llama 3 8B *(no module)*	45.47	58.52	64.61	52.42	37.53	48.97	27.97	35.64
+ QLoRA AG News	**68.53**	**85.13**	**90.72**	**79.54**	**58.02**	**71.11**	**42.03**	60.92
+ QLoRA DBPedia	52.61	69.94	71.67	68.21	41.66	55.40	28.58	41.00
+ QLoRA Combined data	62.76	74.42	81.61	67.24	56.27	68.73	37.50	**62.58**
Llama 3.1 8B *(no module)*	45.86	58.61	66.11	51.11	37.47	46.35	29.89	36.17
+ QLoRA AG News	**67.85**	**85.52**	**91.33**	**79.72**	**57.21**	**73.10**	36.28	**62.25**
+ QLoRA DBPedia	52.63	69.86	70.94	68.77	41.64	54.92	28.11	41.89
+ QLoRA Combined data	65.08	80.65	89.06	72.25	55.01	66.51	**39.97**	58.56
Mistral 7B *(no module)*	44.14	55.96	62.00	49.91	37.48	49.76	29.92	32.75
+ QLoRA AG News	**69.05**	**88.52**	**93.17**	**83.87**	**57.14**	**73.97**	**39.69**	**57.75**
+ QLoRA DBPedia	52.92	67.05	67.50	66.61	43.76	54.13	32.89	44.28
+ QLoRA Combined data	64.64	81.74	89.00	74.47	53.87	67.54	39.22	54.86

Table 6.7: Control Effectiveness for **Topic Control**, training on *single* and *combined* datasets. All values are averages over 3 runs. Bold = highest score in column and section; underline = train and test set from same dataset.

that QLoRA fine-tuning for topic control introduces a slight fluency trade-off in this model.

Regarding **control effectiveness**, the pattern across models is considerably clearer than for diversity and fluency. The AG News-trained module consistently outperforms all other configurations across all three models, both in-domain and out-of-domain (bold scores in Table 6.7), achieving average CE scores of 68.53%, 67.85%, and 69.05% for LLaMA 3 8B, LLaMA 3.1 8B, and Mistral 7B respectively. The DBPedia-trained module performs considerably worse despite being trained on a larger dataset, with average CE scores around 52–53% across all models. This gap is attributable to two compounding factors. First, the classifiers used to compute CE are all trained on AG News: since these classifiers have learned to identify topic-relevant language patterns from AG News texts specifically, a module trained on AG News is likely to generate texts that better match those patterns, giving it a systematic advantage in the CE metric over a module trained on DBPedia. Second, the mapping from DBPedia to AG News labels introduced in Section 6.3.1 is imperfect: because DBPedia entity categories do not map cleanly onto AG News topic labels, some DBPedia training instances are assigned incorrect or ambiguous topic labels, introducing noise into the training signal that reduces the quality of the learned

topic representations. Together, these two factors mean that the CE gap between AG News and DBPedia modules should not be interpreted as a straightforward reflection of module quality. To assess the extent to which classifier bias specifically distorts the results, we conduct an additional evaluation using an instruction-tuned LLM as an independent classifier. These results are reported in Section 6.5.5 rather than here because that section presents the full set of topic control results including composition techniques. The LLM classifier evaluation confirms that the overall trends observed here are not an artefact of classifier training data.

The combined dataset module achieves intermediate CE scores between the AG News and DBPedia single modules in all in-domain settings, reflecting the mixed quality of the training signal from the two datasets. For out-of-domain testing, the combined dataset module outperforms the AG News module in one case: STS proc T for LLaMA 3 8B, where it achieves 62.58% compared to 60.92% for AG News. It also achieves the best STS T result for LLaMA 3.1 8B (39.97%, compared to 36.28% for AG News), representing a somewhat larger margin. In the table headers, the suffix T denotes topic control evaluation, distinguishing these out-of-domain columns from the sentiment control columns (suffix S) reported in Table 6.5. As with sentiment control, the raw models show limited ability to follow topic control prompts without fine-tuning, with average CE scores of around 44–46% on the in-domain test sets (Avg column of Table 6.7), and QLoRA fine-tuning consistently provides improvements across all settings.

Taken together, the results across sentiment and topic control show that QLoRA fine-tuning consistently improves attribute control effectiveness over the raw model baseline, with performance varying notably across training datasets and, in the case of combined-dataset training, across host models. The key findings and their implications for the remainder of the chapter are discussed in the summary below.

6.4.6 Summary of Single-Attribute Modules Training

This section described the design and evaluation of single-attribute specialised QLoRA modules for sentiment and topic control. These modules serve as the building blocks for the subsequent experimental settings in two concrete ways: the individually trained modules are composed at inference time using Compositional QLoRA in Section 6.5; and the findings from this setting, specifically the identification of the best-performing training dataset per attribute, directly inform the portability experiments in Section 6.6, where new modules are trained on those datasets and ported across model architectures. Several findings emerge from the experiments in this setting that are relevant to both of those investigations.

QLoRA fine-tuning improves control effectiveness over the raw model baseline across all settings, with average CE gains of approximately 20–25 points, consistently across host models and training datasets. The choice of training dataset has a notable impact on performance: for sentiment control, Yelp-trained modules achieve the highest average CE and the strongest out-of-domain results across all three models; for topic control, AG News-trained modules consistently outperform DBPedia-trained ones, a gap partly attributable to label mapping noise in the DBPedia data and to the fact that all CE classifiers are trained on AG News. Training on combined datasets generally improves or maintains CE for the LLaMA models but not consistently for Mistral 7B, indicating that the relationship between training data diversity and adaptation quality is model-dependent.

In the broader context of this thesis, the results provide evidence towards **RQ3.1** (How can we create effective single-attribute PEFT modules?), showing that attribute-specialised QLoRA modules can achieve single-attribute control in open-ended generation settings across three different host models. This extends the finding from Chapter 5 that LoRA-based fine-tuning is effective for CTG to a more challenging generation scenario: whereas Chapter 5 addressed data-to-text generation, where the content to be expressed is explicitly specified by structured input triples, the present setting involves open-ended generation from short text prompts, with no ex-

plicit content constraints beyond the target attribute label. Controlling generation in this setting is more demanding because the model has greater freedom in what it generates, making it harder to reliably steer outputs towards a target attribute without also constraining other aspects of generation. The modular structure of the trained modules, each a self-contained parameter block that can be attached to and detached from its host model independently, is the property that makes the composition and portability investigations of the following sections feasible. The combined-dataset modules additionally provide an important point of comparison for those experiments, representing the performance achievable by a single module exposed to all relevant training data simultaneously.

The modules presented in this section are trained exclusively using supervised fine-tuning; the extent to which alternative training paradigms, such as reinforcement learning from attribute classifier feedback, might yield modules with different composability or portability properties remains a direction for future work.

6.5 Multiple-Attribute Control via Module Composition

The results for the single-attribute modules from Section 6.4 demonstrate that QLoRA fine-tuning is an effective means of specialising a pretrained language model for a specific attribute control task. However, real-world applications often require simultaneous control over multiple attributes, for instance, generating text that is both positive in sentiment and focused on a specific topic. A natural question is therefore whether separately trained single-attribute modules can be combined at inference time to achieve joint multiple-attribute control, without incurring the cost of training a new module for every possible attribute combination.

The modular nature of PEFT techniques makes this an appealing prospect. Since parameter blocks such as QLoRA modules can be attached, detached, and replaced within the same host model as needed (Sabry and Belz, 2024), they hold out the

vision of full plug-and-playability, where individually trained modules are combined at will to achieve specific model behaviours. Sabry and Belz (2024) demonstrate this modularity through porting between base and instruction-tuned variants of the same model (e.g. T5 to Flan-T5), but do not investigate porting across different model architectures, nor whether independently trained modules can be composed to achieve joint control over multiple attributes. Realising this vision in practice, however, requires that module composition preserves the task-specific knowledge encoded in each individual module while simultaneously achieving the combined functionality. This is a non-trivial requirement, as naive combination strategies risk interfering with the learned representations of individual modules.

In this section, we examine the extent to which two properties of composability can be achieved when QLoRA modules are combined within a host model:

1. **Generalisation across related datasets**: whether a composition of modules trained on different datasets for the same control attribute (e.g. Yelp, IMDb, and SST-2 for sentiment) can match or exceed the performance of a single module trained on any one of those datasets, when each is evaluated on its own test set;

2. **Functional composability**: whether a composition of single-attribute modules can achieve effective control over multiple attributes simultaneously.

To this end, in this section we introduce and evaluate three composition techniques, weight averaging, output averaging, and output summing, and compare them against the single-module and combined-dataset baselines established in Section 6.4. We test all composition techniques on the tasks of sentiment control, topic control, and multiple-attribute control, across the three host models introduced in Section 6.3.1. The composition techniques themselves are described in Section 6.5.1, followed by the study structure and testing protocol in Section 6.5.2, and the results in Section 6.5.4.

6.5.1 Composition Techniques

We evaluate three techniques for composing trained QLoRA modules at inference time, shown in Figure 6.3: element-wise averaging of module weights, element-wise summation of module outputs, and element-wise averaging of module outputs. All three techniques operate without any post-composition training or optimisation, and can be applied to any number of modules N.

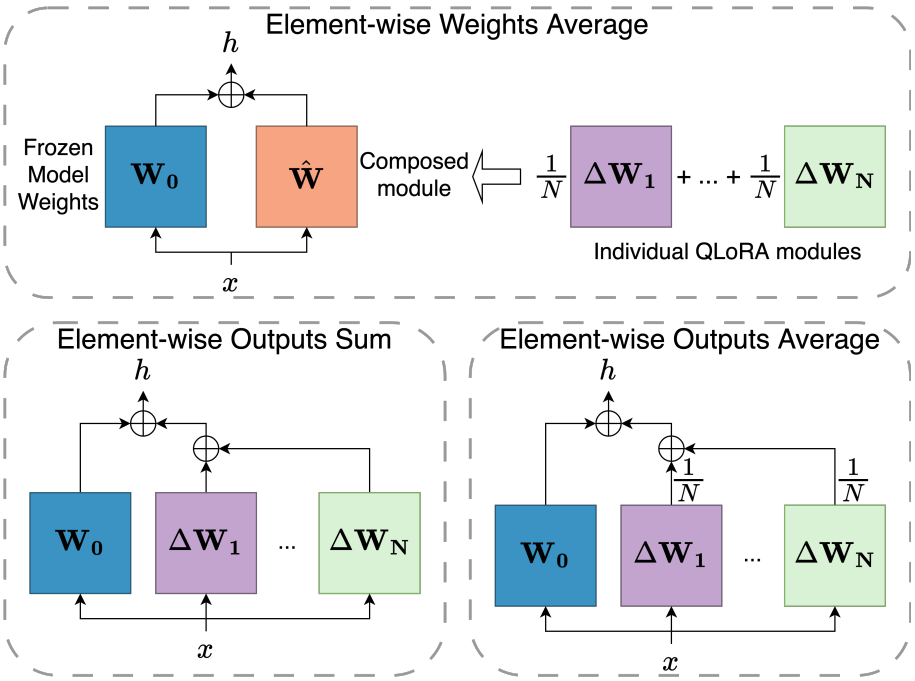


Figure 6.3: Diagram of our three QLoRA module composition techniques: (Top) Element-wise Weights Average; (Bottom-left) Element-wise Outputs Summing; (Bottom-right) Element-wise Outputs Averaging.

Weight Averaging. The first technique, **weight averaging**, composes N QLoRA modules by computing the element-wise average of their weight matrices:

$$\hat{\mathbf{W}} = \frac{1}{N} \sum_{i=1}^{N} \Delta\mathbf{W_i} \tag{6.3}$$

where $\Delta\mathbf{W}_i$ are the weights of the i-th QLoRA module and $\hat{\mathbf{W}}$ is the single composed module. The output of the composed module is then computed in the same way as a single QLoRA module (Equation 6.2):

$$h = \mathbf{W_0}\, x + \frac{\alpha}{r}\hat{\mathbf{W}}x \tag{6.4}$$

where $\mathbf{W}_0$ are the frozen model weights, $\frac{\alpha}{r}$ is the normalised scaling factor, and $\mathbf{x}$ is the input. Weight averaging reduces the N modules to a single composed module, and therefore maintains the same inference cost as a single module. However, as discussed below, this efficiency comes at the cost of losing the individual low-rank structures of the constituent modules.

Output Summing. The second technique, **output summing**, instead computes the output of each module separately before combining them. The outputs are first scaled by their respective scaling factors ($\frac{\alpha}{r} = \frac{16}{64}$ in all experiments, see Appendix D.1.1), and their element-wise sum is then added to the frozen model output:

$$h = \mathbf{W_0}\, x + \sum_{i=1}^{N} \frac{\alpha_i}{r_i}(\Delta\mathbf{W_i}\, x) \tag{6.5}$$

Output summing scales the total adapter contribution linearly with the number of modules N, since each module's output is added in full. In the experiments reported in this section, we compose between 2 and 5 modules; the behaviour of output summing with substantially larger numbers of modules remains an open question and is discussed further in Section 6.5.6.

Output Averaging. The third technique, **output averaging**, computes the element-wise average of module outputs before adding the result to the frozen model output:

$$h = \mathbf{W_0}\, x + \frac{1}{N}\sum_{i=1}^{N} \frac{\alpha_i}{r_i}(\Delta\mathbf{W_i}\, x) \tag{6.6}$$

Unlike output summing, output averaging normalises the total adapter contribution by N, keeping the magnitude of the composed output independent of the number of modules. It is important to note that weight averaging and output averaging are not mathematically equivalent, despite both involving a linear averaging operation,

as explained below.

Weight Averaging vs Output Averaging. Although both weight averaging and output averaging involve linear averaging operations, they are not mathematically equivalent. The distinction arises from how the averaging interacts with the low-rank structure of the QLoRA modules. In our implementation, weight averaging computes the average of the low-rank factors $\mathbf{A}$ and $\mathbf{B}$ separately:

$$\mathbf{A}_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{A}_i, \qquad \mathbf{B}_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_i \tag{6.7}$$

and then computes the adapted output as $\mathbf{A}_{\text{avg}}\mathbf{B}_{\text{avg}}^{\top}\mathbf{x}$. Expanding this product reveals that it is not equivalent to averaging the full module transformations $\mathbf{A}_i\mathbf{B}_i^{\top}$, because the product of averaged factors introduces cross-terms:

$$\left(\sum_i \mathbf{A}_i\right)\left(\sum_j \mathbf{B}_j^{\top}\right) = \sum_i \mathbf{A}_i\mathbf{B}_i^{\top} + \sum_{i\neq j}\mathbf{A}_i\mathbf{B}_j^{\top} \tag{6.8}$$

The cross-terms $\mathbf{A}_i\mathbf{B}_j^{\top}$ $(i \neq j)$ represent combinations of the down-projection from module i with the up-projection from module j, components that were never trained to work together. These cross-terms are absent in output averaging, which only combines the trained transformations $\mathbf{A}_i\mathbf{B}_i^{\top}$ before averaging. Furthermore, averaging the factors separately results in an effective $\frac{1}{N^2}$ scaling of the composed output, compared to $\frac{1}{N}$ for output averaging and no normalisation for output summing. For N modules, weight averaging therefore produces N^2 interaction terms while output averaging produces only N terms, all of which correspond to trained transformations. The presence of these untrained cross-terms in weight averaging provides a theoretical explanation for why output composition methods consistently outperform weight averaging in our experiments, as discussed in Section 6.5.4.

Theoretical Motivation. The theoretical advantage of output composition over weight composition lies in the preservation of the learned low-rank structure of each individual module. As outlined above, weight averaging introduces untrained cross-

term interactions between module components, potentially destroying the individual low-rank decompositions that encode task-specific information. Output summing and averaging instead compute each module's full transformation before combining the results, preserving the task-specific representational directions learned during fine-tuning. This is consistent with findings from representation engineering and activation steering (Turner et al., 2023; Zou et al., 2023), which demonstrate that model behaviour can be effectively controlled by additively combining learned directions in activation space. Each QLoRA module can therefore be understood as encoding a task-specific steering direction, and output composition as linearly integrating these directions.

This theoretical advantage comes with a computational cost: output summing and averaging require N adapter forward passes for N modules, so inference cost scales linearly with the number of composed modules. Weight averaging, by contrast, collapses all modules into a single block and therefore maintains the same inference cost regardless of N, but as shown in Section 6.5.4, this efficiency consistently comes at the expense of control effectiveness.

6.5.2 Study Overview

The composition experiments presented in this section are designed to assess two properties of QLoRA module composability. The first is **cross-dataset performance retention**: whether composing modules trained on different datasets for the same control attribute, for example, Yelp, IMDb, and SST-2 for sentiment control, can match or exceed the performance of each individual module when evaluated on its own single-attribute test set, demonstrating that composition does not degrade the task-specific capability of the constituent modules. The second is **functional composability**: whether composing single-attribute modules can achieve effective joint control over multiple attributes simultaneously. Together, these two properties capture the extent to which a fully plug-and-play composition framework is realisable in practice.

To assess both properties, we train QLoRA modules on individual datasets, compose them using each of the three techniques described in Section 6.5.1, and evaluate the composed models on both individual and composite tasks. For each pretrained host model Θ, set of individual datasets $\mathbf{D} = \{D_1, \ldots, D_m\}$, and set of composition techniques $\mathbf{C} = \{C_1, \ldots, C_p\}$, results are reported according to the schema shown in Table 6.8.

QLoRA-tuned model (composition)	**# rows**
Raw model Θ	1
+ $\Delta\Theta_{D_i}$ trained on D_i	m
+ $\Delta\Theta_{\mathbf{D}}$ trained on combined $D_1 \cup ...D_m$	1
+ C_1 applied to subsets of $\Delta\Theta_{D_1}$, ...$\Delta\Theta_{D_m}$	$\sum_{k=2}^{m} \binom{m}{k}$
+ C_2 applied to subsets of $\Delta\Theta_{D_1}$, ...$\Delta\Theta_{D_m}$	$\sum_{k=2}^{m} \binom{m}{k}$
+ ...	
+ C_p applied to subsets of $\Delta\Theta_{D_1}$, ...$\Delta\Theta_{D_m}$	$\sum_{k=2}^{m} \binom{m}{k}$

Table 6.8: Study structure in terms of raw model and QLoRA (composed) modules. D_i = individual datasets; C_i = composition techniques. For other details, see text.

This study structure provides multiple points of comparison in terms of the performance of (i) the raw model on each dataset; (ii) the raw model + QLoRA module finetuned on each of the individual datasets, tested (a) on the data set it was finetuned on, and (b) on other related datasets; (iii) the raw model + QLoRA module finetuned on all individual datasets combined, tested on each separate data set; and (iv) the raw model + different types of composition applied to separately finetuned QLoRA modules, tested on the individual data sets.

6.5.3 Experimental Setup

The composition experiments follow the general experimental setup described in Section 6.3, using the same host models (Section 6.3.1), task construal (Section 6.3.2), and evaluation metrics (Section 6.3.3) as the single-module experiments in Section 6.4. The modules composed in these experiments are the same as the indi-

vidually trained QLoRA modules described in Section 6.4, requiring no additional training or optimisation beyond what has already been described.

We test the three composition techniques from Section 6.5.1 as per the study structure from Table 6.8, evaluating two properties of composability:

1. **Cross-dataset performance retention**: whether composing modules trained on different datasets for the same control attribute can match or exceed the performance of each individual module when evaluated on its own single-attribute test set, demonstrating that composition does not degrade the task-specific capability of the constituent modules. This is assessed separately for **sentiment control**, where subsets of the Yelp, IMDb, and SST-2 trained modules are composed and each composition is evaluated on the test sets of its constituent datasets; and for **topic control**, where the AG News and DBPedia trained modules are composed and evaluated on both the AG News and DBPedia test sets.

2. **Functional composability**: whether composing single-attribute modules trained on different control attributes can achieve effective joint control over **multiple attributes** simultaneously. For this setting, we test two distinct composition strategies: (i) composing the two single-attribute modules trained on the Combined Sentiment and Combined Topic datasets respectively; and (ii) composing all five single-attribute modules trained on the individual datasets simultaneously.

The two strategies for multiple-attribute composition allow us to assess whether the source of the composed modules has a material impact on joint control performance: specifically, whether the broader training signal of the combined-dataset modules offers an advantage over composing a larger number of more narrowly trained individual modules.

Code and results are available on GitHub.[4]

[4] `https://github.com/michelalorandi/plug_and_play_qlora`

6.5.4 Results

6.5.4.1 Sentiment control

Tables 6.9 and 6.10 present diversity, fluency, and control effectiveness results for sentiment control across the three host models, following the row structure introduced in Table 6.8. The single-module rows (i.e. the first five rows in each of the three parts of a table, showing results for raw model, individual dataset modules, and combined dataset module) are carried over from Section 6.4 for ease of direct comparison with the composition results.

Regarding **diversity**, the trends observed for single modules in Section 6.4.5 persist in the composition setting: compositions that include the SST-2 module tend to produce more diverse outputs than those that do not, and compositions that include the IMDb module tend to produce higher fluency scores. Among the composition techniques, weights averaging most closely matches the high diversity of SST-2-trained modules when the SST-2 module is included: for example, Averaged Weights(Yelp, SST-2) achieves the highest Distinct-3 scores among all composition configurations for LLaMA 3 8B (0.71) and LLaMA 3.1 8B (0.74), and the highest Distinct-3 among composition configurations for Mistral 7B (0.57). Output summing and averaging produce more moderate diversity scores that broadly reflect the mixture of constituent modules.

Regarding **fluency**, one consistent trend emerges across all three models: within each composition technique group, the combination of IMDb and Yelp modules consistently achieves the highest SLOR score, for example Output Summing(IMDb, Yelp) achieves 10.41, 10.33, and 10.51 for LLaMA 3 8B, LLaMA 3.1 8B, and Mistral 7B respectively, and Averaged Weights(IMDb, Yelp) achieves 10.83, 10.78, and 11.19. This is consistent with the finding in Section 6.4.5 that IMDb-trained modules produce the highest fluency scores among single modules, and suggests that the fluency advantage of the IMDb module is preserved when it is composed with other modules.

The most notable finding from Table 6.10 is that three-module output sum-

CTG Technique	Distinct-n↑			SLOR↑
	dist-1	dist-2	dist-3	
Llama 3 8B	0.03	0.09	0.12	9.44
+ QLoRA Yelp	0.09	0.24	0.34	9.87
+ QLoRA IMDB	0.04	0.13	0.19	**10.86**
+ QLoRA SST-2	**0.34**	**0.64**	**0.71**	8.24
+ QLoRA Combined Sentiment dataset	0.11	0.27	0.35	10.07
+ QLoRA Output Summing(IMDB, SST-2)	0.15	0.36	0.46	9.43
+ QLoRA Output Summing(IMDB, Yelp)	0.07	0.21	0.31	10.41
+ QLoRA Output Summing(Yelp, SST-2)	0.21	0.46	0.56	8.93
+ QLoRA Output Summing(IMDB, Yelp, SST-2)	0.13	0.34	0.45	9.83
+ QLoRA Output Averaging(IMDB, SST-2)	0.12	0.27	0.35	9.49
+ QLoRA Output Averaging(IMDB, Yelp)	0.05	0.16	0.23	10.30
+ QLoRA Output Averaging(Yelp, SST-2)	0.13	0.31	0.38	9.16
+ QLoRA Output Averaging(IMDB, Yelp, SST-2)	0.07	0.19	0.27	9.77
+ QLoRA Averaged Weights(IMDB, SST-2)	0.31	0.61	0.69	8.37
+ QLoRA Averaged Weights(IMDB, Yelp)	0.04	0.12	0.19	10.83
+ QLoRA Averaged Weights(Yelp, SST-2)	0.33	**0.64**	**0.71**	8.28
+ QLoRA Averaged Weights(IMDB, Yelp, SST-2)	0.11	0.28	0.36	9.54
Llama 3.1 8B	0.04	0.10	0.13	9.66
+ QLoRA Yelp	0.09	0.26	0.36	9.94
+ QLoRA IMDB	0.04	0.14	0.21	**10.81**
+ QLoRA SST-2	**0.36**	**0.69**	**0.76**	8.24
+ QLoRA Combined Sentiment dataset	0.12	0.27	0.35	10.00
+ QLoRA Output Summing(IMDB, SST-2)	0.09	0.23	0.31	9.91
+ QLoRA Output Summing(IMDB, Yelp)	0.07	0.21	0.31	10.33
+ QLoRA Output Summing(Yelp, SST-2)	0.16	0.38	0.47	9.16
+ QLoRA Output Summing(IMDB, Yelp, SST-2)	0.10	0.27	0.37	9.91
+ QLoRA Output Averaging(IMDB, SST-2)	0.08	0.21	0.28	9.81
+ QLoRA Output Averaging(IMDB, Yelp)	0.07	0.22	0.31	10.24
+ QLoRA Output Averaging(Yelp, SST-2)	0.16	0.36	0.44	9.05
+ QLoRA Output Averaging(IMDB, Yelp, SST-2)	0.07	0.18	0.26	10.01
+ QLoRA Averaged Weights(IMDB, SST-2)	0.33	0.64	0.71	8.38
+ QLoRA Averaged Weights(IMDB, Yelp)	0.04	0.14	0.21	10.78
+ QLoRA Averaged Weights(Yelp, SST-2)	0.35	0.67	0.74	8.27
+ QLoRA Averaged Weights(IMDB, Yelp, SST-2)	0.08	0.20	0.27	9.87
Mistral 7B	0.04	0.09	0.12	7.45
+ QLoRA Yelp	0.03	0.07	0.10	10.44
+ QLoRA IMDB	0.02	0.04	0.06	**11.25**
+ QLoRA SST-2	**0.27**	**0.51**	**0.58**	7.65
+ QLoRA Combined Sentiment dataset	0.03	0.06	0.09	10.83
+ QLoRA Output Summing(IMDB, SST-2)	0.22	0.41	0.47	7.80
+ QLoRA Output Summing(IMDB, Yelp)	0.03	0.07	0.10	10.51
+ QLoRA Output Summing(Yelp, SST-2)	0.18	0.35	0.40	8.22
+ QLoRA Output Summing(IMDB, Yelp, SST-2)	0.08	0.16	0.19	9.13
+ QLoRA Output Averaging(IMDB, SST-2)	0.11	0.21	0.25	8.49
+ QLoRA Output Averaging(IMDB, Yelp)	0.02	0.06	0.09	10.80
+ QLoRA Output Averaging(Yelp, SST-2)	0.05	0.10	0.13	8.84
+ QLoRA Output Averaging(IMDB, Yelp, SST-2)	0.02	0.05	0.07	10.54
+ QLoRA Averaged Weights(IMDB, SST-2)	0.25	0.49	0.55	7.69
+ QLoRA Averaged Weights(IMDB, Yelp)	0.02	0.04	0.07	11.19
+ QLoRA Averaged Weights(Yelp, SST-2)	**0.27**	0.50	0.57	7.66
+ QLoRA Averaged Weights(IMDB, Yelp, SST-2)	0.10	0.20	0.24	8.64

Table 6.9: **Sentiment Control:** Diversity and Fluency for the model + QLoRA module combinations explained in Table 6.8. Here, e.g. Output Summing(data1, data2) refers to the output summation module composition technique. All values are averages over 3 runs. Bold (shaded) = (second) highest score in column/section.

ming achieves the highest average **Control Effectiveness** (CE) scores among all composition techniques for both LLaMA models on the in-domain test sets (Avg column), with Output Summing(IMDb, Yelp, SST-2) achieving 85.66% and 87.35% for LLaMA 3 8B and LLaMA 3.1 8B, respectively. This is not the case for all individual columns or for Mistral 7B, where the picture is more mixed, as discussed below. Crucially, for the LLaMA models these scores not only outperform all other composition techniques but also exceed the best individual single-module results

reported in Section 6.4.5, demonstrating that three-module output summing does not merely preserve single-task performance but can improve upon it in this setting. A more detailed analysis of this finding is presented in Section 6.5.4.2.

CTG Technique	Control Effectiveness↑								
	Avg All	Avg	Yelp	IMDB	SST-2	Out-Of-Domain Avg	PPLM S	STS S	STS proc S
Llama 3 8B	59.93	63.70	68.00	57.56	65.56	57.16	64.60	52.83	54.06
+ QLoRA Yelp	85.11	91.44	92.78	90.67	90.89	79.49	88.41	**67.78**	82.28
+ QLoRA IMDB	83.07	92.07	92.11	89.67	94.44	73.61	80.95	62.00	77.89
+ QLoRA SST-2	77.80	85.96	84.11	82.44	91.33	70.73	82.86	57.39	71.94
+ QLoRA Combined Sentiment dataset	82.99	91.07	92.22	92.00	89.00	75.88	87.46	63.50	76.67
+ QLoRA Output Summ(IMDB, SST-2)	82.12	90.85	89.78	89.56	93.22	74.85	88.89	61.72	73.94
+ QLoRA Output Summ(IMDB, Yelp)	83.98	91.22	90.22	89.33	94.11	77.49	87.46	64.44	80.56
+ QLoRA Output Summ(Yelp, SST-2)	81.98	90.74	92.44	90.22	89.56	74.96	**90.00**	58.33	76.56
+ QLoRA Output Summ(IMDB, Yelp, SST-2)	**85.66**	**92.48**	**94.44**	**92.56**	90.44	**79.54**	88.89	67.00	**82.72**
+ QLoRA Output Avg(IMDB, SST-2)	79.29	88.30	86.33	87.44	91.11	71.24	83.65	58.44	71.61
+ QLoRA Output Avg(IMDB, Yelp)	83.70	92.26	91.11	89.67	**96.00**	75.67	86.19	63.17	77.67
+ QLoRA Output Avg(Yelp, SST-2)	81.96	89.74	89.00	88.56	91.67	74.69	84.29	62.61	77.17
+ QLoRA Output Avg(IMDB, Yelp, SST-2)	81.04	89.81	88.78	89.00	91.67	73.03	84.60	60.50	74.00
+ QLoRA Avg Weights(IMDB, SST-2)	77.50	85.93	85.33	81.89	90.56	69.16	77.94	57.50	72.06
+ QLoRA Avg Weights(IMDB, Yelp)	81.96	90.74	91.89	87.22	93.11	73.65	84.13	60.67	76.17
+ QLoRA Avg Weights(Yelp, SST-2)	77.94	85.93	85.33	81.89	90.56	71.44	84.76	57.50	72.06
+ QLoRA Avg Weights(IMDB, Yelp, SST-2)	80.29	90.04	88.89	88.78	92.44	71.17	83.17	58.72	71.61
Llama 3.1 8B	58.32	61.52	66.22	54.22	64.11	56.41	64.29	51.56	53.39
+ QLoRA Yelp	86.68	91.52	91.00	89.00	94.56	**82.93**	91.75	**71.28**	**85.78**
+ QLoRA IMDB	83.46	92.26	89.67	91.44	**95.67**	75.04	85.24	63.33	76.56
+ QLoRA SST-2	78.29	85.48	82.33	82.89	91.22	72.49	84.76	58.17	74.56
+ QLoRA Combined Sentiment dataset	84.75	92.67	94.89	91.44	91.67	77.86	89.52	62.28	81.78
+ QLoRA Output Summ(IMDB, SST-2)	84.98	92.93	93.44	90.89	94.44	78.68	**92.54**	62.78	80.72
+ QLoRA Output Summ(IMDB, Yelp)	85.50	91.52	91.22	90.00	93.33	80.26	89.05	68.06	83.67
+ QLoRA Output Summ(Yelp, SST-2)	85.38	92.48	92.78	91.44	93.22	79.07	89.05	68.94	79.22
+ QLoRA Output Summ(IMDB, Yelp, SST-2)	**87.35**	**94.70**	**96.33**	**92.67**	95.11	80.61	90.16	66.89	84.78
+ QLoRA Output Avg(IMDB, SST-2)	81.14	89.74	89.22	87.22	92.78	72.95	83.02	61.50	74.33
+ QLoRA Output Avg(IMDB, Yelp)	83.45	92.30	93.00	90.22	93.67	75.16	86.03	63.22	76.22
+ QLoRA Output Avg(Yelp, SST-2)	80.54	89.00	89.67	86.89	90.44	73.06	85.08	60.94	73.17
+ QLoRA Output Avg(IMDB, Yelp, SST-2)	79.79	89.15	88.56	87.11	91.78	71.69	85.56	58.39	71.11
+ QLoRA Avg Weights(IMDB, SST-2)	76.87	84.07	81.00	81.89	89.33	71.21	83.97	56.67	73.00
+ QLoRA Avg Weights(IMDB, Yelp)	82.24	91.19	90.67	88.89	94.00	73.57	83.49	61.89	75.33
+ QLoRA Avg Weights(Yelp, SST-2)	76.83	84.07	81.00	81.89	89.33	71.00	83.33	56.67	73.00
+ QLoRA Avg Weights(IMDB, Yelp, SST-2)	81.48	89.67	91.67	86.67	90.67	74.19	85.56	60.72	76.28
Mistral 7B	57.54	60.00	59.33	55.00	65.67	56.23	62.86	52.50	53.33
+ QLoRA Yelp	**84.33**	92.30	**94.78**	**90.56**	91.56	**76.42**	84.60	**61.33**	**83.33**
+ QLoRA IMDB	80.24	90.63	89.67	88.56	93.67	70.38	82.70	58.78	69.67
+ QLoRA SST-2	78.17	85.67	83.11	81.67	92.22	71.87	83.65	55.94	76.00
+ QLoRA Combined Sentiment dataset	78.28	86.30	87.11	85.67	86.11	71.32	83.17	58.22	72.56
+ QLoRA Output Summ(IMDB, SST-2)	79.28	87.11	85.56	85.44	90.33	73.12	86.98	57.11	75.28
+ QLoRA Output Summ(IMDB, Yelp)	82.53	90.44	93.33	86.33	91.67	75.59	86.98	58.28	81.50
+ QLoRA Output Summ(Yelp, SST-2)	80.17	88.26	89.00	82.33	93.44	74.13	**89.68**	59.22	73.50
+ QLoRA Output Summ(IMDB, Yelp, SST-2)	83.06	**92.81**	93.78	89.44	**95.22**	74.09	86.67	58.89	76.72
+ QLoRA Output Avg(IMDB, SST-2)	79.09	88.89	88.56	85.33	92.78	69.93	82.06	58.39	69.33
+ QLoRA Output Avg(IMDB, Yelp)	82.23	92.63	93.56	90.11	94.22	72.35	84.60	56.78	75.67
+ QLoRA Output Avg(Yelp, SST-2)	80.80	90.74	92.44	86.33	93.44	71.96	85.87	60.67	69.33
+ QLoRA Output Avg(IMDB, Yelp, SST-2)	81.31	92.37	91.67	**90.56**	94.89	70.99	84.76	57.67	70.56
+ QLoRA Avg Weights(IMDB, SST-2)	78.22	85.85	82.89	82.22	92.44	71.98	84.60	56.28	75.06
+ QLoRA Avg Weights(IMDB, Yelp)	80.71	91.33	90.22	89.89	93.89	70.72	83.65	59.56	68.94
+ QLoRA Avg Weights(Yelp, SST-2)	78.35	85.85	82.89	82.22	92.44	72.61	86.51	56.28	75.06
+ QLoRA Avg Weights(IMDB, Yelp, SST-2)	79.19	88.26	88.11	84.11	92.56	71.67	86.35	58.00	70.67

Table 6.10: **Sentiment Control:** Control Effectiveness for the model + QLoRA module combinations explained in Table 6.8. Here, e.g. Output Summing(data1, data2) refers to the output summation module composition technique. All values are averages over 3 runs. Bold (shaded) = (second) highest score in column/section; underline = train/test on same dataset.

A clear overall trend emerges whereby output summing consistently achieves the highest CE scores among composition techniques, followed by output averaging, with weights averaging performing worst in the large majority of settings. This

confirms the theoretical motivation presented in Section 6.5.1: preserving the low-rank structure of individual modules through output composition is more effective than merging their weight matrices directly. The one exception is Mistral 7B, where results present a more mixed picture, though output summing still achieves the best or second-best CE scores in out-of-domain testing across all models.

Regarding the comparison between composition and the combined-dataset baseline, output summing matches or exceeds the combined-dataset module in the majority of settings for both LLaMA models. For Mistral 7B, the combined-dataset module is less competitive than observed for the LLaMA models in Section 6.4.5, and composition techniques more frequently match or surpass it, suggesting that module composition is a particularly attractive alternative to combined-dataset training for this model in the SFT setting.

For out-of-domain testing, the Yelp-trained module remains the strongest single-module configuration across all models, as established in Section 6.4.5. However, three-module output summing is competitive with or exceeds the Yelp module in several out-of-domain settings, particularly for LLaMA 3 8B, where on PPLM, Output Summing(IMDb, Yelp, SST-2) achieves a score of 88.89% compared to 88.41% for the Yelp module alone. This indicates that composition not only preserves but can enhance out-of-domain generalisation by combining the complementary strengths of modules trained on different data distributions.

6.5.4.2 Module Composition Analysis: Does Composition Outperform Specialised Modules?

The sentiment control results suggest that three-module output summing not only preserves single-task performance but can improve upon it in some settings. To examine this finding more closely, Table 6.11 presents a direct comparison between single-dataset training and 2-module and 3-module output summing compositions, where each configuration is evaluated on the same individual test dataset D_a as the single-dataset module. This isolates the question of whether composition provides

a genuine advantage over a task-specialised module on its own test set, rather than merely on aggregate averages. The combined dataset module is not included in this comparison because it is trained on all three datasets simultaneously rather than composing separately trained modules: it therefore addresses a different question (what a single module can achieve with access to all training data) and is not directly comparable to the composition configurations shown here, which combine modules that have never seen each other's training data. Its performance relative to the composition techniques is reported in Table 6.10.

	test	training / output summing			
Model	D_a	D_a	D_a, D_b	D_a, D_c	D_a, D_b, D_c
L3-8B	IMDB	89.67	89.33	89.56	**92.56**
	SST2	91.33	89.56	**93.22**	90.44
	Yelp	92.78	90.22	92.44	**94.44**
L3.1-8B	IMDB	91.44	90.00	90.89	**92.67**
	SST2	91.22	93.22	94.44	**95.11**
	Yelp	91.00	91.22	92.78	**96.33**
Mist7B	IMDB	88.56	85.44	86.33	**89.44**
	SST2	92.22	90.33	93.44	**95.22**
	Yelp	**94.78**	89.00	93.33	93.78
Average		91.44	89.81	91.83	**93.33**

Table 6.11: Sentiment control effectiveness (%) comparing single-dataset training with multi-module output summing. Bold=best; grey=second-best.

The results show a very clear trend: three-module output summing outperforms single-dataset training in 7 of the 9 scenarios, achieving an average CE of 93.33% compared to 91.44% for single-task modules. This improvement demonstrates two complementary effects: first, generalisation to the sentiment control task beyond individual datasets, whereby exposure to multiple data distributions during composition apparently broadens the module's coverage of sentiment expressions; and second, an advantage in terms of individual dataset performance that results directly from this generalisation, whereby the richer combined signal improves performance even on the dataset for which the single module was specifically trained.

The two exceptions, LLaMA 3 8B on SST-2 (-0.89%) and Mistral 7B on Yelp (-1.00%), represent marginal differences that do not change the core finding. No-

tably, two-module compositions do not consistently outperform single-dataset training, with an average CE of 89.81% and 91.83% for two-module compositions compared to 91.44% for single-task modules, suggesting that the generalisation benefit scales with the number of modules composed and is most pronounced in the three-module setting. Taken together, these results provide strong evidence that output composition does not merely match specialised module performance but actively enhances it through cross-dataset generalisation, while simultaneously enabling plug-and-play multi-task functionality.

6.5.4.3 Topic control

Tables 6.12 and 6.13 present diversity, fluency, and control effectiveness results for topic control, following the same structure and notation as Tables 6.9 and 6.10. As only two individual topic datasets are available (AG News and DBPedia), the composition results are limited to a single two-module combination per technique, rather than the full range of two- and three-module combinations evaluated for sentiment control.

Diversity scores are generally higher than those observed for sentiment control across all configurations. Output summing consistently achieves the highest diversity scores across all three models, with Distinct-3 scores of 0.80, 0.78, and 0.59 for LLaMA 3 8B, LLaMA 3.1 8B, and Mistral 7B respectively, suggesting that combining the complementary vocabulary of AG News and DBPedia-trained modules encourages more varied outputs. Fluency scores are very similar across all settings and models, with no notable trends emerging, consistent with the pattern observed for sentiment control.

For **control effectiveness**, the AG News-trained module achieves the highest average CE scores on the in-domain test sets (Avg column of Table 6.13) across all three models, with scores of 68.53%, 67.85%, and 69.05% for LLaMA 3 8B, LLaMA 3.1 8B, and Mistral 7B, respectively. For out-of-domain testing, the picture is more mixed: the AG News module achieves the highest out-of-domain average for all three

CTG Technique	Distinct-n↑			SLOR↑
	dist-1	dist-2	dist-3	
Llama 3 8B	0.07	0.17	0.22	8.45
+ QLoRA AG News	0.25	0.53	0.62	9.39
+ QLoRA DBPedia	0.32	0.60	0.68	8.96
+ QLoRA Combined Topic dataset	0.30	0.60	0.70	8.92
+ QLoRA Output Summing(AG News, DBPedia)	**0.35**	**0.70**	**0.80**	9.33
+ QLoRA Output Averaging(AG News, DBPedia)	0.27	0.55	0.64	**9.57**
+ QLoRA Averaged Weights(AG News, DBPedia)	0.33	0.62	0.71	8.98
Llama 3.1 8B	0.05	0.12	0.17	**9.45**
+ QLoRA AG News	0.26	0.55	0.65	9.40
+ QLoRA DBPedia	0.31	0.59	0.68	8.74
+ QLoRA Combined Topic dataset	0.31	0.62	0.73	9.29
+ QLoRA Output Summing(AG News, DBPedia)	**0.35**	**0.68**	**0.78**	9.12
+ QLoRA Output Averaging(AG News, DBPedia)	0.30	0.59	0.68	8.98
+ QLoRA Averaged Weights(AG News, DBPedia)	0.32	0.61	0.71	8.74
Mistral 7B	0.07	0.13	0.17	7.98
+ QLoRA AG News	0.18	0.37	0.44	9.48
+ QLoRA DBPedia	**0.26**	0.48	0.56	8.81
+ QLoRA Combined Topic dataset	0.21	0.43	0.52	9.09
+ QLoRA Output Summing(AG News, DBPedia)	**0.26**	**0.50**	**0.59**	9.26
+ QLoRA Output Averaging(AG News, DBPedia)	0.18	0.36	0.42	**9.60**
+ QLoRA Averaged Weights(AG News, DBPedia)	**0.26**	0.48	0.56	8.87

Table 6.12: Diversity and Fluency for **Topic Control**, training on *single* and *combined* datasets, and composition of modules trained on single datasets, e.g. Output Summing(data1, data2). All values are averages over 3 runs. Bold (shaded) = (second) highest score in column and section.

models, but the combined dataset module outperforms it on STS proc T for LLaMA 3 8B (62.58% vs 60.92%) and on STS T for LLaMA 3.1 8B (39.97% vs 36.28%). As discussed in Section 6.4.5, the strong in-domain performance of the AG News module might reflect a combination of classifier bias and label mapping noise rather than a fundamental difference in module quality. To assess the extent of classifier bias specifically, we conduct an additional evaluation using an instruction-tuned LLM as an independent classifier in Section 6.5.5. That evaluation shows that the ranking of configurations, which systems are best and second-best, remains consistent with the results in Table 6.13, and that the CE scores obtained with the LLM classifier correlate strongly with those from the original classifiers, with Pearson's correlations

CTG Technique	Control Effectiveness↑							
	Avg All	Avg	AG News	DBPedia	Out-Of-Domain			
					Avg	PPLM T	STS T	STS proc T
Llama 3 8B	45.47	58.52	64.61	52.42	37.53	48.97	27.97	35.64
+ QLoRA AG News	**68.53**	**85.13**	**90.72**	**79.54**	**58.02**	**71.11**	**42.03**	60.92
+ QLoRA DBPedia	52.61	69.94	71.67	68.21	41.66	55.40	28.58	41.00
+ QLoRA Combined Topic dataset	62.76	74.42	81.61	67.24	56.27	68.73	37.50	**62.58**
+ QLoRA Output Summ(AG News, DBPedia)	63.61	82.57	88.78	76.35	52.59	**71.11**	32.81	53.86
+ QLoRA Output Avg(AG News, DBPedia)	60.97	78.33	83.78	72.88	50.70	66.98	33.81	51.31
+ QLoRA Avg Weights(AG News, DBPedia)	53.90	70.61	72.11	69.12	45.50	66.59	28.06	41.86
Llama 3.1 8B	45.86	58.61	66.11	51.11	37.47	46.35	29.89	36.17
+ QLoRA AG News	**67.85**	**85.52**	**91.33**	**79.72**	**57.21**	**73.10**	36.28	**62.25**
+ QLoRA DBPedia	52.63	69.86	70.94	68.77	41.64	54.92	28.11	41.89
+ QLoRA Combined Topic dataset	65.08	80.65	89.06	72.25	55.01	66.51	**39.97**	58.56
+ QLoRA Output Summ(AG News, DBPedia)	63.41	82.91	88.39	77.44	51.64	69.05	29.39	56.47
+ QLoRA Output Avg(AG News, DBPedia)	58.85	76.42	83.44	69.40	47.74	61.67	29.81	51.75
+ QLoRA Avg Weights(AG News, DBPedia)	52.59	69.52	70.94	68.09	43.06	60.71	27.61	40.86
Mistral 7B	44.14	55.96	62.00	49.91	37.48	49.76	29.92	32.75
+ QLoRA AG News	**69.05**	**88.52**	**93.17**	**83.87**	**57.14**	**73.97**	**39.69**	**57.75**
+ QLoRA DBPedia	52.92	67.05	67.50	66.61	43.76	54.13	32.89	44.28
+ QLoRA Combined Topic dataset	64.64	81.74	89.00	74.47	53.87	67.54	39.22	54.86
+ QLoRA Output Summ(AG News, DBPedia)	65.46	85.27	88.89	81.65	53.19	69.76	36.36	53.44
+ QLoRA Output Avg(AG News, DBPedia)	58.01	76.01	82.00	70.03	46.64	60.95	33.89	45.08
+ QLoRA Avg Weights(AG News, DBPedia)	53.58	66.53	66.06	67.01	46.72	61.75	33.47	44.94

Table 6.13: Control Effectiveness for **Topic Control**, training on *single* and *combined* datasets, and composition of modules trained on single datasets, e.g. Output Summing(data1, data2). All values are averages over 3 runs. Bold (shaded) = (second) highest score in column and section; underline = train and test set from same dataset.

of 0.915–0.955 and Spearman's correlations of 0.883–0.989 across all datasets. This confirms that the overall trends reported here are not an artefact of classifier training data.

Among the composition techniques, output summing achieves the second highest CE scores in the majority of settings across all three models, confirming the consistent trend observed for sentiment control. Specifically, output summing obtains second-best scores in 11 out of the available comparison cases, with the combined-dataset module also achieving second-best scores in 11 cases, indicating that the two approaches are broadly competitive as alternatives to the AG News single module. Output averaging performs consistently below output summing, while weights averaging performs worst among the three composition techniques, again in line with the theoretical expectations established in Section 6.5.1.

It is worth noting that the gap between the best single-module configuration and the composition techniques is more pronounced for topic control than for sentiment control. Unlike the sentiment setting, where three-module output summing was able to exceed the performance of the best single module, the two-module composition

for topic control does not surpass the AG News module in any setting. This is likely attributable to the asymmetry between the two topic modules: the AG News module is substantially stronger than the DBPedia module due to the label mapping noise discussed above, and composing a strong module with a weaker one dilutes rather than amplifies the control signal. This observation motivates the investigation of weighted composition strategies as a direction for future work, discussed further in Section 6.5.6.

6.5.4.4 Multiple-attribute control

The single-attribute results presented in previous sections shed light on the ability of module composition to generalise across individual datasets within the same control attribute. The results presented in this section address the more demanding question of functional composability.

Tables 6.14 and 6.15 present the multiple-attribute control results. In terms of **diversity**, modules trained on the Combined Topic dataset alone achieve the highest scores across all three models, with Distinct-3 scores of 0.59, 0.68, and 0.36 for LLaMA 3 8B, LLaMA 3.1 8B, and Mistral 7B, respectively, while weights averaging of the Combined Sentiment and Combined Topic modules achieves the second-highest diversity scores in most cases. In terms of fluency, all models perform best with the module trained on the Combined Sentiment dataset, achieving SLOR scores of 10.81, 10.73, and 11.60, respectively, consistent with the high-fluency advantage of sentiment-trained modules observed earlier.

For *Control Effectiveness* overall, fine-tuning consistently performs above raw model baseline for the Llama models, by big margins. For Mistral, fine-tuned models in some cases perform worse.

For multiple-control, results present a mixed picture. One perspective is provided by comparing modules trained on the Combined datasets with those composing modules trained on individual datasets. Here, results are dataset and model-dependent: for Llama, Output Summing is best for STS M, Output Averaging is best for PPLM

CTG Technique	Distinct-n↑			SLOR↑
	dist1	dist2	dist3	
Llama 3 8B	0.04	0.11	0.16	9.77
+ QLoRA Combined Sentiment dataset	0.03	0.09	0.14	**10.81**
+ QLoRA Combined Topic dataset	**0.24**	**0.50**	**0.59**	8.52
+ QLoRA Output Summing(S, T)	0.09	0.23	0.31	9.81
+ QLoRA Output Summing(Ind mod)	0.18	0.42	0.51	9.11
+ QLoRA Output Averaging(S, T)	0.10	0.24	0.31	9.26
+ QLoRA Output Averaging(Ind mod)	0.08	0.20	0.26	9.55
+ QLoRA Averaged Weights(S, T)	0.21	0.46	0.55	8.56
+ QLoRA Averaged Weights(Ind mod)	0.09	0.23	0.30	9.45
Llama 3.1 8B	0.05	0.12	0.18	9.84
+ QLoRA Combined Sentiment dataset	0.03	0.11	0.17	**10.73**
+ QLoRA Combined Topic dataset	**0.28**	**0.58**	**0.68**	8.58
+ QLoRA Output Summing(S, T)	0.10	0.27	0.36	9.80
+ QLoRA Output Summing(Ind mod)	0.13	0.29	0.35	8.63
+ QLoRA Output Averaging(S, T)	0.11	0.29	0.37	9.67
+ QLoRA Output Averaging(Ind mod)	0.08	0.22	0.29	9.36
+ QLoRA Averaged Weights(S, T)	0.24	0.52	0.62	8.74
+ QLoRA Averaged Weights(Ind mod)	0.10	0.25	0.33	9.14
Mistral 7B	0.03	0.07	0.10	9.77
+ QLoRA Combined Sentiment dataset	0.01	0.03	0.05	**11.60**
+ QLoRA Combined Topic dataset	**0.12**	**0.27**	**0.36**	8.92
+ QLoRA Output Summing(S, T)	0.09	0.22	0.30	9.50
+ QLoRA Output Summing(Ind mod)	0.09	0.20	0.24	8.94
+ QLoRA Output Averaging(S, T)	0.08	0.18	0.23	9.63
+ QLoRA Output Averaging(Ind mod)	0.02	0.05	0.08	10.75
+ QLoRA Averaged Weights(S, T)	0.10	0.24	0.32	9.04
+ QLoRA Averaged Weights(Ind mod)	0.03	0.07	0.10	10.18

Table 6.14: Diversity and Fluency for **Multi-attribute Control** alongside single-attribute control results for comparison. All values are averages over 3 runs. S=the Combined Sentiment dataset, T=the Combined Topic dataset, M=Multiple, p = processed, Ind mod=composition is on all 5 individually trained modules. Bold (shaded) = (second) highest score in column and section.

M, and Combined dataset training is best for STS p M; for Mistral generally, Weights Averaging is best.

The fine-tuned models in this section were all task-specifically created for multiple-attribute control, so the above are the main results in this section. However, we also wanted to see to what extent the multi-attribute control models retain their single-attribute control performance, shown in the last six columns in Table 6.14. A clear

CTG Technique	Control Effectiveness ↑ Out-Of-Domain									
		Multiple			Sentiment			Topic		
	Avg	PPLM M	STS M	STS p M	PPLM S	STS S	STS p S	PPLM T	STS T	STS p T
Llama 3 8B	19.22	29.29	14.54	20.38	64.60	52.83	54.06	48.97	27.97	35.64
+ QLoRA Combined Sent dataset	**25.66**	37.14	16.50	**30.79**	87.46	**63.50**	**76.67**	51.67	27.69	38.36
+ QLoRA Combined Topic dataset	22.64	36.55	14.96	25.46	55.71	50.78	54.83	**68.73**	**37.50**	**62.58**
+ QLoRA Output Sum(S, T)	23.85	35.60	**17.75**	25.83	**87.62**	59.56	72.72	62.46	29.42	50.25
+ QLoRA Output Sum(Ind mod)	24.08	33.69	16.79	28.00	79.68	59.22	69.39	43.97	25.22	37.75
+ QLoRA Output Avg(S, T)	22.32	36.55	14.50	25.17	76.35	54.56	63.83	57.86	29.39	42.11
+ QLoRA Output Avg(Ind mod)	23.30	**37.50**	15.46	26.17	78.41	57.17	67.61	57.94	26.58	43.22
+ QLoRA Avg Weights(S, T)	22.85	36.55	15.62	25.29	74.92	50.78	54.83	67.70	36.83	59.86
+ QLoRA Avg Weights(Ind mod)	23.58	35.71	15.79	27.12	73.65	54.50	63.17	64.84	32.11	46.33
Llama 3.1 8B	19.95	28.57	15.00	21.88	64.29	51.56	53.39	46.35	29.89	36.17
+ QLoRA Combined Sent dataset	26.44	37.50	17.54	**31.46**	**89.52**	62.28	**81.78**	49.13	30.72	36.86
+ QLoRA Combined Topic dataset	23.95	35.12	17.54	26.46	52.06	51.39	51.83	66.51	**39.97**	58.56
+ QLoRA Output Sum(S, T)	**26.76**	34.40	**19.58**	31.25	75.56	**62.89**	71.83	58.17	34.97	49.75
+ QLoRA Output Sum(Ind mod)	20.78	30.24	16.29	21.96	84.60	60.94	72.94	35.00	24.42	28.92
+ QLoRA Output Avg(S, T)	25.89	**40.83**	16.50	30.04	73.81	59.83	63.72	59.44	29.61	43.08
+ QLoRA Output Avg(Ind mod)	23.17	37.38	15.21	26.17	69.84	54.50	64.89	45.87	27.03	39.42
+ QLoRA Avg Weights(S, T)	24.43	37.98	17.29	26.83	73.49	51.39	51.83	**70.08**	39.61	**61.11**
+ QLoRA Avg Weights(Ind mod)	23.26	40.36	15.46	25.08	72.22	53.61	64.06	50.40	28.28	39.81
Mistral 7B	20.43	26.07	16.12	22.75	62.86	52.50	53.33	49.76	29.92	32.75
+ QLoRA Combined Sent dataset	21.47	30.60	14.75	25.00	83.17	**58.22**	72.56	46.83	27.44	38.33
+ QLoRA Combined Topic dataset	26.06	35.12	20.33	28.62	50.16	51.50	52.33	67.54	39.22	54.86
+ QLoRA Output Sum(S, T)	25.53	34.05	19.58	28.50	59.84	51.33	57.56	63.33	39.75	**60.88**
+ QLoRA Output Sum(Ind mod)	22.77	36.19	16.25	24.58	**87.78**	57.33	**75.83**	57.30	31.44	42.14
+ QLoRA Output Avg(S, T)	22.73	34.88	15.54	25.67	63.17	52.83	57.17	60.87	29.92	43.97
+ QLoRA Output Avg(Ind mod)	23.01	35.95	14.08	27.42	76.19	54.78	66.89	48.17	27.81	37.61
+ QLoRA Avg Weights(S, T)	**26.24**	27.14	**21.25**	**30.92**	59.05	51.50	52.33	**70.00**	**39.92**	55.78
+ QLoRA Avg Weights(Ind mod)	21.70	**36.31**	14.42	23.88	77.46	54.28	62.44	52.86	28.00	38.78

Table 6.15: Control Effectiveness for **Multi-attribute Control** alongside single-attribute control results for comparison. All values are averages over 3 runs. S=the Combined Sentiment dataset, T=the Combined Topic dataset, M=Multiple, p = processed, Ind mod=composition is on all 5 individually trained modules. Bold (shaded) = (second) highest score in column and section; underline = train and test set from same dataset.

trend is that best results are worse than in single-attribute control (Tables 6.9 and 6.12). Which composition methods work best also follows clear trends: for retaining sentiment control performance, it is the module trained on the Combined Topic dataset combined with Weights Averaging. For retaining topic control performance, it is the module trained on the Combined Sentiment dataset combined with Output Summing.

For retention of topic control performance, Weights Averaging yields best or second-best results across all models. However, with Mistral, our new Summing approach achieves comparable or higher performance than Weights Averaging, suggesting that output-based composition is also competitive in this scenario under certain conditions.

6.5.5 Validation of Control Effectiveness Metric for Topic Control

As noted in Section 6.5.4.3, all three classifiers used to compute the CE metric for topic control are trained on AG News, which raises the concern that the evaluation may be biased in favour of systems trained on AG News data. Using a general-purpose instruction-tuned LLM as an independent classifier provides a useful sanity check precisely because, unlike the AG News-trained classifiers, it has not been fine-tuned for any topic classification task: its ability to identify topic labels derives from broad pretraining and instruction following rather than from exposure to AG News data specifically, making it less susceptible to the same potential bias. To this end, we evaluate using Command R+ quantised to 4-bit precision,[5] prompted using a simple classification prompt (Table 6.16) following Cohere's recommended template for classification.[6]

System message	You are a helpful assistant that classifies texts by topic.
User message	Classify the following text into one of these topic categories: SPORTS, BUSINESS, WORLD, SCIENCE/TECHNOLOGY. Only reply with one of the possible topic categories. Do not include any other category or text. {text}

Table 6.16: Prompt used for the instruction-tuned LLM classifier (Command R+ quantised to 4-bit precision).

Table 6.17 reports the CE scores obtained using the LLM as classifier across all models and configurations. The rankings of configurations, specifically which settings are identified as best or second-best within each model section, are the same as those obtained using the original classifier ensemble in Tables 6.12 and 6.13 in all but a small number of cases, indicating that the AG News-trained classifiers do not appear to have materially distorted the relative ordering of systems. In particular, the AG News-trained module retains its position as the strongest configuration

[5] https://huggingface.co/CohereForAI/c4ai-command-r-plus-4bit
[6] https://cohere.com/llmu/use-case-patterns#classifying

across all three models, and output summing achieves the second-highest CE scores among composition techniques in the majority of settings under both evaluation setups.

CTG Technique	Control Effectiveness↑							
	Avg All	Avg	AG News	DBPedia	Out-Of-Domain			
					Avg	PPLM	STS	STS proc
Llama 3 8B	54.49	73.22	75.67	70.77	43.25	60.24	27.92	41.58
+ QLoRA AG News	**68.82**	**84.56**	**84.50**	**84.62**	**59.72**	**75.24**	**42.67**	61.25
+ QLoRA DBPedia	59.43	78.31	78.83	77.78	48.21	65.71	31.17	47.75
+ QLoRA Combined Topic dataset	63.51	75.19	75.50	74.87	58.22	75.00	37.00	**62.67**
+ QLoRA Output Sum(AG News, DBPedia)	64.44	83.02	84.33	81.71	53.87	72.86	35.17	53.58
+ QLoRA Output Avg(AG News, DBPedia)	63.05	81.60	81.67	81.54	52.76	72.62	32.83	52.83
+ QLoRA Avg Weights(AG News, DBPedia)	60.64	79.31	80.50	78.12	50.58	71.67	31.25	48.83
Llama 3.1 8B	53.08	68.65	71.83	65.47	44.81	62.86	32.92	38.67
+ QLoRA AG News	**68.09**	**84.38**	**85.00**	83.76	**58.41**	**73.57**	36.92	**64.75**
+ QLoRA DBPedia	59.17	76.73	75.17	78.29	49.34	67.86	32.42	47.75
+ QLoRA Combined Topic dataset	66.31	82.60	83.50	81.71	56.35	70.48	**39.17**	59.42
+ QLoRA Output Sum(AG News, DBPedia)	64.68	83.64	82.83	**84.44**	53.45	71.19	30.50	58.67
+ QLoRA Output Avg(AG News, DBPedia)	62.43	81.17	82.00	80.34	51.48	69.52	30.08	54.83
+ QLoRA Avg Weights(AG News, DBPedia)	58.20	75.89	74.17	77.61	48.51	67.86	30.92	46.75
Mistral 7B	49.63	65.81	66.50	65.13	42.48	66.43	29.00	32.00
+ QLoRA AG News	**69.06**	**88.63**	**87.00**	**90.26**	**56.89**	73.10	**40.75**	**56.83**
+ QLoRA DBPedia	59.00	72.41	72.00	72.82	51.67	66.43	35.33	53.25
+ QLoRA Combined Topic dataset	65.04	82.62	82.33	82.91	54.39	70.00	37.08	56.08
+ QLoRA Output Sum(AG News, DBPedia)	66.01	83.98	83.00	84.96	56.08	**75.48**	39.75	53.00
+ QLoRA Output Avg(AG News, DBPedia)	60.81	80.34	80.17	80.51	49.73	69.76	33.00	46.42
+ QLoRA Avg Weights(AG News, DBPedia)	59.13	72.41	72.17	72.65	52.23	68.10	35.17	53.42

Table 6.17: Control Effectiveness calculated using an LLM as classifier for **Topic Control**, training on *single* and *combined* datasets, and composition of modules trained on single datasets Sum(data1, data2), Average(data1, data2), and Weights Average(data1, data2). Bold (shaded) = (second) highest score in column and section; underline = train and test set from same dataset.

To quantify the agreement between the two evaluation setups, we compute Pearson's and Spearman's correlation coefficients between the CE scores obtained from the instruction-tuned LLM and those from the original classifier ensemble.

Dataset	Pearson	Spearman
AG News	0.915 (0.000)	0.944 (0.000)
DBPedia	0.955 (0.000)	0.989 (0.000)
PPLM Prompt	0.917 (0.000)	0.928 (0.000)
STS	0.919 (0.000)	0.883 (0.000)
STS proc	0.941 (0.000)	0.944 (0.000)

Table 6.18: Pearson's and Spearman's correlations between the LLM as classifier and the other three classifiers used; p-value in parentheses.

The results in Table 6.18 show strong correlations across all datasets, with Pearson's correlation ranging from 0.915 for AG News to 0.955 for DBPedia, and Spearman's correlation ranging from 0.883 for STS to 0.989 for DBPedia. These high

correlation values confirm that the instruction-tuned LLM behaves consistently with the original classifiers, reinforcing the reliability of the CE metric as used throughout this chapter and indicating that the AG News classifier bias does not materially affect the conclusions drawn from the topic control results.

6.5.6 Limitations

The experiments reported in this section are subject to some limitations.

Model scope. The composition experiments are conducted on three pretrained models of similar parameter count, layer depth, and architecture type. This controlled setting allows the effects of composition to be isolated, but means that the findings may not generalise to models with substantially different architectures, parameter scales, or numbers of layers.

Task and attribute scope. The evaluation covers two control attributes, sentiment and topic, using datasets from related domains. This focused scope allows a systematic comparison of composition techniques under controlled conditions, but leaves open the question of whether the observed patterns extend to other attributes.

Composition scale. The experiments compose between 2 and 5 modules trained on related tasks within the same domain. Scaling to larger numbers of modules, or to modules trained on heterogeneous tasks, introduces challenges not addressed here. These include the linear growth of inference cost with the number of active modules, the potential for output summing to push activation magnitudes outside the range seen during training as N increases, interference effects between modules trained on unrelated tasks, and the absence of a mechanism for selecting which modules to activate for a given input. Addressing these challenges would likely require additional components such as dynamic module selection, normalisation schemes, or learned weighting strategies, which are left for future work.

Training data scale. The stratified sampling procedure described in Section 6.4.1 ensures that all modules are trained on datasets of equal size, enabling fair comparison across composition techniques. However, this design does not cap-

ture how composition behaves when the constituent modules have been trained on datasets of different sizes, which is a common scenario in practice.

PEFT technique. This section investigates composition using QLoRA exclusively. This choice is motivated by the consistent finding, both in the literature (Hu et al., 2022) and in our own experiments in Chapter 5, that LoRA-based methods outperform alternative PEFT techniques such as adapters and prefix tuning for text generation tasks. Nevertheless, whether the composition techniques introduced here generalise to other PEFT methods, or to settings where modules produced by different PEFT techniques are composed together, is not explored and represents a direction for future investigation.

6.5.7 Summary of Module Composition

This section investigated whether independently trained single-attribute QLoRA modules can be composed at inference time to achieve two properties of composability: generalisation over related tasks and functional composability for multiple-attribute control. The results across three host models, five datasets, and three composition techniques yield several clear take-home findings.

Output summing achieves the highest control effectiveness scores among composition techniques for sentiment and topic control in the large majority of settings. Notably, three-module output summing not only preserves but improves upon single-module performance, achieving an average sentiment CE of 93.33% compared to 91.44% for single-task modules. This cannot be attributed to access to more training data, since modules are combined at inference time with no additional training; it instead reflects cross-dataset generalisation across complementary data distributions. For single-attribute control, weight averaging consistently underperforms output-based methods, explained by the untrained cross-term interactions introduced when low-rank factors are averaged separately. However, this picture partially reverses for multiple-attribute control, where weight averaging yields the best or second-best topic control retention results across all models, indicating that the relative merits

of the techniques are task-dependent.

For multiple-attribute control, the picture is more nuanced. No single composition technique dominates across all models and evaluation settings, and overall performance is considerably lower than in single-attribute control, reflecting the inherent difficulty of satisfying multiple attribute constraints simultaneously. Nevertheless, module composition was shown to successfully combine the functionalities of modules trained on different task types, and provides a viable and practically attractive alternative to training a dedicated multiple-attribute module from scratch, since it requires no additional training and can be applied to any combination of pre-existing single-attribute modules.

These findings have a direct bearing on the broader thesis narrative. The success of output summing demonstrates that the task-specific knowledge encoded in independently trained PEFT modules is sufficiently self-contained to be combined additively in activation space, realising the plug-and-play vision described in Section 6.1 without any post-composition optimisation. At the same time, the limitations observed for multiple-attribute control, particularly the model-dependent results for Mistral 7B and the weaker topic control retention, identify boundaries of the current approach and motivate directions for future work.

6.6 Cross-Model Module Portability

The single-attribute modules developed in Section 6.4 and composed in Section 6.5 are self-contained parameter blocks that can be attached to a host model directly only when the source and target share the same hidden dimensions, vocabulary, layer depth, and attention head configuration. The first two stages of investigation in this chapter operated within this constraint, training and combining modules on the same host model. This section addresses the third stage: whether a trained module can be transferred to a different host model, one that may differ in any of these architectural properties, without retraining from scratch, by algebraically transforming the adapter weights prior to attachment.

This question has practical relevance. As LLM backbones are updated and new model families emerge, retraining attribute-specialised modules from scratch for each new host model incurs both computational cost and storage overhead. If modules can instead be ported algebraically from one model to another, the need for repeated retraining is reduced.

This section investigates **module portability**: the transfer of a trained LoRA module from its original host model to a different target model without retraining from scratch. Because the adapter matrices are tied to the specific weight dimensions and layer structure of the source model, they cannot be directly attached to a target model with different architectural properties and must first be transformed algebraically, for instance, by projecting the adapter weights into the target model's representation space using embedding-based transformation matrices, and by remapping layers based on representational similarity. The specific framework used is described in Section 6.6.2. We investigate two post-porting regimes: **zero-shot porting**, in which the transferred module is used directly with no additional training, and **few-step porting**, in which a small number of additional fine-tuning steps are performed using 10% of the original training dataset. We do not additionally compare against a no-transfer baseline trained from scratch on this same 10% subset: Sabry and Belz (2024) (Section 3.5) test this comparison in their simpler single-task setting and find that training from scratch on such a small fraction of data performs poorly relative to porting, and we take this finding as established rather than re-testing it in the more complex attribute-control setting studied here.

6.6.1 Study Overview

The portability experiments are structured to assess module transfer across three conditions of increasing difficulty, defined by the relationship between the source and target models in terms of architectural family and parameter count.

Same family, different size. The source and target models share the same architectural family but differ in the number of parameters. This is the least difficult

condition, as the two models share the same tokeniser, attention mechanism, and general layer structure, with differences arising primarily from depth and width. We test this condition by transferring modules between LLaMA 3.2 3B ↔ LLaMA 3.1 8B and Qwen3 4B ↔ Qwen3 8B, in each case in both directions.

Different family, comparable size. The source and target models belong to different architectural families but have similar parameter counts. This condition introduces additional challenges, including vocabulary mismatches, different tokenisation schemes, and potentially different attention configurations, while keeping the representational capacity of the two models broadly comparable. We test this condition with LLaMA 3.2 3B ↔ Qwen3 4B and LLaMA 3.1 8B ↔ Qwen3 8B.

Different family, different size. The most difficult condition: the source and target models belong to different families *and* differ substantially in size. Here, architectural dissimilarity compounds representational capacity mismatch, making algebraic transfer most challenging. We test this condition with LLaMA 3.2 3B ↔ Qwen3 8B and LLaMA 3.1 8B ↔ Qwen3 4B.

The four host models used throughout are LLaMA 3.2 3B (AI@Meta, 2024), LLaMA 3.1 8B (AI@Meta, 2024), Qwen3 4B (Team, 2025), and Qwen3 8B (Team, 2025). For each source model, a single-attribute LoRA module is first trained on the sentiment and topic control datasets described in Section 6.3.1, following the SFT procedure of Section 6.4. Each trained module is then ported to all other host models using the portability framework described in Section 6.6.2, covering all directed transfer pairs within and across families. This yields a full matrix of source-target combinations, enabling a systematic comparison of portability across all three conditions and both transfer regimes.

The core question addressed in this section is whether module portability can serve as a practical alternative to retraining, both in the zero-shot setting, where the ported module must stand on its own with no additional learning, and in the few-step setting, where a specific amount of training investment is applied. The results are presented in Section 6.6.4, following a description of the portability framework

in Section 6.6.2 and the experimental setup in Section 6.6.3.

6.6.2 Portability Framework

The portability framework adapts LoRASuite (Li et al., 2025), a modular approach originally designed to transfer LoRA weights across successive versions of the same model family, and applies it to the attribute-controlled text generation setting described in this chapter. While the original LoRASuite work focuses on same-family, same-architecture upgrades (e.g. LLaMA 2 to LLaMA 3), we adapt if or, and apply it to, the three transfer conditions of increasing difficulty introduced in Section 6.6.1, which additionally include cross-family transfer between the LLaMA and Qwen model families. In the zero-shot porting regime, the framework operates without any retraining on the target model, transferring the LoRA weights algebraically through a pipeline of four sequential steps, shown in pseudo-code in Algorithm 4.

Algorithm 4: Pseudocode of LoRASuite portability pipeline, adapted from Li et al. (2025).

$\mathbf{B}_o, \mathbf{A}_o = \texttt{load\_peft\_weights}(\texttt{ORIGINAL\_LORA\_PATH})$
Step 1: Embedding transformation
$\mathbf{W}_h =$
$\texttt{embedding\_transform}(\text{ORIGINAL\_TOKENIZER}, \text{NEW\_TOKENIZER})$
Step 2: CKA-based layer mapping, where $S_{i,j}$ is the CKA similarity between layer i of the original model and layer j of the upgraded model
$L_{\text{dict}} = \texttt{cka\_layer\_mapping}(\mathbf{S})$
for *each* $(i,\ j)$ *in* L_{dict} **do**
  # Step 3: Attention head mapping, computing the Hungarian-based head mapping between the i-th projection layer of the original model $\mathbf{W}_{o,i}$ and the j-th layer of the new model $\mathbf{W}_{n,j}$
  $\Delta\mathbf{W}_{o,i} = \mathbf{B}_{o,i}\,\mathbf{A}_{o,i}$
  $H_{\text{dict}} = \texttt{hungarian\_head\_mapping}(\mathbf{W}_{o,i},\ \mathbf{W}_{n,j})$ **for** *each* $(h_o,\ h_n)$ *in* H_{dict} **do**
    $\Delta\mathbf{W}_{n,j}^{h_n} = \mathbf{W}_h^{\top}\,\Delta\mathbf{W}_{o,i}^{h_o}\,(\mathbf{W}_{o,i}^{h_o})^{\top}\,\mathbf{W}_h\,\mathbf{W}_{n,j}^{h_n}$
  # Step 4: LoRA weight transformation
  $\mathbf{B}_{n,j},\ \mathbf{A}_{n,j} = \texttt{SVD}(\Delta\mathbf{W}_{n,j})$

6.6.2.1 Step 1: Embedding Transformation

The first step addresses dimensional mismatches between the source and target model's hidden representation spaces, which arise when the two models differ in vocabulary size, hidden size, or both. Based on LoRASuite (Li et al., 2025), we build a linear mapping $\mathbf{W}_h \in \mathbb{R}^{d_{\text{src}} \times d_{\text{tgt}}}$ that converts representations from the source model's input space into the target model's input space. It is computed from the embedding weight matrices of both models as:

$$\mathbf{W}_h = \mathbf{E}_{\text{src}}^{+} \mathbf{E}_{\text{tgt}} \tag{6.9}$$

where $\mathbf{E}_{\text{src}}$ and $\mathbf{E}_{\text{tgt}}$ are the input embedding matrices of the source and target models, respectively, and $\mathbf{E}_{\text{src}}^{+}$ is the Moore-Penrose pseudoinverse of $\mathbf{E}_{\text{src}}$, obtained by solving a least-squares problem. Concretely, this finds the matrix $\mathbf{W}_h$ that makes $\mathbf{E}_{\text{src}}\mathbf{W}_h$ best approximate $\mathbf{E}_{\text{tgt}}$ in the least-squares sense, so that applying $\mathbf{W}_h$ maps source embeddings to the closest matching vectors in the target embedding space. When the two models use different vocabularies, as is the case for cross-family transfer between LLaMA and Qwen, only the shared token subset is used to estimate $\mathbf{W}_h$. If the hidden sizes are identical and the vocabulary fully overlaps, $\mathbf{W}_h$ reduces to the identity matrix and this step is skipped. For up- and down-projection layers whose intermediate dimensions differ, a separate transformation matrix $\mathbf{W}_i$ is derived from the projection weights of both models using the same least-squares approach.

6.6.2.2 Step 2: CKA-Based Layer Mapping

The second step addresses the structural mismatch arising when the source and target models have different numbers of transformer layers. Since different layers capture representations at different levels of abstraction, a simple sequential alignment (mapping source layer i to target layer i) is insufficient when the two models differ in depth. Following Li et al. (2025), we use Centred Kernel Alignment (CKA) (Kornblith et al., 2019) to measure the representational similarity between all pairs of layers across the two models, constructing a similarity matrix $\mathbf{S} \in \mathbb{R}^{L_{\text{src}} \times L_{\text{tgt}}}$ where

$S_{i,j}$ quantifies how similar the representations of source layer i and target layer j are across a shared set of inputs. CKA is computed using the unbiased minibatch HSIC estimator of Nguyen et al. (2021), which makes the similarity values independent of batch size. A calibration corpus of 256 texts from WikiText-2[7] is used to collect the layer activations, running 32 minibatches through both models simultaneously.

Algorithm 5: CKA-based Layer Mapping

Input:
(a) CKA similarity matrix $S \in \mathbb{R}^{l_o \times l_n}$
(b) original and upgrade layer depth l_o and l_n
Output: $path[i][j]$ with the highest total CKA

```
Initialise dp[i][j] ← −∞ for all i, j
Initialise path[i][j] ← 0 for all i, j
for j = 0 to |l_n − l_o| do
    dp[0][j] ← S[0][j]
for i = 1 to l_o − 1 do
    for j = i to i + |l_n − l_o| do
        max_v ← −∞
        max_i ← −1
        for k = i − 1 to j − 1 do
            if dp[i − 1][k] + S[i][j] > max_v then
                max_v ← dp[i − 1][k] + S[i][j]
                max_i ← k
        dp[i][j] ← max_v
        path[i][j] ← max_i
```

Given **S**, the optimal layer mapping is determined by a dynamic programming algorithm (Algorithm 5), which finds the strictly order-preserving assignment of source layers to target layers that maximises the total CKA similarity, subject to a maximum offset constraint Δ_{layer} that limits how far from the diagonal the mapping can deviate. The offset is set to 2 for same-family pairs (where layers are architecturally more similar) and to 4 for cross-family pairs (where greater divergence is expected). The CKA matrix is precomputed offline for each model pair and cached, so it doesn't need to be recomputed for each ported module.

[7] `https://huggingface.co/datasets/Salesforce/wikitext`

6.6.2.3 Step 3: Attention Head Mapping

The third step handles mismatches in the number of attention heads between the source and target models. Within each mapped layer pair (i, j), the attention heads of the source layer are matched to those of the target layer using the Hungarian algorithm (Munkres, 1957), which finds the optimal one-to-one assignment minimising total assignment cost. Each attention head is characterised by two input-independent interaction matrices:

$$\mathbf{W}_{QK}^{h} = \mathbf{W}_{Q}^{h}(\mathbf{W}_{K}^{h})^{\top}, \qquad \mathbf{W}_{VO}^{h} = \mathbf{W}_{V}^{h}(\mathbf{W}_{O}^{h})^{\top} \tag{6.10}$$

where $\mathbf{W}_{Q}^{h}$, $\mathbf{W}_{K}^{h}$, $\mathbf{W}_{V}^{h}$, and $\mathbf{W}_{O}^{h}$ are the query, key, value, and output projection weights of head h respectively. $\mathbf{W}_{QK}^{h}$ captures the attention strength between token pairs, while $\mathbf{W}_{VO}^{h}$ models how attending to specific tokens influences the subsequent hidden state. When the hidden sizes differ across models, the interaction matrices are projected to a common space using $\mathbf{W}_{h}$ before computing cosine similarity. The resulting similarity matrix is passed to the Hungarian algorithm to obtain the optimal head-to-head assignment.

6.6.2.4 Step 4: LoRA Weight Transformation

Given the layer mapping from Step 2 and the head mapping from Step 3, the LoRA weights are transformed algebraically for each mapped layer pair $(i \to j)$ and each mapped head pair $(h_{\text{src}} \to h_{\text{tgt}})$. For attention projection layers, the weight update $\Delta\mathbf{W}$ for the source head is transformed to the target head's representation space as follows:

$$(\Delta\mathbf{W}^{h_{\text{tgt}}})_{\text{tgt}} = \mathbf{W}_{h}^{\top} \cdot (\Delta\mathbf{W}^{h_{\text{src}}})_{\text{src}} \cdot (\mathbf{W}^{h_{\text{src}}})_{\text{src}}^{\top} \cdot \mathbf{W}_{h} \cdot (\mathbf{W}^{h_{\text{tgt}}})_{\text{tgt}} \tag{6.11}$$

incorporating the original pretrained weights of both models to improve numerical stability. For MLP layers, the intermediate-dimension transformation $\mathbf{W}_{i}$ is applied analogously. After all layer and head transformations are completed, a single SVD

step decomposes the transformed weight update $\Delta\mathbf{W}_{\text{tgt}}$ back into its low-rank factors $\mathbf{A}_{\text{tgt}}$ and $\mathbf{B}_{\text{tgt}}$, restoring the LoRA structure required for attachment to the target model. The ported module is then attached to the frozen target model and evaluated without any further training.

The complete portability pipeline is summarised in Algorithm 4. This training-free design is central to the plug-and-play philosophy of the framework: a module trained once on a source model can be ported to any compatible target model without access to the original training data or any gradient computation. In the few-step regime, the same algebraic transfer is performed first, followed by a short fine-tuning phase on the target model.

6.6.3 Experimental Setup

The portability experiments use a dedicated set of single-attribute LoRA modules trained specifically for this section, for two reasons. First, the modules trained in Section 6.4 use QLoRA with 4-bit quantisation, while in the portability framework we use standard LoRA without quantisation, keeping all model weights in full precision. Second, rather than training modules on all dataset combinations explored in Section 6.4, we train on the best-performing dataset per attribute identified in Section 6.4.5: Yelp for sentiment control and AG News for topic control, using the same sampled versions of these datasets described in Section 6.4.1. This choice ensures that the modules entering the portability experiments are as strong as possible, so that any performance loss observed after transfer can be attributed to the porting process rather than to weak source modules. One module per attribute is trained on each of the four host models described in Section 6.2: LLaMA 3.2 3B, LLaMA 3.1 8B, Qwen3 4B, and Qwen3 8B.

The task construal and evaluation setup follow Sections 6.3.2 and 6.3.3, respectively. Given that portability is primarily a question of whether a ported module retains useful task-specific knowledge when transferred to a new host, we evaluate exclusively on the out-of-domain datasets (PPLM Prompts and STS) described in

Section 6.3.1.

LoRA hyperparameters. We set rank $r = 32$, scaling factor $\alpha = 64$, and dropout to 0.1. The down-projection matrix $\mathbf{A}$ is initialised using the Kaiming uniform distribution with $a = \sqrt{5}$, and the up-projection matrix $\mathbf{B}$ is initialised to all zeros. LoRA blocks are attached to all linear layers of the pretrained model, specifically the query, key, value, output, gate, up, and down projections.

Training. All modules are trained on a single NVIDIA A100 80GB GPU using the Paged AdamW optimiser (Loshchilov, 2017) with a cosine learning rate scheduler. The remaining hyperparameters are: learning rate 1×10^{-4}, batch size 16, gradient accumulation steps 8, maximum gradient norm 1.0, warmup ratio 0.05, weight decay 0.05, and maximum sequence length 1024 tokens. Training runs for up to 2 epochs, corresponding to 400 steps for sentiment control and 1,840 steps for topic control, reflecting the different sizes of the Yelp and AG News training sets. Rather than using the final checkpoint, we select the checkpoint with the lowest evaluation loss on the validation set.

Portability conditions. Results are reported for the following configurations:

- **Raw model**: the pretrained model with no LoRA module attached and no fine-tuning of any kind, serving as a lower-bound baseline.
- **Identity**: the trained module attached to the model it was originally trained on, with no porting, serving as an upper-bound reference for the ported results.
- **Zero-shot porting**: the module is ported to the target model using the algebraic transfer described in Section 6.6.2 and used directly for inference with no additional training.
- **Few-step porting**: following algebraic transfer, a small number of additional training steps are performed on the target model, covering 10% of the original training data: 20 steps for sentiment control and 100 steps for topic control.

All experiments are repeated three times with different random seeds (4613, 7097, and 89723), and results are reported as averages across the three runs.

Code and results are available on GitHub.[8]

6.6.4 Results

This section presents the results for the experimental setup described above (Section 6.6.1), covering sentiment and topic control across the four host models and three transfer conditions. For each control attribute, results are reported separately for zero-shot porting, in which no post-porting training is performed, and few-step porting, in which 10% of the training data is used for adaptation after transfer. Each table includes the raw model baseline (no module) and the identity configuration (module on its original host) as reference points for the lower and upper bounds, respectively.

6.6.4.1 Sentiment control

Tables 6.19 and 6.20 present the portability results for sentiment control under zero-shot and few-step porting, respectively, across all four target models and all transfer conditions. In both tables, the identity configuration is displayed using the same arrow notation as the ported results (e.g. LLaMA 3.2 3B → LLaMA 3.2 3B) for consistency, but represents simply the fine-tuned module attached to the model it was originally trained on, with no porting involved. It serves as the upper-bound reference for the ported configurations in each block.

Zero-shot porting. The identity configurations (module attached to its original host model) achieve average Control Effectiveness (CE) scores between 84.1% and 85.2% across all four target models (bold rows in each section of Table 6.19), compared to raw model baselines of 56.0%–68.4% (*no module* rows), confirming that the trained LoRA modules provide a substantial improvement over the untuned models and establishing the reference against which ported results should be interpreted.

[8] `https://github.com/michelalorandi/portable_lora`

Sentiment control zero-step porting									
Setting	**Control Effectiveness ↑**				**Distinct-n ↑**			**SLOR ↑**	**Params**
	Avg all	**PPLM**	**STS**	**STS proc**	**dist-1**	**dist-2**	**dist-3**		
Target: LLaMA 3.2 3B									
LLaMA 3.2 3B *(no module)*	60.6	*71.0*	*58.7*	59.0	0.306	0.720	0.815	*9.77*	—
LLaMA 3.2 3B → LLaMA 3.2 3B	**85.2**	**90.0**	**80.1**	**88.7**	0.263	**0.760**	**0.892**	**10.02**	1.0×
LLaMA 3.1 8B → LLaMA 3.2 3B	*60.8*	68.4	57.7	61.3	0.312	0.727	0.822	9.73	0.4×
Qwen3 4B → LLaMA 3.2 3B	60.7	63.7	58.4	*61.9*	*0.317*	0.729	0.819	9.70	0.8×
Qwen3 8B → LLaMA 3.2 3B	60.3	66.0	58.4	60.2	**0.317**	*0.741*	*0.830*	9.75	0.4×
Target: LLaMA 3.1 8B									
LLaMA 3.1 8B *(no module)*	56.0	62.9	53.2	56.5	*0.326*	**0.764**	*0.867*	9.65	—
LLaMA 3.1 8B → LLaMA 3.1 8B	**84.1**	**90.6**	**78.4**	**87.6**	0.267	0.761	**0.886**	**9.90**	1.0×
LLaMA 3.2 3B → LLaMA 3.1 8B	*57.4*	*63.0*	53.9	*59.0*	0.323	0.744	0.833	9.68	2.7×
Qwen3 8B → LLaMA 3.1 8B	50.0	48.7	50.6	49.9	0.273	0.707	0.812	9.46	1.0×
Qwen3 4B → LLaMA 3.1 8B	57.1	59.4	*55.0*	58.4	**0.329**	*0.761*	0.852	*9.70*	2.0×
Target: Qwen3 4B									
Qwen3 4B *(no module)*	68.4	68.1	65.3	*71.7*	0.078	*0.438*	*0.656*	*9.14*	—
Qwen3 4B → Qwen3 4B	**84.7**	**90.6**	**81.0**	**86.4**	**0.173**	**0.661**	**0.844**	**9.84**	1.0×
Qwen3 8B → Qwen3 4B	68.4	*70.5*	*67.2*	68.9	*0.079*	0.428	0.643	9.11	0.5×
LLaMA 3.2 3B → Qwen3 4B	*68.7*	66.2	66.9	71.3	0.076	0.419	0.634	9.10	1.3×
LLaMA 3.1 8B → Qwen3 4B	68.4	68.4	66.8	70.0	0.079	0.423	0.641	9.06	0.5×
Target: Qwen3 8B									
Qwen3 8B *(no module)*	58.5	*66.2*	56.5	57.8	0.111	0.554	0.739	9.23	—
Qwen3 8B → Qwen3 8B	**85.1**	**93.5**	**80.8**	**86.4**	**0.214**	**0.699**	**0.856**	**9.73**	1.0×
Qwen3 4B → Qwen3 8B	57.9	62.4	58.2	56.1	0.111	0.561	0.736	9.30	2.0×
LLaMA 3.1 8B → Qwen3 8B	56.8	62.2	55.6	56.2	*0.120*	*0.584*	*0.752*	*9.49*	1.0×
LLaMA 3.2 3B → Qwen3 8B	*60.9*	65.9	*58.7*	*61.2*	0.114	0.563	0.738	9.32	2.7×

Table 6.19: Portability results with 0 post-porting learning steps for sentiment control, grouped by target model. Each block shows the raw base model (no module), the identity upper bound, and all ported configurations. CE values are percentages averaged over seeds; Distinct-n and SLOR are averaged over test sets and seeds. Params = size of the target model relative to the source model. **Bold** = best; *Italic* = second best within each block.

Under zero-shot porting, results are close to the raw model baseline across all transfer conditions and target models. For transfers within the LLaMA family, porting from LLaMA 3.1 8B to LLaMA 3.2 3B achieves an average CE of 60.8%, and porting from LLaMA 3.2 3B to LLaMA 3.1 8B achieves 57.4%, compared to raw model baselines of 60.6% and 56.0%, respectively. These scores represent little to no improvement over the untuned models. Cross-family transfers to the LLaMA models show a similar pattern, with CE scores of 50.0%–60.7% across all source models, remaining near the raw model baselines. The transfer from Qwen3 8B to LLaMA 3.1 8B achieves only 50.0%, which falls below the raw model baseline of 56.0%, suggesting some degradation beyond simply failing to transfer the task-specific signal.

For transfers targeting the Qwen models, zero-shot porting similarly produces

results at or near the raw model baseline. When porting to Qwen3 4B, source models achieve CE scores of 68.4%–68.7%, indistinguishable from the Qwen3 4B raw model baseline of 68.4%, indicating that the algebraic transfer introduces no meaningful task-specific signal in this setting. Transfers to Qwen3 8B yield CE scores of 56.8%–60.9%, close to the raw model baseline of 58.5%. The highest ported CE score in the Qwen3 8B section is achieved by LLaMA 3.2 3B → Qwen3 8B at 60.9% (italic in Table 6.19), which is marginally above the baseline but still far below the identity configuration of 85.1%.

Diversity and fluency scores under zero-shot porting remain broadly in line with the raw model baselines across all target models, with no consistent improvement attributable to the ported module. Taken together, the zero-shot porting results indicate that algebraic transfer alone does not reliably preserve the sentiment control capability added by fine-tuning, across any of the transfer conditions tested here.

Few-step porting. The picture changes substantially when post-porting lightweight fine-tuning (LFT) steps are added (Table 6.20). Under the few-step condition, CE scores increase considerably across all transfer conditions and target models compared to zero-shot porting, with average CE scores ranging from 52.4% to 81.3%. This represents a meaningful recovery of the gap between the raw model baseline and the identity configuration in most settings, though the degree of recovery varies considerably across transfer pairs.

For transfers targeting LLaMA 3.2 3B, few-step porting achieves average CE scores of 76.2%–78.7% across all source models, with Qwen3 4B → LLaMA 3.2 3B achieving the highest ported CE of 78.7% (italic in Table 6.20), representing a remaining gap of approximately 6.5 points to the identity configuration of 85.2%. For transfers targeting LLaMA 3.1 8B, most source models achieve CE scores of 80.7%–81.3%, within approximately 3–4 points of the identity configuration of 84.1%. The exception is Qwen3 8B → LLaMA 3.1 8B, which achieves only 52.4% even after few-step porting, falling below the raw model baseline of 56.0%. This outlier result, already observed under zero-shot porting, is not resolved by the lightweight fine-

Sentiment control few-step porting									
Setting	**LFT**	**Control Effectiveness ↑**				**Distinct-n ↑**			**SLOR ↑**
		Avg all	**PPLM**	**STS**	**STS proc**	**dist-1**	**dist-2**	**dist-3**	
Target: LLaMA 3.2 3B									
LLaMA 3.2 3B *(no module)*	✗	60.6	71.0	58.7	59.0	**0.306**	*0.720*	*0.815*	9.77
LLaMA 3.2 3B → LLaMA 3.2 3B	✗	**85.2**	**90.0**	**80.1**	**88.7**	*0.263*	**0.760**	**0.892**	**10.02**
LLaMA 3.1 8B → LLaMA 3.2 3B	✓	76.2	80.3	72.9	78.2	0.187	0.642	0.812	*9.84*
Qwen3 4B → LLaMA 3.2 3B	✓	*78.7*	80.5	*74.9*	*81.9*	0.184	0.636	0.810	9.83
Qwen3 8B → LLaMA 3.2 3B	✓	77.4	*82.7*	73.9	78.9	0.186	0.640	0.812	9.82
Target: LLaMA 3.1 8B									
LLaMA 3.1 8B *(no module)*	✗	56.0	62.9	53.2	56.5	**0.326**	**0.764**	*0.867*	9.65
LLaMA 3.1 8B → LLaMA 3.1 8B	✗	**84.1**	**90.6**	**78.4**	**87.6**	*0.267*	*0.761*	**0.886**	**9.90**
LLaMA 3.2 3B → LLaMA 3.1 8B	✓	80.7	88.1	73.7	*85.1*	0.188	0.643	0.814	*9.81*
Qwen3 8B → LLaMA 3.1 8B	✓	52.4	53.2	50.1	54.3	0.149	0.524	0.706	9.51
Qwen3 4B → LLaMA 3.1 8B	✓	*81.3*	*89.4*	*75.4*	84.3	0.183	0.632	0.805	9.81
Target: Qwen3 4B									
Qwen3 4B *(no module)*	✗	68.4	68.1	65.3	71.7	0.078	0.438	0.656	9.14
Qwen3 4B → Qwen3 4B	✗	**84.7**	**90.6**	**81.0**	**86.4**	**0.173**	**0.661**	**0.844**	9.84
Qwen3 8B → Qwen3 4B	✓	75.7	81.4	72.2	77.3	*0.172*	*0.606*	0.782	9.89
LLaMA 3.2 3B → Qwen3 4B	✓	77.5	83.8	*73.3*	79.6	0.164	0.604	*0.786*	**9.95**
LLaMA 3.1 8B → Qwen3 4B	✓	*78.1*	*86.0*	73.1	*80.2*	0.163	0.601	0.783	*9.92*
Target: Qwen3 8B									
Qwen3 8B *(no module)*	✗	58.5	66.2	56.5	57.8	0.111	0.554	0.739	9.23
Qwen3 8B → Qwen3 8B	✗	**85.1**	**93.5**	**80.8**	**86.4**	**0.214**	**0.699**	**0.856**	9.73
Qwen3 4B → Qwen3 8B	✓	73.6	81.7	69.4	75.1	*0.191*	*0.652*	*0.818*	*9.85*
LLaMA 3.1 8B → Qwen3 8B	✓	*76.0*	*84.1*	69.4	*79.6*	0.176	0.631	0.812	9.79
LLaMA 3.2 3B → Qwen3 8B	✓	72.4	83.3	*69.8*	71.2	0.184	0.643	0.816	**9.88**

Table 6.20: Portability results with 20 post-porting learning steps (10% of training data) for sentiment control grouped by target model. Each block shows the raw base model (no module), the identity upper bound, and all source models ported into the target. All values are averaged over 3 seeds. CE values are percentages; Distinct-n and SLOR are weighted averages across test sets and seeds. **Bold** = best within each transfer condition, *Italic* = second best. LFT = post-porting Lightweight Fine-Tuning.

tuning steps and suggests a fundamental compatibility issue between this particular source-target pair.

For transfers targeting the Qwen models, few-step porting achieves CE scores of 72.4%–76.0% for Qwen3 8B and 75.7%–78.1% for Qwen3 4B across all source models, representing improvements of approximately 12–16 points over the zero-shot condition. These scores remain 7–12 points below the respective identity configurations of 85.1% and 84.7%, indicating that few-step porting partially but not fully recovers the task-specific adaptation quality of the original module. For Qwen3 8B, the highest ported CE is achieved by LLaMA 3.1 8B → Qwen3 8B at 76.0%, while for Qwen3 4B the highest is LLaMA 3.1 8B → Qwen3 4B at 78.1%.

Diversity and fluency scores under few-step porting show a consistent shift relative to the zero-shot condition: Distinct-n scores decrease compared to the raw

model baselines and move closer to the identity configuration levels, while SLOR scores remain broadly stable. This pattern suggests that the post-porting learning steps specialise the model towards the target attribute at the cost of some output diversity, consistent with the diversity-control trade-off observed in Section 6.4.5.

6.6.4.2 Topic control

Tables 6.21 and 6.22 show the portability results for topic control under zero-shot and few-step porting, respectively. The identity configurations achieve CE scores between 56.2% and 75.2% across the four target models, compared to raw model baselines of 26.2%–35.6%, establishing the upper and lower bounds for the ported results, respectively. The gap between these bounds is larger than that observed for sentiment control, reflecting the greater difficulty of topic control as established in Section 6.4.5.

Zero-shot porting. Zero-shot porting (Table 6.21) produces average CE scores that are at or below the raw model baseline across all target models and transfer conditions. Ported module CE scores range from 25.1% to 34.1% across all settings, closely tracking the corresponding raw model baselines of 26.2%–35.6%.

Two settings produce notably low results that fall below their respective raw model baselines. Qwen3 8B ported to LLaMA 3.1 8B achieves a CE of 25.1% with a SLOR of 5.16, well below the LLaMA 3.1 8B raw model baseline of 35.6% and with substantially degraded fluency. Similarly, LLaMA 3.1 8B ported to Qwen3 8B achieves a CE of 25.4% with a SLOR of 7.39, below the Qwen3 8B raw model baseline of 26.2%. Both results suggest that the algebraic transfer introduces degradation beyond simply failing to transfer the task-specific signal, as even the raw model without any module outperforms these ported configurations.

For transfers targeting the Qwen models, ported CE scores of 25.4%–26.3% for Qwen3 8B and 26.8%–27.2% for Qwen3 4B are essentially indistinguishable from their respective raw model baselines of 26.2% and 26.7%, indicating that the algebraic transfer introduces no meaningful task-specific signal in these settings. For

Topic control zero-step porting									
Setting	**Control Effectiveness ↑**				**Distinct-n ↑**			**SLOR ↑**	**Params**
	Avg all	**PPLM**	**STS**	**STS proc**	**dist-1**	**dist-2**	**dist-3**		
Target: LLaMA 3.2 3B									
LLaMA 3.2 3B *(no module)*	33.2	36.9	*30.7*	34.4	*0.314*	*0.750*	*0.840*	9.38	—
LLaMA 3.2 3B → LLaMA 3.2 3B	**74.6**	**85.8**	**63.8**	**81.5**	0.263	0.629	0.723	9.20	1.0×
LLaMA 3.1 8B → LLaMA 3.2 3B	31.8	36.2	29.4	32.7	0.296	0.731	0.823	*9.45*	0.4×
Qwen3 4B → LLaMA 3.2 3B	*33.5*	35.2	30.4	*36.1*	**0.320**	**0.764**	**0.853**	**9.49**	0.8×
Qwen3 8B → LLaMA 3.2 3B	31.7	*37.8*	28.4	32.8	0.306	0.745	0.839	9.39	0.4×
Target: LLaMA 3.1 8B									
LLaMA 3.1 8B *(no module)*	*35.6*	38.3	30.8	*39.4*	*0.304*	**0.748**	**0.843**	*9.35*	—
LLaMA 3.1 8B → LLaMA 3.1 8B	**69.8**	**85.5**	**56.9**	**77.2**	0.267	0.626	0.718	9.14	1.0×
LLaMA 3.2 3B → LLaMA 3.1 8B	34.1	*41.0*	*31.0*	34.8	0.303	0.731	0.822	9.31	2.7×
Qwen3 8B → LLaMA 3.1 8B	25.1	25.6	25.1	25.0	0.054	0.279	0.495	5.16	1.0×
Qwen3 4B → LLaMA 3.1 8B	34.5	40.3	30.9	35.9	**0.308**	*0.742*	*0.835*	**9.43**	2.0×
Target: Qwen3 4B									
Qwen3 4B *(no module)*	26.7	28.6	*25.7*	27.0	0.095	0.506	0.733	9.17	—
Qwen3 4B → Qwen3 4B	**67.0**	**84.8**	**52.6**	**75.2**	**0.275**	**0.642**	*0.737*	9.07	1.0×
Qwen3 8B → Qwen3 4B	*27.2*	29.1	25.4	26.6	*0.123*	*0.555*	**0.768**	**9.26**	0.5×
LLaMA 3.2 3B → Qwen3 4B	26.8	*29.4*	25.3	*27.4*	0.097	0.496	0.722	*9.22*	1.3×
LLaMA 3.1 8B → Qwen3 4B	26.8	29.2	25.4	27.3	0.093	0.491	0.720	9.18	0.5×
Target: Qwen3 8B									
Qwen3 8B *(no module)*	26.2	27.9	25.2	26.6	0.128	*0.634*	**0.798**	*9.39*	—
Qwen3 8B → Qwen3 8B	**55.6**	**77.1**	**38.7**	**65.1**	**0.297**	**0.696**	*0.791*	9.17	1.0×
Qwen3 4B → Qwen3 8B	*26.3*	*27.9*	25.3	*26.7*	0.128	0.623	0.786	9.32	2.0×
LLaMA 3.1 8B → Qwen3 8B	25.4	25.6	25.3	25.5	0.058	0.389	0.607	7.39	1.0×
LLaMA 3.2 3B → Qwen3 8B	26.2	27.3	*25.3*	26.7	*0.132*	0.625	0.785	**9.51**	2.7×

Table 6.21: Portability results with 0 post-porting learning steps for topic control grouped by target model. Each block shows the raw base model (no module), the identity upper bound, and all source models ported into the target. All values are averaged over 3 seeds. CE values are percentages; Distinct-n and SLOR are weighted averages across test sets and seeds. **Bold** = best within each transfer condition, *Italic* = second best.

transfers targeting the LLaMA models, results are similarly at baseline level, with CE scores of 31.7%–33.5% for LLaMA 3.2 3B and 25.1%–34.5% for LLaMA 3.1 8B against raw model baselines of 33.2% and 35.6%, respectively. These results indicate that zero-shot porting provides no meaningful topic control capability in any of the settings tested here.

Few-step porting. Few-step porting (Table 6.22) produces substantial improvements over zero-shot porting across all transfer conditions and target models. Average CE scores under few-step porting range from 26.3% to 59.2%, with the majority of settings falling between 49.6% and 59.2% and recovering a meaningful portion of the gap to the identity configuration.

For transfers targeting LLaMA 3.2 3B, few-step porting achieves CE scores of 54.9%–57.1% across all source models, with LLaMA 3.1 8B → LLaMA 3.2 3B achiev-

Topic control few-step porting									
Setting	**LFT**	**Control Effectiveness ↑**				**Distinct-n ↑**			**SLOR ↑**
		Avg all	**PPLM**	**STS**	**STS proc**	**dist-1**	**dist-2**	**dist-3**	
Target: LLaMA 3.2 3B									
LLaMA 3.2 3B *(no module)*	×	33.2	36.9	30.7	34.4	**0.314**	**0.750**	**0.840**	9.38
LLaMA 3.2 3B → LLaMA 3.2 3B	×	**74.6**	**85.8**	**63.8**	**81.5**	0.263	0.629	0.723	9.20
LLaMA 3.1 8B → LLaMA 3.2 3B	✓	*57.1*	*75.6*	*41.6*	*66.1*	0.270	*0.697*	*0.815*	*9.49*
Qwen3 4B → LLaMA 3.2 3B	✓	55.3	75.2	39.5	64.2	0.264	0.683	0.799	9.42
Qwen3 8B → LLaMA 3.2 3B	✓	54.9	73.6	40.2	63.0	*0.271*	0.695	0.811	**9.50**
Target: LLaMA 3.1 8B									
LLaMA 3.1 8B *(no module)*	×	35.6	38.3	30.8	39.4	**0.304**	**0.748**	**0.843**	**9.35**
LLaMA 3.1 8B → LLaMA 3.1 8B	×	**69.8**	**85.5**	**56.9**	**77.2**	0.267	0.626	0.718	9.14
LLaMA 3.2 3B → LLaMA 3.1 8B	✓	57.5	76.3	41.6	66.8	0.264	0.648	0.754	9.20
Qwen3 8B → LLaMA 3.1 8B	✓	26.3	26.2	26.0	26.7	0.143	0.323	0.399	4.15
Qwen3 4B → LLaMA 3.1 8B	✓	*59.2*	*79.5*	*43.7*	*67.5*	*0.276*	*0.673*	*0.780*	*9.35*
Target: Qwen3 4B									
Qwen3 4B *(no module)*	×	26.7	28.6	25.7	27.0	0.095	0.506	0.733	9.17
Qwen3 4B → Qwen3 4B	×	**67.0**	**84.8**	**52.6**	**75.2**	**0.275**	0.642	0.737	9.07
Qwen3 8B → Qwen3 4B	✓	*50.6*	67.8	*38.4*	*56.9*	0.258	0.682	0.812	9.56
LLaMA 3.2 3B → Qwen3 4B	✓	49.6	*68.5*	37.0	55.5	*0.264*	**0.695**	**0.824**	**9.61**
LLaMA 3.1 8B → Qwen3 4B	✓	50.0	66.5	37.6	56.6	0.264	*0.695*	*0.824*	*9.56*
Target: Qwen3 8B									
Qwen3 8B *(no module)*	×	26.2	27.9	25.2	26.6	0.128	0.634	0.798	9.39
Qwen3 8B → Qwen3 8B	×	**55.6**	**77.1**	38.7	**65.1**	**0.297**	**0.696**	0.791	9.17
Qwen3 4B → Qwen3 8B	✓	51.3	66.3	38.1	59.4	0.254	0.654	0.769	9.40
LLaMA 3.1 8B → Qwen3 8B	✓	*54.4*	*73.7*	**41.1**	61.1	0.254	0.683	**0.814**	*9.49*
LLaMA 3.2 3B → Qwen3 8B	✓	53.6	73.3	*39.0*	*61.4*	*0.268*	*0.693*	*0.813*	**9.54**

Table 6.22: Portability results with 100 post-porting learning steps (10% of training data) for topic control grouped by target model. Each block shows the raw base model (no module), the identity upper bound, and all source models ported into the target. All values are averaged over 3 seeds. CE values are percentages; Distinct-n and SLOR are weighted averages across test sets and seeds. **Bold** = best within each transfer condition, *Italics* = second best. LFT = post-porting Lightweight Fine-Tuning.

ing the highest ported CE of 57.1% (italic in Table 6.22), representing a remaining gap of approximately 17.5 points to the identity configuration of 74.6%. For transfers targeting LLaMA 3.1 8B, most source models achieve CE scores of 57.5%–59.2%, within approximately 10–12 points of the identity configuration of 69.8%. The exception is Qwen3 8B → LLaMA 3.1 8B, which achieves only 26.3% CE even after few-step porting, with a very low SLOR of 4.15, again falling well below the raw model baseline of 35.6%. This outlier, already observed under zero-shot porting, is not resolved by the lightweight fine-tuning steps.

For transfers targeting Qwen3 4B, few-step porting achieves CE scores of 49.6%–50.6% across all source models, compared to the identity configuration of 67.0%, leaving a remaining gap of approximately 16–17 points. For transfers targeting Qwen3 8B, few-step porting achieves CE scores of 51.3%–54.4%, compared to the

identity configuration of 55.6%. The highest ported CE for Qwen3 8B is achieved by LLaMA 3.1 8B → Qwen3 8B at 54.4%, within approximately 1 point of the identity configuration, representing the closest recovery observed in the topic control portability experiments.

Diversity and fluency scores under few-step porting show a pattern consistent with the sentiment control results: Distinct-n scores decrease relative to the raw model baseline and move closer to the identity configuration levels, while SLOR scores remain broadly stable with the exception of the Qwen3 8B → LLaMA 3.1 8B outlier. The overall picture for topic control is that few-step porting provides meaningful but incomplete recovery of task-specific adaptation in most settings, with the Qwen3 8B to LLaMA 3.1 8B transfer pair representing a persistent failure case that neither porting regime resolves.

6.6.5 Limitations

The portability experiments are subject to some limitations. First of all, the evaluation covers only two control attributes, sentiment and topic, using a single training dataset per attribute. Whether the observed portability patterns extend to other attributes, more complex generation tasks, or modules trained on different datasets remains to be established and is left to future work.

The portability framework is applied here to four models across two families, covering three transfer conditions of increasing difficulty. While this design enables a structured comparison across conditions, the number of model pairs tested is limited, and results may not generalise to other model families or to transfers involving larger architectural differences than those considered here.

The few-step porting condition uses a fixed proportion of the original training data (10%) and a fixed number of learning steps for each attribute. The sensitivity of the results to these choices is not explored, and different data scales or learning schedules may yield different outcomes.

6.6.6 Summary of Cross-Model Portability Experiments

This section investigated whether attribute-specialised LoRA modules trained on one host model can be transferred to a different target model using an adaptation of the LoRASuite algebraic portability framework (Li et al., 2025), without retraining from scratch. Results were reported for both sentiment and topic control, across three transfer conditions of increasing difficulty: same-family, different size; cross-family, comparable size; and cross-family, different size.

Under zero-shot porting, transferred modules consistently perform at or near the raw model baseline across all transfer conditions and both control attributes, indicating that algebraic transfer alone does not reliably preserve task-specific adaptation in the settings tested here. This holds for both same-family and cross-family transfers, with two transfer pairs (Qwen3 8B to LLaMA 3.1 8B in both sentiment and topic control) producing results that fall below the raw model baseline after transfer, suggesting that the algebraic transformation introduces some degradation in these cases rather than simply failing to transfer the task-specific signal.

Few-step porting substantially improves performance across all transfer conditions and both control attributes, recovering a meaningful portion of the gap between the raw model baseline and the identity configuration in most settings. For sentiment control, few-step ported modules achieve average CE scores of 72.4%–81.3% across all transfer conditions, compared to identity configurations of 84.1%–85.2%, with the Qwen3 8B $\rightarrow$ LLaMA 3.1 8B transfer pair as an outlier at 52.4%. For topic control, recovery is less complete in most settings, with few-step ported modules reaching 49.6%–59.2% against identity configurations of 55.6%–74.6%, though LLaMA 3.1 8B $\rightarrow$ Qwen3 8B represents an exception, achieving 54.4% against an identity of 55.6%, within approximately 1 point. The Qwen3 8B $\rightarrow$ LLaMA 3.1 8B transfer pair remains an outlier under few-step porting for both attributes, with CE scores of 52.4% for sentiment and 26.3% for topic control, and anomalously low SLOR scores of 9.51 and 4.15, respectively, that few-step adaptation does not resolve.

In the context of **RQ5** (Can attribute-specialised PEFT modules transfer across model architectures?), these results provide partial evidence for the feasibility of cross-model module portability. Zero-shot porting in its current form does not achieve the goal of training-free module reuse across the model pairs tested here. Few-step porting is more promising, demonstrating that a small amount of post-porting adaptation can recover substantial task-specific performance at a fraction of the cost of full retraining, though a residual gap to the identity configuration remains in most settings. Whether this gap can be further reduced, for instance through improved algebraic transfer strategies or more targeted fine-tuning, is a direction for future work.

6.7 Chapter Summary

This chapter investigated three progressive stages of a plug-and-play framework for attribute-controlled text generation using PEFT modules: the design and evaluation of single-attribute specialised modules (Section 6.4), their composition for multiple-attribute control (Section 6.5), and their portability across model architectures (Section 6.6). Together, these investigations address **RQ4** and **RQ5**, providing empirical evidence on the extent to which attribute-specialised PEFT modules can enable effective and reusable controlled text generation.

Single-attribute modules (RQ4.1). QLoRA fine-tuning consistently improves attribute control effectiveness over the raw model baseline across all models, datasets, and control tasks evaluated, with average CE gains of approximately 20–25 points. The choice of training dataset has a notable impact on performance: Yelp-trained modules achieve the strongest sentiment control results, particularly for out-of-domain generalisation, while AG News-trained modules consistently outperform DBPedia-trained ones for topic control, partly due to label mapping noise in the DBPedia data. The effect of combined-dataset training varies across models, benefiting the LLaMA models in most settings but not Mistral 7B, suggesting that the rela-

tionship between training data diversity and adaptation quality is model-dependent. These findings provide evidence towards RQ4.1, showing that attribute-specialised QLoRA modules can achieve effective single-attribute control in open-ended generation settings.

Module composition (RQ4.2). The composition experiments exploit the plug-and-play functionality of our QLoRA reimplementation (Section 6.4.2), which allows multiple independently trained modules to be attached to the same frozen host model simultaneously and composed at inference time using one of three techniques, without any modification to the modules themselves or to the host model weights. The experiments show that output summing is overall the most effective of the three composition techniques evaluated, outperforming output averaging and weight averaging in the large majority of settings for both sentiment and topic control. Notably, three-module output summing achieves an average sentiment CE of 93.33% across all models, compared to 91.44% for single-task modules, demonstrating that composition can improve upon the performance of individually trained modules in this setting. This improvement is attributable to cross-dataset generalisation: composing modules trained on complementary data distributions broadens the coverage of the control attribute beyond what any individual module captures alone. Weight averaging consistently underperforms output-based methods, a finding explained by the untrained cross-term interactions introduced when low-rank factors are averaged separately. For multiple-attribute control, results are more mixed and model-dependent, with no single composition technique dominating across all settings, and overall CE scores considerably lower than in single-attribute control, reflecting the inherent difficulty of simultaneously satisfying multiple attribute constraints. These results provide evidence towards RQ4.2, showing that independently trained single-attribute modules can be composed at inference time without further training to achieve both generalisation over related tasks and a degree of functional multiple-attribute control.

Cross-model portability (RQ5). The portability experiments show that zero-shot algebraic transfer does not reliably preserve task-specific adaptation across the model pairs tested, with ported modules performing at or near the raw model baseline in the large majority of settings. Few-step porting substantially improves this picture, with ported sentiment control modules achieving average CE scores of 72.4%–81.3% against identity configurations of 84.1%–85.2%, and ported topic control modules reaching 49.6%–61.7% against identity configurations of 56.2%–75.2%. Two transfer pairs involving Qwen3 8B and LLaMA 3.1 8B produce anomalously low results that few-step adaptation does not resolve, suggesting compatibility limitations for certain cross-family pairs that warrant further investigation. Taken together, the portability results provide partial evidence towards RQ5: few-step porting demonstrates that modules can be transferred across model architectures at a fraction of the cost of full retraining, but zero-shot transfer in its current form does not yet achieve the goal of training-free module reuse.

Broader implications. Viewed together, the three stages of investigation in this chapter provide a first empirical characterisation of the composability and portability of attribute-specialised PEFT modules for controlled text generation. The results show that the modular structure of LoRA-based adapters supports meaningful plug-and-play behaviour within a single host model: individually trained modules can be composed at inference time to generalise across related tasks and to achieve a degree of multiple-attribute control without additional training. Cross-model portability is feasible with minimal post-porting adaptation, though it currently falls short of fully training-free transfer. These findings contribute to the broader thesis goal of understanding the conditions under which PEFT-based approaches can serve as practical, reusable components for controlled text generation, complementing the evaluation framework developed in Chapter 4 and the single-model PEFT findings of Chapter 5. The limitations identified (the restriction to two control attributes, three composition techniques, and a small set of model pairs) represent concrete directions for extending this work, as discussed in more detail in the following chapter.

Chapter 7

Conclusion

This thesis aimed to address a central challenge in natural language generation: how to achieve reliable, efficient, and reusable controlled text generation (CTG) in large language models. The work was motivated by a set of interconnected limitations in the existing CTG literature, namely the lack of a systematic framework for organising and understanding CTG approaches, the absence of standardised evaluation protocols, the computational cost of full fine-tuning, and the lack of mechanisms for isolating and composing attribute-specific control signals. The research programme was structured around five research questions that progressively built from systematically categorising the CTG landscape, to establishing principled evaluation foundations, to comparing control implementation strategies, and finally to developing and transferring modular, attribute-specialised control mechanisms.

In this concluding chapter, we summarise the principal contributions of the thesis (Section 7.1), revisit and answer each research question in light of our empirical findings (Section 7.2), reflect on the limitations of the work (Section 7.3), outline promising directions for future research (Section 7.4), and close with broader concluding remarks (Section 7.5).

7.1 Contributions

This thesis makes eight main contributions, each summarised below.

1. Multi-dimensional categorisation scheme for CTG techniques. Chapter 2 introduced a systematic three-dimensional categorisation scheme for analysing and comparing CTG approaches. The scheme was developed and validated through manual annotation of 140 papers from the ACL Anthology.

2 Automatic annotation system for CTG papers. To enable the categorisation scheme to scale beyond manual annotation, Chapter 2 additionally developed an LLM-based automatic annotation system combining reasoning-augmented supervised fine-tuning with reinforcement learning refinement. The system achieves annotation quality sufficient for systematic literature analysis while substantially reducing annotation time compared to fully manual approaches.

3. Systematic survey of CTG techniques. Chapter 3 applied the categorisation framework to a structured survey of the state of the art in controlled text generation, organising the literature according to the three-dimensional scheme. This survey situated the contributions of the thesis within the broader research landscape, identified under-explored areas, provided a structured vocabulary for positioning new contributions, and laid the foundations for the empirical chapters of the thesis, particularly the attribute-specialised parameter-efficient fine-tuning (PEFT) approach developed in Chapters 5 and 6.

4. Evaluation framework for controlled text generation. Chapter 4 developed a principled, level-playing-field (LPF) evaluation framework for comparing CTG systems fairly and reproducibly. The framework identifies and systematically addresses four key sources of evaluation bias: system selection, metric selection, dataset selection, and output-generation protocols. It then applies standardised choices at each stage. A meta-evaluation of 29 automatic metrics for semantic consistency assessment in data-to-text generation established which families of metrics are appropriate for content control tasks, showing that traditional textual similarity metrics and pre-Transformer neural metrics are uniformly unsuitable, while

LLM-based and embedding-based metrics such as BLEURT and SBERT provide the strongest correlations with human judgements. As a secondary contribution, the LPF evaluation framework was applied to a comparative evaluation of 12 CTG systems across four control attributes and four datasets, demonstrating that individual scores and system rankings vary substantially with methodological choices, confirming the unreliability of single-classifier control effectiveness assessment, and establishing the unsuitability of perplexity as a fluency measure for LLM-generated text.

5. Empirical comparison of control implementation strategies. Chapter 5 provided an empirical comparison of prompting-based and fine-tuning-based (PEFT) approaches for content-controlled text generation, using data-to-text generation as a controlled testbed. Prompting-based approaches were first evaluated in a multilingual setting through participation in the WebNLG 2023 shared task, where our system achieved first place, firmly establishing the promise of prompting-based approaches to data-to-text generation. Building on this, a systematic comparison using GPT-3.5 for prompting and LoRA, prefix-tuning, and prompt-tuning for parameter-efficient fine-tuning demonstrated that LoRA-adapted Mistral 7B can achieve performance within 1% of GPT-3.5 in terms of BLEU despite a 25-fold difference in parameter count. This result has important implications for resource-constrained settings, demonstrating that effective task specialisation through PEFT can largely compensate for differences in model scale.

6. Training of attribute-specialised PEFT modules. Section 6.4 of Chapter 6 developed and evaluated the training of effective single-attribute QLoRA modules for sentiment and topic control. QLoRA fine-tuning was shown to improve control effectiveness over raw model baselines by approximately 20–25 percentage points consistently across host models (LLaMA 3 8B, LLaMA 3.1 8B, and Mistral 7B) and training datasets. The analysis also revealed that dataset choice has a systematic impact on out-of-domain generalisation, with Yelp-trained modules yielding

the strongest sentiment control results and AG News-trained modules consistently outperforming DBPedia-trained ones for topic control.

7. Strategies for compositional multiple-attribute control. Section 6.5 of Chapter 6 developed and evaluated three composition techniques, namely output summing, output averaging, and weight averaging, for combining independently trained single-attribute modules at inference time, without any joint retraining. Output summing was found to be the consistently strongest technique, and crucially, three-module output summing was shown not merely to preserve single-module performance but actively to improve upon it, achieving an average sentiment control effectiveness of 93.33% compared to 91.44% for individually trained modules. This improvement arises from cross-dataset generalisation: composing modules trained on complementary data distributions broadens the coverage of each control attribute beyond what any single module can achieve independently.

8. Method for cross-model module transfer. Section 6.6 of Chapter 6 investigated whether attribute-specialised LoRA modules can be ported across model architectures using an adaptation of the LoRASuite algebraic transfer framework (Li et al., 2025), covering same-family and cross-family transfer conditions. Zero-shot algebraic transfer alone did not reliably preserve task-specific adaptation. However, few-step porting, applying a small number of gradient updates on 10% of the original training data after algebraic transfer, substantially improved performance, with ported sentiment modules achieving control effectiveness scores of 72.4%–81.3% against identity configurations of 84.1%–85.2%, at a fraction of the cost of full retraining. These findings provide the first empirical characterisation of the conditions under which PEFT modules can be reused across model architectures for CTG.

7.2 Research Questions Revisited

We now revisit the four research questions introduced in Chapter 1 and summarise the answers provided by the empirical work of the thesis.

RQ1: How can we systematically categorise and understand existing controlled text generation approaches?

RQ1.1: How can we develop a multi-dimensional framework for categorising CTG approaches?

The categorisation scheme introduced in Chapter 2 organises CTG approaches along three orthogonal dimensions: **Control attributes**, what is being controlled (e.g. sentiment, topic, or factual content); **Control integration** where control signals enter the system (e.g. input addition, model injection, or system addition); **Control implementation** how control is achieved during generation (e.g. prompting, fine-tuning, or modification of token distribution).

The framework was developed through iterative refinement and validated via manual annotation of 140 papers drawn from the ACL Anthology. The resulting scheme provides a structured vocabulary for describing and comparing CTG systems, identifies under-explored research areas, and situates the contributions of this thesis within the broader landscape of the field.

RQ1.2: Can we effectively automate the categorisation of CTG papers using language models with human validation?

The manual annotation process, while reliable, does not scale to the rapidly growing CTG literature. We therefore developed an LLM-based automatic annotation system combining reasoning-augmented supervised fine-tuning with reinforcement learning refinement, achieving annotation quality sufficient for systematic literature analysis while substantially reducing annotation time compared to fully manual approaches. Applied to 178 papers spanning 2024–2025, the system revealed key trends in the field, including the dominance of single-attribute control and the rise of input-addition and prompting-based methods following the advent of instruction-tuned LLMs. Together, the manual framework and automated system demonstrate

that principled, scalable categorisation of the CTG literature is achievable.

RQ2: How can we effectively evaluate control in text generation systems?

RQ2.1: What evaluation approaches are appropriate for measuring different types of control?

Content control requires that generated text faithfully preserves the meaning of a structured input, making semantic consistency the key quality criterion. Achieving semantic consistency in turn requires metrics sensitive to meaning preservation rather than surface overlap. The meta-evaluation reported in Section 4.3 established that traditional metrics such as BLEU and ROUGE are inadequate for this purpose, and that LLM-based judges and embedding-based metrics (BLEURT, SBERT) provide stronger proxy measures for human judgements, albeit with correlation coefficients that leave room for improvement (best observed $r < 0.73$). For control attributes, such as sentiment and topic, classifier-based control effectiveness (CE) was used as the primary evaluation metric. The LPF framework established in Section 4.4 demonstrated that relying on a single classifier produces unreliable assessments, motivating the use of multiple independent classifiers whose scores are aggregated to mitigate individual classifier bias. Perplexity has been argued in the literature to be an unsuitable fluency measure for LLM-generated text, as it conflates fluency with the model's own training distribution, artificially inflating scores for text that merely resembles the model's training data rather than reflecting genuine linguistic quality. This motivated the use of SLOR, a length- and unigram-normalised alternative, as a more reliable fluency proxy.

RQ2.2: How can we design an evaluation framework that enables fair, reproducible comparison?

The LPF framework introduced in Section 4.4 provides a principled answer to this question. By standardising system selection, metric selection, dataset selection, and output-generation protocols, the framework makes evaluation outcomes comparable across studies. The empirical application of the framework to 12 CTG

techniques demonstrated that performance assessments can vary substantially under the standardised conditions provided by the LPF approach compared to the non-standardised conditions of the original papers, with some systems appearing markedly stronger or weaker than their original publications suggest. The LPF framework is made available[1] as a reusable resource for future CTG evaluation.

RQ3: What control implementation strategies are effective for content-controlled text generation?

RQ3.1: How effective are different prompting strategies for out-of-the-box LLMs in content-controlled generation?

Prompting strategy selection (zero-shot versus few-shot) was found to have limited impact when applied to models that have undergone extensive instruction tuning and reinforcement learning from human feedback (RLHF), such as GPT-3.5 (see Section 5.3.4). The substantial investment in instruction tuning yields robust out-of-the-box performance that prompting variations cannot meaningfully improve. For non-aligned models, however, in-context learning via few-shot prompting was found to be important. In low-resource multilingual settings, pipeline approaches combining a strong pre-trained large language model (LLM) with machine translation achieved competitive performance, though effectiveness depended critically on the quality of the underlying machine translation system.

RQ3.2: How does PEFT compare to prompting-based approaches for enabling content control?

PEFT-based specialisation of smaller open-source models achieved performance comparable to much larger instruction-tuned LLMs (within 1% BLEU of GPT-3.5 for a model 25 times smaller), while offering significant practical advantages in terms of cost, deployment flexibility, and independence from proprietary APIs. The choice between prompting-based and PEFT-based approaches therefore depends less on achievable quality, which appears comparable for well-defined tasks with available training data, and more on practical considerations including resource availability,

[1] `https://github.com/michelalorandi/comparative_study_of_ctg`

deployment constraints, and customisation needs. Among PEFT methods, LoRA substantially outperformed prefix-tuning and prompt-tuning, suggesting that the mechanism and location of parameter adaptation matter considerably for structured generation tasks.

RQ4: Can attribute-specialised PEFT modules enable effective controlled text generation?

RQ4.1: How can we create effective single-attribute PEFT modules?

QLoRA fine-tuning was shown to create effective single-attribute control modules for both sentiment and topic control, improving over raw model baselines by approximately 20–25 percentage points in control effectiveness across all evaluated models and datasets. The results demonstrated that training dataset choice is an important design decision, with out-of-domain generalisation varying significantly across datasets, and that the relationship between training data diversity and adaptation quality is model-dependent.

RQ4.2: Can multiple single-attribute modules be composed to achieve coordinated multiple-attribute control without joint retraining?

Output summing was identified as the most effective composition strategy, outperforming both output averaging and weight averaging across most evaluation settings. Composing three single-attribute modules via output summing achieved higher sentiment control effectiveness than any individual module trained on a single dataset, demonstrating that composition can provide cross-dataset generalisation benefits in addition to enabling multiple-attribute control. For multiple-attribute control, results were more mixed and model-dependent, with no single composition technique dominating across all settings, and overall control effectiveness scores considerably lower than in single-attribute control. Nevertheless, composition without retraining provides a practically attractive alternative to training dedicated multiple-attribute modules, as it preserves the modularity of the framework, avoids the combinatorial explosion of training a separate model for every possible combination of control attributes, and allows new attributes to be added by training a single

additional module rather than retraining from scratch.

RQ5: Can attribute-specialised PEFT modules transfer across model architectures?

RQ5.1: Can PEFT modules trained on one model architecture be effectively applied to different models?

Zero-shot algebraic transfer using the LoRASuite framework did not reliably preserve task-specific adaptation in the settings tested, with ported modules performing at or near the raw model baseline across all transfer conditions. Few-step porting, however, substantially improved performance across all conditions and both control attributes, demonstrating that a small amount of post-porting adaptation can recover a meaningful portion of task-specific performance at a fraction of the cost of full retraining.

RQ5.2: How do architectural differences between source and target models affect module transferability?

Transferability was found to degrade with increasing architectural distance, with same-family different-size transfers yielding the most complete performance recovery under few-step porting, and cross-family transfers generally achieving lower CE scores. One cross-family transfer pair (Qwen3 8B to LLaMA 3.1 8B) produced anomalously low results that few-step adaptation did not resolve, suggesting that compatibility constraints exist for certain cross-family pairs that the current algebraic transfer framework does not fully address. These findings indicate that the conditions under which zero-shot transfer is feasible remain to be characterised, and that cross-family portability currently requires at minimum a small amount of post-porting adaptation.

7.3 Limitations

Each chapter discusses its own limitations in detail; here we summarise the principal limitations of the thesis as a whole, grouped under five broad themes.

Scope of control attributes. The empirical investigations in Chapters 5 and 6 focused on three control attributes: semantic content (via data-to-text generation), sentiment, and topic. These attributes are well-defined, have clear training signal, and can be assessed with automatic metrics. Whether the findings generalise to more complex or less tractable attributes, such as formality gradations, rhetorical style, or creative voice, is an open question. Attributes involving continuous or subjective dimensions, or those for which classifier-based evaluation is not available, may require different training strategies and evaluation approaches.

Scope of PEFT techniques. The composition and portability experiments are conducted exclusively with LoRA, which represents the default and generally best-performing PEFT method currently available, making it the natural choice for this investigation. Nevertheless, the question of whether the composition strategies introduced in Section 6.5 generalise to other PEFT families, such as adapters or prefix tuning, or to settings where modules produced by different PEFT techniques are composed together, remains unexplored. It is possible that the modular properties of LoRA's low-rank decomposition are specific to its parameterisation and may not transfer to other adaptation methods.

Model and composition scale. The composition experiments involve models of broadly similar parameter count and architecture type (decoder-only transformers in the 7B–8B range), and compose between two and five modules trained on related tasks within the same domain. The extent to which the observed composition patterns hold for models of substantially different scale, for modules trained on heterogeneous tasks, or when composing larger numbers of modules, is not addressed. At large composition scales, output summing may push activation magnitudes outside the range seen during training, and the absence of a principled module selection mechanism may introduce interference effects not observed in the controlled settings studied here.

Scope of portability experiments. The portability framework is applied to four models across two model families, covering three transfer conditions of increasing architectural distance and six transfer directions in total. This provides a structured comparison but remains a limited sample of the space of possible model pairings. Results for the Qwen3 8B to LLaMA 3.1 8B transfer pair, a cross-family transfer between two models of comparable parameter count but different architectural lineage, suggest that there may be cross-family configurations that present compatibility challenges the current algebraic transfer framework does not resolve, but the root cause of this failure is not fully characterised. Moreover, the few-step porting condition uses a fixed data proportion (10%) and a fixed number of learning steps; the sensitivity of results to these hyperparameter choices was not explored.

7.4 Future Work

The limitations identified above, together with the broader findings of the thesis, suggest several concrete directions for future research.

Extending the attribute and task scope. The most immediate extension of the empirical work is to apply attribute-specialised PEFT modules to a broader range of control attributes and generation tasks. Attributes such as formality, reading level, discourse structure, or domain-specific style would test whether the training and composition strategies developed here generalise beyond sentiment and topic. Similarly, applying the framework to more open-ended generation tasks, creative writing, dialogue, or long-form document generation, would probe the robustness of the modular approach in less structured settings.

Improving cross-model portability. The partial success of few-step porting motivates further investigation of algebraic transfer strategies, and represents perhaps the most promising avenue for future work in this area. Richer alignment techniques, such as representation alignment between source and target models prior

to transfer, or the use of shared intermediate representations, may improve zero-shot portability and reduce the amount of post-porting adaptation required. An important empirical question is whether the anomalous failure of the Qwen3 8B to LLaMA 3.1 8B transfer pair reflects a general limitation of cross-family transfer at comparable parameter scales, or a specific incompatibility between these architectures that could be addressed through targeted alignment. A deeper challenge underlying portability is the question of encapsulation: for a module to transfer reliably, it should ideally encode all of the task-specific knowledge required for the control attribute, rather than relying on knowledge already present in the host model that may not be available in the target. Strengthening encapsulation, so that attribute knowledge is fully self-contained within the module rather than distributed between the module and its host, is therefore a key objective for future work on portable control mechanisms.

Generalisation to other PEFT methods. While LoRA represents the default and generally best-performing PEFT method currently available, extending the composition and portability framework to other PEFT families would establish whether the plug-and-play properties observed are specific to low-rank decomposition or are a more general property of modular adaptation. Investigating the composition of modules produced by different PEFT techniques, for example combining a LoRA sentiment module with an adapter-based topic module, would be particularly relevant for settings where existing deployed modules have been trained with heterogeneous methods.

LLM-assisted annotation of the CTG literature. The automated annotation system developed in Chapter 2 could be applied to continuously update the systematic survey as the CTG literature grows. Incorporating retrieval-augmented approaches, or fine-tuning the annotation model on new papers as they appear, would allow the categorisation framework to serve as a living knowledge base for the field. This would also enable longitudinal analysis of trends in CTG research,

for example, tracking the adoption of new PEFT methods or the emergence of new control attribute types.

7.5 Concluding Remarks

Controlled text generation sits at the intersection of two pressing demands on modern language technology: the desire for highly capable, flexible generation on the one hand, and the practical need for reliable, efficient, and deployable systems on the other. Large language models have raised the baseline for generation quality, but they have also made the problem of principled control more complex: the sheer scale and opacity of these systems makes it difficult to understand, audit, or systematically adjust their behaviour.

This thesis has approached that challenge from a modular perspective. Rather than treating control as a property of a monolithic system, we have investigated whether it is possible to isolate individual control attributes into independent, reusable components that can be composed and transferred without retraining. The results are encouraging. Attribute-specialised QLoRA modules trained on clearly scoped tasks can (i) achieve strong single-attribute control, (ii) be composed at inference time to broaden generalisation and realise a degree of multiple-attribute control, and (iii) be ported across model architectures through modest post-porting adaptation. These findings are evidence of the feasibility of a plug-and-play paradigm for CTG, in which control mechanisms are developed as reusable components rather than as non-separable aspects of monolithic systems.

At the same time, the work has surfaced important constraints. Multiple-attribute control remains a substantially harder problem than single-attribute control: while single-attribute modules achieve control effectiveness scores consistently above 90%, multiple-attribute control scores fall considerably lower, and the gap between the two highlights the fundamental difficulty of satisfying simultaneous, potentially competing constraints in open-ended generation. Cross-model portability, while feasible with few-step adaptation, falls short of fully training-free reuse, a

gap that points to a deeper question about the extent to which learned control representations capture model-agnostic linguistic properties or remain tied to specific architectures.

The categorisation scheme introduced in Chapter 2 similarly addresses an infrastructural gap, providing the CTG field with a shared vocabulary for describing and situating new contributions. As the CTG landscape continues to grow rapidly, tools for organising and navigating the literature become increasingly valuable, both for researchers seeking to position new work and for practitioners seeking to identify methods appropriate for their particular control requirements.

Underpinning all of the empirical work is the evaluation framework developed in Chapter 4, which addressed a foundational problem in the field: the absence of standardised protocols that prevents fair comparison across systems. Without principled evaluation, it is impossible to determine which approaches genuinely advance the state of the art, and it is easy to mistake methodological artefacts for substantive progress. The LPF framework is perhaps the most broadly applicable contribution of the thesis, as its principles apply not only to future CTG research regardless of the specific control mechanisms under investigation, but extend beyond CTG to any area of NLP research where fair and reproducible comparison of systems is required.

Taken together, the contributions of this thesis represent steps towards a more systematic, principled, and efficient approaches to controlled text generation. The modular framework developed here is not a complete solution, but it demonstrates that attribute-specialised, composable, and transferable control mechanisms are achievable in practice, and it lays the foundation for future work aimed at making controlled generation a reliable and accessible capability across a broad range of language technology applications.

Bibliography

Aghajanyan, A., Gupta, S., and Zettlemoyer, L. (2021). Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 7319–7328.

Ahmad, M. A., Yaramis, I., and Roy, T. D. (2023). Creating trustworthy llms: Dealing with hallucinations in healthcare ai. *arXiv preprint arXiv:2311.01463*.

AI@Meta (2024). Llama 3 model card.

Almazrouei, E., Alobeidli, H., Alshamsi, A., Cappelli, A., Cojocaru, R., Alhammadi, M., Daniele, M., Heslow, D., Launay, J., Malartic, Q., Noune, B., Pannier, B., and Penedo, G. (2023). The falcon series of language models: Towards open frontier models.

Asadi, N., Beitollahi, M., Khalil, Y. H., Li, Y., Zhang, G., and Chen, X. (2024). Combining pre-trained lora modules improves few-shot adaptation of foundation models to new tasks. In *ICML 2024 Workshop on Foundation Models in the Wild*.

Asgari, E., Montaña-Brown, N., Dubois, M., Khalil, S., Balloch, J., Yeung, J. A., and Pimenta, D. (2025). A framework to assess clinical safety and hallucination rates of llms for medical text summarisation. *NPJ digital medicine*, 8(1):274.

Axelsson, A. and Skantze, G. (2023). Using large language models for zero-shot natural language generation from knowledge graphs. In Gatt, A., Gardent, C.,

Cripwell, L., Belz, A., Borg, C., Erdem, A., and Erdem, E., editors, *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 39–54, Prague, Czech Republic. Association for Computational Linguistics.

Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Goldstein, J., Lavie, A., Lin, C.-Y., and Voss, C., editors, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Belz, A., Kow, E., Viethen, J., and Gatt, A. (2009). Generating referring expressions in context: The grec task evaluation challenges. In *Conference of the European Association for Computational Linguistics*, pages 294–327. Springer.

Belz, A., Mille, S., and Thomson, C. (2025). The qcet taxonomy of standard quality criterion names and definitions for the evaluation of nlp systems. *arXiv preprint arXiv:2509.22064*.

Betti, F., Ramponi, G., and Piccardi, M. (2020). Controlled text generation with adversarial learning. In Davis, B., Graham, Y., Kelleher, J., and Sripada, Y., editors, *Proceedings of the 13th International Conference on Natural Language Generation*, pages 29–34, Dublin, Ireland. Association for Computational Linguistics.

Bowman, S. R. and Dahl, G. (2021). What will it take to fix benchmarking in natural language understanding? In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y., editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4843–4855, Online. Association for Computational Linguistics.

Büyükakyüz, K. (2024). Olora: Orthonormal low-rank adaptation of large language models. *arXiv preprint arXiv:2406.01775*.

Carlsson, F., Öhman, J., Liu, F., Verlinden, S., Nivre, J., and Sahlgren, M. (2022). Fine-grained controllable text generation using non-residual prompting. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6837–6857, Dublin, Ireland. Association for Computational Linguistics.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In Bethard, S., Carpuat, M., Apidianaki, M., Mohammad, S. M., Cer, D., and Jurgens, D., editors, *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., and Kurzweil, R. (2018). Universal sentence encoder for English. In Blanco, E. and Lu, W., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Chang, H.-S., Yuan, J., Iyyer, M., and McCallum, A. (2021). Changing the mind of transformers for topically-controllable language generation. In Merlo, P., Tiedemann, J., and Tsarfaty, R., editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2601–2611, Online. Association for Computational Linguistics.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. In *arXiv preprint arXiv:1312.3005*.

Chen, M., Yu, X., Shi, W., Awasthi, U., and Yu, Z. (2023a). Controllable mixed-initiative dialogue generation through prompting. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 951–966, Toronto, Canada. Association for Computational Linguistics.

Chen, Q., Qin, L., Liu, J., Peng, D., Guan, J., Wang, P., Hu, M., Zhou, Y., Gao, T., and Che, W. (2025). Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *CoRR*.

Chen, Q., Wu, W., and Li, S. (2023b). Exploring in-context learning for knowledge grounded dialog generation. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10071–10081, Singapore. Association for Computational Linguistics.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In Palmer, M., Hwa, R., and Riedel, S., editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E., Lam, J., Licht, D., Maillard, J., et al. (2022). No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Cripwell, L., Belz, A., Gardent, C., Gatt, A., Borg, C., Borg, M., Judge, J., Lorandi, M., Nikiforovskaya, A., and Soto Martinez, W. (2023). The 2023 WebNLG shared task on low resource languages. overview and evaluation results (WebNLG 2023). In Gatt, A., Gardent, C., Cripwell, L., Belz, A., Borg, C., Erdem, A., and Erdem, E., editors, *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 55–66, Prague, Czech Republic. Association for Computational Linguistics.

Cui, W., Li, Z., Lopez, D., Das, K., Malin, B. A., Kumar, S., and Zhang, J. (2024). Divide-conquer-reasoning for consistency evaluation and automatic improvement of large language models. In Dernoncourt, F., Preoţiuc-Pietro, D., and Shimorina, A., editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 334–361, Miami, Florida, US. Association for Computational Linguistics.

Dahl, M., Magesh, V., Suzgun, M., and Ho, D. E. (2024). Large legal fictions: Profiling legal hallucinations in large language models. *Journal of Legal Analysis*, 16(1):64–93.

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. (2019). Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

Ding, H., Pang, L., Wei, Z., Shen, H., Cheng, X., and Chua, T.-S. (2023). MacLaSa: Multi-aspect controllable text generation via efficient sampling from compact latent space. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Associ-*

ation for Computational Linguistics: EMNLP 2023, pages 4424–4436, Singapore. Association for Computational Linguistics.

Dou, S., Zhou, E., Liu, Y., Gao, S., Shen, W., Xiong, L., Zhou, Y., Wang, X., Xi, Z., Fan, X., Pu, S., Zhu, J., Zheng, R., Gui, T., Zhang, Q., and Huang, X. (2024). LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945, Bangkok, Thailand. Association for Computational Linguistics.

Dušek, O. and Kasner, Z. (2020). Evaluating semantic accuracy of data-to-text generation with natural language inference. In Davis, B., Graham, Y., Kelleher, J., and Sripada, Y., editors, *Proceedings of the 13th International Conference on Natural Language Generation*, pages 131–137, Dublin, Ireland. Association for Computational Linguistics.

Ethayarajh, K. and Jurafsky, D. (2020). Utility is in the eye of the user: A critique of NLP leaderboards. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4846–4853, Online. Association for Computational Linguistics.

Fang, X., Li, J., Shang, L., Jiang, X., Liu, Q., and Yeung, D.-Y. (2022). Controlled text generation using dictionary prior in variational autoencoders. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 97–111, Dublin, Ireland. Association for Computational Linguistics.

Farhadzadeh, F., Das, D., Borse, S., and Porikli, F. (2025). LoRA-x: Bridging foundation models with training-free cross-model adaptation. In *The Thirteenth International Conference on Learning Representations*.

Feng, W., Hao, C., Zhang, Y., Han, Y., and Wang, H. (2024). Mixture-of-LoRAs: An efficient multitask tuning method for large language models. In Calzolari, N., Kan, M.-Y., Hoste, V., Lenci, A., Sakti, S., and Xue, N., editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 11371–11380, Torino, Italia. ELRA and ICCL.

Ferreira, T. C., Gardent, C., Ilinykh, N., Van Der Lee, C., Mille, S., Moussallem, D., and Shimorina, A. (2020). The 2020 bilingual, bi-directional webnlg+ shared task overview and evaluation results (webnlg+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*.

Gatt, A. and Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Gokaslan, A. and Cohen, V. (2019). Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

González Corbelle, J., Bugarín-Diz, A., Alonso-Moral, J., and Taboada, J. (2022). Dealing with hallucination and omission in neural natural language generation: A use case on meteorology. In Shaikh, S., Ferreira, T., and Stent, A., editors, *Proceedings of the 15th International Conference on Natural Language Generation*, pages 121–130, Waterville, Maine, USA and virtual meeting. Association for Computational Linguistics.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Gu, Y., Feng, X., Ma, S., Wu, J., Gong, H., and Qin, B. (2022a). Improving controllable text generation with position-aware weighted decoding. In Muresan, S.,

Nakov, P., and Villavicencio, A., editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3449–3467, Dublin, Ireland. Association for Computational Linguistics.

Gu, Y., Feng, X., Ma, S., Zhang, L., Gong, H., and Qin, B. (2022b). A distributional lens for multi-aspect controllable text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1023–1043, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Gu, Y., Feng, X., Ma, S., Zhang, L., Gong, H., Zhong, W., and Qin, B. (2023). Controllable text generation via probability density estimation in the latent space. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12590–12616, Toronto, Canada. Association for Computational Linguistics.

Guo, Q., Jin, Z., Qiu, X., Zhang, W., Wipf, D., and Zhang, Z. (2020). CycleGT: Unsupervised graph-to-text and text-to-graph generation via cycle training. In Castro Ferreira, T., Gardent, C., Ilinykh, N., van der Lee, C., Mille, S., Moussallem, D., and Shimorina, A., editors, *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 77–88, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Han, J., Wang, Q., Guo, Z., Xu, B., Zhang, L., and Mao, Z. (2024). Disentangled learning with synthetic parallel data for text style transfer. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15187–15201, Bangkok, Thailand. Association for Computational Linguistics.

Hao, Y., Yu, H., and You, J. (2025). Beyond facts: Evaluating intent hallucination in large language models. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T., editors, *Proceedings of the 63rd Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 7046–7069, Vienna, Austria. Association for Computational Linguistics.

Harkous, H., Groves, I., and Saffari, A. (2020). Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. In Scott, D., Bel, N., and Zong, C., editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2410–2424, Barcelona, Spain (Online). International Committee on Computational Linguistics.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

He, X. (2021). Parallel refinements for lexically constrained text generation with BART. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8653–8666, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hewitt, J., Thickstun, J., Manning, C., and Liang, P. (2023). Backpack language models. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9103–9125, Toronto, Canada. Association for Computational Linguistics.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J. W., and Sifre, L. (2022). Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Hong, L., Luo, P., Blanco, E., and Song, X. (2024). Outcome-constrained large language models for countering hate speech. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N., editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4523–4536, Miami, Florida, USA. Association for Computational Linguistics.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for NLP. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Howcroft, D., Belz, A., Clinciu, M., Gkatzia, D., Hasan, S., Mahamood, S., Mille, S., Santhanam, S., van Miltenburg, E., and Rieser, V. (2020). Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Natural Language Generation Conference*.

Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2022). Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Huang, C., Liu, Q., Lin, B. Y., Pang, T., Du, C., and Lin, M. (2024). Lorahub: Efficient cross-task generalization via dynamic loRA composition. In *First Conference on Language Modeling*.

Huang, X., Liu, Z., Li, P., Li, T., Sun, M., and Liu, Y. (2023). An extensible plug-and-play method for multi-aspect controllable text generation. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15233–15256, Toronto, Canada. Association for Computational Linguistics.

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow,

A., Welihinda, A., Hayes, A., Radford, A., et al. (2024). Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jelinek, F., Mercer, R. L., Bahl, L. R., and Baker, J. K. (1977). Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. (2023). Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. (2023). Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Kalajdzievski, D. (2023). A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*.

Kang, H. and Liu, X.-Y. (2023). Deficiency of large language models in finance: An empirical examination of hallucination. *arXiv preprint arXiv:2311.15548*.

Kann, K., Rothe, S., and Filippova, K. (2018). Sentence-level fluency evaluation: References help, but can be spared! In Korhonen, A. and Titov, I., editors,

Proceedings of the 22nd Conference on Computational Natural Language Learning, pages 313–323, Brussels, Belgium. Association for Computational Linguistics.

Ke, P., Guan, J., Huang, M., and Zhu, X. (2018). Generating informative responses with controlled sentence function. In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1499–1508, Melbourne, Australia. Association for Computational Linguistics.

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Kim, H. and Cho, H. (2023). GTA: Gated toxicity avoidance for LM performance preservation. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14747–14763, Singapore. Association for Computational Linguistics.

Kim, M., Lee, H., Yoo, K. M., Park, J., Lee, H., and Jung, K. (2023). Critic-guided decoding for controlled text generation. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4598–4612, Toronto, Canada. Association for Computational Linguistics.

Kitchenham, B. et al. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. (2019). Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMlR.

Krause, B., Gotmare, A. D., McCann, B., Keskar, N. S., Joty, S., Socher, R., and Rajani, N. F. (2021). GeDi: Generative discriminator guided sequence generation. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Findings of*

the Association for Computational Linguistics: EMNLP 2021, pages 4929–4952, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kumar, S., Paria, B., and Tsvetkov, Y. (2022). Gradient-based constrained sampling from language models. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2251–2277, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Landsman, D., Chen, J. Z., and Zaidi, H. (2022). BeamR: Beam reweighing with attribute discriminators for controllable text generation. In He, Y., Ji, H., Li, S., Liu, Y., and Chang, C.-H., editors, *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pages 422–437, Online only. Association for Computational Linguistics.

Lango, M. and Dusek, O. (2023). Critic-driven decoding for mitigating hallucinations in data-to-text generation. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2853–2862, Singapore. Association for Computational Linguistics.

Laurençon, H., Saulnier, L., Wang, T., Akiki, C., Villanova del Moral, A., Le Scao, T., Von Werra, L., Mou, C., González Ponferrada, E., Nguyen, H., et al. (2022). The bigscience roots corpus: A 1.6 tb composite multilingual dataset. *Advances in Neural Information Processing Systems*, 35:31809–31826.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Levenshtein, V. I. et al. (1966). Binary codes capable of correcting deletions, inser-

tions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2015). A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Li, Y., Meng, F., Zhang, M., Zhu, S., Wang, S., and Xu, M. (2025). LoRASuite: Efficient loRA adaptation across large language model upgrades. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Lialin, V., Deshpande, V., and Rumshisky, A. (2023). Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Liang, X., Wang, H., Song, S., Hu, M., Wang, X., Li, Z., Xiong, F., and Tang, B. (2024a). Controlled text generation for large language model with dynamic attribute graphs. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5797–5814, Bangkok, Thailand. Association for Computational Linguistics.

Liang, X., Wang, H., Wang, Y., Song, S., Yang, J., Niu, S., Hu, J., Liu, D., Yao, S., Xiong, F., et al. (2024b). Controllable text generation for large language models: A survey. *arXiv preprint arXiv:2408.12599*.

Lin, B. Y., Zhou, W., Shen, M., Zhou, P., Bhagavatula, C., Choi, Y., and Ren, X. (2020). CommonGen: A constrained text generation challenge for generative commonsense reasoning. In Cohn, T., He, Y., and Liu, Y., editors, *Findings of*

the Association for Computational Linguistics: EMNLP 2020, pages 1823–1840, Online. Association for Computational Linguistics.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Liu, A., Sap, M., Lu, X., Swayamdipta, S., Bhagavatula, C., Smith, N. A., and Choi, Y. (2021). DExperts: Decoding-time controlled text generation with experts and anti-experts. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online. Association for Computational Linguistics.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. (2024a). Dora: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, pages 32100–32121. PMLR.

Liu, X., Khalifa, M., and Wang, L. (2023a). BOLT: Fast energy-based controlled text generation with tunable biases. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 186–200, Toronto, Canada. Association for Computational Linguistics.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. (2024b). Gpt understands, too. *AI Open*, 5:208–215.

Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., and Zhu, C. (2023b). G-eval: NLG evaluation using gpt-4 with better human alignment. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Liu, Z., Patwary, M., Prenger, R., Prabhumoye, S., Ping, W., Shoeybi, M., and Catanzaro, B. (2022). Multi-stage prompting for knowledgeable dialogue generation. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1317–1337, Dublin, Ireland. Association for Computational Linguistics.

Lorandi, M. and Belz, A. (2023a). Data-to-text generation for severely under-resourced languages with GPT-3.5: A bit of help needed from Google Translate (WebNLG 2023). In Gatt, A., Gardent, C., Cripwell, L., Belz, A., Borg, C., Erdem, A., and Erdem, E., editors, *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*, pages 80–86, Prague, Czech Republic. Association for Computational Linguistics.

Lorandi, M. and Belz, A. (2023b). How to control sentiment in text generation: A survey of the state-of-the-art in sentiment-control techniques. In Barnes, J., De Clercq, O., and Klinger, R., editors, *Proceedings of the 13th Workshop on Computational Approaches to Subjectivity, Sentiment, & Social Media Analysis*, pages 341–353, Toronto, Canada. Association for Computational Linguistics.

Lorandi, M. and Belz, A. (2024). DCU-NLG-PBN at the GEM'24 data-to-text task: Open-source LLM PEFT-tuning for effective data-to-text generation. In Mille, S. and Clinciu, M.-A., editors, *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 76–83, Tokyo, Japan. Association for Computational Linguistics.

Loshchilov, I. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Luo, Y., Lu, M., Liu, G., and Wang, S. (2022). Few-shot table-to-text generation with prefix-controlled generator. In Calzolari, N., Huang, C.-R., Kim, H., Pustejovsky, J., Wanner, L., Choi, K.-S., Ryu, P.-M., Chen, H.-H., Donatelli, L., Ji,

H., Kurohashi, S., Paggio, P., Xue, N., Kim, S., Hahm, Y., He, Z., Lee, T. K., Santus, E., Bond, F., and Na, S.-H., editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6493–6504, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., and Zhang, Y. (2025). An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Madotto, A., Ishii, E., Lin, Z., Dathathri, S., and Fung, P. (2020). Plug-and-play conversational models. In Cohn, T., He, Y., and Liu, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2422–2433, Online. Association for Computational Linguistics.

Maheshwari, A., Jyothi, P., and Ramakrishnan, G. (2024). DictDis: Dictionary constrained disambiguation for improved NMT. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N., editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10991–11004, Miami, Florida, USA. Association for Computational Linguistics.

Majumder, B. P., Jhamtani, H., Berg-Kirkpatrick, T., and McAuley, J. (2022). Achieving conversational goals with unsupervised post-hoc knowledge injection. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3140–3153, Dublin, Ireland. Association for Computational Linguistics.

Mille, S., Dasiopoulou, S., and Wanner, L. (2019). A portable grammar-based nlg system for verbalization of structured data. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1054–1056, New York, NY, USA. Association for Computing Machinery.

Mille, S., Ricci, J., Shvets, A., and Belz, A. (2023). A pipeline for extracting abstract dependency templates for data-to-text natural language generation. In Rambow, O. and Lareau, F., editors, *Proceedings of the Seventh International Conference on Dependency Linguistics (Depling, GURT/SyntaxFest 2023)*, pages 91–101, Washington, D.C. Association for Computational Linguistics.

Mille, S., Sabry, M., and Belz, A. (2024a). DCU-NLG-small at the GEM'24 data-to-text task: Rule-based generation and post-processing with t5-base. In Mille, S. and Clinciu, M.-A., editors, *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 84–91, Tokyo, Japan. Association for Computational Linguistics.

Mille, S., Sedoc, J., Liu, Y., Clark, E., Axelsson, A. J., Clinciu, M. A., Hou, Y., Mahamood, S., Obonyo, I. N., and Zhang, L. (2024b). The 2024 GEM shared task on multilingual data-to-text generation and summarization: Overview and preliminary results. In Mille, S. and Clinciu, M.-A., editors, *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 17–38, Tokyo, Japan. Association for Computational Linguistics.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

Nguyen, T., Raghu, M., and Kornblith, S. (2021). Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*.

Osuji, C. C., Huidrom, R., Adebayo, K. J., Castro Ferreira, T., and Davis, B. (2024). DCU-ADAPT-modPB at the GEM'24 data-to-text generation task: Model hy-

bridisation for pipeline data-to-text natural language generation. In Mille, S. and Clinciu, M.-A., editors, *Proceedings of the 17th International Natural Language Generation Conference: Generation Challenges*, pages 66–75, Tokyo, Japan. Association for Computational Linguistics.

Ou, Y. and Jian, P. (2024). Effective integration of text diffusion and pre-trained language models with linguistic easy-first schedule. In Calzolari, N., Kan, M.-Y., Hoste, V., Lenci, A., Sakti, S., and Xue, N., editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5551–5561, Torino, Italia. ELRA and ICCL.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Pei, J., Yang, K., and Klein, D. (2023). PREADD: Prefix-adaptive decoding for controlled text generation. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10018–10037, Toronto, Canada. Association for Computational Linguistics.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. (2023). The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. (2021). AdapterFusion: Non-destructive task composition for transfer learning. In Merlo, P.,

Tiedemann, J., and Tsarfaty, R., editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Popović, M. (2017). chrF++: words helping character n-grams. In Bojar, O., Buck, C., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., and Kreutzer, J., editors, *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Post, M. (2018). A call for clarity in reporting BLEU scores. In Bojar, O., Chatterjee, R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Monz, C., Negri, M., Névéol, A., Neves, M., Post, M., Specia, L., Turchi, M., and Verspoor, K., editors, *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Poth, C., Sterz, H., Paul, I., Purkayastha, S., Engländer, L., Imhof, T., Vulić, I., Ruder, S., Gurevych, I., and Pfeiffer, J. (2023). Adapters: A unified library for parameter-efficient and modular transfer learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 149–160.

Prabhumoye, S., Black, A. W., and Salakhutdinov, R. (2020). Exploring controllable text generation techniques. *arXiv preprint arXiv:2005.01822*.

Pu, D. and Demberg, V. (2023). ChatGPT vs Human-authored Text: Insights into Controllable Text Summarization and Sentence Style Transfer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 1–18, Toronto, Canada. Association for Computational Linguistics.

Qian, J., Dong, L., Shen, Y., Wei, F., and Chen, W. (2022). Controllable natural

language generation with contrastive prefixes. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2912–2924, Dublin, Ireland. Association for Computational Linguistics.

Qiao, L., Yan, J., Meng, F., Yang, Z., and Zhou, J. (2020). A sentiment-controllable topic-to-essay generator with topic knowledge graph. In Cohn, T., He, Y., and Liu, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3336–3344, Online. Association for Computational Linguistics.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rei, R., Stewart, C., Farinha, A. C., and Lavie, A. (2020). Comet: A neural framework for mt evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Rogers, D. J. and Tanimoto, T. T. (1960). A computer program for classifying plants. *Science*, 132(3434):1115–1118.

Rohatgi, S., Qin, Y., Aw, B., Unnithan, N., and Kan, M.-Y. (2023). The ACL OCL corpus: Advancing open science in computational linguistics. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10348–10361, Singapore. Association for Computational Linguistics.

Sabry, M. and Belz, A. (2023). Peft-ref: A modular reference architecture

and typology for parameter-efficient finetuning techniques. *arXiv preprint arXiv:2304.12410*.

Sabry, M. and Belz, A. (2024). Assessing the portability of parameter matrices trained by parameter-efficient finetuning methods. In Graham, Y. and Purver, M., editors, *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1548–1556, St. Julian's, Malta. Association for Computational Linguistics.

Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. (2018). Relational recurrent neural networks. *Advances in neural information processing systems*, 31.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., et al. (2022). Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Schmidtova, P., Mahamood, S., Balloccu, S., Dusek, O., Gatt, A., Gkatzia, D., Howcroft, D. M., Platek, O., and Sivaprasad, A. (2024). Automatic metrics in natural language generation: A survey of current evaluation practices. In Mahamood, S., Minh, N. L., and Ippolito, D., editors, *Proceedings of the 17th International Natural Language Generation Conference*, pages 557–583, Tokyo, Japan. Association for Computational Linguistics.

Sclar, M., Choi, Y., Tsvetkov, Y., and Suhr, A. (2024). Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, volume 2024, pages 25055–25083.

Sedoc, J., Mille, S., Clinciu, M. A., Liu, Y., Dhole, K., and Mahamood, S. (2025). The 2024 GEM shared task on multilingual data-to-text generation: English and Spanish qualitative evaluation results. In Mille, S., editor, *Proceedings of the 18th International Natural Language Generation Conference: Generation Challenges*, pages 1–36, Hanoi, Vietnam. Association for Computational Linguistics.

Sellam, T., Das, D., and Parikh, A. (2020). BLEURT: Learning robust metrics for text generation. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., et al. (2024). Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Sharma, R., Allen, J., Bakhshandeh, O., and Mostafazadeh, N. (2018). Tackling the story ending biases in the story cloze test. In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 752–757, Melbourne, Australia. Association for Computational Linguistics.

Shimorina, A., Gardent, C., Narayan, S., and Perez-Beltrachini, L. (2018). *WebNLG challenge: Human evaluation results*. PhD thesis, Loria & Inria Grand Est.

Shin, S., Kim, D., and Ko, Y. (2025). ECO decoding: Entropy-based control for controllability and fluency in controllable dialogue generation. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V., editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 28309–28321, Suzhou, China. Association for Computational Linguistics.

Shu, L., Papangelis, A., Wang, Y.-C., Tur, G., Xu, H., Feizollahi, Z., Liu, B., and Molino, P. (2020). Controllable text generation with focused variation. In Cohn, T., He, Y., and Liu, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3805–3817, Online. Association for Computational Linguistics.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical*

Papers, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In Yarowsky, D., Baldwin, T., Korhonen, A., Livescu, K., and Bethard, S., editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Tang, Z., Wang, P., Zhou, K., Li, J., Cao, Z., and Zhang, M. (2023). Can diffusion model achieve better performance in text generation ? bridging the gap between training and inference ! In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11359–11386, Toronto, Canada. Association for Computational Linguistics.

Team, Q. (2025). Qwen3 technical report.

Thomson, C., Reiter, E., and Anya, B. (2024). Common flaws in running human evaluation experiments in nlp. *Computational Linguistics*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Turner, A. M., Thiergart, L., Udell, D., Leech, G., Mini, U., and MacDiarmid, M. (2023). Activation addition: Steering language models without optimization. *CoRR*.

Wang, J., Zhang, C., Zhang, D., Tong, H., Yan, C., and Jiang, C. (2024a). A recent survey on controllable text generation: a causal perspective. *Fundamental Research*.

Wang, R., Ghosh, S., Cox, D., Antognini, D., Oliva, A., Feris, R., and Karlinsky, L. (2024b). Trans-LoRA: towards data-free transferable parameter efficient finetuning. *Advances in Neural Information Processing Systems*, 37:61217–61237.

Wang, Y., Deng, J., Sun, A., and Meng, X. (2022). Perplexity from plm is unreliable for evaluating text quality. *arXiv preprint arXiv:2210.05892*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Whitehouse, C., Huot, F., Bastings, J., Dehghani, M., Lin, C.-C., and Lapata, M. (2024). Low-rank adaptation for multilingual summarization: An empirical study. In Duh, K., Gomez, H., and Bethard, S., editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1202–1228, Mexico City, Mexico. Association for Computational Linguistics.

Xia, F., Liao, M., Fang, Y., Li, D., Xie, Y., Li, W., Li, Y., Xia, D., and Huang, J. (2025). Cross-lora: A data-free lora transfer framework across heterogeneous llms. *arXiv preprint arXiv:2508.05232*.

Xie, Y., Hu, Y., Li, Y., Bi, G., Xing, L., and Peng, W. (2022). Psychology-guided controllable story generation. In Calzolari, N., Huang, C.-R., Kim, H., Pustejovsky, J., Wanner, L., Choi, K.-S., Ryu, P.-M., Chen, H.-H., Donatelli, L., Ji, H., Kurohashi, S., Paggio, P., Xue, N., Kim, S., Hahm, Y., He, Z., Lee, T. K., Santus, E., Bond, F., and Na, S.-H., editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6480–6492, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Xu, C., Li, J., Li, P., and Yang, M. (2023). Topic-guided self-introduction generation for social media users. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11387–11402, Toronto, Canada. Association for Computational Linguistics.

Xu, P., Patwary, M., Shoeybi, M., Puri, R., Fung, P., Anandkumar, A., and Catanzaro, B. (2020). MEGATRON-CNTRL: Controllable story generation with external knowledge using large-scale language models. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2831–2845, Online. Association for Computational Linguistics.

Yadav, P., Tam, D., Choshen, L., Raffel, C. A., and Bansal, M. (2023). Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115.

Yang, E., Bai, C., Xiong, D., Zhang, Y., Meng, Y., Xu, J., and Chen, Y. (2022). Learning structural information for syntax-controlled paraphrase generation. In Carpuat, M., de Marneffe, M.-C., and Meza Ruiz, I. V., editors, *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2079–2090, Seattle, United States. Association for Computational Linguistics.

Yang, E., Liu, M., Xiong, D., Zhang, Y., Meng, Y., Hu, C., Xu, J., and Chen, Y. (2021). Syntactically-informed unsupervised paraphrasing with non-parallel data. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2594–2604, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yang, E., Wang, Z., Shen, L., Liu, S., Guo, G., Wang, X., and Tao, D. (2024). Adamerging: Adaptive model merging for multi-task learning. In *The Twelfth International Conference on Learning Representations*.

Yang, K. and Klein, D. (2021). FUDGE: Controlled text generation with future discriminators. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y., editors, *Proceedings of the 2021 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, pages 3511–3535, Online. Association for Computational Linguistics.

Yang, K., Klein, D., Peng, N., and Tian, Y. (2023). DOC: Improving long story coherence with detailed outline control. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3378–3465, Toronto, Canada. Association for Computational Linguistics.

Yeo, E., Tong, Y., Niu, X., Neubig, G., and Yue, X. (2025). Demystifying long chain-of-thought reasoning in llms. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*.

Yu, D., Yu, Z., and Sagae, K. (2021). Attribute alignment: Controlling text generation from pre-trained language models. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t., editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2251–2268, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., et al. (2025). Dapo: An open-source llm reinforcement learning system at scale. *CoRR*.

Yu, S., Lee, C., Lee, H., and Yoon, S. (2024). Controlled text generation for black-box language models via score-based progressive editor. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14215–14237, Bangkok, Thailand. Association for Computational Linguistics.

Zhang, H. and Song, D. (2022). DisCup: Discriminator cooperative unlikelihood prompt-tuning for controllable text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3392–3406, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhang, H., Song, H., Li, S., Zhou, M., and Song, D. (2023). A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56(3):1–37.

Zhang, S., Wu, L., Gong, C., and Liu, X. (2024). LanguageFlow: Advancing diffusion language generation with probabilistic flows. In Duh, K., Gomez, H., and Bethard, S., editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3893–3905, Mexico City, Mexico. Association for Computational Linguistics.

Zhang*, T., Kishore*, V., Wu*, F., Weinberger, K. Q., and Artzi, Y. (2020). Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zhao, J., Wang, T., Abid, W., Angus, G., Garg, A., Kinnison, J., Sherstinsky, A., Molino, P., Addair, T., and Rishi, D. (2024). Lora land: 310 fine-tuned llms that rival gpt-4, a technical report. *arXiv preprint arXiv:2405.00732*.

Zhao, Z., Shen, T., Zhu, D., Li, Z., Su, J., Wang, X., and Wu, F. (2025). Merging loRAs like playing LEGO: Pushing the modularity of loRA to extremes through rank-wise clustering. In *The Thirteenth International Conference on Learning Representations*.

Zhou, J., Pang, L., Shen, H., and Cheng, X. (2023). SimOAP: Improve coherence and consistency in persona-based dialogue generation via over-sampling and post-evaluation. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 9945–9959, Toronto, Canada. Association for Computational Linguistics.

Zhu, K., Tian, Z., Wei, J., Luo, R., Song, Y., and Mao, X. (2024). StyleFlow: Disentangle latent representations via normalizing flow for unsupervised text style transfer. In Calzolari, N., Kan, M.-Y., Hoste, V., Lenci, A., Sakti, S., and Xue, N., editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 15384–15397, Torino, Italia. ELRA and ICCL.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Zingale, J. and Kalita, J. (2024). Language model sentence completion with a parser-driven rhetorical control method. In Graham, Y. and Purver, M., editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 193–203, St. Julian's, Malta. Association for Computational Linguistics.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. (2023). Representation engineering: A top-down approach to ai transparency. *CoRR*.

Appendix A

Appendix of the Categorisation Scheme for CTG

A.1 Automatic Annotation of CTG Papers

A.1.1 Hyperparameters

We use LLaMa3.3 70B Instruct (Grattafiori et al., 2024) to generate the reasoning traces of our datasets using three NVIDIA A100 GPUs (80 GB each). We use 2048 maximum generated tokens, temperature 0.6, and top p 0.95.

All training runs are performed on two NVIDIA A100 GPUs (80 GB each). Only for RL training for all the evaluation categorisation settings we use three NVIDIA A100 GPUs (80 GB each). We use Qwen3-8B[1] (Team, 2025) as the base model, fine-tuned with LoRA, which is configured with a rank of 32, scaling factor $\alpha = 64$, dropout 0.05, and applied to all attention projection matrices.

For SFT, we train for up to 30 steps with batch size 1, gradient accumulation 8, learning rate 2×10^{-4}, `paged_adamw_32bit` optimizer, a constant learning-rate scheduler, and weight decay 0.01.

For RL, we train for up to 3 epochs with batch size 1, gradient accumulation 8, number of generations per sample 8, learning rate 1×10^{-5}, `paged_adamw_32bit`

[1] `https://huggingface.co/Qwen/Qwen3-8B`

optimizer, a constant learning-rate scheduler, and DAPO loss type. When using the combination of 3 reward functions, we use weights 0.7, 0.25, 0.05 for correctness, JSON format, and format reward, respectively. When using the combination of 2 reward functions, we use weights 0.7 and 0.3 for correctness and format reward, respectively.

For inference of all our trained models, we use one NVIDIA RTXA6000 with 48 GB. We use 2048 maximum generated tokens, temperature 0.6, top p 0.95, top k 20, and min p 0, as recommended by Qwen3 8B.[2]

A.1.1.1 Baselines

Regarding the baselines, we tested two families of models: LLaMa3 (8B and 70B) without reasoning training (Grattafiori et al., 2024) and Qwen 3 (8B, 14B, and 32B) with reasoning training (Team, 2025). More specifically, we tested LLaMa3.1 8B Instruct[3], LLaMa3.3 70B Instruct,[4] Qwen3 8B,[5] Qwen3 14B,[6] and Qwen3 32B.[7] For all models, we use 1024 maximum generated tokens, temperature 0.6, top p 0.95, top k 20, and min p 0.

A.1.2 CTG Papers Prompts and Questions Definition

This section presents the prompts used for reasoning trace generation, model training and inference, along with annotated examples illustrating the output quality of different approaches.

Question definition. Based on the categorisation scheme described in Section 2.3, we define the questions that specify the annotation task for the model. Figure A.1 shows the complete multi-field question that defines all four annotation dimensions (Control Type, Control Attributes, Control Implementation, and Control Integra-

[2] `https://huggingface.co/Qwen/Qwen3-8B`
[3] `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`
[4] `https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct`
[5] `https://huggingface.co/Qwen/Qwen3-8B`
[6] `https://huggingface.co/Qwen/Qwen3-14B`
[7] `https://huggingface.co/Qwen/Qwen3-32B`

tion) jointly, including the possible categories for each field and the expected JSON output format.

Reasoning trace generation. Figure 2.7 shows the prompt used to generate reasoning traces for training data. As described in Section 2.5.2, Dataset creation, we use LLaMa3.3 70B Instruct to produce detailed step-by-step reasoning that leads to the gold annotation for each paper in our dataset. An example of a complete generated reasoning trace is shown in Figure 2.6.

Model training and inference. During both SFT training and inference, we use the prompt shown in Figure A.2. The model is trained to generate outputs in a structured format containing both reasoning and annotation, enclosed within `<think>` and `<answer>` tags, respectively.

Output format and parsing. Figure A.3 shows an example output from our best multi-field model (SFT+RL with EM+F+JSON rewards). The output contains both the reasoning trace and the final annotation expressed as a JSON object. To extract the structured annotation, we parse the content between `<answer>` and `</answer>` tags as JSON. For evaluation, we extract all expected fields from the parsed JSON and compare them against the gold annotation using the metrics described in Section 2.5.3, Evaluation setup.

Figure 2.9 shows an example of generated annotations comparing the gold annotation, the annotation generated using our best model and the annotation generated using LLaMa3.3 70B Instruct.

A.1.3 Evaluation Papers Prompts and Questions Definition

We adapt the prompts from CTG annotation (Appendix A.1.2) by modifying task and question definitions to reflect the evaluation categorisation scheme (Appendix A.2.1).

Question definition. Figure A.4 shows the measure-level evaluation type question, which asks the model to identify and classify each evaluation measure. Figure A.5 shows the measure-level QCET node question, requiring classification across three hierarchy levels.

Reasoning trace generation. We use LLaMA3.3 70B Instruct with the same prompt structure as Figure 2.7, substituting evaluation-specific questions and task description.

Model training and inference. We use the same prompt structure as Figure A.2 with evaluation-specific questions. Outputs contain `<think>` blocks (reasoning) and `<answer>` blocks (JSON annotations). For measure-level tasks, JSON outputs are arrays of measures with their classifications.

A.2 QCET Evaluation Methodology Annotation

To demonstrate generalisation, we test our approach to new task types not seen during development. We apply it to a second annotation task type focused on evaluation methods.

A.2.1 Task Description

This scheme categorises papers based on two aspects. **Evaluation type** identifies the methodology used to assess system output quality. Evaluation type can be annotated at two levels of granularity: (i) paper-level, where a multi-label classification identifies the set of evaluation types used in the paper; (ii) measure-level, where each distinct evaluation measure or criterion used in the paper is identified and individually classified.

At the measure level, we additionally annotate evaluation measures using the Quality Criteria for Evaluation Taxonomy (**QCET**) (Belz et al., 2025), which provides a hierarchical framework for characterising what quality aspects evaluation

measures assess. QCET consists of three main hierarchy levels that progressively narrow the scope of the quality criterion. More details in Appendix A.2.2.

We evaluate three classification settings of increasing complexity: (1) **paper-level evaluation type**; (2) **measure-level evaluation type**; and (3) **measure-level QCET classes**. Combining evaluation type and QCET level annotations is unlikely to be helpful for the evaluation method annotation tasks, as they are entirely orthogonal.

A.2.2 Evaluation categorisation scheme

Our evaluation annotation scheme categorises papers along two dimensions: **evaluation type**, which identifies the methodology used to assess system output quality, and **QCET node classification**, which characterises what quality aspects are being assessed.

Evaluation type. This dimension classifies evaluation measures according to the methodology used to produce assessments. We distinguish four types:

- **None found**: No evaluation of system output quality is reported in the paper;
- **Human**: Evaluations where scores, annotations, or other assessments are assigned by human participants, or where human participants interact with a system (other aspects may be automated);
- **Metric**: Fully automatic evaluations, such as BLEU, perplexity, or other metrics calculated automatically;
- **Hybrid**: Evaluations where scores, annotations, or assessments are determined by a combination of human and metric techniques (other aspects may be either manual or automated).

Evaluation type can be annotated at two levels of granularity: (i) *paper-level*, where multi-label classification identifies all evaluation approaches used anywhere

in the paper (e.g., a paper might use both Human and Metric evaluations); (ii) *measure-level*, where each distinct evaluation measure or criterion is identified and individually classified (e.g., "BLEU score" → Metric, "human fluency ratings" → Human).

We exclude from annotation any evaluations of computational efficiency (e.g., inference time, memory usage) or evaluations of human-created texts rather than system outputs.

QCET node classification. At the measure level, we additionally annotate each evaluation measure using the Quality Criteria for Evaluation Taxonomy (QCET) (Belz et al., 2025). QCET provides a hierarchical framework with three levels that progressively narrow the characterisation of what quality aspect an evaluation measure assesses:

QCET Level 1 (Frame of Reference) captures the frame of reference relative to which system output quality is assessed. The four classes are:

- **Output in its own right**: Quality criteria that assess outputs based solely on the output itself, without reference to inputs or gold standards (e.g., fluency ratings, readability scores);
- **Output relative to input**: Quality criteria that assess outputs relative to both, and only, the output and the input (e.g., faithfulness to source, relevance to query);
- **Output relative to target outputs**: Quality criteria that assess outputs relative to in-distribution gold-standard reference outputs, optionally also considering the input (e.g., BLEU, ROUGE, exact match against gold answers);
- **Output relative to external frame**: Quality criteria that assess outputs relative to a system-external frame of reference beyond input and target outputs (e.g., factual accuracy verified against knowledge bases, alignment with external guidelines).

QCET Level 2 (Type of Quality) captures the type of quality being assessed. The three classes are:

- **Correctness**: Quality criteria based on countable errors or deviations from a standard (e.g., word error rate, grammatical error count, exact match);
- **Goodness**: Quality criteria that assess overall quality holistically, typically considering multiple factors without distinguishing them (e.g., overall quality ratings, coherence scores, helpfulness);
- **Features**: Quality criteria that measure properties without an inherent good/bad orientation; either end of the scale can be preferred depending on context (e.g., length, diversity, specificity).

QCET Level 3 (Aspect of Outputs) captures what aspect of system outputs is being assessed. The three classes are:

- **Form**: Quality criteria that assess only the surface form of outputs (e.g., spelling, grammar, formatting);
- **Content**: Quality criteria that assess only the meaning or content of outputs (e.g., semantic similarity, factual correctness, relevance);
- **Outputs as a whole**: Quality criteria that assess outputs holistically without distinguishing form from content (e.g., overall quality, adequacy, naturalness).

Each evaluation measure is classified along all three QCET levels, producing a three-dimensional characterisation.

A.2.3 Data and Experimental Setup

Following the same procedure as in Section 2.5.2.1, we created an evaluation method annotation dataset with 109 training and 30 test papers. Papers are sampled by a systematic process, annotated with gold labels for evaluation type and QCET categories, and augmented with synthetic reasoning traces.

A.2.3.1 Paper selection process

The QCET dataset consists of three samples of NLP papers. For the first sample, 60 ACL 2022–2024 main conference papers were selected by downloading the ACL Anthology as BibTex from `https://aclanthology.org` in Feb 2025, extracting all ACL main conference items for 2022, 2023, and 2024, then removing any entries which were not papers, e.g. frontmatter, before randomly sampling 20 papers from each year.

The second sample is another 60 papers, also randomly, but independently, sampled using the sample method, and ensuring no overlap.

The third sample consists of the first 20 papers from a reannotation of the 190 papers from the 20Years Survey Howcroft et al. (2020). Full details of the sampling methods and annotation scheme can be found in Belz et al. (2025).

Due to extraction errors in our preprocessing phase, 1 paper was discarded. This resulted in 139 papers in our dataset.

Altogether, the three samples had 569 individual evaluation measures in them, corresponding to an average of 4.09 per paper. When we refer to a measure-level classification task in the main part of the paper, this is what we refer to: the task is to identify all evaluation measures for which results are reported in a paper and then categorise them according to evaluation type and QCET Levels 1–3.

A.2.3.2 Evaluation papers dataset

Data collection and annotation. Following the same procedure used for our CTG dataset (Section 4.3.3.1) and the papers selection process described in Section A.2.3.1, we construct a dataset of 139 papers. Each paper is annotated by three of the co-authors following the categorisation scheme detailed in Appendix A.2.2.

Reasoning trace generation. For each annotated paper, we generate synthetic reasoning traces that demonstrate the step-by-step annotation process. We create one set of reasoning traces for each setting: (1) paper-level evaluation type clas-

sification, (2) measure-level evaluation type classification, (3) measure-level QCET classification, and (4) measure-level evaluation type and QCET classification. All trace types follow the same reasoning structure, systematically analysing paper content to justify label assignments, but differ in scope.

Dataset statistics. Table A.1 shows statistics for our evaluation annotation dataset. The dataset is split into 99 training papers, 10 validation papers, and 30 test papers. Paper lengths range from approximately 400 to 15,900 tokens (mean: ∼8,500 tokens for training), measured using Qwen3 8B tokeniser. Reasoning traces in the training and validation sets average ∼405 and ∼449 tokens respectively, with lengths ranging from 171 to 838 tokens depending on the number and complexity of evaluation measures in each paper. Test set papers do not include reasoning traces, as the model generates these during inference.

	# samples	Paper text			Reasoning trace		
		Max token	Min token	Avg token	Max token	Min token	Avg token
Train	99	15911	4121	8528.8	838	171	404.7
Val	10	11394	3278	8267.4	770	260	448.6
Test	30	13924	418	9117	-	-	-

Table A.1: Dataset statistics for the evaluation annotation task. Paper text and reasoning trace lengths are measured in tokens using the Qwen3 8B tokeniser. Test set reasoning traces are not used (model generates them during inference).

A.2.3.3 Hyperparameters

We use LLaMa3.3 70B Instruct (Grattafiori et al., 2024) to generate the reasoning traces of our datasets using three NVIDIA A100 GPUs (80 GB each). We use 2048 maximum generated tokens, temperature 0.6, and top p 0.95.

All training runs are performed on two NVIDIA A100 GPUs (80 GB each). Only for RL training for all the evaluation categorisation settings we use three NVIDIA A100 GPUs (80 GB each). We use Qwen3-8B[8] (Team, 2025) as the base model, fine-tuned with LoRA, which is configured with a rank of 32, scaling factor $\alpha = 64$, dropout 0.05, and applied to all attention projection matrices.

[8] `https://huggingface.co/Qwen/Qwen3-8B`

For SFT, we train for up to 30 steps with batch size 1, gradient accumulation 8, learning rate 2×10^{-4}, `paged_adamw_32bit` optimiser, a constant learning-rate scheduler, and weight decay 0.01.

For RL, we train for up to 3 epochs with batch size 1, gradient accumulation 8, number of generations per sample 8, learning rate 1×10^{-5}, `paged_adamw_32bit` optimiser, a constant learning-rate scheduler, and DAPO loss type. When using the combination of 3 reward functions, we use weights 0.7, 0.25, 0.05 for correctness, JSON format, and format reward, respectively. When using the combination of 2 reward functions, we use weights 0.7 and 0.3 for correctness and format reward, respectively.

For inference of all our trained models, we use one NVIDIA RTXA6000 with 48 GB. We use 2048 maximum generated tokens, temperature 0.6, top p 0.95, top k 20, and min p 0, as recommended by Qwen3 8B.[9]

Baselines Regarding the baselines, we tested two families of models: LLaMa3 (8B and 70B) without reasoning training (Grattafiori et al., 2024) and Qwen 3 (8B, 14B, and 32B) with reasoning training (Team, 2025). More specifically, we tested LLaMa3.1 8B Instruct[10], LLaMa3.3 70B Instruct,[11] Qwen3 8B,[12] Qwen3 14B,[13] and Qwen3 32B.[14] For all models, we use 1024 maximum generated tokens, temperature 0.6, top p 0.95, top k 20, and min p 0.

A.2.4 Prompts and questions definition

We adapt the prompts from CTG annotation (Appendix A.1.2) by modifying task and question definitions to reflect the evaluation categorisation scheme (Appendix A.2.3.1).

[9] `https://huggingface.co/Qwen/Qwen3-8B`
[10] `https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct`
[11] `https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct`
[12] `https://huggingface.co/Qwen/Qwen3-8B`
[13] `https://huggingface.co/Qwen/Qwen3-14B`
[14] `https://huggingface.co/Qwen/Qwen3-32B`

Question definition. Figure A.4 shows the measure-level evaluation type question, which asks the model to identify and classify each evaluation measure. Figure A.5 shows the measure-level QCET node question, requiring classification across three hierarchy levels.

Reasoning trace generation. We use LLaMA3.3 70B Instruct with the same prompt structure as Figure 2.7, substituting evaluation-specific questions and task description.

Model training and inference. We use the same prompt structure as Figure A.2 with evaluation-specific questions. Outputs contain `<think>` blocks (reasoning) and `<answer>` blocks (JSON annotations). For measure-level tasks, JSON outputs are arrays of measures with their classifications.

A.2.5 Results

Table A.2 shows results for paper-level evaluation type, the least complex of the evaluation method categorisation tasks. Interestingly, larger pre-trained models do not always show better instruction-following: Qwen3 32B achieves only 6.7% EM due to frequent failures to generate outputs in the required format, while the smaller Qwen3 8B and 14B variants successfully follow formatting instructions. This highlights a key challenge in deploying larger models for structured output tasks without task-specific training. Our SFT approach addresses this issue, boosting Qwen3 8B to 75.3% EM with perfect format compliance, and adding RL further improves performance to 78.7% EM.

Table A.3 shows results for measure-level evaluation type classification. This variable-length task proves substantially harder: pre-trained models achieve only 13.3-19.3% EM compared to 53.3-71.3% for the paper-level version of the task. Our SFT approach reaches 28.0% EM, surpassing LLaMA3.3 70B by 8.7 points, with strong partial-credit metrics indicating correct classification of many individual measures. RL refinement further improves performance to 30.7% EM, achieving the

Model	R	Evaluation Type		
		EM	F1	J
Pre-trained LLMs				
Qwen3 8B	✓	68.7	80.3	75.7
Qwen3 14B	✓	71.3	81.9	78.7
Qwen3 32B	✓	6.7	20.7	10.1
LLaMa3.1 8B I	×	4.0	48.7	30.1
LLaMa3.3 70B I	×	53.3	74.3	70.4
Ours				
Qwen3 8B SFT only	✓	*75.3*	*84.7*	*81.7*
Qwen3 8B SFT+RL	✓	**78.7**	**85.4**	**83.0**

Table A.2: Paper-level evaluation type classification results, averaged over 5 runs. **Bold** indicates best result, *italics* second-best. R=Reasoning trace; EM=Exact Match; J=Jaccard score (intersection-over-union).

highest exact match while maintaining competitive F1 and Jaccard scores.

Model	R	Evaluation Type		
		EM	F1	J
Pre-trained LLMs				
Qwen3 8B	✓	13.3	55.4	41.8
Qwen3 14B	✓	15.3	46.2	35.6
Qwen3 32B	✓	17.3	44.8	34.3
LLaMa3.1 8B I	×	3.3	31.1	17.9
LLaMa3.3 70B I	×	19.3	*66.8*	54.7
Ours				
Qwen3 8B SFT only	✓	*28.0*	**67.3**	**58.2**
Qwen3 8B SFT+RL	✓	**30.7**	65.4	*57.8*

Table A.3: Measure-level evaluation type classification results, averaged over 5 runs. **Bold** indicates best result, *italics* second-best. R=Reasoning trace; EM=Exact Match; J=Jaccard score (intersection-over-union).

Table A.4 shows results for measure-level QCET classification across three hierarchical levels. The task's complexity is evident: none of the pre-trained models achieve any exact matches (all three levels correct for all measures). Even LLaMA3.3 70B, despite reasonable Level 1 performance, fails completely on Level 3, indicating particular difficulty with fine-grained quality aspect classification. Our SFT approach achieves 14.0% complete EM, the only model to achieve non-zero EM, and consistently outperforms baselines across the hierarchy: +8.0 points on Level

1, +10.7 points on Level 2, and +21.3 points on Level 3. Partial-credit metrics (51.7-55.0% F1) demonstrate strong performance on individual measures despite the challenging multi-level structure. RL refinement substantially improves performance to 18.0% complete EM (+4.0 points), with particularly strong gains on Level 3 (+12.0 points to 33.3% EM). Unlike Tables A.2 and A.3, RL optimisation improves both exact match and partial-credit metrics on this hierarchical task (40.8% F1 complete, 58.8-68.8% F1 across levels), demonstrating effective learning across the full structure.

Model	**R**	**QCET node**											
		Complete			**Level 1**			**Level 2**			**Level 3**		
		EM	**F1**	**J**	**EM**	**F1**	**J**	**EM**	**F1**	**J**	**EM**	**F1**	**J**
Pre-trained LLMs													
Qwen3 8B	✓	0.0	0.3	0.0	12.0	25.4	20.1	6.7	21.9	15.7	*0.0*	2.3	0.7
Qwen3 14B	✓	0.0	0.0	0.0	2.7	4.1	3.6	2.0	3.8	2.8	*0.0*	0.3	0.1
Qwen3 32B	✓	0.0	0.3	0.0	14.7	34.0	28.5	7.3	30.6	22.9	*0.0*	4.1	0.9
LLaMa3.1 8B I	×	0.0	0.5	0.2	3.3	17.5	10.8	4.0	24.9	16.1	*0.0*	2.9	0.9
LLaMa3.3 70B I	×	0.0	0.9	0.2	18.0	52.1	43.3	8.0	*53.0*	40.4	0.0	7.5	2.1
Ours													
Qwen3 8B SFT only	✓	*14.0*	*33.5*	*29.8*	*26.0*	*55.0*	*48.7*	*18.7*	51.7	*42.1*	*21.3*	*53.8*	*45.8*
Qwen3 8B SFT+RL	✓	**18.0**	**40.8**	**35.9**	**28.0**	**58.8**	**51.9**	**19.3**	**54.1**	**43.2**	**33.3**	**68.8**	**61.2**

Table A.4: Measure-level QCET node classification results, averaged over 5 runs. **Bold** indicates best result, *italics* second-best. R=Reasoning trace; EM=Exact Match; J=Jaccard score (intersection over union).

Multi-field question for CTG categorisation

```
Based on the given paper, you need to identify the control type, the control implementation, the control integration, and the control attributes used in the proposed system.

Control type is whether the implemented system is able to control only one attribute at a time, or multiple attributes at once and it must be one of the following categories:
- Single-attribute, i.e. the implemented system controls only one attribute type at a time;
- Multiple-attribute, i.e. the implemented system controls multiple attribute types at once, for example sentiment and topic.

Control attributes are the high-level semantic or structural properties of the generated text that a controllable text generation system aims to influence or constrain or control. They define the target aspect of control, such as sentiment, topic, length, style, knowledge content, or discourse function.
You need to return a list of all the control attributes used, for example: {"control_attributes": ["Sentiment", "Keywords", "Length", "Dialogue act", "Knowledge elements"]}.

Control implementation is the method by which control capabilities are enabled in a CTG system and it must be one or more of the following categories:
- LM Prompting, i.e. use input prompts to steer the generation of the model;
- Text Selection, i.e. select the text that most represents the given control attribute;
- Modification of Token Distribution, i.e. modify the token distribution of the pre-trained model to control the generated text;
- Complete Training, i.e. train the model from scratch;
- Fine-Tuning, i.e. fine-tune the model on a specific dataset.
In case of multiple implementations, you need to specify the combination of implementations as list, for example: {"control_implementation": ["LM Prompting", "Modification of Token Distribution"]}.

Control integration is the level at which the control signal is applied within the system architecture: at the input (Input Addition), within the model (Model Injection), or as an external module (System Addition). Different combinations of integrations are possible. It must be one or more of the following categories:
- Input Addition, i.e. the control signal is added at the input of the system, for example adding a control code to the input;
- Model Injection, i.e. the control signal is injected within the model, for example injecting a PEFT module encapsulating the control signal inside the model;
- System Addition, i.e. the control signal is added as an external module, for example adding a classifier that modifies the output of the model;
In case of multiple integrations, you need to specify the combination of integrations as list, for example: {"control_integration": ["Input Addition", "Model Injection"]}.

You need to return the control type, the control attributes, the control implementation, and the control integration categories as JSON object between <answer> and </answer> tags, for example: <answer>{"control_type": "Single-attribute", "control_attributes": ["Sentiment", "Keywords", "Length", "Dialogue act", "Knowledge elements"], "control_implementation": "LM Prompting", "control_integration": "Input Addition"}</answer>.
```

Figure A.1: Definition of the multi-field question for CTG categorisation. The question defines all 4 fields for the CTG categorisation: *Control Type*, *Control Attributes*, *Control Implementation*, and *Control Integration*.

Training and Inference prompt

```
<paper> {paper_text} </paper>

<question> {question} </question>

Analyze the paper to classify the controllable text generation system according to the question.

Think step by step: identify the relevant information in the paper, cite specific sections or quotes that support your answer, and explain how they lead to your classification choice.

<think> [Your reasoning process] </think>
<answer> [Your final classification] </answer>
```

Figure A.2: Prompt used for training and inference of CTG papers annotation. Placeholders are shown in {braces}.

Output example using our multi-field SFT+RL best model

```
<think>
To answer this question, we first need to understand the key components of a controllable text generation (CTG) system and how they relate to the control type, control attributes, control implementation, and control integration.

1. **Initial Analysis**: The question asks us to identify the control type, control attributes, control implementation, and control integration used in the proposed system based on the given paper. The control type can be either single-attribute or multiple-attribute. Control attributes are the aspects of the text that the system aims to control, such as sentiment, topic, or length. Control implementation refers to how the control capabilities are enabled, which could be through LM prompting, text selection, modification of token distribution, complete training, fine-tuning, or a hybrid approach. Control integration pertains to where the control signal is applied, which could be through input addition, model injection, system addition, or a hybrid method.

2. **Systematic Paper Review**:
    - The abstract and introduction indicate that the system focuses on controlling the simplicity of text, specifically for grade-specific text simplification. This suggests that the primary control attribute is "Simplicity".
    - The methodology section explains that the system predicts control tokens based on source text features and the desired target grade level, indicating a fine-tuning approach for implementation.
    - The system description and results show that the control signal is integrated at the input level, as the control tokens are added to the input sequence to guide the generation of the output.

3. **Evidence Collection**:
    - "We introduce a Control Predictor module (CP) that predicts a vector of control token values V for each input X at inference time." This supports the idea of fine-tuning and input addition for control integration.
    - "The control tokens are added to the source text as a side constraint in the format, W_{} C_{} L_{} WR_{} DTD_{} {Source_ text}." This confirms that control attributes are integrated at the input level.
    - The system aims to control the simplicity of the text, as indicated by the focus on grade-specific simplification and the use of metrics like SARI.

4. **Category Matching**:
    - **Control Type**: Given that the primary focus is on controlling simplicity, it appears to be a single-attribute control, as the system does not explicitly mention controlling multiple attributes simultaneously.
    - **Control Attributes**: The primary attribute controlled is
    "Simplicity".
    - **Control Implementation**: The system is implemented through fine-tuning, as it involves training a control predictor to predict control tokens and using these to guide the generation.
    - **Control Integration**: The control signal is integrated through input addition, as control tokens are added to the input sequence to influence the output.

5. **Verification**:
    - The evidence supports a fine-tuning implementation and input addition for control integration.
    - The control attribute is clearly simplicity, and the system is designed to control this attribute effectively.

</think> <answer> {"control_type": "Single-attribute", "control_implementation":
"Fine-Tuning", "control_integration": "Input Addition", "control_attributes":
["Simplicity"]} </answer><|im_end|>
```

Figure A.3: Example of generated output using our best model (multi-field SFT+RL with EM+F+JSON rewards). The output contains the reasoning trace and the final answer expressed as JSON object.

Measure-level evaluation type question for evaluation categorisation

```
Based on the given paper, you need to identify each evaluation measure used in the research reported in the paper in order to assess the output quality of the proposed systems. For each identified evaluation measure, you further need to identify its type. Include only evaluations of system output quality; do not include evaluations of efficiency, or evaluations of human-created texts.

More specifically, your task is to list each distinct evaluation measure used in the paper, and for each one, classify it into one of the following types:
- None found, i.e. no evaluation of system output quality is found in the paper;
- Human, i.e. evaluations where scores, annotations or other assessments are assigned by human participants, our where human participants interact with a system (other aspects may be automated);
- Metric, i.e. fully automatic evaluations, such as BLEU or perplexity calculated with a given model;
- Hybrid, i.e. evaluations where scores, annotations or other assessments are determined by a combination of human and metric techniques (other aspects may be either manual or automated).

For each evaluation measure identified, create one item in the output list. You need to return the results as a JSON array of objects where each object (key-value pair) represents one evaluation measure, for example [{"evaluation_type": "Metric"}, {"evaluation_type": "Human"}, {"evaluation_type": "Metric"}], or [{"evaluation_type": "None found"}] if no evaluation measures are found. The array must contain one key-value pair for each evaluation measure identified in the paper, and each value must be exactly one label (one of "Metric", "Human", "Hybrid", "None found").
```

Figure A.4: Definition of the measure-level evaluation type question for evaluation categorisation.

Measure-level QCET node question for evaluation categorisation

```
Based on the given paper, you need to identify each evaluation measure used in the research reported in the paper in order to assess the output quality of the proposed systems. For each identified evaluation measure, you further need to classify the evaluation measure in terms of its frame of reference, the type of quality it assesses, and the aspect of system outputs it assesses. Include only evaluations of system output quality; do not include evaluations of efficiency, or evaluations of human-created texts.

More specifically, your task is to list each distinct evaluation measure used in the paper, and for each one, to classify the quality criterion the evaluation assesses in terms of the following three dimensions:

1. QCET Level 1, consisting of four Frame-of-Reference classes each capturing one of four possible frames of reference relative to which system output quality can be assessed. The four classes are the following:

- Output in its own right (output_in_its_own_right): choose this class for quality criteria that assess system output quality relative to just the output; i.e. the measured values these quality criteria produce are obtained on the basis of the system outputs alone;
- Output relative to input (output_relative_to_input): choose this class for quality criteria that assess system output quality relative to both, and only, the output and the input; i.e. the measured values these quality criteria produce are obtained on the basis of system outputs and system inputs alone;
- Output relative to in-distribution target outputs +/-input (output_relative_to_target_outputs): choose this class for quality criteria that assess system output quality relative to target outputs sampled from the same distribution as the training data, and optionally also relative to the input; i.e. the measured values these quality criteria produce are obtained on the basis of system outputs and gold-standard reference outputs (plus, optionally, system inputs);
- Output relative to a system-external frame of reference +/-input (output_relative_to_external_frame): choose this class for quality criteria that assess system output quality relative to a system-external frame of reference, and optionally also the input; i.e. the measured values produced by an evaluation with this type of quality criterion are obtained on the basis of system outputs and gold-standard reference outputs (plus, optionally, system inputs).

2. QCET Level 2, consisting of three Type-of-Quality classes each capturing one of three possible types of system output quality assessed by a quality criterion. The three classes are the following:

- Correctness (correctness): choose this class for quality criteria that assess the correctness of system outputs; i.e. the measured values these quality criteria produce are based on countable errors or deviations;
- Goodness (goodness): choose this class for quality criteria that assess the goodness of system outputs; i.e. the measured values these quality criteria produce are not primarily based on notions of countable errors, instead typically taking multiple factors into account without naming or distinguishing them;
- Features (features): choose this class for quality criteria that assess features of system outputs; i.e. quality criteria of this type have no natural good end of the scale; while Correctness and Goodness criteria align with (i) what systems are trained to be good at, and/or (ii) common-sense notions of desirable properties in systems; in Feature-type criteria, instead, either end of the scale can be the preferred end depending on evaluation context.

3. QCET Level 3, consisting of three Aspect-of-Outputs classes each capturing one of three possible aspects of system outputs assessed by a quality criterion. The three classes are the following:
- Form (form): choose this class for quality criteria that assess the form of system outputs alone;
- Content (content): choose this class for quality criteria that assess the content/meaning of system outputs alone;
- Outputs as a whole (outputs_as_a_whole): choose this class for quality criteria that assess system outputs without distinguishing form from content.

In choosing the above classes, bear in mind that the name a paper gives an evaluation measure may not be fully aligned with what it actually assesses. If the paper provides a definition, gloss, or the question put to evaluators for an evaluation measure, then use that as the basis for choosing classes, rather than the name.

For each evaluation measure identified, create one item in the output list. You need to return the results as a JSON array of objects where each object (key-value pair) represents one evaluation measure, for example [{"qcet_level_1": "output_relative_to_input", "qcet_level_2": "goodness", "qcet_level_3": "content"}, {"qcet_level_1": "output_relative_to_target_outputs", "qcet_level_2": "correctness", "qcet_level_3": "outputs_as_a_whole"}].
```


Figure A.5: Definition of the measure-level QCET node question for evaluation categorisation. The question defines all 3 fields for the QCET node categorisation:

Appendix B

A Fair Comparison of Controlled Text Generation Techniques Appendix

B.1 LLMs Prompts

Tables B.1 and B.2 show the zero-shot and few-shot prompts used with Falcon 40B Instruct model, while Tables B.3 and B.4 show the prompts used with LLaMa 2 70B chat model.

B.2 Hyperparameters

All the scripts and all the library versions used to execute the experiments can be found on our GitHub repo.[1]

B.2.1 CTG Techniques

We executed all the experiments on our own GPUs.

[1] https://github.com/michelalorandi/comparative_study_of_ctg

Falcon 40B Instruct Zero-Shot prompt	

Template for PPLM Prompts and OWT neutral prompts:	`System: You are a text generator. Reply with only one turn.`
	`User: Write a fluent English text starting with "{`*text*`}"`
	`with {`*control attribute value*`} {`*control attribute*`}`
	`Falcon:`

Template for STS benchmark test and Cloze 2018 test:	`System: You are a story writer. Reply with only one turn.`
	`User: Write a fluent English story starting with "{`*text*`}"`
	`with {`*control attribute value*`} {`*control attribute*`}`
	`Falcon:`

Multiple-attribute control instruction template:	`User: Write a fluent English {`*story or text*`} starting with "{`*text*`}" with {`*sentiment value*`} sentiment and {`*topic value*`} topic`

Table B.1: Template of the Zero-Shot prompt for Falcon 40B Instruct model.

CTRL We used `Salesforce/ctrl` model on HuggingFace both for the tokeniser and the model. We set *max length*=256. CTRL was executed on a Nvidia RTXA6000 GPU with 48GB RAM.

C BART We used the C BART large One-Billion-Words model provided on GitHub[2]. We set the hyperparameters as follows *decoder chain*=5, *encoder loss type*=0, *max insert label*=1, *temperature*=1, *do sample*=0, *top k*=0, *top p*=0.9, *refinement steps*=10, *max refinement steps*=30, *adaptive*=False, *repetition penalty*=2, *threshold*=0, and *max length*=20. C BART was executed on a Nvidia RTXA6000 GPU with 48GB RAM.

Multi CTG We used `bert-base-uncased` and `gpt2-medium` models on HuggingFace as the pretrained encoder and pretrained decoder, respectively. We also used the checkpoint-30000 provided by the authors[3]. The hyperparameters were set as follows: *latent size*=768, *latent num*=1, *sequence length per latent*=20, *variation*=1e-

[2] `https://github.com/nlpcode/cbart`
[3] `https://github.com/HappyGu0524/MultiControl`

Falcon 40B Instruct Few-Shot prompt	
Template for PPLM Prompts and OWT neutral prompts:	System: You are a text generator. Reply with only one turn. User: Write a fluent English text starting with "The new research reveals relatively recent tectonic activity connected to" with {*positive or science/technology or "lunar", "cooling"*} {*sentiment or topic or keywords*} Falcon: The new research reveals relatively recent tectonic activity connected to the long-lived cooling and associated contraction of the lunar interior. User: Write a fluent English text starting with "This test article is a quarter the size of" with {*negative or business or "2016", "company", "sail"*} {*sentiment or topic or keywords*} Falcon: This test article is a quarter the size of the sail the company plans to fly in 2016. User: Write a fluent English text starting with "{*text*}" with a {*control attribute value*} {*control attribute*}
Template for STS benchmark test and Cloze 2018 test:	System: You are a story writer. Reply with only one turn. User: Write a fluent English story starting with "{*text*}" with a {*control attribute value*} {*sentiment or topic or keywords*} Falcon: Jack is tired of writing emails on his tablet computer. He decides he needs to invest in a laptop. He researches different laptops until he has found the perfect one. Finally he buys it. Now Jack actually enjoys replying to emails. User: Write a fluent English story starting with "Gertrude was notoriously clumsy and awkward." with a {*negative or sports or "soccer", "ball", "school", "team"*} {*sentiment or topic or keywords*} Falcon: Gertrude was notoriously clumsy and awkward. Her friends thought that participating in a sport may help her. She decided to try out for the soccer team at school. She ran out on the field and unexpectedly tripped over the ball. Naturally, she was not accepted to the soccer team. User: Write a fluent English story starting with "Jack is tired of writing emails on his tablet computer." with a {*positive or science/technology or "emails", "laptop"*} {*sentiment or topic or keywords*}
Multiple-attribute control instruction template:	User: Write a fluent English {*story or text*} starting with "{*text*}" with {*sentiment value*} sentiment and {*topic value*} topic

Table B.2: Template of the Few-Shot prompt for Falcon 40B Instruct model.

3, *number centers*=1000, *number output centers*=[[1,1,5,1], [10,10,5,1]], *top k*=200, *batch*=5, *max iter*=15, *strategy*=none, *temperature*=50, *SDM reinit*=True, *rp*=1.2, *weight*={"default": [2,7,1], "01": [2,4,1], "02": [2,8,1], "03": [3,1,3], "10": [2,12,1], "11": [3,5.5,1], "12": [2,9,1], "13": [3,1,1]}, and *max length*=50. Multi CTG was

LLaMa2 70B chat Zero-Shot prompt	

Template for PPLM Prompts and OWT neutral prompts:	`<s> [INST] <<SYS>>`
	`You are a text generator. Reply with only one turn.`
	`<</SYS>>`
	`Write a fluent English text starting with "{`*text*`}" with a` {*control attribute value*} {*control attribute*} `[/INST]`

Template for STS benchmark test and Cloze 2018 test:	`<s> [INST] <<SYS>>`
	`You are a story generator. Reply with only one turn.`
	`<</SYS>>`
	`Write a fluent English story starting with "{`*text*`}" with a` {*control attribute value*} {*control attribute*} `[/INST]`

Multiple-attribute control instruction template:	`Write a fluent English` {*story or text*} `starting with "{`*text*`}" with` {*sentiment value*} `sentiment and` {*topic value*} `topic [/INST]`

Table B.3: Template of the Zero-Shot prompt for LLaMa2 70B chat model.

executed on a Nvidia RTXA6000 GPU with 48GB RAM.

Prior CTG We used `bert-base-uncased` and `gpt2-medium` models on HuggingFace as the pretrained encoder and pretrained decoder, respectively. We also used the checkpoint-30000 and the configuration files (*config*= generate_config_combine.json and *optim config*=generate_config_combine_optim.json) provided by the authors[4]. All other hyperparameters were set as follows *latent size*=768, *latent num*=1, *sequence length per latent*=20, *variation*=0, *prior*=True, *flow num*=8, *prior num*=8, *std*=1, *is extend*=True, *is constrained*=True, *weight*=[1,5,1], *optim weight*=[1,1,1], and *max length*=50. Prior CTG was executed on a Nvidia RTXA6000 GPU with 48GB RAM.

CAT-PAW We used the `gpt2-medium` model on HuggingFace as the PLM. We set the hyperparameters as follows *stepsize*=0.02, *temperature*=1.0, *top k*=10, *fusion gm scale*=0.9, *fusion kl scale*=0.01, *number iterations*=3, *gradient length*=10000,

[4] `https://github.com/HappyGu0524/MultiControl`

LLaMa2 70B chat Few-Shot prompt	
Template for PPLM Prompts and OWT neutral prompts:	<s> [INST] <<SYS>> You are a text generator. Reply with only one turn. <</SYS>> Write a fluent English text starting with "The new research reveals relatively recent tectonic activity connected to" with {*positive or science/technology or "lunar", "cooling"*} {*sentiment or topic or keywords*} [/INST] The new research reveals relatively recent tectonic activity connected to the long-lived cooling and associated contraction of the lunar interior. [/INST] Write a fluent English text starting with "This test article is a quarter the size of" with {*negative or business or "2016", "company", "sail"*} {*sentiment or topic or keywords*} [/INST] This test article is a quarter the size of the sail the company plans to fly in 2016. [/INST] Write a fluent English text starting with "{*text*}" with a {*control attribute value*} {*control attribute*} [/INST]
Template for STS benchmark test and Cloze 2018 test:	<s> [INST] <<SYS>> You are a story writer. Reply with only one turn. <</SYS>> Write a fluent English story starting with "{*text*}" with a {*control attribute value*} {*sentiment or topic or keywords*} [/INST] Jack is tired of writing emails on his tablet computer. He decides he needs to invest in a laptop. He researches different laptops until he has found the perfect one. Finally he buys it. Now Jack actually enjoys replying to emails. [/INST] Write a fluent English story starting with "Gertrude was notoriously clumsy and awkward." with a {*negative or sports or "soccer", "ball", "school", "team"*} {*sentiment or topic or keywords*} [/INST] Gertrude was notoriously clumsy and awkward. Her friends thought that participating in a sport may help her. She decided to try out for the soccer team at school. She ran out on the field and unexpectedly tripped over the ball. Naturally, she was not accepted to the soccer team. [/INST] Write a fluent English story starting with "Jack is tired of writing emails on his tablet computer." with a {*positive or science/technology or "emails", "laptop"*} {*sentiment or topic or keywords*}
Multiple-attribute control instruction template:	Write a fluent English {*story or text*} starting with "{*text*}" with {*sentiment value*} sentiment and {*topic value*} topic [/INST]

Table B.4: Template of the Few-Shot prompt for LLaMa2 70B chat model.

number samples=1, *horizon length*=1, *window length*=0, *decay*=False, *gamma*=1.5, *sample*=True, *activate alter scale*=False, *require origin*=False, *active size*=0.01, *classifier type*=attn, *annotator type*=dis, *loss type*=1, and *max length*=50. CAT-PAW was executed on a Nvidia RTXA6000 GPU with 48GB RAM.

PPLM We used `gpt2-medium` model on HuggingFace as the PLM and we used the models provided by the authors as discriminators[5]. The common hyperparameters for all control attributes were set as *number samples*=1, *top k*=10, *sample*=True, *gradient length*=10000, *horizon length*=1, *window length*=0, *decay*=False, *kl scale*=0.01, *discriminator weights*=null and *discriminator meta*=null. Regarding sentiment and multiple-attribute control, we set *stepsize*=0.1, *temperature*=0.85, *number iterations*=3, *gamma*=1.5, and *gm scale*=0.9. For topic control we set *stepsize*=0.03, *temperature*=0.9, *number iterations*=10, *gamma*=1.0, *gm scale*= 0.95, and *max length* =50. PPLM was executed on a Nvidia RTXA6000 GPU with 48GB RAM.

Falcon 40B Instruct We used `tiiuae/falcon-40b-instruct` model on HuggingFace. We set the hyperparameters as *early stopping*=False, *max time*=300, *do sample*=True, *num_beam*=1, *num_beam_groups*=1, *use cache*=True, *temperature*=1.0, *top k*=50, *top p*=0.95, *typical p*=1, *epsilon cutoff*=0, *eta cutoff*=0, *diversity penalty*=0, *repetition penalty*=1, *encoder repetition penalty*=1, *length penalty*=1, *no repeat ngram size*=0, *renormalize logits*=False, *number return sequences*=1, *output attentions*=False, *output hidden states*=False, *output scores*=False, *return dict in generate*=True, and *max length*=100. Falcon was executed on a Nvidia A100 GPU with 80GB RAM.

LLaMa2 70B chat We used `meta-llama/Llama-2-70b-chat-hf` model on HuggingFace. We set the hyperparameters as *early stopping*=False, *max time*=300, *do sample*=True, *number beans*=1, *number beans groups*=1, *use cache*=True, *temperature*=1.0, *top k*=50, *top p*=0.95, *typical p*=1, *epsilon cutoff*=0, *eta cutoff*=0, *diver-*

[5] `https://github.com/uber-research/PPLM`

sity penalty=0, *repetition penalty*=1, *encoder repetition penalty*=1, *length penalty*=1, *no repeat ngram size*=0, *renormalize logits*=False, *number return sequences*=1, *output attentions*=False, *output hidden states*=False, *output scores*=False, *return dict in generate*=True, and *max length*=100. LLaMa2 was executed on a Nvidia A100 GPU with 80GB RAM.

GeDi We used `gpt2-xl` model on HuggingFace as the frozen PLM and the GeDi sentiment and GeDi topic as GeDi models provided by the authors[6] for sentiment and topic control, respectively. The hyperparameters were set as follows *pad lens*=null, *temperature*=1.0, *top k*=50, *top p*=1.0, *repetition penalty*=1.2, *repetition penalty scale*=10.0, *pad token id*=0, *do sample*=True, *penalize condition*=True, *discriminator weight*=30.0, *filter p*=0.8, *target p*=0.8, *class bias*=0.0, and *max length*=100. GeDi was executed on a Nvidia RTXA6000 GPU with 48GB RAM.

DisCup We used `gpt2-large` model on HuggingFace as the PLM and `distill_tuning_negative_(6,6)` and `distill_tuning_positive_(5,5)` checkpoints provided by the authors[7] as discriminator for sentiment control. We set the hyperparameters as *template*=(5,5) for positive and (6,6) for negative, *tuning name*=distill_tuning, *pseudo token*=xxx, *use lm finetune*=False, *lstm dropout*=0.0, *discriminator embedding checkpoint*=null, *prompt pad length*=10, *ranking scope*=50, *temperature*=0.01, *beta*=0.4, and *max length*=20. DisCup was executed on a Nvidia RTXA6000 GPU with 48GB RAM.

DExperts We used `gpt2-large` model on HuggingFace as the PLM and the `finetuned_ gpt2_positive` and `finetuned_gpt2_negative` models provided by the authors[8] as experts and anti-experts models. We set the hyperparameters as *model type*=dexperts, *n*=1, *alpha*=0.0, *p*=1.0, *filter p*=0.9, *resume*=False, *use dataset*=True, and *max length*=20. DExperts was executed on a Nvidia RTXA6000 GPU

[6] `https://github.com/salesforce/GeDi`

[7] `https://github.com/littlehacker26/Discriminator-Cooperative-Unlikelihood-Prompt-Tuning`

[8] `https://github.com/alisawuffles/DExperts`

with 48GB RAM.

BOLT We used `gpt2-large` model and `gpt2` tokenizer on HuggingFace and the discriminator checkpoint (`replaced_vocab_roberta_for_yelp_ polarity`) for sentiment control provided by the authors[9]. The hyperparameters were set as follows *weight decay*=0.01, *init from vocabulary*=True, *learning rate*=0.025 for sentiment and 0.4 for keywords control, and *max length*=50. BOLT was executed on a Nvidia RTXA6000 GPU with 48GB RAM.

B.2.2 Evaluation Metrics

SLOR We used `gpt2-xl` and `bigscience/bloom-1b7` models from HuggingFace to compute sentence and unigram probabilities.

Control Effectiveness To detect sentiment, we used `distilbert/distilbert-base-uncased-finetuned-sst-2-english`[10] and `michelecafagna26/t5-base-finetuned-sst2-sentiment`[11] from HuggingFace, while for topic classification, we used `textattack/distilbert-base-uncased-ag-news`[12] and `fabriceyhc/bert-base-uncased-ag_news`[13]. To compute Word Inclusion Coverage, we used Spacy with `en_core_web_sm` model to extract lemmas of tokens.

Perplexity We used the `evaluate`[14] library of HuggingFace using both `gpt2-xl` and `bigscience/bloom-1b7` models.

B.3 Scientific artifacts and licensing

CTRL and GeDi are licensed under BSD-3-Claude license. Multi CTG and Prior CTG are licensed under MIT license. PPLM, Falcon 40B Instruct, PPLM Prompts,

[9] `https://github.com/launchnlp/BOLT`
[10] `https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english`
[11] `https://huggingface.co/michelecafagna26/t5-base-finetuned-sst2-sentiment`
[12] `https://huggingface.co/textattack/distilbert-base-uncased-ag-news`
[13] `https://huggingface.co/fabriceyhc/bert-base-uncased-ag_news`
[14] `https://huggingface.co/docs/evaluate/en/index`

Yelp review dataset and AGnews dataset are licensed under Apache-2.0 license. LLaMa2 70B chat is licensed under a commercial license.[15] BLOOM 1B7 is licensed under RAIL License v1.0.[16] C BART, CAT-PAW, DisCup, DExperts, BOLT and Cloze 2018 test were shared without explicitly specified license. Bookcorpus dataset is licensed under Smashwords terms of service.[17] One-Billion-Words is licensed under CC0: Public Domain license. OWT neutral prompts is licensed under Creative Commons CC0 license ("no rights reserved"). STS benchmark test provides a license for each included dataset.[18] The usage of all listed artifacts is consistent with their licenses.

[15] `https://ai.meta.com/resources/models-and-libraries/llama-downloads/`
[16] `https://huggingface.co/spaces/bigscience/license`
[17] `https://www.smashwords.com/about/tos`
[18] `http://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark`

Appendix C

Enable Content Control with Data-to-Text Generation Appendix

C.1 Out-of-the-Box LLMs for Data-to-Text Generation

C.1.1 Hyperparameters and APIs

We executed all the experiments either via API or on our own GPUs. We used the paid-for OpenAI API to access text-davinci-003[1] (GPT-3.5), while we used the free inference API of HuggingFace to access BLOOM 176B[2] and falcon-180B-chat[3]. On the other hand, we downloaded and executed Llama-2-70b-chat-hf[4] on a Nvidia A100 GPU with 80GB RAM.

To use the three explored Machine Translation engines, we used the pay-as-you-go APIs of Google Cloud[5] and Alibaba Cloud[6], and we downloaded and executed

[1] https://platform.openai.com/docs/models/gpt-3-5
[2] https://huggingface.co/bigscience/bloom
[3] https://huggingface.co/tiiuae/falcon-180B-chat
[4] https://huggingface.co/meta-llama/Llama-2-70b-chat-hf
[5] https://cloud.google.com/translate
[6] https://www.alibabacloud.com/product/machine-translation

NLLB Costa-jussà et al. (2022) on a Nvidia A100 GPU with 80GB RAM.

For all used models, we set *maximum length* to 500 with Zero-Shot Minimal Instruction and 1000 with Few-Shot In Context. All generated texts are post-processed as described above.

GPT-3.5 In all experiments involving GPT-3.5, we set text-davinci-003 parameters to *temperature*=0, *top p*=1 (default), *frequency penalty*=0 and *presence penalty*=0 (default), *best of*=1 (default) to get only 1 completion for each prompt.

BLOOM We used bigscience/bloom model with HuggingFace's Inference Client API setting the parameters to *temperature*=0.7, *top p*=0.9, *frequency penalty*=0 and *presence penalty*=0.

LLaMa2-chat We used meta-llama/Llama-2-70b-chat-hf model on HuggingFace setting the parameters to *temperature*=1 (default), *top p*=1 (default), *repetition penalty*=1 (default) and *diversity penalty*=0 (default), *num return sequences*=1.

Falcon-chat We used tiiuae/falcon-180b-chat model with HuggingFace's Inference Client API setting the parameters to *temperature*=0.7, *top p*=0.9, *frequency penalty*=0 and *presence penalty*=0.

NLLB We used facebook/nllb-200-1.3B model on HuggingFace setting the languages to *mlt_Latn*, *cym_Latn*, and *gle_Latn*, respectively for Maltese, Welsh, and Irish.

COMET We used the Unbabel/wmt22-comet-da model on HuggingFace.

C.1.2 Prompts

We provide the prompts we used to execute all our experiments. In Table C.1 Zero-Shot Minimal Instruction prompt is shown, while in Table C.2 Few-Shot In Context prompt is shown with the examples used for each language tested.

Zero-Shot Minimal Instruction	

Template:	`Write the following triples as fluent` {*English* | *Irish* | *Welsh* | *Maltese* | *Breton*} `text.`
	`Triples: """`
	{*set of triples in format subject predicate object, each triple on a new line*}
	`"""`
	`Text:` *[MODEL]*

Table C.1: Template of the Zero-Shot Minimal Instruction prompt.

C.2 Data-to-Text Generation using PEFT Techniques

C.2.1 Prompts

Table C.3 shows the input template of our fine-tuned Mistral 7B Instruct. Tables C.4 and C.5 show the Zero-shot prompts used with Mistral 7B Instruct and Falcon 40B Instruct, while Tables C.6 and C.7 show the Few-shot prompts used with Mistral 7B Instruct and Falcon 40B Instruct.

C.2.2 Hyperparameters

We executed our experiments using the transformer library[7] of HuggingFace and the paid-for Google Translate API[8] in late March/early April 2024. The systems are tested using the six datasets described in Section 5.4.1. All generated texts are post-processed as described in Section 5.4.3.2. All systems are executed on a Nvidia A100 GPU with 80GB RAM.

Prompting. We set mistralai/Mistral-7B-Instruct-v0.2 parameters to *max seq length*= 512, *seed*= 6787, and *use 4bit*= True.

Model Fine-tuning. We use the PEFT library[9] of HuggingFace to create and load LoRA modules. We set mistralai/Mistral-7B-v0.1 parameters to *max steps*=

[7] `https://huggingface.co/docs/transformers/index`
[8] `https://cloud.google.com/translate`
[9] `https://huggingface.co/docs/peft/index`

Few-Shot In Context	

Template:	`Write the following triples as fluent` {*English* | *Irish* | *Welsh* | *Maltese* | *Breton*} `text.`
	`Triple 1: """`
	{*set of triples in format subject predicate object, each triple on a new line*}
	`"""`
	`Text 1:` {*verbalisation of Triple 1*}
	`##`
	`Triple 2: """`
	{*set of triples in format subject predicate object, each triple on a new line*}
	`"""`
	`Text 2:` {*verbalisation of Triple 2*}
	`##`
	`Triple 3: """`
	{*set of triples in format subject predicate object, each triple on a new line*}
	`"""`
	`Text 3:` *[MODEL]*

English, Irish, and Breton Triples:	`Triple 1: Adolfo_Suárez_Madrid--Barajas_Airport runwayName "14R/32L"`
	`Triple 2: American_Journal_of_Mathematics abbreviation "Am. J. Math."`
	`American_Journal_of_Mathematics firstPublicationYear 1878`
	`American_Journal_of_Mathematics issnNumber "1080-6377"`

English texts:	Text 1: 14R/32L is the runway name of Adolfo Suárez Madrid-Barajas Airport. Text 2: The American Journal of Mathematics was first published in 1878 and is also known by the abbreviated title of Am. J. Math. It has an ISSN number of 1080-6377.
Irish texts:	Text 1: 14R/32L is ainm do rúidbhealach Aerfort Adolfo Suárez Madrid-Barajas Text 2: Foilsíodh an American Journal of Mathematics don chéad uair in 1878 agus aithnítear leis an ainm giorraithe Am. J. Math. chomh maith é. Tá an uimhir ISSN 1080-6377 aige.
Breton texts:	Text 1: Anv leurenn bradañ aerborzh Adolfo Suárez Madrid-Barajas zo 14L/32R. Text 2: Finland zo bro ar Finniz hag hini ar skorndorrer Aleksey Chirikov bet savet e chanter-bigi Arctech en Helsinki.

Maltese and Welsh Triples:	`Triple 1: Albennie_Jones birthPlace Errata,_Mississippi`
	`Triple 2: GMA_New_Media industry Entertainment`
	`GMA_New_Media type Media_company`
	`GMA_New_Media product World_Wide_Web`

Maltese texts:	Text 1: Albennie Jones twieldet f'Errata Mississippi. Text 2: GMA New Media hija kumpanija tal-midja tal-industrija tad-divertiment li toffri servizzi li jikkonċernaw il-World Wide Web.
Welsh texts:	Text 1: Ganed Albennie Jones yn Errata, Mississippi. Text 2: Mae GMA New Media yn gwmni cyfryngau yn y diwydiant adloniant sy'n cynnig gwasanaethau sy'n ymwneud â'r We Fyd Eang.

Table C.2: Few-Shot In Context prompt. **Top:** Template of the prompt. **Centre:** Examples' triple set and texts in English, Irish, and Breton. **Bottom:** Examples' triple set and texts in Maltese and Welsh.

10000, *learning rate*= 2e-4, *max grad norm*= 0.3, *weight decay*= 0.001, *lora alpha*= 16, *lora dropout*= 0.1, *lora r*= 64, *max seq length*= 512, *seed*= 6787, *use 4bit*= True,

Input template fine-tuned Mistral 7B	
Template:	`[INST] Write the following triples as fluent English text. Triples: """` {*set of triples in format subject predicate object, separated by ;*} `"""` `[/INST] [ANS]` {*MODEL*} `[/ANS]`
Example:	`[INST] Write the following triples as fluent English text. Triples: """ Alison_O'Donnell genre Jazz; Alison_O'Donnell recordLabel Deram_Records; Deram_Records location London; Alison_O' Donnell recordLabel Fruits_de_Mer_Records """ [/INST] [ANS]`
Model output:	*Alison O'Donnell, who performs jazz music, was once signed to the record label Fruits de Mer Records. She is now signed with Deram Records, which is based in London* `[/ANS]`

Table C.3: Input template for our fine-tuned Mistral 7B. **Top:** Template of the input. **Bottom:** Complete example in English.

Zero-Shot prompt Mistral 7B Instruct	
Template:	`[INST] Write the following triples as fluent English text. Triples: """` {*set of triples in format subject predicate object, separated by ;*} `"""` `[/INST] Text:` {*MODEL*}
Example:	`[INST] Write the following triples as fluent English text. Triples: """ Alison_O'Donnell genre Jazz; Alison_O'Donnell recordLabel Deram_Records; Deram_Records location London; Alison_O' Donnell recordLabel Fruits_de_Mer_Records """ [/INST] Text:`
Model output:	*Alison O'Donnell is a jazz artist. She is signed under the record label Deram Records. Deram Records is based in London. Alternatively, Alison O'Donnell has also recorded under the record label Fruits de Mer Records.*

Table C.4: Zero-shot prompt for Mistral 7B Instruct. **Top:** Template of the input. **Bottom:** Complete example in English.

Zero-Shot prompt Falcon 40B Instruct	
Template:	`>>QUESTION<< Write the following triples as fluent English text. Triples: """` {*set of triples in format subject predicate object, separated by ;*} `"""` `>>ANSWER<< Text:` {*MODEL*}

Table C.5: Zero-shot prompt for Falcon 40B Instruct.

Few-Shot prompt Mistral 7B Instruct	

Template:	`[INST] Write the following triples as fluent English text.`
	`Triple 1: """` {*set of triples in format subject predicate object, separated by ;*} `"""`
	`Text 1:` {*verbalisation of Triple 1*}
	`##`
	`Triple 2: """` {*set of triples in format subject predicate object, separated by ;*} `"""`
	`Text 2:` {*verbalisation of Triple 2*}
	`##`
	`Triple 3: """` {*set of triples in format subject predicate object, separated by ;*} `"""`
	`[/INST] Text 3:` {*MODEL*}

Fixed examples:	`Triple set 1: Adolfo_Suárez_Madrid--Barajas_Airport runwayName "14R/32L"`
	Text 1: 14R/32L is the runway name of Adolfo Suárez Madrid-Barajas Airport.
	`Triple set 2: American_Journal_of_Mathematics abbreviation "Am. J. Math.";`
	`American_Journal_of_Mathematics firstPublicationYear 1878;`
	`American_Journal_of_Mathematics issnNumber "1080-6377"`
	Text 2: The American Journal of Mathematics was first published in 1878 and is also known by the abbreviated title of Am. J. Math. It has an ISSN number of 1080-6377.

Example Prompt:	`[INST] Write the following triples as fluent English text.`
	`Triple 1: """ Adolfo_Suárez_Madrid--Barajas_Airport runwayName "14R/32L" """`
	Text 1: 14R/32L is the runway name of Adolfo Suárez Madrid-Barajas Airport.
	`##`
	`Triple 2: """ American_Journal_of_Mathematics abbreviation "Am. J. Math.";`
	`American_Journal_of_Mathematics firstPublicationYear 1878;`
	`American_Journal_of_Mathematics issnNumber "1080-6377" """`
	Text 2: The American Journal of Mathematics was first published in 1878 and is also known by the abbreviated title of Am. J. Math. It has an ISSN number of 1080-6377.
	`##`
	`Triple 3: """ Alison_O'Donnell genre Jazz; Alison_O'Donnell recordLabel`
	`Deram_Records; Deram_Records location London; Alison_O' Donnell recordLabel`
	`Fruits_de_Mer_Records """ [/INST] Text 3:`

Model output:	*Jazz artist Alison O'Donnell has previously recorded for Deram Records in London, and currently records for Fruits de Mer Records.*

Table C.6: Few-Shot prompt for Mistral 7B Instruct. **Top:** Template of the prompt. **Middle:** Fixed examples used during testing. **Bottom:** Complete example in English.

Few-Shot prompt Falcon 40B Instruct	
Template:	>>QUESTION<< Write the following triples as fluent English text. Triple 1: """ {*set of triples in format subject predicate object, separated by ;*} """ Text 1: {*verbalisation of Triple 1*} ## Triple 2: """ {*set of triples in format subject predicate object, separated by ;*} """ Text 2: {*verbalisation of Triple 2*} ## Triple 3: """ {*set of triples in format subject predicate object, separated by ;*} """ >>ANSWER<< Text 3: {*MODEL*}
Fixed examples:	Triple set 1: Adolfo_Suárez_Madrid--Barajas_Airport runwayName "14R/32L" Text 1: 14R/32L is the runway name of Adolfo Suárez Madrid-Barajas Airport. Triple set 2: American_Journal_of_Mathematics abbreviation "Am. J. Math."; American_Journal_of_Mathematics firstPublicationYear 1878; American_Journal_of_Mathematics issnNumber "1080-6377" Text 2: The American Journal of Mathematics was first published in 1878 and is also known by the abbreviated title of Am. J. Math. It has an ISSN number of 1080-6377.

Table C.7: Few-Shot prompt for Falcon 40B Instruct. **Top:** Template of the prompt. **Bottom:** Fixed examples used during testing.

and *warmup ratio*= 0.3. We use the checkpoint at step 6000 at inference time as it has the lowest loss based on the validation set. WebNLG 2020 train set is used for the model fine-tuning. The fine-tuning is defined as an instruction-based task where the task description and input are delimited by special instruction tokens ([INST] and [/INST]), and the output is delimited by special answer tokens ([ANS] and [/ANS]).

Appendix D

Appendix of Attribute-specialised PEFT modules for CTG

D.1 Design of Single-Attribute Specialised Modules

D.1.1 Hyperparameters

D.1.1.1 Trained Models

We train all our QLoRA modules with the following pre-trained raw models: `meta-llama/Meta-Llama-3-8B`, `meta-llama/Llama-3.1-8B`, and `mistralai/Mistral-7B-v0.3`. Each model is trained on 3 different seeds (8989, 79817, 794323). All the executions are done on a NVIDIA A100 with 80GB.

We perform a small grid search for each trained module investigating the learning rate (5e-6, 2e-4) and learning rate scheduler (cosine, constant). We use Paged AdamW (Loshchilov, 2017) as the optimiser.

We set all the other hyperparameters as *optimizer*=paged_adamw_32bit, *batch size*=4, *gradient accumulation steps*=4, *maximum gradient norm*=1.0, *warmup ratio*=0.1, *weight decay*=0.5, *group by length*=true, *fp16*=false, *bf16*=true, *maximum sequence length*=1024, *padding*=right, *save strategy*=epoch and we trained the mod-

ules for up to 3 epochs. During inference we set *batch size*=64.

Regarding model quantization, we set *bnb 4bit compute dtype*=bfloat16, *use 4bit*=True, *bnb 4bit quant type*=nf4, and *use nested quant*=False.

Regarding QLoRA hyperparameters, we follow the hyperparameters setting used in the vanilla QLoRA by Dettmers et al. (Appendix B.2). We set *rank*=64, *alpha*=16, *dropout*=0.1 and we attach QLoRA at every linear layer of the pre-trained model ([“q_proj”, “k_proj”, “v_proj”, “o_proj”, “gate_proj”, “up_proj”, “down_proj”, “lm_head”]).

To get the best trained module for each run, we evaluate Control Effectiveness of each saved checkpoint using the validation portion of the train dataset. We determine the best model for each grid search based on the run with highest Control Effectiveness. Tables D.1 and D.2 show the hyperparameters of the best-performing module for each executed grid search.

Model	**Seed**	**Hyperparameters**
LLaMa 3 8B	8989	*checkpoint*=4562, *learning rate scheduler*=cosine, *learning rate*=5e-6
	79817	*checkpoint*=9124, *learning rate scheduler*=cosine, *learning rate*=5e-6
	794323	*checkpoint*=13686, *learning rate scheduler*=constant, *learning rate*=5e-6
LLaMa 3.1 8B	8989	*checkpoint*=4562, *learning rate scheduler*=cosine, *learning rate*=5e-6
	79817	*checkpoint*=13686, *learning rate scheduler*=cosine, *learning rate*=2e-4
	794323	*checkpoint*=13686, *learning rate scheduler*=constant, *learning rate*=5e-6
Mistral 7B	8989	*checkpoint*=13686, *learning rate scheduler*=constant, *learning rate*=5e-6
	79817	*checkpoint*=9124, *learning rate scheduler*=constant, *learning rate*=5e-6
	794323	*checkpoint*=4562, *learning rate scheduler*=cosine, *learning rate*=5e-6

Table D.1: Hyperparameters of the best-performing modules for sentiment control, selected through grid search. The modules were trained on the combined datasets.

Model	Seed	Hyperparameters
LLaMa 3 8B	8989	*checkpoint*=27997, *learning rate scheduler*=constant, *learning rate*=5e-6
	79817	*checkpoint*=27997, *learning rate scheduler*=constant, *learning rate*=5e-6
	794323	*checkpoint*=27997, *learning rate scheduler*=constant, *learning rate*=5e-6
LLaMa 3.1 8B	8989	*checkpoint*=27997, *learning rate scheduler*=constant, *learning rate*=5e-6
	79817	*checkpoint*=13998, *learning rate scheduler*=cosine, *learning rate*=5e-6
	794323	*checkpoint*=41995, *learning rate scheduler*=constant, *learning rate*=2e-4
Mistral 7B	8989	*checkpoint*=13998, *learning rate scheduler*=cosine, *learning rate*=2e-4
	79817	*checkpoint*=13998, *learning rate scheduler*=cosine, *learning rate*=5e-6
	794323	*checkpoint*=41995, *learning rate scheduler*=constant, *learning rate*=2e-4

Table D.2: Hyperparameters of the best-performing modules for topic control, selected through grid search. The modules were trained on the combined datasets.

D.1.1.2 Evaluation Metrics

SLOR We used `gpt2-xl` and `bigscience/bloom-1b7` models from HuggingFace to compute sentence and unigram probabilities.

Control Effectiveness For sentiment control, we used `distilbert/distilbert-base-uncased-finetuned-sst-2-english`[1] and `michelecafagna26/t5-base-finetuned-sst2-sentiment`[2] from HuggingFace, and DeBERTa fine-tuned on Yelp (Gu et al., 2023). For topic control, we used `textattack/distilbert-base-uncased-ag-news`,[3] and `fabriceyhc/bert-base-uncased-ag_news`[4] from HuggingFace, and DeBERTa (Gu et al., 2023). For topic control, we further used a general-purpose instruction-tuned LLM as classifier, namely `CohereForAI/c4ai-command-r-plus-4`

[1] `https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english`
[2] `https://huggingface.co/michelecafagna26/t5-base-finetuned-sst2-sentiment`
[3] `https://huggingface.co/textattack/distilbert-base-uncased-ag-news`
[4] `https://huggingface.co/fabriceyhc/bert-base-uncased-ag_news`

bit.[5]

D.2 Scientific artifacts and licensing

Mistral 7B v0.3 and PPLM prompts are licensed under the Apache-2.0 license. LLaMa 3 8B[6] and LLaMa 3.1 8B[7] are licensed under a commercial license. Yelp Reviews dataset[8] is licensed under a commercial license. The DBPedia ontology classification dataset is licensed under the terms of the Creative Commons Attribution-ShareAlike License and the GNU Free Documentation License. IMDB, SST-2, and AG News datasets do not specify a license. STS benchmark test provides a license for each included dataset.[9] Regarding the classifiers used in the evaluation, DistilBERT and T5 fine-tuned on SST-2 are licensed under the Apache-2.0 license. BERT fine-tuned on AG News is licensed under the Apache-2.0 license, while DeBERTa models fine-tuned by Gu et al. (2023) are licensed under the MIT license. Command R plus is licensed under the Creative Commons Attribution Non Commercial 4.0. The usage of all listed artifacts is consistent with their licenses.

[5] https://huggingface.co/CohereForAI/c4ai-command-r-plus-4bit

[6] https://huggingface.co/meta-llama/Meta-Llama-3-8B/blob/main/LICENSE

[7] https://huggingface.co/meta-llama/Llama-3.1-8B/blob/main/LICENSE

[8] https://s3-media3.fl.yelpcdn.com/assets/srv0/engineering_pages/bea5c1e92bf3/assets/vendor/yelp-dataset-agreement.pdf

[9] http://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark