Building Reliable Surface Realization Systems with Sentence Plans

Henry Elder

B.A., M.Sc.

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University
School of Computing

Supervisors:
Dr. Jennifer Foster
Dr. Alexander O'Connor, Autodesk

May 2021

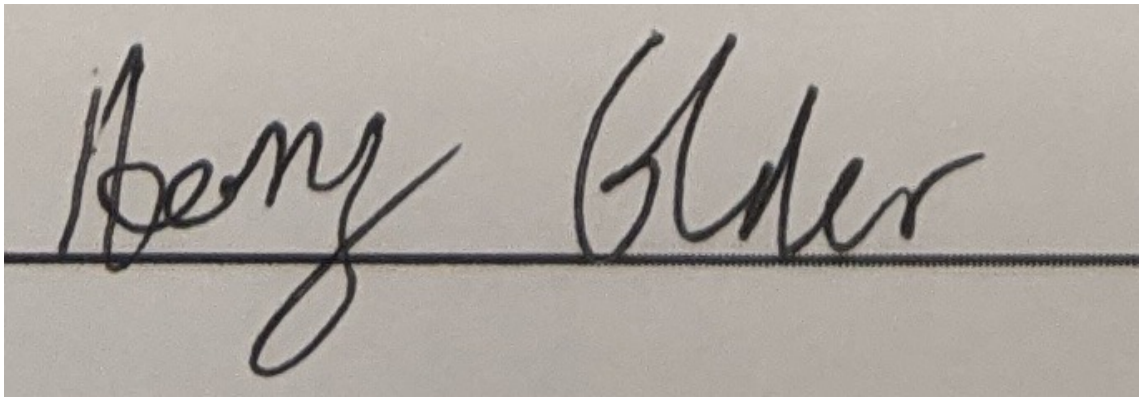
Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D. is entirely my own work, that I have exercised reasonable care to ensure that the work is original, 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 and to the extent that such work has been cited and acknowledged within the text of my work.

Henry Elder

ID No.: 16213103

Date: 17th May 2021



Contents

List of Figures

List of Tables

Building Reliable Surface Realization Systems with Sentence Plans

Henry Elder

Abstract

Neural network-based language models have been shown to generate remarkably fluent and human-like text. Our goal is to incorporate these language models into real life applications, such as surface realization in task-oriented dialogue systems. However these language models cannot be trusted to produce outputs with 100% accuracy. Even in the best case scenario — with large datasets, on relatively simple tasks — neural network-based language models communicate incorrect information in 5% - 10% of cases. Therefore, our research focuses on how to guarantee accurate output. We present experiments and analysis on the use of sentence plans, which we believe are key to improving the performance of neural network-based language models on surface realization tasks. These insights are a key contribution towards the development of more reliable surface realization systems in task-oriented dialogue.

Chapter 1

Introduction

Open up any ecommerce website today and you will likely see a messenger box in the corner of the screen.[1] The messenger box is a text interface, in which the website owners can send messages to greet you and ask if you have any questions about their products. If you choose to respond to these messages, the messenger box may attempt to connect you to a live agent. These are customer service agents, who will try to answer any questions you may have about their products or services. However, customer service agents are a limited resource. Not all websites are able to provide this service 24 hours a day, or to handle a large volume of queries at once. If you ask a question and human assistance is not available, the messenger box may default to an automated system, this is called a chatbot.

Chatbots can respond to simple queries and provide an automated interface to information available on the website. Chatbots vary in the complexity of tasks they can handle. The most basic chatbots act as a missed message service. They collect questions for human agents to respond to once they are available. However, chatbots can be built using more complex systems (Gao et al., 2019). These chatbots are able to respond more appropriately to common customer queries. Queries such as finding clothes sizes on a fashion website or details about company return policies.

[1] https://en.wikipedia.org/wiki/Intercom_(company)

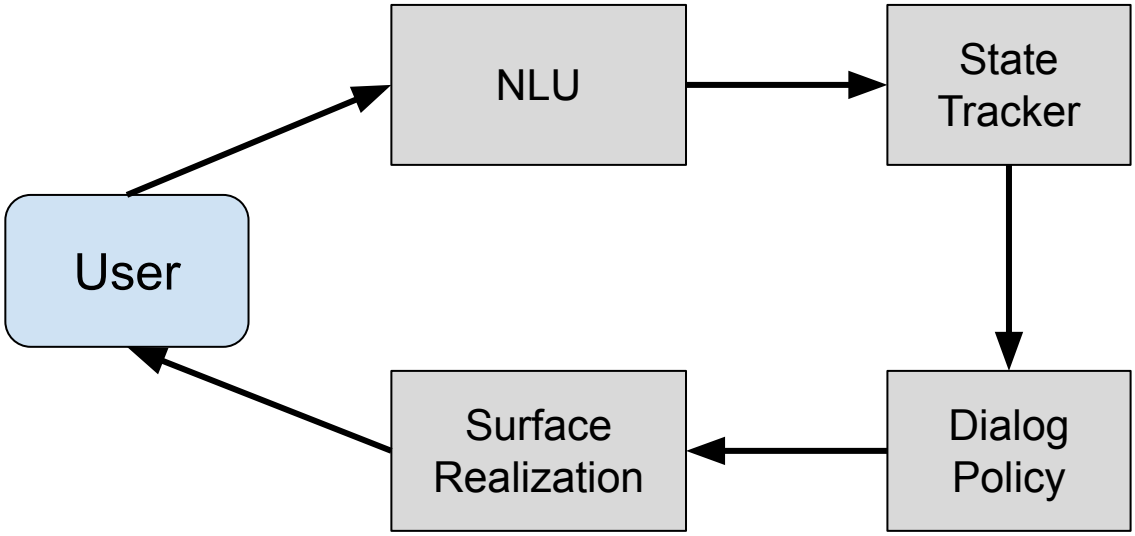


Figure 1.1: A pipeline task-oriented dialogue system

Chatbots with these kinds of narrow focus can be built using task-oriented dialogue systems (Chen et al., 2017). A task-oriented dialogue system (see Figure 1.1) is typically a pipeline system composed of four sequential components: natural language understanding (NLU), state tracking, dialogue policy, and surface realization. NLU analyzes the user's message; it guesses their intent and extracts any key pieces of information from their message. Given the context of previous messages sent by the user, state tracking tracks and summarizes the user's intent and key pieces of information. Dialogue policy determines how the system should respond and retrieves any information needed from its knowledge base to return to the user. Surface realization[2] modules generate a text-based response to the user containing any relevant information from the dialogue policy step.

Task-oriented dialogue systems have seen a vast increase in the complexity of tasks they can handle. This is due to advancements in the use of neural networks for natural language processing (NLP). Two of the four components used in task-oriented dialogue systems have seen major improvements in their performance due to neural networks: NLU (Devlin et al., 2019) and State tracking (Kim et al., 2021). However, surface realization systems have largely stayed the same, unaffected by

[2]Surface realization is a task in natural language generation (NLG)

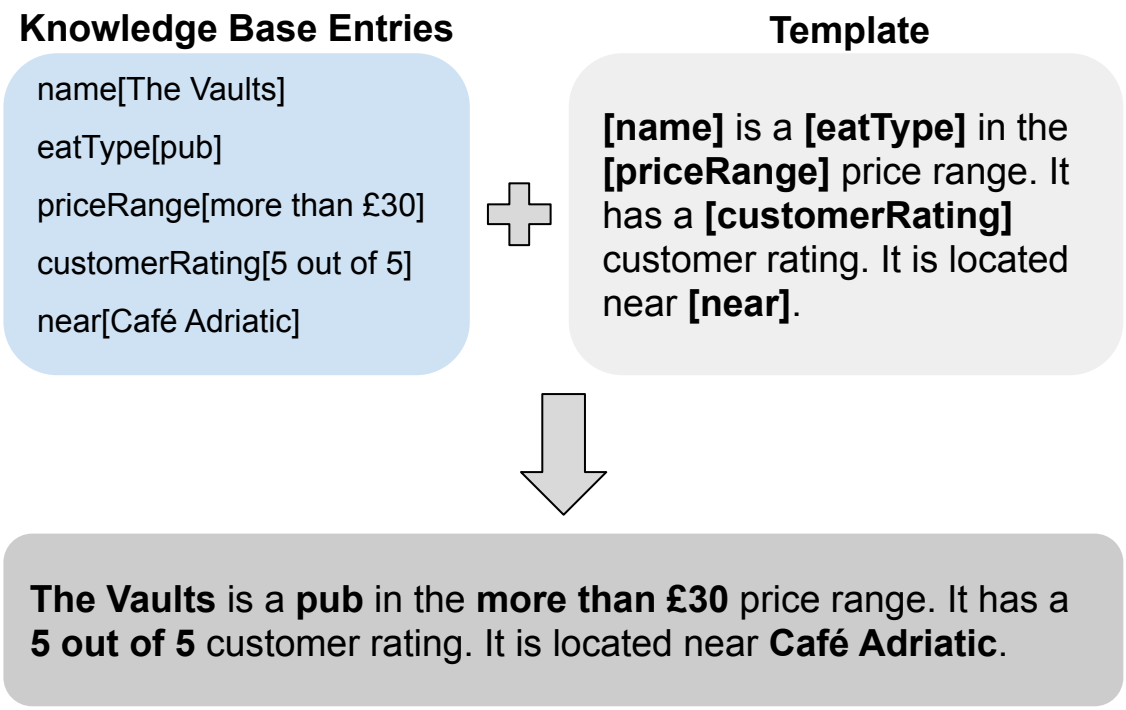


Figure 1.2: A template surface realization system. The template is based on an example from Puzikov and Gurevych (2018).

these advancements (Dale, 2020).

Surface realization is an NLG task (Reiter and Dale, 2000). NLG can be defined as the task of generating text from non-linguistic input (Gatt and Krahmer, 2018). In the dialogue surface realization task, the non-linguistic input is information provided by the dialogue policy component. Information retrieved by the dialogue policy component may come in the form of a set of knowledge base entries. These entries contain semantic information that has been stored in a machine-readable format. Surface realization systems have to perform the following task: generate text containing all and only the information from the provided knowledge base entries; furthermore, this text must sound natural and fluent, as if written by a human.

The most basic surface realization system[3] is a template. This is a pre-written sentence with *slots* for missing information to be filled in. In Figure 1.2, a template sentence has been written with slots, each slot has been labelled for the specific type

[3]Aside from selecting from a list of uneditable canned responses.

of knowledge base entry that can fill it. In the figure, each knowledge base entry is an attribute-value pair, which follows the format *attribute[value]*. The slots in the template can then be filled in by whichever knowledge base entries the dialogue policy manager retrieves. Most, if not all, commercial task-oriented dialogue systems use surface realization systems based on templates, or more complex rule-based variants of templates (Dale, 2020).

Surface realization systems can also be built using neural networks. Instead of templates, neural network-based systems generate text using a language model. A language model is a probability distribution over a sequence of words (Bengio et al., 2003). To generate text from a language model, words are sampled from the probability distribution, one after another, to construct a full sentence or piece of text. Sequence-to-sequence (seq2seq) models (Sutskever et al., 2014) are comprised of a pair of source and target sequence-based neural network language models. Seq2seq models can be trained to do surface realization using example pairs of knowledge base entries and text, see Figure 1.3. We call the input to a neural surface realization system the *source representation*, and the direct input to a seq2seq model the *source sequence*. The source sequence is obtained by converting the source representation into a vector representation that can interact with the seq2seq model. The exact format of the source representation varies between task-oriented dialogue systems, based on type of knowledge base entry, while the utterances are generally human-authored texts containing the corresponding information from the knowledge base entries (Gardent et al., 2017b; Dušek et al., 2020).

Unlike the other task-oriented dialogue components, neural network-based surface realization systems have not seen adoption in commercial chatbots (Dale, 2020). Surface realization systems are required to generate text that is *adequate* and *fluent*. The text is considered adequate if it contains all the relevant information. Fluency has many aspects; the text must be easy to read, sound natural in the given context, and *"flow well"* (Howcroft et al., 2020). While neural surface realization systems

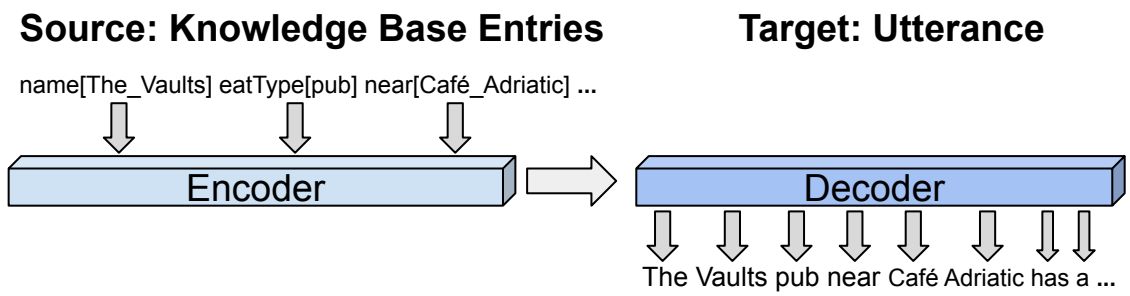


Figure 1.3: A neural surface realization system using a sequence-to-sequence model.

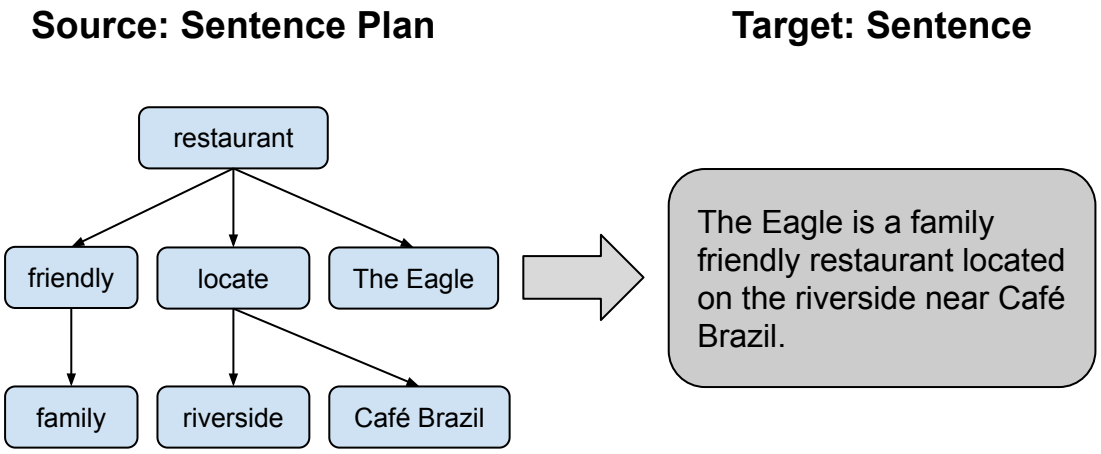


Figure 1.4: A sentence plan source representation and target sentence. The example source-target pair comes from the E2E NLG Challenge Dataset.

can generate fluent text — typically outperforming templates in human evaluation of fluency (Shimorina et al., 2018; Dušek et al., 2020) — the generated text often lacks adequacy. Generated text is deemed to be inadequate when it is missing key information and/or contains incorrect additional information. The reason it is even possible for neural models to generate incorrect information is due to the probabilistic nature of neural language models; the level of control that is normally provided by templates isn't available when generating text by sampling from neural language models. Developing more reliable surface realization systems remains an open problem (Gehrmann et al., 2021).

We hypothesize that the lack of adequacy in neural surface realization systems may stem from how the training data for a task is defined, consisting of pairs of knowledge-base entries and text. We propose that, instead of using knowledge-base

entries as the source representation, *sentence plans* could be used. A sentence plan is an abstract representation of a sentence, see Figure 1.4. Sentence plans contain detailed instructions as to *how* the information is to be expressed in the text, as well as the information itself. Whereas, a set of knowledge base entries contains only the information. We believe that neural surface realization systems with source representations built around sentence plans could provide a number of advantages over knowledge-base entries:

1. Sentence plans contain lower-level details about the desired text than sets of knowledge base entries. These additional details provide more guidance for the model.

2. Sentence plans can be derived from representations designed for parsing tasks, such as the abstract meaning representation (AMR) (Banarescu et al., 2013) and the universal dependency (UD) tree (McDonald et al., 2013). The access to state-of-the-art parsers allows us to automatically annotate sentences, thus obtaining additional training data (Qi et al., 2019). This same feature isn't easily available to knowledge-base entries-to-text tasks. The prime example of this issue being the WebNLG dataset (Gardent et al., 2017b), which was created using manual annotation. Others have not been able to easily create new data for the task through automatically parsing (Guo et al., 2020). It may be possible to address this issue by creating a new parsing system, as done by Oraby et al. (2019), but this approach requires hand crafting and ends up being somewhat more domain specific and limited than more generic approaches involving universal dependency parsers (Elder et al., 2020a).

3. Furthermore, sentence plans may also contain the tokens intended to appear in the target utterance. These tokens can be used in conjunction with restricted decoding in order to ensure that the text has been correctly generated.

However, the sentence plans themselves must still be constructed from the set

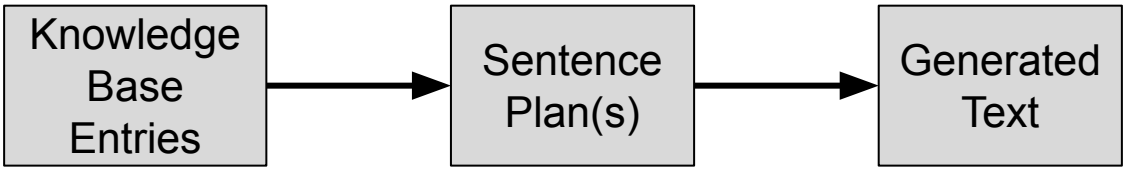


Figure 1.5: The proposed pipeline surface realization system.

of knowledge base entries provided by the dialogue policy component. Generating the sentence plan from a set of knowledge base entries requires adding an extra step to the surface realization system. Depending on the style of chatbot this system is used in, dialogue acts may also be needed to guide the generation of the text from a sentence plan (Stolcke et al., 2000). However, we do not directly address the issue of dialogue acts or their generation in this thesis and leave it to future work.

The focus on sentence plans, as the source representation for the seq2seq models, provides us with the proposed framework for designing surface realization systems. The framework takes the form of a pipeline (see Figure 1.5) similar to rule-based approaches to text generation, but enhanced by the fluency and variety of neural approaches. The pipeline stages are as follows: first the system receives a set of knowledge base entries, this is the information that needs to be included in the generated text. From this set of knowledge base entries, a sentence plan, or plans, needs to be generated. Finally, each sentence plan is transformed into text.

This new framework acts as the basis for our research. There are three main research questions that arise from shifting focus towards sentence plans:

1. What is the best way to use seq2seq models for the sentence-plan-to-text task?

2. Can automatically parsed sentence plans significantly benefit generation?

3. How can a sentence plan be generated from a set of knowledge base entries?

In this thesis, Chapter 2 provides an outline of the different surface realization tasks, an analysis of the implicit alignment present in source-target pairs used for training seq2seq models, and an overview of the different approaches to dataset

creation. Chapter 3 covers how we model the task and evaluate the generated text. In Chapter 4, we discuss our method for the surface realization shared task (SRST) shallow task, and investigate the value of automatically created datasets in the task. Chapter 5 iterates on some of the work done in Chapter 4, to design a sentence plan — based on one of the source representations used by the surface realization shared task — that could be used as part of a neural surface realization system for task-oriented dialogue. Chapter 6 takes the lessons learned from Chapter 5 and designs a pipeline system with a rule based approach to generating sentence plans, then investigates how this affects the reliability and adequacy of the new system. Finally, Chapter 7 discusses the reasons and evidence presented in the thesis for our claim that sentence plans could be used to design better surface realization systems.

In this chapter, we have introduced the problem of designing neural surface realization systems for task-oriented dialogue systems. Neural surface realization systems lack adequacy. We propose to investigate ways to solve this problem by focusing on *sentence plan-based neural generation*.

A list of our first author publications whose work and content is discussed in this thesis.

Chapter 2

- Henry Elder, Sebastian Gehrmann, Alexander O'Connor, and Qun Liu. 2018. E2E NLG Challenge Submission: Towards Controllable Generation of Diverse Natural Language. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 457–462. Association for Computational Linguistics

Chapter 4

- Henry Elder and Chris Hokamp. 2018. Generating High-Quality Surface Realizations Using Data Augmentation and Factored Sequence Models. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 49–53, Stroudsburg, PA, USA. Association for Computational Linguistics

- Henry Elder, Robert Burke, Alexander O'Connor, and Jennifer Foster. 2020a. Shape of Synth to Come: Why We Should Use Synthetic Data for English Surface Realization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7465–7471, Stroudsburg, PA, USA. Association for Computational Linguistics

- Henry Elder. 2020. ADAPT at SR'20: How Preprocessing and Data Augmentation Help to Improve Surface Realization. In *Proceedings of the Third Workshop on Multilingual Surface Realisation*, pages 30–34, Barcelona, Spain (Online). Association for Computational Linguistics

Chapter 5

- Henry Elder, Jennifer Foster, James Barry, and Alexander O'Connor. 2019. Designing a Symbolic Intermediate Representation for Neural Surface Realization. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 65–73, Stroudsburg, PA, USA. Association for Computational Linguistics

Chapter 6

- Henry Elder, Alexander O'Connor, and Jennifer Foster. 2020b. How to Make Neural Natural Language Generation as Reliable as Templates in Task-Oriented Dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2877–2888, Stroudsburg, PA, USA. Association for Computational Linguistics

Chapter 2

Neural Surface Realization

In order to build more reliable neural surface realization systems, we must first understand surface realization tasks, as they are defined for seq2seq models. While the specific modelling details are addressed in Chapter 3, this chapter focuses on the design of neural surface realization tasks. There are three main considerations to be made when designing such a task; how the task is defined, what the source-target alignment is, and how the training data will be created. We categorize neural surface realization tasks by the source representations they use: either a set of knowledge base entries or a sentence plan. Source-target alignment addresses which parts of the source correspond to which parts of the target; alignment affects the quality of trained models and can greatly influence task design choices. Finally, it is important to consider how a dataset will be created for the task; we discuss two alternative approaches, human created or automatically generated.

2.1 Defining the Task

A neural surface realization task can be characterized by its source representation. As previously mentioned, a task-oriented dialogue system's surface realization module uses a set of dialogue acts or knowledge base entries as its input. This set

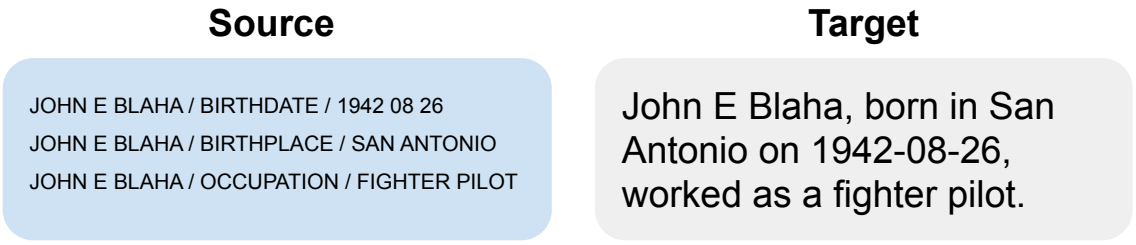


Figure 2.1: Source-target pair from the WebNLG dataset

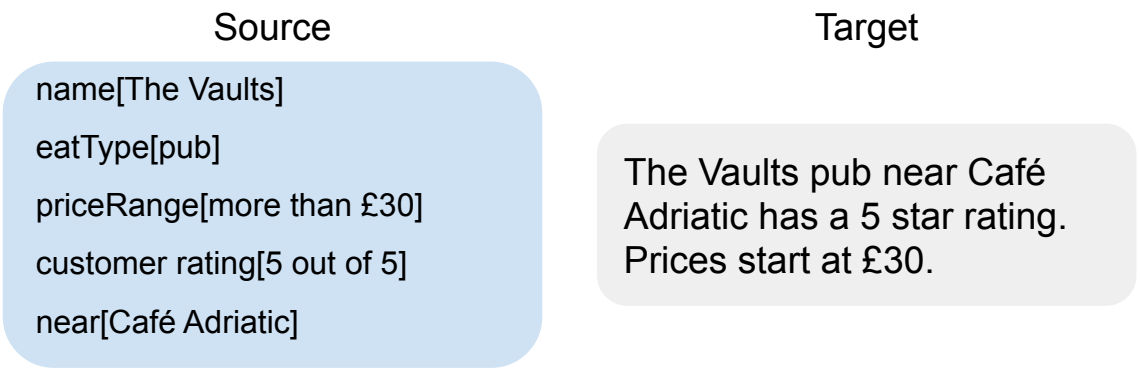


Figure 2.2: Source-target pair from the E2E NLG Challenge dataset

of knowledge base entries is generated by the dialogue policy component of the task-oriented dialogue system (Chen et al., 2017). We call this the *knowledge-base-entries-to-text* task. The other main surface realization task we focus on in this thesis is *sentence-plan-to-text*. In sentence-plan-to-text tasks, the source representation is an abstract representation of a sentence.

Knowledge-Base-Entries-to-Text While knowledge base entries can be returned in any number of user-defined formats, the most common formats used in surface realization are semantic triples and attribute-value pairs (Wen et al., 2015, 2016; Gardent et al., 2017b; Dušek et al., 2020; Gehrmann et al., 2021). We look at two knowledge-base-entries-to-text datasets, the WebNLG Challenge (Gardent et al., 2017b) and E2E NLG Challenge (Dušek et al., 2020); both released as part of shared tasks that were run in 2017. Part of the reason we focus on these two datasets in particular is that we participated in both shared tasks early on in our PhD, this

influenced subsequent research decisions made during the PhD. In the WebNLG Challenge dataset, the source representation is a set of resource description framework (RDF) triples, from the Wikipedia database DBPedia,[1] and the target is a piece of text, containing one or more sentences (see Figure 2.1). In the E2E NLG Challenge dataset, the source representation is a set of attribute-value pairs inspired by a task-oriented dialogue system focused on restaurant descriptions and the target is a descriptive piece of text, containing one or more sentences (see Figure 2.2).

The RDF triples, used by the WebNLG dataset, are composed of three parts; subject, property and object. In DBPedia, *"the subject is a URI (Uniform Resource Identifier), the property is a binary relation and the object is either a URI or a literal value such as a string, a date or a number"* (Gardent et al., 2017b). Figure 2.1 shows a source representation containing three triples; in the first triple, *John E Blaha* is the subject, *birthdate* is the property, and *26/08/1942* is the object.

The attribute-value pairs used in the E2E dataset have eight different types of attributes, such as *restaurant name*, *customer rating* and *price range*. There is a wide range of values these attributes can take. Values use different data types, such as verbatim string, boolean, dictionary or enumerable. For example, the name attribute is a verbatim string and the customer rating attribute is an enumerable — 1 of 5, 3 of 5 or 5 of 5 stars.

In the WebNLG and E2E datasets, the source contains only factual information in the form of knowledge base entries. Stylistic choices about how the text is generated — such as the order of the information, length of the sentences or descriptive vocabulary — are all learned by the model during training from the target text.

Sentence-Plan-to-Text The source representation of a sentence-plan-to-text task is an abstract representation of a sentence. This source representation is often based on a semantic or syntactic parse representation; for instance PropBank semantic

[1] http://wiki.dbpedia.org/dbpedia-dataset-version-2015-10

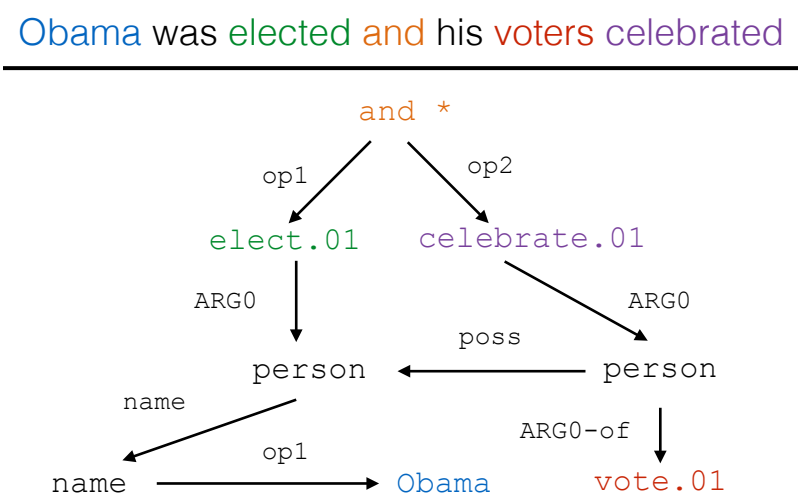


Figure 2.3: "*An example sentence and its corresponding Abstract Meaning Representation (AMR). AMR encodes semantic dependencies between entities mentioned in the sentence, such as 'Obama' being the 'arg0' of the verb 'elected'.*" Graph and quote from Konstas et al. (2017).

roles (Fan et al., 2019) or the Penn Treebank (Belz et al., 2011). We focus on two parse representations in particular: Abstract Meaning Representation graphs (AMR) (Banarescu et al., 2013) and Universal Dependency trees (UD) (Zeman et al., 2017). We chose to focus on the UD representation as we also participated in a UD-based shared task early in the PhD and the AMR representation because of the paper written by Konstas et al. (2017) which influenced a lot of our thinking and choices early on when designing surface realization systems.

AMR was initially created for semantic parsing, but in 2017 the AMR-to-text task (May and Priyadarshi, 2017) was introduced, using a version of the AMR 2.0 dataset[2] with the source and targets switched. Since then a number of works have used AMR-to-text as the basis for generation (Dohare et al., 2018; Fan and Gardent, 2020). *"The AMR is a rooted directed acyclical graph. It contains nodes whose names correspond to sense-identified verbs, nouns, or AMR specific concepts, for example* `elect.01`*,* `Obama`*, and* `person` *in Figure 2.3. One of these nodes is a distinguished root, for example, the node* `and` *in Figure 2.3. Furthermore, the graph*

[2] https://catalog.ldc.upenn.edu/LDC2017T10

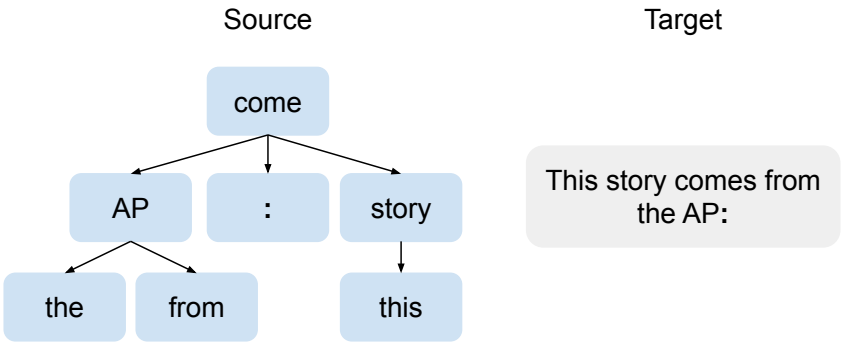


Figure 2.4: SRST shallow task source-target pair: a UD tree and reference sentence

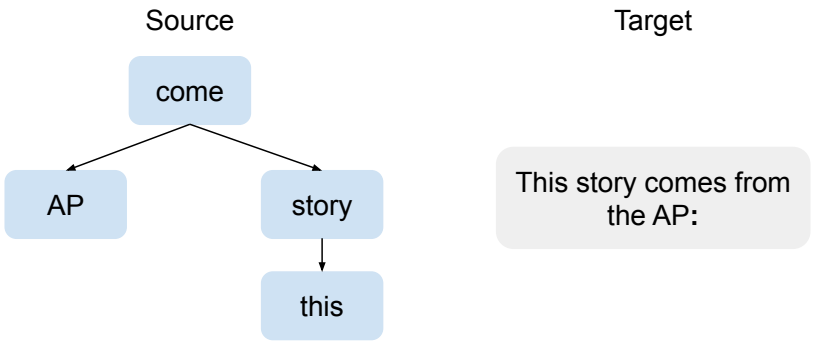


Figure 2.5: SRST deep task source-target pair: an underspecified UD tree and reference sentence

contains labeled edges, which correspond to PropBank-style (Palmer et al., 2005) semantic roles for verbs or other relations introduced for AMR, for example, `arg0` *or* `op1` *in Figure 2.3."*(Konstas et al., 2017)

UD, a dependency annotation framework, was originally used as the basis for the CoNLL parsing shared task (Zeman et al., 2017). Then, in 2018, the surface realization shared task (SRST) workshop introduced two new tasks — each with sentence plans derived from a UD parse tree — the shallow task and the deep task (Mille et al., 2018a). The source representation of the shallow task is an unordered dependency tree with lemmatised nodes; linguistic information is also available, namely part-of-speech tags, dependency relations and morphological features. emph"functional words (in particular, auxiliaries, functional prepositions and conjunctions), and surface-oriented morphological and syntactic information have ad-

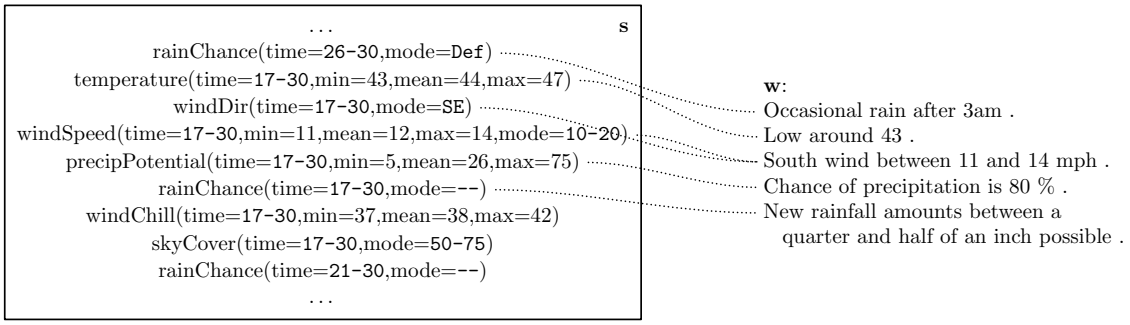


Figure 2.6: Explicit source-target alignments in the WeatherGov dataset (Liang et al., 2009)

ditionally been removed" (Mille et al., 2020). The SRST deep task is also known as the underspecified universal dependency (UUD) representation (Mille et al., 2018b). For the English language portion of the task, the 2018 dataset was created using the English Web Treebank dataset (Silveira et al., 2014). Examples of the shallow task and deep task from this dataset appear in Figures 2.4 and 2.5 respectively.

2.2 Source-Target Alignment

In the previous section, we described several source representations used in neural surface realization tasks. However, these source representations are missing an important feature — one that used to be a requirement for model-based NLG systems (Gatt and Krahmer, 2018) — explicit alignments between tokens in the source and target sequences.

Figure 2.6 shows a training example from the WeatherGov dataset (Liang et al., 2009). In the example, fine-grained alignments have been annotated between the source and target. Explicit alignments have to be included in datasets for most non-neural model-based NLG systems,[3] which adds an extra burden to dataset creation. A major advantage of seq2seq models is that they do not require explicit alignments. This was demonstrated by Mei et al. (2016), when they trained a neural network

[3]Some exceptions to this are Dušek and Jurcicek (2015); Lampouras and Vlachos (2016), though the processes involved can be cumbersome.

model without explicit alignments.[4] Not only did they manage to train their model without alignments, but their system outperformed the prior state-of-the-art on the available datasets.

Seq2seq models are capable of learning these alignments automatically, using mechanisms such as attention (Bahdanau et al., 2015). Nevertheless, it is important to consider what implicit alignments are present in the source-target pairs of the training data, as these are what the model learns from. Poor quality implicit alignments may be a contributing factor to the poor adequacy of a neural surface realization system. Thus, there are three main questions to consider about how the source and target sequence are implicitly aligned with each other, each of which is addressed in the next three subsections;

1. What do the source tokens represent in the target sequence? Section 2.2.1
2. What determines the order of the tokens in the source sequence? Section 2.2.2
3. What is the relationship between tokens in the source sequence? Section 2.2.3

2.2.1 What Do The Source Tokens Align To?

Depending on the surface realization task, there are three main categories for what a token in the source sequence represents in the target sequence:

- a single token
- a phrase or group of tokens
- a stylistic attribute of the text

Source representations typically contain tokens corresponding to one, or more, of these categories.

[4] Although Wen et al. (2015) is an earlier example of a neural network model trained without explicit alignments, they trained mostly on their own dialogue focused datasets. Mei et al. (2016) on the other hand was a more direct continuation of the work in Liang et al. (2009) and used many of the same datasets as Liang et al. (2009).

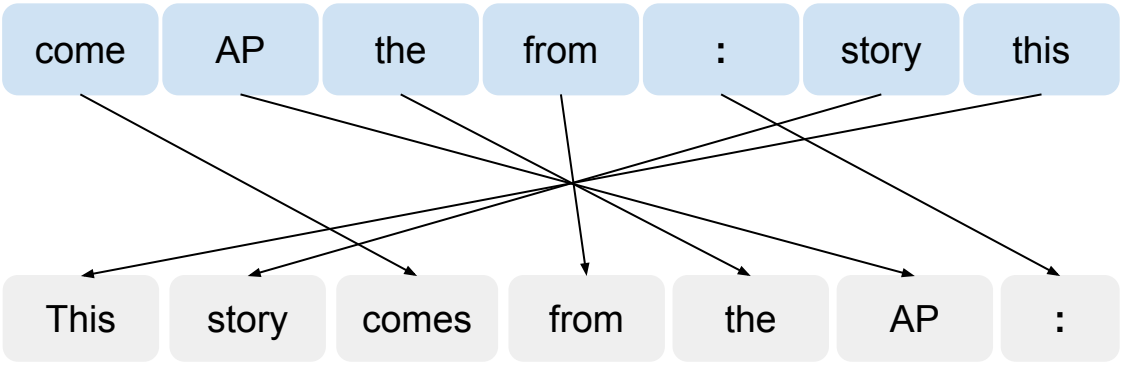


Figure 2.7: Source-target alignment in the SRST shallow task

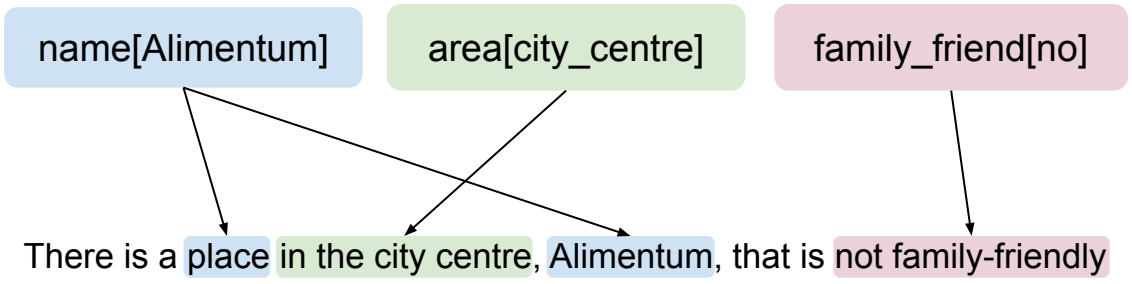


Figure 2.8: Source-target alignment in the E2E dataset

Single Token In this category, a token in the source sequence corresponds directly to another token in the target sequence. In the previous section, we introduced the SRST shallow task. Figure 2.7 shows an example from the SRST shallow task in which tokens from the source sequence appear in the target sequence, either identically or as a conjugated verb — e.g. *AP* to *AP*, *come* to *comes*. There is a 1:1 mapping between tokens in the source sequence and the target sequence.

Phrase or Group of Tokens In this category, a source token corresponds to a particular phrase, or group of tokens, in the target sequence. Figure 2.8 shows an example from the E2E dataset. In this example, each attribute-value pair is a single token but corresponds to a phrase containing multiple tokens in the target sequence. Note that target tokens corresponding to the source token can appear in multiple locations in the sentence, as is the case with the source token *name[Alimentum]* in figure 2.8.

Stylistic Attribute Across All Text In this category, a source token corresponds to a general attribute of the target sequence, such as its style or length. Stylistic attributes are not usually found in surface realization datasets. However, they are worth mentioning as they can offer additional control over generation in surface realization systems (Oraby et al., 2019). We illustrate this with an example from Fan et al. (2018) as the stylistic attributes they use in the source sequence are cleaner and simpler than those used in Oraby et al. (2019).

Fan et al. (2018) introduced a *controllable abstractive text summarization* system with two types of text-wide stylistic attribute tokens; one for controlling the length (short, medium, long), the other for influencing the style of the generated text (Daily Mail or CNN). In this approach, target sequences were labelled before training for the particular stylistic attribute. Then during training, tokens for length and style were added to the start of the source sequence. Finally, by adding different stylistic control tokens to the source sequence at test time, text could be generated in different styles.

2.2.2 What Determines the Order of the Source Tokens?

In most — but not all — seq2seq models, source tokens are passed to the model as a linear sequence. However, the source representations used by many surface realization tasks — such as knowledge base entries or tree-based sentence plans — do not contain any inherent assumptions of linearity. In the E2E dataset, the source is an unordered collection of attribute-value pairs. A decision must be made by the researcher during experiments as to what order to use. Using the E2E dataset's source representation as an illustrative example, we analyze approaches to source ordering. There are three main ways source tokens can be ordered:

- fixed order
- random order

- target sequence order

Fixed Order In this approach, a heuristic is used to determine a consistent order for all examples. Balakrishnan et al. (2019) used an alphabetical order, based on the first letter of each attribute, to decide the order attribute-value pairs should appear in the source sequence. Whereas Juraska et al. (2018) used the default ordering, the order in which attributes appear as provided by the dataset creators. Both their models, each using a different fixed order, achieved broadly similar results on the E2E dataset.

Random Order When using random ordering, the collection of attribute-value pairs is shuffled before being linearized. Every source-target training example in the dataset is shuffled. Both Kedzie and McKeown (2020) and Juraska et al. (2018) found that training a model using randomly ordered attribute-value pairs performed worse than models trained with a fixed order.

Target-Sequence Order In this ordering — also referred to as *alignment training* by Kedzie and McKeown (2020) — attribute-value pairs in the source sequence are given an order corresponding to the order they appear in the target sequence. This ordering requires the researcher to first find the phrase or group of tokens in the target sequence to which each attribute-value pair corresponds, in other words, to perform the alignment. Kedzie and McKeown found that this type of ordering results in the best performing systems. However, there is a significant issue with this type of ordering: it is not possible to know the sentence order of an unseen dataset at test time. Thus, a heuristic or model based solution is required to determine the order of attribute-value pairs at test time. Using an externally determined ordering introduces a further issue; if the attribute-value pairs are put in an order which the seq2seq model has not seen during training, it is more likely to generate poor quality text (Elder et al., 2018).

2.2.3 What is the Relationship between Source Tokens?

In the previous subsection, we demonstrated how there are a number of valid fixed orderings for attribute-value pairs from the E2E dataset, such as the default from the dataset or alphabetical. However, sentence plans contain additional information that influences the order in which tokens should appear in the source sequence. This additional information comes from the graph or tree structure that many parse representations use. While some knowledge-base-entries-to-text tasks do contain a graph structure, this is not as impactful on source ordering as in sentence plans. For instance, in the WebNLG challenge, loose connections could be made between knowledge base entries,[5] though in reality the entries can be treated as independent of each other without affecting performance on the task (Gardent et al., 2017c). Therefore, this section focuses on how the relationships between tokens in sentence plans affect the preprocessing required to convert them into sequence-based representations. Relationships between tokens have two aspects:

1. Edges: a functional feature which indicates which tokens are connected to each other. E.g. parent-child relationship in a tree structure.

2. Labels: these describe the type of relationship between two tokens that share an edge. E.g. dependency relations in universal dependency (UD) trees.

The sentence plans we have introduced so far use a tree or graph structure: AMR-to-text uses a graph and the SRST shallow task uses a tree. The labels used by AMR-to-text are semantic relations, and in the SRST shallow task, the labels are dependency relations. To maximise performance, this structure needs to be taken into account when converting from a sentence plan to the source sequence used by a seq2seq model (Konstas et al., 2017; Elder and Hokamp, 2018).

Edges Konstas et al. (2017) described the methods they used to take structure

[5] In the WebNLG dataset, individual RDF triples in a set often share properties or subjects.

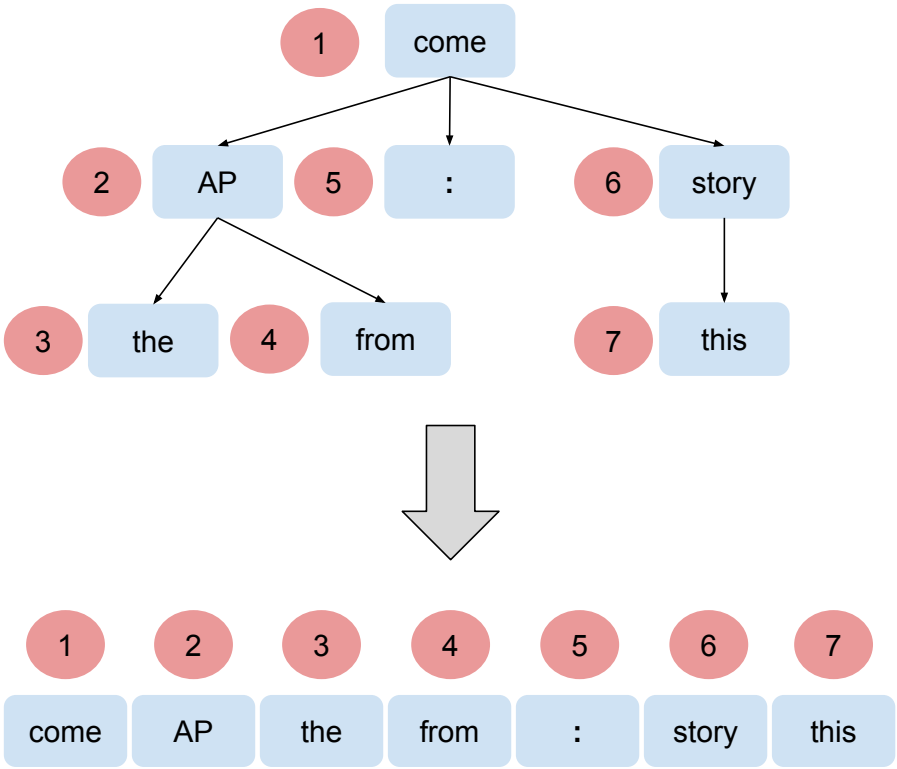


Figure 2.9: Depth-first linearization of a UD tree from the SRST shallow task.

into account for the AMR-to-text task. We also applied these ideas to our own work on the SRST shallow task (Elder and Hokamp, 2018). To capture the information about edges in the source representation there are two main preprocessing steps; depth-first linearization and scoping brackets.

Depth-first linearization is used to convert a graph or tree into a linear sequence. Figure 2.9 shows depth-first linearization for an example from the SRST shallow task. This linearization begins at the root node and adds each subsequent child to the sequence, before returning to the highest node not yet added. There is also a decision to be made when there are multiple children at the same level: in what order should they be explored? Konstas et al. found that the best order to use was the one *"presented by human authored AMR annotations"*. The second best order to use was a globally consistent, heuristic-based ordering. Finally they found that a random order performed the worst. This lines up with findings of Kedzie and McKeown (2020), they also determined the performance rank of orders as: sentence

ID	LEMMA	FORM	UPOS	XPOS	FEATS	HEAD	DEPREL	DEPS	MISC
1	story	_	NOUN	NN	Number=Sing	3	nsubj	_	_
2	this	_	DET	DT	Number=Sing, PronType=Dem	1	det	_	_
3	come	_	VERB	VBZ	Mood=Ind, Number=Sing, Person=3, Tense=Pres, VerbForm=Fin	0	root	_	_
4	the	_	DET	DT	Definite=Def, PronType=Art	5	det	_	_
5	AP	_	PROPN	NNP	Number=Sing	3	obl	_	_
6	from	_	ADP	IN	_	5	case	_	_
7	:	_	PUNCT	:	_	3	punct	_	_

Table 2.1: CoNLL-U format source representation from the SRST shallow task

based order, consistent order, random order. However, both of these experiments missed a key alternative approach to ordering, *permanently random linearizations*. In this ordering, every time the model sees a training example of the same tree its order will have been reshuffled using a different random depth-first linearization. We will discuss this approach more in Chapter 4.

Konstas et al. also used *scoping brackets* to provide additional context to the model about the order of the tokens. A left bracket '(' was added to the sequence after each node with children, once the graph had been fully traversed a right bracket ')' was added. By including scoping brackets in the sequence, they could indicate parent-child relationships. Although, to limit the length of the source sequence, they did not use scoping brackets around nodes with a single child node.

Labels These describe the type of relationship between tokens. The AMR source representation uses semantic relations, in the style of PropBank relations (Palmer et al., 2005). These are used to describe the semantic connection between tokens in a sentence. The SRST shallow task uses dependency relations, which describe the relationship between tokens, such as whether one token is a subject, object or modifier of another. It is important to include these labels in the source sequence as they contain key information about *how* the sentence ought to be generated. There are two main approaches to including labels in the source. The approach

used by Konstas et al. (2017) was to add labels to the source sequence by inserting them as tokens next to their corresponding nodes and edges during the depth-first linearization. However, this approach increases the length of the source sequence and grows linearly with the number of tokens in the source representation. This is an issue for seq2seq models which often have a fixed upper input length limit that can be approached by annotated token sequences. Another approach is to include labels as source token features. This is a special modelling feature introduced by Sennrich and Haddow (2016) which we will discuss in Section 3.2.

While most knowledge base entry source representations do not contain information about relationships between the entries themselves, it is possible to specify this information manually. Balakrishnan et al. (2019) proposed relationship types, based on conjunctions, between entries in the E2E source representation to give more detail about the sentence to be generated. The additional information provided by these conjunction tokens gave them more control over the expressibility of the generated text. Balakrishnan et al. (2019) released an annotated E2E dataset with these new conjunction tokens, as well as a new dataset, WeatherE2E, which also used this representation.[6]

2.3 Creating the Dataset

The key to training reliable seq2seq models is having large, high quality datasets. Datasets consist of example pairs of source representations and target text.

As shown in Figure 2.10, creating datasets for surface realization is a two step process; starting with either the source or target, a complementary target or source is created. A complementary source or target can be created using human authors and annotators, or they can be created automatically, using models and heuristics. In the knowledge-base-entries-to-text task, the source is constructed by sampling a

[6] https://github.com/facebookresearch/TreeNLG

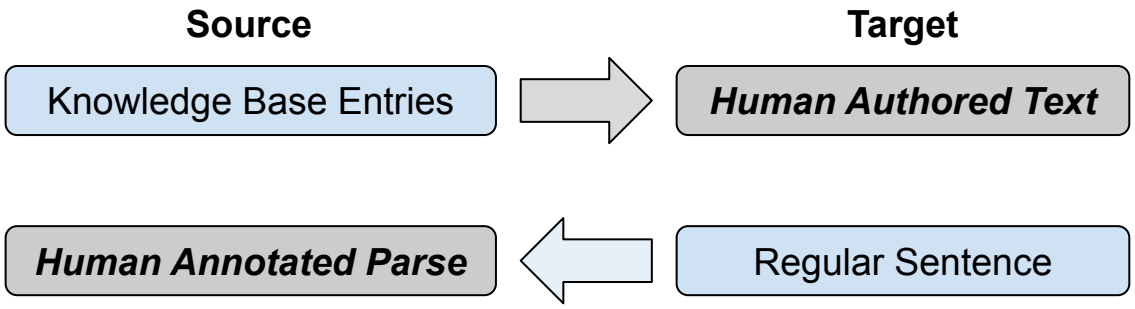


Figure 2.10: Creating source-target pairs by generating the complementary source or target

set of entries from the knowledge base; this set is then given to a person who writes the target text. The reverse occurs in the sentence-plan-to-text task; a pre-existing human-authored sentence is selected as the target, this sentence is then given to a person to annotate in the format of the source representation.

2.3.1 Human Created Complements

Knowledge-Base-Entries-to-Text To create datasets for knowledge-base-entries-to-text tasks, we first begin with the source representation. The E2E dataset creators chose an unordered set of attribute-value pairs from the restaurant domain as their source representation (Dušek et al., 2020). This representation had been used previously by other task-oriented dialogue datasets, also created for seq2seq models (Wen et al., 2015). Whereas, the WebNLG dataset creators chose RDF triples from DBPedia as their source representation (Gardent et al., 2017a). This meant they had a wide number of domains to choose from when constructing source representations for their dataset. Gardent et al. first used a probabilistic model to construct miniature graphs of related entities and properties, and then converted the graphs to their source representation. By sampling from this model, they could generate additional source representations.

Once the dataset creators had established how source representations would be generated, writing the target text was outsourced to the crowdworking platform

Amazon Mechanical Turk. By using human authors, the dataset creators expected to collect texts that were both adequate and fluent; namely, texts that included all relevant information from the source representation, and were written fluently. However, there are issues of quality control when using crowdworking platforms (Daniel et al., 2018). Crowdworkers may not be legitimate, they may try to exploit the data collection system to obtain payment without fulfilling the task correctly. Additionally, even if a crowdworker is legitimate, they may genuinely misunderstand the task or accidentally write poor quality text. Therefore, the E2E and WebNLG dataset creators each used a unique data collection method to ensure that the human-authored texts met their dataset creation goals.

The creators of the E2E dataset noted that in previous crowd-sourced task-oriented dialogue datasets, e.g. Wen et al. (2015), the human-authored target texts were rather stilted and repetitive. Crowdworkers often copied text directly from the attribute-value pair into a sentence. The goal for the E2E dataset, however, was to create a rich and diverse set of target texts. Therefore, instead of giving crowdworkers text-based prompts, they experimented with providing a visual image of the source representation (Dušek et al., 2020). The creators of the E2E dataset found this led the crowdworkers to write more diverse text. Ultimately, they used these visual representations to collect 20% of the full dataset, with textual representations used for the remainder.

The WebNLG dataset was created to test the limits of an automated system's ability to learn surface realization; the WebNLG creators wanted a dataset which challenged NLG systems to do *"lexicalisation, aggregation, surface realisation, referring expression generation and sentence segmentation"* (Gardent et al., 2017b). In order to do this, they needed a dataset that would mimic the act of combining knowledge base entries into an utterance. Thus, instead of asking crowdworkers to write text for a set of RDF triples, the crowdworkers wrote text for a single RDF triple. Then, in subsequent rounds of data collection, the crowdworkers would write

utterances that combined the text of these individual RDF triples. This resulted in an utterance with a source representation of two or more RDF triples based on the original pieces of text. This sequential approach enabled the WebNLG creators to ensure the accuracy of crowdworker written utterances. They could now verify that there was sufficient word overlap between the original pieces of text — written for the single RDF triples — and the newly written utterance.

The E2E NLG Challenge dataset[7] contains 51,426 examples of attribute-value pairs and utterances: 42,061 training, 4,672 validation, 4693 test. The *seen* portion of the WebNLG Challenge 2017 dataset[8] contains 19,944 examples of knowledge graphs and utterances: 18,102 training, 871 development, 971 test.

Sentence-Plan-to-Text Now we look at the opposite problem, starting with a human-authored sentence, how can we annotate the sentence to get a source representation? While the E2E and WebNLG datasets were created with neural surface realization in mind, many sentence plan representations used in surface realization (Belz et al., 2011; Mille et al., 2018a) were originally designed solely for parsing tasks. Expert linguistic annotators use a strict set of annotation rules and guidelines to annotate sentences in a corpus. Many of these annotators are also researchers or co-authors on the papers, so their incentives are aligned to create high quality data. They may still make mistakes due to conflicts in guidelines, human error or inherent ambiguity of the data itself, but intentional subversion, as sometimes occurs when using Mechanical Turk, tends to be absent. Task-specific annotation tools and detailed instruction / training are provided to the annotators.

Both the AMR 2.0[9] and Universal Dependencies English Web Treebank (EWT)[10] (Silveira et al., 2014) datasets were created using this approach. The AMR 2.0 dataset contains 39,260 sentences: 36,521 training, 1,368 validation, 1,371 test. It

[7] https://github.com/tuetschek/e2e-dataset
[8] https://gitlab.com/shimorina/webnlg-dataset
[9] https://catalog.ldc.upenn.edu/LDC2017T10
[10] https://github.com/UniversalDependencies/UD_English-EWT

used *"English natural language sentences from broadcast conversations, newswire, weblogs and web discussion forums."* The EWT dataset contains 16,622 sentences: 12,543 training, 2,002 validation, 2,077 test. The sentences were *"taken from various web media including weblogs, newsgroups, emails, reviews, and Yahoo! answers"*.

2.3.2 Automatically Generated Complements

Whether using crowd-sourcing or linguistic annotators, collecting large datasets remains a significant challenge. Each of the datasets discussed so far — E2E, WebNLG, AMR 2.0 and EWT — contains only tens of thousands of examples. Compared with machine translation, where public datasets for popular language pairs can reach millions of examples (Bojar et al., 2014), these surface realization datasets are relatively small. Therefore, it is necessary to investigate any means by which the available datasets can be augmented, or new datasets created.

In this subsection, we look at how complements can be automatically generated for the knowledge-base-to-text and sentence-plan-to-text tasks. Complements can be generated by a model or a set of heuristics. Knowledge-base-entries-to-text tasks will have a harder time automatically generating complements, as this requires generating text, the task for which the models are already being trained and need additional data to improve. Automatically annotating new training data is easier for sentence-plan-to-text tasks, because they come from parsing tasks which are designed to generate annotations from new sentences. However, they may still suffer from possible issues of parsing quality. One of the main issues with automatically generating complements for any task is that of data quality. It may be possible to generate large datasets, but this data is not very useful if the implicit alignments are not clear to the model. The WebNLG creators noted such a problem with the WikiBio dataset[11] (Lebret et al., 2016), *"we manually examined 50 input/output*

[11]The WikiBio dataset was created by automatically matching tables of biography data with the first two or three sentences of the Wikipedia entry

pairs randomly extracted from this dataset and did not find a single example where data and text matched" (Gardent et al., 2017b). Similarly Wiseman et al. (2018) paired basketball-game summaries with a corresponding set of game statistics to create a large data-to-text dataset. However, due to the automated nature of the dataset creation, the target text does not always align well with the source data.

Knowledge-Base-Entries-to-Text To automatically generate data for the knowledge-base-entries-to-text task, one must either take a source representation and generate text for it, or take an existing sentence(s) and annotate it with a source representation. The approach used by Kedzie and McKeown (2019) was to generate sentences from randomly sampled source representations. They passed these new source representations to a seq2seq model trained on the original dataset, and let it generate text. To ensure the generated text was accurate, they applied a rule-based parser to the generated text to validate that all of the attribute-value pairs from the source representation had appeared in the text. This allowed Kedzie and McKeown to filter out utterances that did not express all of the attribute-value pairs, and maintain a higher quality dataset. They used this method to automatically generate an additional 500,000 training examples. It was found that a model trained on this 500,000 example dataset achieves much higher adequacy than when trained on the original dataset. The key to their success was the use of a rule based parser, created using a large set of hand-crafted regular expressions.

There are limitations to the approach of generating new utterances from a trained model. The main issue is that the new model trained on the generated data only learns what the other model already knows. In Kedzie and McKeown (2019), they relied on an accurate rule-based parser. If this did not exist, the generated text would have contained more errors and been lower quality. This limitation was highlighted particularly by the case of WeatherGov (Liang et al., 2009), which used a rule based system to generate training data. Any model trained on this data, would

only learn to recreate what the rule based system had generated, and wouldn't necessarily provide any new value (Reiter, 2017). However, research from Arun et al. (2020) and Stevens-Guille et al. (2020) show the case may be slightly different for seq2seq models, where self-training could help with bootstrapping from small, human-authored datasets to larger datasets.

An alternative approach to creating data for knowledge-base-entries-to-text tasks is to classify existing sentences, and create source representations for those sentences. This was attempted by the CycleGT team from Amazon, in their submission to the 2020 WebNLG shared task (Guo et al., 2020). They increased the amount of their training data by annotating unseen sentences from Wikipedia by aligning them with database entries from DBpedia (Jin et al., 2020). However, this method of aligning database entries and sentences relied on n-gram overlap between keywords that appeared in both the text and the database entries. This approach to obtaining source-targets provides little guarantees of accuracy and may suffer from similar issues of data quality as Lebret et al. (2016).

Sentence-Plan-to-Text Automatically generating training data for sentence plan tasks is comparatively easier, as these representations were originally designed for parsing. As such, generating data involves finding existing sentences and annotating these sentences using a parser. We found no examples of the complementary approach; creating additional training data by generating target sentences from an existing sentence plan. This approach would involve generating sentence plans from a model or set of rules, then using a pretrained model to generate utterances corresponding to the input.

Konstas et al. (2017) automatically annotated an additional 20 million sentences for their AMR-to-text system. Though they found additional data provided additional gains, there were decreasing marginal returns to the use of this data — the more data that was added, the smaller the relative increase in performance. Konstas

et al. performed experiments in which 200,000, 2,000,000 and 20,000,000 automatically generated examples are used to train a model. Each time a new, larger dataset was used the benefit to performance became smaller and smaller. Our own work, discussed in Chapter 4, utilizes automatic annotation to create an augmented dataset, also known as silver data, silver parse data, or silver standard data (Filannino and Di Bari, 2015).

Semantic or syntactic parsers can also provide a method for researchers to create datasets on new topics. An example of a dataset created this way is YelpNLG (Oraby et al., 2019). YelpNLG was created by parsing millions of human-authored sentences from the Yelp reviews corpus. Similar to E2E, they defined their attributes up front: *"the lexicons include five attribute types prevalent in restaurant reviews: restaurant-type, cuisine, food, service, and staff collected from Wikipedia and DBpedia"*. Their values were taken as words with a particular dependency relation to the attribute token(s). They were then able to filter for sentences containing examples of these pairs. Afterwards, a heuristic was applied to extract attribute-value pairs using the parser annotations.

2.4 Summary

In this chapter, we discussed the neural surface realization task. We demonstrated how tasks can be defined by their source representations and introduced examples of knowledge-base-entries-to-text and sentence-plan-to-text datasets. We then discussed the different ways source-target pairs can be implicitly aligned, and how it may affect the quality of the training. Finally, we addressed the issue of dataset creation. Datasets are of great importance when training seq2seq models. While human-created datasets may have the highest quality, it's important to consider the available methods for generating additional training data. Though we introduced the AMR and WebNLG datasets, in the experiment chapters of the thesis we focus

on the E2E dataset as a proxy for a task-oriented dialogue system, and use the SRST source representations as a core focus. We chose to focus on both the E2E dataset and SRST source representations as we had familiarity with them from their respective shared tasks and found them to have decent quality datasets with which we were able to do efficient experimental work.

Chapter 3

Modelling and Evaluation

Thus far, we have discussed how to define and design surface realization tasks. The main focus of our research is the design of surface realization systems that are more reliable at generating adequate text. A key to achieving this goal is understanding the models trained for the task.

In this chapter, we introduce the main neural network modelling paradigm that we follow, sequence-to-sequence (seq2seq) modelling. Following that, we highlight the modelling features which are most important for surface realization. Finally, we address the problematic task of evaluating generated text.

3.1 Seq2seq Modelling

Neural network modelling for surface realization follows the same paradigm as neural machine translation (NMT) — seq2seq models. In this section, we provide a brief overview of three key components; word embeddings, seq2seq models and attention. For a more detailed technical introduction to neural network modelling, given in the context of natural language generation, we recommend the book *Deep Learning Approaches to Text Production* by Narayan and Gardent (2020).

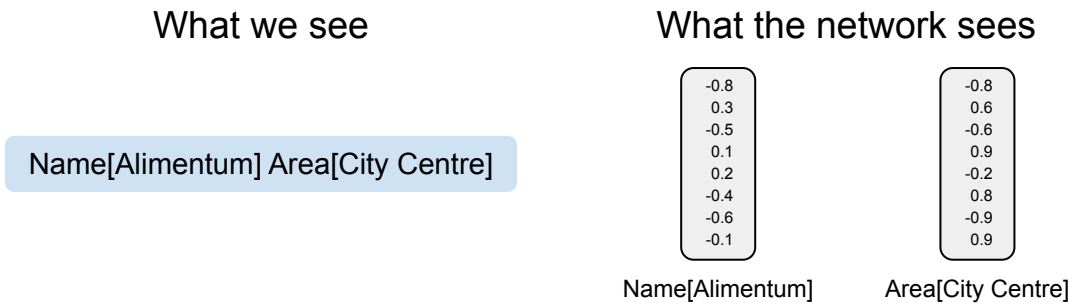


Figure 3.1: Word Embeddings

Word Embeddings In neural network-based modelling, text is mapped to N-dimensional feature vectors, known as word embeddings (Bengio et al., 2003; Mikolov et al., 2013) (see Figure 3.1).

Word embeddings can capture semantic and syntactic relationships between tokens. Semantic relationships are indicated by the closeness of embeddings in the vector space. Similar nouns tend to be grouped together. For instance *horse*, *cow* and *pig* are closer to each other in the vector space than to *cat* or *dog*. Syntactic relationships are revealed by directional vectors. Directional vectors between different verb tenses are similar to each other: *walked* to *walking* and *swam* to *swimming*.

During modelling, text from source-target pairs is divided up into a sequence of tokens, each of which is assigned an embedding. However, tokenization can be flexible. In Figure 3.1, each attribute-value pair has an embedding, e.g. *Name[Alimentum]* and *Area[CityCentre]*, alternatively, each word in the attribute-value pair could be assigned a word embedding, e.g. Name, Alimentum, Area, City, Centre. The choice of tokenization is important, particularly in the source sequence. Source sequence tokenization can affect the use of modelling features, such as restricted decoding, which we will discuss in Section 3.2.

Seq2seq Models When used with seq2seq models, the source representation and target text are sequences of tokens. These sequences of tokens are converted into word embeddings, and can be modelled using recurrent neural networks (RNNs) with

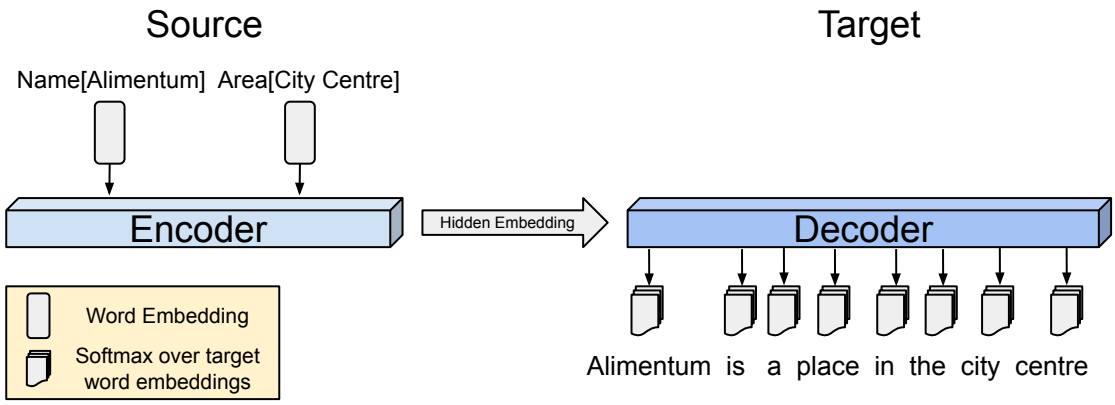


Figure 3.2: Sequence-to-sequence model

a long short-term memory (LSTM) cell (Hochreiter and Schmidhuber, 1997).[1] RNNs can capture long range dependencies between tokens (Khandelwal et al., 2018). Neural surface realization systems use RNNs in an encoder-decoder framework known as a seq2seq model, popularised by NMT, to convert source representations into text (Sutskever et al., 2014), see Figure 3.2. The encoder converts tokens into context-aware fixed-length vector representations known as hidden embeddings. The decoder network is a language model with a probability distribution over a vocabulary of tokens. Text is generated by sampling from a language model, a single token at a time.

Attention Attention is a modelling feature that enables seq2seq models to focus on particular parts of the source sequence during decoding, see Figure 3.3. The mechanism, originally developed for NMT, greatly improves the performance of seq2seq models (Bahdanau et al., 2015). Attention works by creating a distribution over each hidden embedding in the encoder. The distribution is obtained by first taking the dot product of a query vector and the hidden embeddings to get a score for each, then a softmax function is applied to the scores, creating the attention distribution.

[1] While numerous variants of the LSTM-RNN have been proposed, Melis et al. (2017) demonstrated that many of these approaches can be outperformed by the standard LSTM architecture, when it is properly regularised.

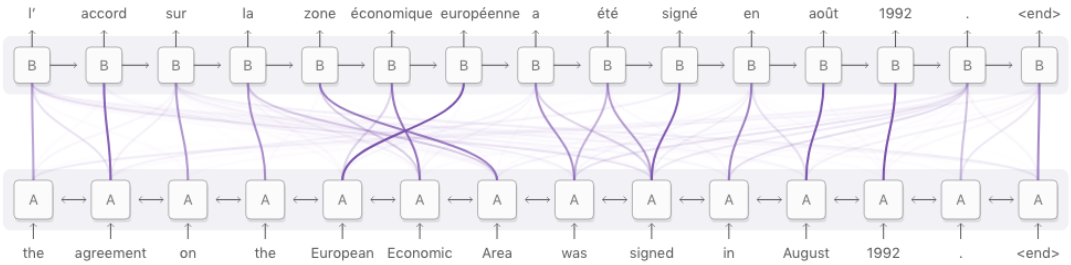


Figure 3.3: *"Attention network diagram. Line thickness indicates the alignment weights between words in the source and target sequences"* (Olah and Carter, 2016).

Since attention was introduced, its importance in language modelling has increased. Vaswani et al. (2017) introduced transformers — encoder-decoder models built entirely out of attention modules — which outperformed LSTM-based seq2seq models on NMT tasks, see Figure 3.4. There are some key differences to how transformers operate. During encoding, each token in sequence is processed simultaneously by the transformer; there is no inherent concept of a sequence in the model. Instead a positional encoding must be used to indicate the order of tokens. Transformers can only look at a fixed number of tokens at a time, compared with LSTM-based seq2seq models which can process an indefinite number of tokens in a sequence, one after the other. Another key difference is the prevalence of subword encoding in transformer based language models. While subwords were used in prior research with LSTM-based seq2seq models (Sennrich et al., 2016), the dominant paradigm is by far to use subword tokenization when training large language models using transformers (Devlin et al., 2019; Raffel et al., 2020). A final key difference is that transformers benefit greatly from pretraining. Previous work had attempted to use pretraining to improve LSTM-based seq2seq model performance (Ramachandran et al., 2017) but the benefit was nowhere near what we have since observed with transfomers.

Devlin et al. (2019) were perhaps the first to demonstrate that a transformer-based encoder model could use pretraining on large amounts of unannotated data to achieve state-of-the-art in many NLP tasks. This began a trend of pretraining

transformer-based models on larger and larger amounts of data and scaling up model size from millions to billions of parameters. Evidence from Brown et al. (2020) indicates that transformers have the potential to continue scaling, both in amount of training data and model size.

Transformers require a lot of training data to perform satisfactorily, otherwise they must first be pretrained on unannotated text and then fine-tuned on a particular task. Research into fine-tuning for seq2seq NLG tasks such as surface realization has been limited, though a number of entries to the WebNLG Challenge 2020 (Castro Ferreira et al., 2020) explored fine-tuning the seq2seq T5 model to some success (Raffel et al., 2020).

Two entries to the 3rd Multilingual Surface Realization Shared Task 2020 utilized pretrained transformers (Mille et al., 2020). Farahnak et al. (2019) fine-tuned a BART model (Lewis et al., 2020) to generate text directly from a set of lemmas and Sobrevilla Cabezudo and Pardo (2020) applied one of the smaller GPT-2 models (Radford et al., 2019) to the deep task. While neither system performed particularly well relative to other participants in the task, it seems evident from the overall performance of transformers in the NLP space that if many of the modelling features discussed in the subsequent section, 3.2, were to be correctly integrated with a pretrained transformer, the state-of-the-art in surface realization could likely be further advanced.

3.2 Surface Realization Seq2seq Extensions

There are three neural network-based modelling mechanisms that we have found very helpful for increasing the quality of generated text from surface realization systems. These mechanisms were originally designed for NMT systems and provide additional functionality that is unavailable when using standard seq2seq models. They are: copy attention, source features, and restricted decoding.

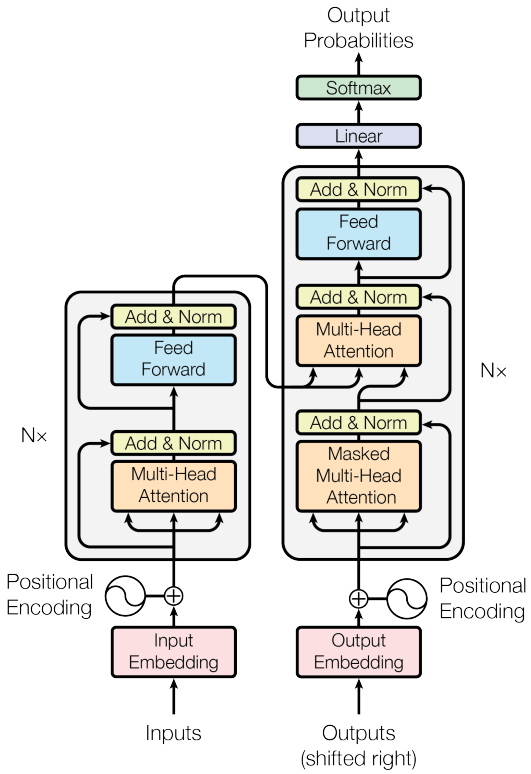


Figure 3.4: Model architecture of a transformer from (Vaswani et al., 2017).

Copy Attention Out of all these mechanisms, copy attention is the most important. It enables a model to handle out-of-vocabulary tokens — tokens which do not appear in its language model — by copying them directly from the source sequence into the generated text. A language model's vocabulary of tokens is limited, both by the available data and the computation required for a larger vocabulary. Originally introduced as pointer networks (Vinyals et al., 2015), copy attention gained popularity after See et al. (2017) applied it to the task of abstractive text summarization.[2] In surface realization, copy attention is particularly relevant with sentence-plan-to-text tasks, as the source sequence may contain tokens that need to be included directly in the generated text but have never been seen by the model before (Elder and Hokamp, 2018).

[2] http://www.abigailsee.com/2017/04/16/taming-rnns-for-better-summarization.html

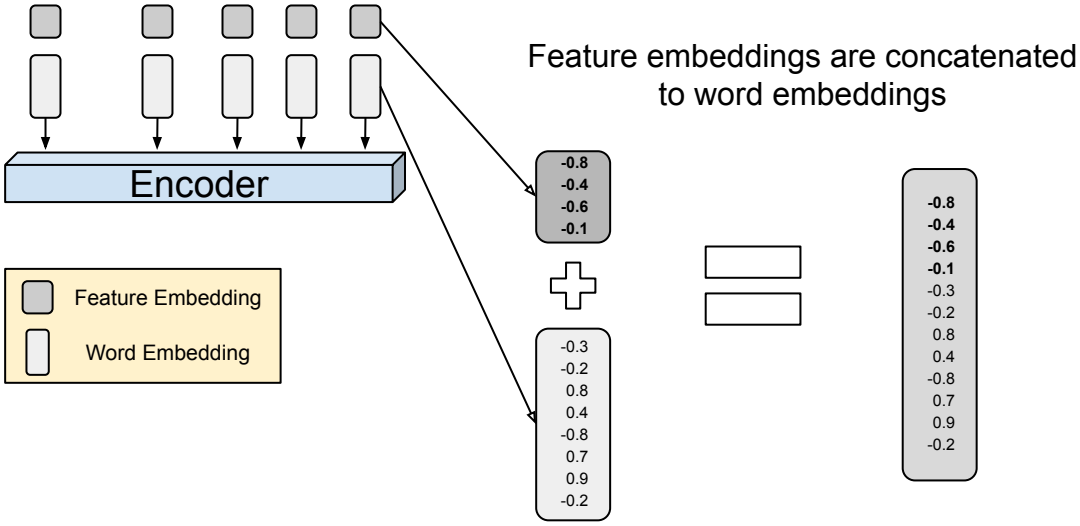


Figure 3.5: Feature embeddings are concatenated to word embeddings, the new embedding is then passed to the encoder.

Source Token Features Source token features, also known as factored sequence models (Sennrich and Haddow, 2016), are another type of embedding. Source feature embeddings are concatenated to word embeddings on a token-by-token basis, see Figure 3.5. Source features were mentioned in Section 2.2.3, in reference to the SRST shallow task, where the source representation contains tokens with additional information attached to them, such as dependency relations. While these tokens could be added directly to the source sequence, this would make the source sequence much longer, something that would impact model performance (Khandelwal et al., 2018).

Restricted decoding beam search Finally, a feature that is not strictly *modelling*, restricted decoding beam search (Hu et al., 2015). Beam search is an algorithm that iteratively generates text by choosing between multiple outputs using conditional probability. At each time step, beam search picks the top N sequences with the highest probability to continue exploring. N is the number of alternatives the beam chooses between, this is called the beam size.

Restricted decoding is a modification to the beam search algorithm that restricts

which words are generated by the model. For example, in the SRST shallow task, the source representation contains a list of tokens. Many of these tokens need to appear in the generated text. Therefore, we use restricted decoding to ensure that the correct tokens are included in the generated text. Restricted decoding has been used to some success on niche NMT tasks, such as when the vocabulary for a subdomain is known or during the post-editing of a sentence (Hokamp and Liu, 2017). We discuss restricted decoding further and also apply it in Chapters 4 and 6.

3.3 Evaluating the Generated Text

The goal of surface realization is to generate text which is both fluent and adequate. To establish whether a model has achieved this goal, we must evaluate the generated text. The gold standard is human evaluation — once again asking crowdworkers or trained linguists to evaluate the model's generated text. However, carrying out human evaluation requires a significant amount of time and effort; it is simply not practical to perform during the experimentation phase of research. Thus, when more efficient methods of evaluation are needed, we commonly use automated evaluation metrics. For an automated evaluation metric to be useful in decision making, its results must be highly correlated with human evaluation metrics. Unfortunately, as we will show in this section, many surface realization tasks do not have automated evaluation metrics that meet the requirement of high correlation with human evaluation metrics.

Our own work in this thesis does not employ any formal, large-scale human evaluation. Hence, we don't go into great detail on human evaluation task types and best practices (van der Lee et al., 2021). The human evaluations we do refer to come from existing shared task evaluations, such as Dušek et al. (2020); Mille et al. (2020).

3.3.1 Human Evaluation

Knowledge-Base-Entries-to-Text To evaluate text generated for the knowledge-base-entries-to-text task, we must ensure that the generated text includes all the information from the provided source representation, that it contains no hallucinated information, and is written fluently. In the WebNLG Challenge human evaluation (Shimorina et al., 2018), crowdworkers evaluated the generated text in the context of the source representation — the set of RDF triples. Crowdworkers rated each text for fluency, grammaticality and semantics. In the E2E NLG Challenge (Dušek et al., 2020), crowdworkers rated groups of generated sentences for naturalness. The naturalness evaluation was performed without the context of the source representation. Separately, crowdworkers also evaluated the overall quality of generated texts. When evaluating overall quality, the source representations of the generated texts were provided as context.

Although arduous, human evaluation can help researchers to gather insights about models, or even the task itself. In the E2E NLG Challenge's human evaluation, Chen et al. (2018)'s submission scored highest in naturalness. However, the same system scored last in human evaluation of quality. This disparity highlights the core issue with seq2seq models that we are trying to solve. Seq2seq models are capable of generating highly fluent text, but on further inspection the text may in fact be inadequate for the task due to missing or incorrect information.

In the WebNLG Challenge human evaluation of the *Seen* subtask, our test set submission (ADAPT) statistically significantly outperformed the human-authored target text on the fluency and semantics ratings, and matched it on grammaticality. It is unusual for text generated by a model to be higher quality than the target text of the original dataset. Aharoni and Goldberg (2018) also noted they achieved unusually good performance on a dataset derived from the WebNLG dataset. Eventually, Shimorina and Gardent (2018) discovered that, due to the method of dataset construction used — discussed in Section 2.3.1 — there was an overlap between

individual triples that appeared in the training and test sets. This led to the release of the WebNLG 2.0 *constrained* dataset, which ensures that individual triples appearing in the training set do not appear in the validation or test sets.

Sentence-Plan-to-Text To evaluate sentence-plan-to-text tasks, we need to assess whether generated sentences sound fluent and have the same semantic meaning as the original sentence from which the sentence representation was parsed. In each of the Multilingual Surface Realization Shared Tasks (Mille et al., 2018a, 2019, 2020), the organizers used crowdworkers to evaluate the readability and meaning similarity of generated sentences. Sentences were evaluated for readability independently of the original sentence or the source representation. However, as in the WebNLG Challenge, the human-authored target sentences were included in the analysis to act as a reference. For example, in the 2018 Shared Task, our system (ADAPT) scored 73.9% in readability, while the human-authored target sentences scored 78.7%. This highlights the fact that human-authored texts aren't always perceived as perfect by evaluators. Sentences were evaluated for meaning similarity with the context of the corresponding human-authored target sentence. The organisers used quality control methods recommended by Bojar et al. (2017) — by checking whether the crowdworkers were *"able to replicate scores for same sentences and scoring damaged sentences lower"* (Mille et al., 2018a). They found that only 31% of crowdworkers passed this quality control, and removed ratings from the remaining 69% from their analysis.

In the 2017 AMR-to-text shared task, human evaluation was carried out by the participants of the shared task (May and Priyadarshi, 2017). Participants were asked to rank generated sentences from a set of randomly chosen systems. Although the human-authored target sentences were provided as context, participants were not asked to rank the human sentences relative to the generated sentences. In a section on qualitative analysis, May and Priyadarshi (2017) noted that generated sentences

from all systems were often disfluent. Similarly, Konstas et al. (2017) performed an error analysis on 50 generated sentences from their AMR-to-text system and found that 23% contained disfluency and 29% had coverage errors, which means that their model had left out key information from the sentence plan. The inability of AMR-to-text systems, even those trained on large datasets, to generate fluent sentences from AMR graphs was a contributing factor in our own decision not to carry out research involving AMR graphs.

3.3.2 Automated Evaluation

In surface realization, automated evaluation metrics can be used to guide experiments. To ensure that the right decisions are made — for instance when choosing between hyperparameter values or modelling mechanisms — automated evaluation metrics must correlate well with human evaluation.

The main type of automated metrics used in surface realization are n-gram metrics, e.g. BLEU score (Papineni et al., 2002). N-gram metrics analyse the overlap between the generated text and human-authored target text. They compare different groups of tokens (n-grams) that appear in both texts. Here we will discuss the relative usefulness of the automated n-gram metrics for the different surface realization tasks, as well as a new type of metric that is emerging.

Knowledge-Base-Entries-to-Text The E2E NLG Challenge used: BLEU (Papineni et al., 2002), NIST (Doddington, 2002), METEOR (Lavie and Agarwal, 2007), ROUGE (Lin, 2004), and CIDEr (Vedantam et al., 2015). The E2E NLG Challenge provides an official scoring script: `https://github.com/tuetschek/e2e-metrics`. The WebNLG challenge used the automated metrics: BLEU, TER (Snover et al., 2006) and METEOR. These n-gram metrics were designed to evaluate an entire system, namely all sentences that are generated from a test set. However, it has also become common to investigate n-gram metrics on a sentence-by-sentence basis

(Novikova et al., 2017). On both the WebNLG and E2E tasks automated metrics had a loose correlation with the human evaluation scores of seq2seq model outputs at a system level but were uncorrelated at a sentence level. This disparity between system level and sentence level correlations has also been highlighted in NMT (Specia et al., 2010).

Sentence-Plan-to-Text In the AMR-to-text shared task, BLEU score was used as the automated evaluation metric. The correlation between the rank of human judgements and BLEU score, on a system level, was similar to that of WebNLG, 0.6988 Pearson correlation. The SRST shallow task was evaluated using BLEU, NIST and normalised edit distance (DIST). DIST is not an n-gram metric, it is the *"character-based string-edit distance"*[3] between the generated text and the reference text. The automated n-gram metrics were found to correlate better with human evaluation than any other task we have mentioned so far. System-level BLEU score correlated most consistently with human evaluations, *"achieving a correlation above 0.95 in all settings"* (Mille et al., 2018a)

Model-Based Automated Metrics An alternative approach to automated metrics is to use datasets based on human evaluation to train a language model that can predict the human evaluation score. Metrics such as ADEM and BLEURT are model-based automated metrics trained to predict human evaluation scores (Lowe et al., 2017; Sellam et al., 2020) However, thus far, none of these metrics have proved more useful than n-gram metrics such as BLEU. For instance, BLEURT was used as an automated evaluation metric in the 2020 SRST shallow task, and was found to produce results that correlated highly with BLEU score, failing to provide any new information (Mille et al., 2020).

[3] http://taln.upf.edu/pages/msr2018-ws/SRST.html

3.4 Summary

In this chapter, we discussed the basic seq2seq models used by neural surface realization systems. We highlighted the modelling features that we believe are key to improving adequacy on the surface realization task. Finally we discussed evaluation of the generated text from these models and the importance of automated metrics that correlate with human evaluation.

Chapter 4

Silver Data and Surface Realization

In Chapter 2, we introduced the surface realization shared task (SRST) shallow task. It is a sentence-plan-to-text task, where the sentence plan is a UD tree with the order and forms removed; what is left is a tree of lemmas with dependency relations, part-of-speech tags, and morphological features. The goal is to reconstruct the sentence in its original order and with the correct forms. Improved performance on the SRST shallow task would facilitate investigation of more complex versions of the task, such as the underspecified Universal Dependency (UUD) representation in which function words are pruned from the tree (Mille et al., 2018b), which may be of more practical use in a pipeline surface realization system (Moryossef et al., 2019; Elder et al., 2019; Castro Ferreira et al., 2019).

In the 2018 SRST (Mille et al., 2018a), our submission came joint first place in the English language track of the shallow task (Elder and Hokamp, 2018). We trained a model on both the gold parse EWT dataset and our own automatically generated silver parse dataset.[1] Training with an automatically generated dataset greatly improved our model's performance. However, the other joint first place

[1]Gold parse datasets are created by human annotators, while silver parse datasets are generated by parsing sentences with a trained model.

submission did not augment their training data. Thus, the organisers decided for the 2019 shared task to disallow the use of silver parse training data in order to *"improve the comparability of the results"*.[2]

This chapter presents two main contributions: first, we describe and analyse how we improved our 2018 system such that it performs on par with the best results on the 2018 SRST English dataset *without* using the silver parse data (Yu et al., 2019b), and second, we demonstrate further improvement from using silver parse data, which increases results from 72.3 to 80.1 BLEU score. We analyse the ways in which the silver parse data enhances performance, finding that longer sentences are particularly improved and more exact reference matches are generated overall.

4.1 Method

We start by describing the baseline system — our submission to the 2018 SRST shallow task (Elder and Hokamp, 2018) — and then describe the improvements made to the system.

4.1.1 Baseline System

Our system has three main parts to it:

1. a seq2seq model
2. data preprocessing to convert the CoNLL formatted parse trees to a source sequence
3. a decoding algorithm for generating text from the model

Seq2seq Model The baseline system uses a bidirectional LSTM encoder-decoder model with copy attention. The model performs both linearization and inflection in a single decoding step.

[2] http://taln.upf.edu/pages/msr2019-ws/SRST.html

Preprocessing involves three main aspects; *depth-first linearization* creates a linear sequence from the parse tree, *source features* add token-level information for each lemma such as dependency relations, part-of-speech tags and morphological features, and *form suggestions* are added to the end of the source sequence, these suggestions help the model to choose the correct form for each token during generation.

Depth-First Linearization We introduced depth-first linearization in Section 2.2.3, where we highlighted its importance in creating a source sequence from graph and tree structures. Depth-first linearization begins at the root node and adds each subsequent child node to the sequence, before returning to the highest node not yet added.[3] Where a node has multiple child nodes, we choose randomly between the children.

In an ablation analysis in Elder and Hokamp (2018), we found that depth-first linearization boosted our bleu score from 23.75 to 43.11, an increase of almost 20 bleu score.

Source Features We append a number of features to each token;[4] treebank-specific part-of-speech tag (XPOS), ID, head ID (HEAD), dependency relations (DepRel), relative linear order with respect to the governor (Lin). This enables us to use factored sequence models (Sennrich and Haddow, 2016), which we introduced in Section 3.2. The embedding sizes of the features are set heuristically by OpenNMT-py, using the heuristic $|embedding_k| = |V_k|^{0.7}$, where $|V_k|$ is the vocabulary size of feature k. To use this modelling feature a special pipe symbol, |, is required between each of the token's features.[5]

[3] https://github.com/Henry-E/surface-realization-shallow-task/blob/master/modules/create_source_and_target.py#L12-L36

[4] We did not experiment with subwords as it would have complicated the modelling of token-level features.

[5] https://github.com/Henry-E/surface-realization-shallow-task/blob/master/modules/create_source_and_target.py#L79-L95

In Elder and Hokamp (2018) we did not perform an ablation analysis of the various source features used, however in early experiments we found that using no source features resulted in poor model performance.

Form Suggestions Finally, we address the problem of generating a token's form when only given its lemma. To do this, we provide the model with form suggestions.[6] To obtain the form suggestions, we use the automatically generated corpus, which will be discussed in Section 4.2.2, to create a dictionary.[7] Form suggestions are a list of possible forms that a lemma, with the same XPOS tag, was observed to take in the corpus. The key-value pair dictionary is structured as such: the key is a concatenated lemma and XPOS tag, the value is a list of possible forms observed in the automatically generated corpus.[8] For example: { *"VBN_bootstrap": "bootstrapped"*}. A list of form suggestions is appended to the source sequence containing the possible forms for each lemma + XPOS tag found in the form suggestions dictionary. For some (**lemma**, **xpos**) pairs there are multiple potential forms, when this occurs we add all potential forms to the input sequence. The mapping was found to cover 98.9% of cases in the validation split of the dataset.

In an ablation analysis in Elder and Hokamp (2018), we found that form suggestions boosted our bleu score from 21.27 to 23.75, an increase of roughly 2.5 bleu score. Although this increase in bleu score is small relative to the other improvements, form suggestions are a key part of our system as they provide the copy attention model with exact tokens of all the possible forms a lemma can take.

Beam search Decoding is done using beam search. The generated sequence length is artificially constrained to contain the same number of tokens as the linearized tree

[6] https://github.com/Henry-E/surface-realization-shallow-task/blob/master/modules/create_source_and_target.py#L101-L123

[7] https://github.com/Henry-E/surface-realization-shallow-task/blob/master/modules/get_form_suggestions.py

[8] Dictionary: https://github.com/Henry-E/surface-realization-shallow-task/blob/master/inflection_dicts/18th_october_tests/lemma_form_dict_sorted.json

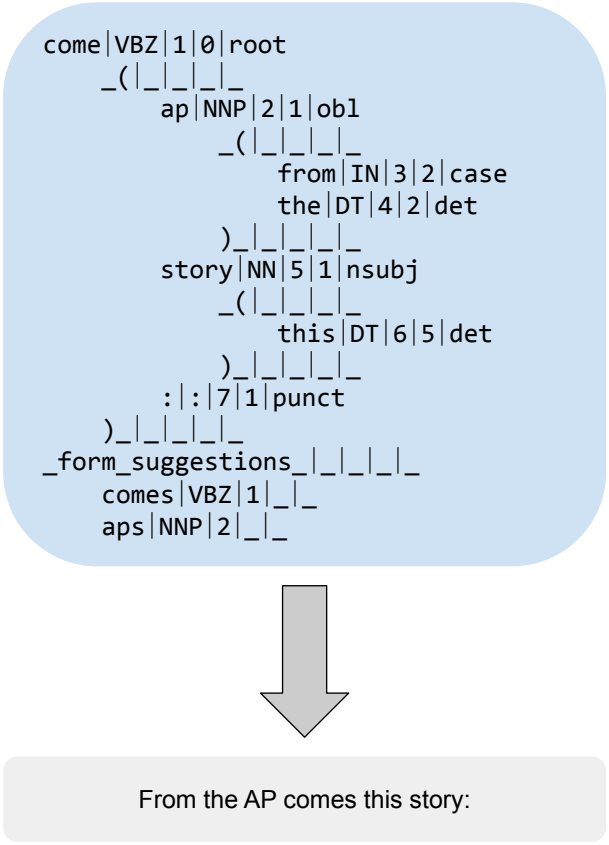


Figure 4.1: Source sequence formatted with newlines and indentation. '|' indicates where a source feature is appended. Source features appear in the order: Lemma | XPOS | ID | Head | DepRel.

in the source sequence.

4.1.2 Improvements to Baseline System

Three improvements were made to the baseline system; two focused on the preprocessing step, *random linearizations* and *scoping brackets*, and one that altered the decoding algorithm, *restricted beam search*. An example source sequence, including the scoping brackets, is shown in Figure 4.1.

Random linearizations In the baseline system, a single random depth-first linearization of the training data is obtained and used repeatedly when training the model. In the improved system, we instead obtain multiple linearizations. Multiple linearizations means that each epoch of training data contains a different lineariza-

tion of the same parse tree.[9] This may help the model to become robust to new linearizations it sees during testing.

Scoping brackets Similar to the approach of Konstas et al. (2017), we apply scoping brackets around child nodes. This provides further indication of the tree structure to the model.[10]

Restricted beam search In an attempt to reduce unnecessary errors during decoding, our beam search algorithm inspects the input sequence tokens[11] and restricts decoder generation to only those tokens from the input. Furthermore, the decoder can only use each token from the input once.[12] This is similar to the approach used by King and White (2018).

4.2 Experimental Setup

4.2.1 Data

We evaluate on the SRST 2018 dataset (Mille et al., 2018a) for English,[13] which was derived from the Universal Dependency English Web Treebank (EWT)[14] (Silveira et al., 2014).

4.2.2 Generating Silver Standard Data

In order to augment the existing training data we create silver standard data by parsing sentences from a corpus of human-authored text. The two corpora we use

[9] However, we do not enforce any uniqueness conditions when linearizing subsequent trees.

[10] https://github.com/Henry-E/surface-realization-shallow-task/blob/master/modules/create_source_and_target.py#L23-L28

[11] https://github.com/Henry-E/OpenNMT-py/blob/master/onmt/translate/translator.py#L467-L496

[12] https://github.com/Henry-E/OpenNMT-py/blob/master/onmt/translate/beam_search.py#L204-L207

[13] http://taln.upf.edu/pages/msr2018-ws/SRST.html

[14] https://github.com/UniversalDependencies/UD_English-EWT

are: Wikitext 103 (Merity et al., 2017) and the CNN stories portion of the DeepMind Q&A dataset (Hermann et al., 2015). We chose these two corpora because they are known to be both large in quantity and contain high quality text.

Each corpus requires cleaning and formatting, after which the corpus can be sentence tokenized using CoreNLP (Manning et al., 2014). Sentences are filtered by length – minimum 5 tokens and maximum 50 tokens – and for vocabulary overlap with the original training data – set to 80% of tokens in a sentence required to appear in the original vocabulary. We use at minimum 5 tokens and maximum 50 tokens to reflect the distribution of sentence lengths in the EWT dataset. We use 80% vocabulary overlap, instead of 100%, to increase the vocabulary size of the silver dataset without compromising on its quality and similarity to our existing training data too much.[15] These sentences are then parsed using the Stanford NLP UD parser (Qi et al., 2018). This leaves us with 2.4 million parsed sentences from the CNN stories corpus and 2.1 million from Wikitext.

To convert a parse tree into the source representation: first, word order information is removed by shuffling the IDs of all nodes in the parse tree, then, the tokens are lemmatised by removing surface form information. This is the same process used by the shared task organizers to create datasets from the UD treebanks.

4.2.3 Training

The system is trained using our fork[16] of the OpenNMT-py framework (Klein et al., 2017), the principal change made was to the beam search decoding code. Hyperparameter details and replication instructions are provided in our project's repository,[17] in particular in the directory *surface-realization-shallow-task/configs*.

Vocabulary size varies based on the datasets in use. It is determined by using

[15]Note, we only ever filter out sentences with less than 80% vocabulary overlap from the silver training data. None of the sentences from the EWT dataset are removed, regardless of whether they happen to have less than 80% vocabulary overlap or not.

[16]https://github.com/Henry-E/OpenNMT-py

[17]https://github.com/Henry-E/surface-realization-shallow-task

any tokens which appear 10 times or more. When using the original shared task dataset, the vocabulary size is 2,193 tokens, training is done for 33 epochs and takes 40 minutes on two Nvidia 1080 Ti GPUs. All hyperparameters stay the same when training with the synthetic data, except for vocabulary size and training time. For the combined Wikitext, CNN and shared task dataset the vocabulary size is 89,233, training time increases to around 2 days, and uses 60 random linearizations of the shared task dataset and 8 of the Wikitext and CNN datasets.[18]

4.2.4 Evaluation

The evaluation is performed on detokenized sentences[19] using the official evaluation script from the 2018 shared task. We focus our analysis on BLEU-4 score (Papineni et al., 2002), which, as mentioned in Section 3.3, was found to be highly correlated with human evaluation scores.

4.3 Results

In Table 4.1, we compare our results on the test set with those reported in Yu et al. (2019b), which include the Yu et al. system (Yu19) and the best 2018 shared task result for English (Elder and Hokamp, 2018) (ST18). Ignoring for now the result with augmented data, we can see that our system is competitive with that of Yu et al (72.3 vs 72.7).

In Section 4.1.2, we described three improvements to our baseline system: random linearization, scoping brackets and restricted beam search. An ablation analysis of these improvements on the validation set is shown in Table 4.2. The largest increase in BLEU score comes from the introduction of random linearizations. How-

[18]These hyperparameters were chosen after experimenting on the development set to see what led to the highest and most reliable performance.

[19]Passing detokenized inputs to BLEU in this task is typically a bad idea because it makes the score very sensitive to the type of detokenization used. This issue was fixed in the 2019 shared task; the evaluation script was altered to use tokenized inputs instead.

	BLEU-4
ST18	69.1
Yu19	**72.7**
Ours	72.3
Ours + Silver Data	**80.1**

Table 4.1: Test set results for the baselines, trained on just the EWT dataset, and the model trained on silver data and the EWT dataset

System	BLEU-4
SR Baseline	57.3
SR + Random Lins	65.1
SR + Random Lins + Scope	69.2
SR + Random Lins + Scope + Restricted Beam	72.2

Table 4.2: Validation set results for an ablation analysis of the baseline system plus improvements; trained only on the EWT dataset

ever, all three improvements make a meaningful, positive contribution.

In Table 4.3 we show a pairwise statistical significance test of the system ablation analysis using compare-mt's BLEU score bootstrap resampling functionality with a sample size of 1,000 pairs (Neubig et al.). The BLEU scores reported by compare-mt differ from those of the official shared task evaluation, likely due to differences in tokenization and smoothing functions used in each evaluation. However, we still see that the relative performance of each system is the same and that the difference between each pair is highly statistically significant; each pair has $p < 0.0001$.

4.3.1 Effect of the Silver Data

The last row of Table 4.1 shows the effect of training with the silver parse data. BLEU score on the test set jumps from 72.3 to 80.1. To help understand why additional data makes such a substantial difference, we perform various analyses on the validation set, including examining the effect of the choice of unlabeled corpus and highlighting interesting differences between the systems trained with and without the silver parse data.

	SR Baseline	SR + Random Lins	Win?
BLEU	53.7924 [51.8591,55.8416]	60.8591 [58.8182,62.8910]	s2>s1 p=0.0000
	SR + Random Lins	SR + Random Lins + Scope	Win?
BLEU	60.8591 [58.9269,62.8725]	64.8524 [62.9106,66.7132]	s2>s1 p=0.0000
	SR + Random Lins + Scope	SR + Random Lins + Scope + Restricted Beam	Win?
BLEU	64.8524 [62.8446,66.8145]	67.5997 [65.6663,69.4095]	s2>s1 p=0.0000

Table 4.3: Statistical significance tests of the system ablation analysis using compare-mt's BLEU score bootstrap resampling functionality with a sample size of 1,000 pairs.

Data used	BLEU-4
Improved SR Baseline (SRST)	72.2
SR + Wikitext	79.8
SR + CNN	80.3
SR + CNN + Wikitext	80.8

Table 4.4: Validation set results for our model trained with the EWT dataset, and then with silver data from the additional corpora

The Role of Corpus Table 4.4 compares the Wikitext corpus as a source of additional training data to the CNN corpus. Both the individual results and the result obtained by combining the two corpora show that there is little difference between the two.

In Table 4.5 we show a pairwise statistical significance test of the dataset ablation analysis using compare-mt's BLEU score bootstrap resampling functionality with a sample size of 1,000 pairs (Neubig et al.). As previously noted, the BLEU scores reported by compare-mt differ from those of the official shared task evaluation. The additional data provides a statistically significant boost to BLEU score but the improvements from the individual datasets are more mixed. The system trained on CNN only has a 71.7% chance of beating the system trained on Wikitext, which implies they are not statistically significantly different. The only statistically signif-

	Improved SR Baseline (SRST)	SR + Wikitext	Win?
BLEU	67.5997 [65.6275,69.4708]	74.8199 [73.0819,76.4832]	s2>s1 p=0.0000
	SR + Wikitext	SR + CNN	Win?
BLEU	74.8199 [73.1214,76.4123]	75.1171 [73.2820,76.8589]	- p=0.2830
	SR + CNN	SR + CNN + Wikitext	Win?
BLEU	75.1171 [73.4311,76.8383]	75.6438 [73.8791,77.4143]	- p=0.1080
	SR + Wikitext	SR + CNN + Wikitext	Win?
BLEU	74.8199 [73.0333,76.5535]	75.6438 [73.8747,77.4310]	s2>s1 p=0.0370

Table 4.5: Statistical significance tests of the dataset ablation analysis using compare-mt's BLEU score bootstrap resampling functionality with a sample size of 1,000 pairs.

icant result is that the two datasets sets combined beat the system trained on just Wikitext with a 96.3% likelihood.

Sentence Length and BLEU Score Using compare-mt (Neubig et al., 2019), we noticed a striking difference between the systems with regards to performance on sentences of different length.[20] This is shown in Figure 4.2.

Even though the silver parse sentences were limited to 50 tokens in length, the model trained with silver data performed equally well for sentence length buckets 50-60 and 60+, while the system trained on the EWT dataset performed relatively worse. It is possible this is due to the silver data model having a larger vocabulary and being exposed to a wider range of commonly occurring phrases, which make up parts of longer sentences.

Error Analysis Table 4.6 lists the number of exact matches, in which the tokenized reference sentence and the generated sentence exactly match. We also detect relatively minor errors, namely punctuation and inflection, in which these are the

[20]These are results for the tokenized versions of the generated and reference sentences, hence the higher numbers.

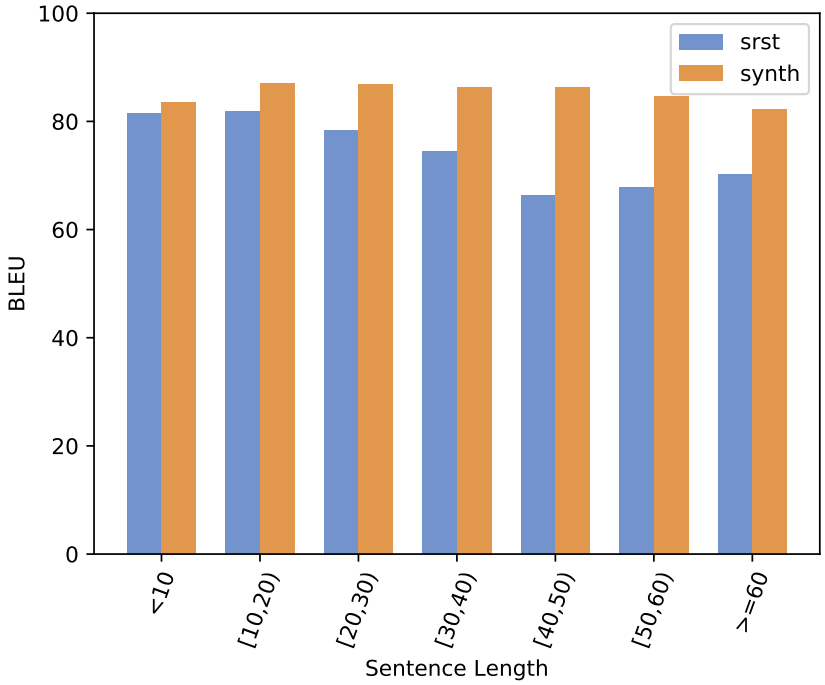


Figure 4.2: BLEU score breakdown by sentence length buckets, comparing our model trained on just the EWT dataset *(SRST, blue)* with a model also trained with the silver data *(SYNTH, orange)*

	SRST	Synth
Exact match	1159	1314
+ Punctuation error only	43	46
+ Inflection error only	123	142
Total (relatively error free)	1325	1502
Remaining Sentences	653	476

Table 4.6: Error analysis breakdown for the 1,978 sentences in the EWT validation set.

only differences between the reference and generated sentences. Punctuation errors are typically minor and there is usually ambiguity about the placement of punctuation.[21] Inflection errors occur when a different inflected form has been chosen by the model than in the reference sentence. These tend to be small differences and are often valid alternatives, e.g. choosing I *'m* over I *am*.

Within the remaining uncategorized sentences are mostly linearization errors. Linearization errors come in two main categories; semantically similar, in which the linearization is different from the reference sentence but is still valid and communicates the same meaning as the reference – see Example 1 below; and semantically dissimilar, where the linearization has clear differences and doesn't contain the same meaning as the reference sentence – see Example 2 below.

1. Semantically similar
    (a) Ref: From the AP comes this story:
    (b) Gen: This story comes from the AP:
2. Semantically dissimilar
    (a) Ref: I ran across this item on the Internet.
    (b) Gen: I ran on the internet across this item.

We advocate for the use of this kind of breakdown in an error analysis in order to help understand the quality of these systems in more absolute terms, since it's the overall number of accurate sentences which matters. This could be more intuitive than comparing BLEU scores relative to prior models when deciding whether to apply a system in a business setting.

Beyond the high level categories of semantically similar or dissimilar, we didn't find any clear trends emerging of common errors made by the system.

[21] In the 2019 shared task an additional feature was provided to indicate the position of punctuation relative to its head token.

4.4 Discussion

Although it is common knowledge that machine learning systems typically benefit from more data, the 7.4 point jump in BLEU seen in Table 4.1 is important and worth emphasizing, considering that the 2019 shared task introduced a new rule which prohibited the use of silver data. The EWT training set contains 12,375 sentences, compared with the 5.5 million silver parse sentences in our augmented dataset. A possible issue with this rule is that systems designed with smaller datasets in mind might not scale to the use of large augmented datasets. Thus, an inadvertent consequence could be that results from the 2019 SRST may be misleading for future research directions.

For instance, the system which was the clear winner of the 2019 SRST, Yu et al. (2019a), used tree-structured long short-term memory (LSTM) networks (Tai et al., 2015) in four of the five steps of their pipeline approach (linearization, completion, inflection, contraction, and detokenization[22]) to generate the final text. In general, tree LSTMs can be slow and difficult to train.[23] Song et al. (2018) utilized a variant of the tree LSTM in a similar NLG task, converting AMR graphs to text. Following the state-of-the-art system (Konstas et al., 2017), which used standard LSTMs, Song et al. augmented their training with silver parse AMR data. Though their system outperformed Konstas et al. at equivalent levels of additional training sentences, it was unable to scale up to the 20 million sentences used by the best Konstas et al. system and ultimately did not outperform them.[24]

Critics of neural NLG approaches[25] emphasise that quality and reliability are at the core of production-ready NLG systems. We argue that if using silver data contributes to producing higher quality and more robust outputs, then we ought to ensure we are designing systems that can take advantage of this automatically

[22]Detokenization was a rule-based system

[23]https://github.com/dasguptar/treelstm.pytorch/issues/6

[24]Song et al.'s best system achieved 33.0 BLEU score with 2 million additional sentences, while Konstas et al. scored 32.3 with 2 million and 33.8 with 20 million (the best overall system).

[25]See, for example, https://ehudreiter.com/2016/12/12/nlg-and-ml/

	BLEU	NIST	DIST
EWT	80.4	13.47	85.5
+ Silver Data	87.5	13.81	90.35

Table 4.7: Test set results on the 2020 SRST EWT dataset - Automated Evaluation metrics

generated data.

4.5 2020 Surface Realization Shared Task Results

System	Ave.	Ave. z	n	N
EWT	72.5	0.32	830	953
+ Silver Data	75.7	0.426	797	913
HUMAN	**75.7**	**0.417**	**669**	**1,402**

Table 4.8: Test set results on the 2020 SRST EWT dataset - Human Evaluation: **Readability**. *Ave. = average score for system; Ave. z = corresponding average standardized score; n = distinct test sentences assessed; N = total number of judgments; HUMAN = original reference texts.*

System	Ave.	Ave. z	n	N
EWT	90.7	0.476	1,685	1,914
+ Silver Data	92.6	0.54	1,698	1,931

Table 4.9: Test set results on the 2020 SRST EWT dataset - Human Evaluation: **Meaning Similarity**. *Ave. = average score received by systems; Ave. z = corresponding average standardized score; n = total number of distinct test sentences assessed; N = total number of human judgments.*

Following the release of the work in this chapter as an ACL paper (Elder et al., 2020a), the 2020 SRST included a track which permitted the use of silver data. In this section, we report our results on the 2020 SRST. A detailed explanation of the evaluation methodology, as well as a comparison with other participants, can be found in the shared task description paper (Mille et al., 2020). Our silver data system ranked first, or joint first, for both automated and human evaluation metrics on the EWT test set.

Table 4.7 contains automated evaluation metrics on the EWT test set. As with the results in this chapter, we find that the augmented dataset greatly improves the

performance of our system. The BLEU scores reported on this test set are higher than in the previous results section due to the difference in tokenization used by the 2020 SRST evaluation script.

Both human evaluation metrics, *readability* and *meaning similarity*, were collected following the same method. Raters were given continuous sliders with the values 0 to 100 (best). When rating for readability, only a single sentence (from one system) was provided at time. When rating for meaning similarity, again one sentence was provided but this time in the context of the original target sentence. To produce standardised scores Mille et al. (2020) report that they: *"map each individual evaluator's scores to their standard scores (or z-scores) computed on the set of all raw scores by the given evaluator using each evaluator's mean and standard deviation. For both raw and standard scores, we compute the mean of sentence-level scores."*

Table 4.8 contains human evaluation results for the readability metric. Rather surprisingly, the readability for our system with the augmented corpora is equivalent to the readability of the original human text, both have an average readability score of 75.7%. Recall, however, that the readability metric only reflects how well written the annotators deemed a sentence to be, it doesn't take into account whether the generated sentence has managed to capture the meaning of the original sentence.

Table 4.9 contains human evaluation results for the meaning similarity metric. This metric describes how successful the system has been at generating sentences with the same meaning as the original sentence. Sentences generated by a model trained on the augmented corpora are on average 92.6% similar in meaning to the original sentence. While this may seem like a strong result,[26] ultimately we are aiming for 100% meaning similarity in order to develop surface realization systems that are reliable enough to be used with real world task-oriented dialogue systems.

[26]The highest recorded meaning similarity on the same test set in the 2019 SRST was 86.6% (Mille et al., 2019)

4.6 Conclusion

In this chapter, we argued for the use of silver data in the English language track of the SRST, justified by the fact that its use gives a significant performance boost on the shallow task, from 72.7 BLEU up to 80.1. While this is not yet at the level of reliability needed for task-oriented dialogue systems to be used commercially, it is a step in the right direction. Assuming the use of silver data, more needs to be investigated in order to fully maximize its benefit on performance. Future work could look more closely at the choice of corpus, construction details of the augmented dataset, as well as the trade-off between training time and accuracy that comes with larger vocabularies. The work described in this study has focused on English. Another avenue of research would be to investigate the effect of silver data on surface realization in other languages.

Chapter 5

Designing a Sentence Plan

In Chapter 4, we described our system for generating text from SRST shallow task sentence plans and demonstrated the effect silver data had on the system's performance. In this chapter, we address the broader issue of using sentence plans as part of a pipeline surface realization system. Our proposed pipeline surface realization system splits apart the typically end-to-end neural system into separate knowledge-base-entries-to-utterance-plan and sentence-plan-to-text models, connected in the middle by an utterance plan — made up of one or more sentence plans. This is similar to other work in the area of multi-stage neural NLG (Dušek and Jurcicek, 2016a; Daniele et al., 2017; Puduppully et al., 2019; Hajdik et al., 2019; Moryossef et al., 2019; Castro Ferreira et al., 2019), though it is largely inspired by more traditional pipeline data-to-text generation (Reiter and Dale, 2000; Gatt and Krahmer, 2018). In particular, we focus on the sentence-plan-to-text task and introduce a new sentence plan based on the underspecified universal dependency (UUD) representation (Mille et al., 2018b). In designing the sentence plan, we are driven by the following constraints:

1. The sentence plan must be suitable for processing with a neural system.
2. It must not make the surface realization task too difficult because we are interested in understanding the limitations of neural generation even under

favorable conditions.

3. It must be possible to parse a sentence into this representation so that a training set could be easily augmented with in-domain silver data.

One of the main contributions of this chapter is to evaluate the suitability of the proposed sentence plan. We use the E2E NLG Challenge dataset, introduced in Chapter 2, as the basis for our experiments. To test the sentence plan, we take target utterances from the E2E dataset, divide the utterances into sentences and then parse them into the sentence plan. We then train a model to generate sentences from the sentence plan. Finally, we compare the generated text with the reference sentence, using both automatic and human evaluation. We find that the quality of the generated text is high, achieving a BLEU score of 82.47. Additionally, we experiment with the use of silver data from the TripAdvisor corpus (Wang et al., 2011) to train the model and find this increases the BLEU score to 83.38. A manual error analysis shows that in only a very small proportion (∼5%) of the output sentences, the meaning of the reference is not fully recovered. This high level of adequacy is expected since the sentence plans are generated directly from the reference sentences. An analysis of a sample of the adequate sentences shows that readability is on a par with the reference sentences.

Having established that surface realization from our new sentence plan achieves sufficiently high performance, we then test its efficacy as part of a pipeline system. On the E2E task, our system scores higher on automated results than the winner of the E2E Challenge (Juraska et al., 2018). The use of the silver data in the sentence plan model results in further gains. These encouraging results suggest that pipelines could work well in a surface realization system.

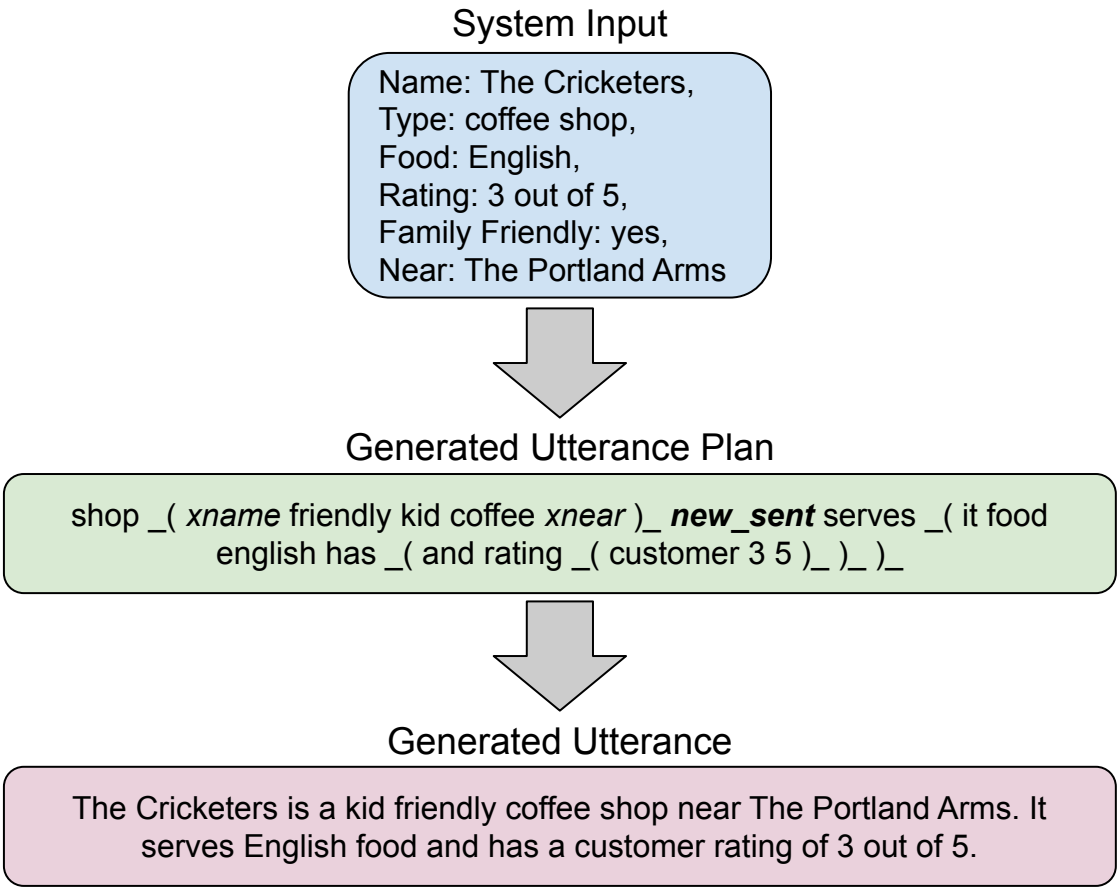


Figure 5.1: Pipeline surface realization system. Both *Generated Utterance Plan* and *Generated Utterance* are real examples, generated by their respective models.

5.1 Method

Our system consists of two distinct seq2seq models. The first is an utterance planning model which takes as input a set of knowledge base entries and generates an utterance plan containing one or more sentence plans. Each sentence plan in the utterance plan is then passed to a second model which generates the output text, one sentence at a time. We use the utterance plan, rather than numeric embeddings used by end-to-end systems, to pass information between the two models. See Figure 5.1 for an example from the E2E dataset.

5.1.1 Sentence Plan

Our sentence plan is based on the underspecified universal dependency (UUD) representation (Mille et al., 2018b). As described in Section 2.1, the UUD representation is a tree *"containing only content words linked by predicate-argument edges in the PropBank/NomBank (Palmer et al., 2005; Meyers et al., 2004) fashion"* (Mille et al., 2018b). Each UUD representation corresponds to a single sentence. The UUD representation was designed to *"approximate the kind of abstract meaning representations used in native NLG tasks"* (Mille et al., 2018b). By this Mille et al. mean that the UUD representation was designed to be similar to the kind of output a rule-based system might generate as part of a pipeline NLG process. However, to the best of our knowledge, no such system has yet been developed or adapted to generate the UUD representation as output. Hence, when designing our sentence plan we made three main changes to the UUD representation;

1. When linearizing the tree, we use a consistent order for child nodes
2. We remove token level features
3. We use forms instead of lemmas as tokens in the source sequence

Linearization Tree representations must be linearized for use with a seq2seq model. As with the SRST shallow task sentence plan, we linearize the tree using depth-first search. Scoping brackets are added before each child node. However, for this sentence plan, when a node has only one child node, we omit scoping brackets. While this could lead to minor ambiguity, it also reduces the length of the utterance plan; this is beneficial as longer sequences can be more difficult for seq2seq models to generate (Khandelwal et al., 2018).

As highlighted in Section 2.2.2, the order of tokens in the source sequence plays a role in the implicit alignments learned by a seq2seq model. When a parent node has two or more child nodes, we have to decide in what order to linearize those child

nodes. In Chapter 4, we chose to have a permanently random order. This choice reflected the absence of a strong basis on which to order children,[1] so instead we used a permanently random ordering to ensure the model would be robust to new linearizations seen at test time. In this task, however, we choose to use a consistent order: the original sentence order of the child tokens. We referred to this in Section 2.2.2 as the target-sequence order. This simplifies the sentence-plan-to-text task by providing the model additional information that it can use when generating the target sequence.

Features Tokens in the UUD representation have a number of additional features: head ID, dependency relations (DepRel), universal part-of-speech tag (UPOS) and lexical features (feats). In Chapter 4, we passed this information to the model using source features. However, this time, we decided to create a sentence plan which does not include these additional features. We chose not to include the additional features in order to simplify the task of generating the utterance plan using a seq2seq model. While token features could be generated using multitask learning (Dalvi et al., 2017), target-side features are a complex undertaking[2] and we leave this for future work.

Lemmas vs. Forms In the UUD representation, source tokens are lemmas. Lemmas are the root of the token. Part-of-speech tags and lexical features can be used by a surface realization system to generate the original form of a lemma. However, as we do not include these features in our sentence plan, we use the original form of the token instead in the sentence plan. This is another simplification of the task.

[1] Others have tried using rule-based heuristics (Fu and White, 2018) or model-based approaches to ordering child nodes (Ferreira et al., 2018).

[2] https://github.com/marian-nmt/marian/issues/207#issuecomment-416837093

5.2 Experimental Setup

Datasets Experiments were performed with the E2E dataset. Training data for the sentence-plan-to-text model was augmented, for some experiments, with the TripAdvisor corpus (Wang et al., 2010), which was filtered for sentences with a 100% vocabulary overlap[3] with the E2E corpus and a sentence length between 5 and 30 tokens,[4] resulting in an additional 209,823 sentences, with an average sentence length of 10 tokens. By comparison the E2E corpus has sentence lengths ranging between 1 and 59 tokens with an average sentence length of 13 tokens.

As in Chapter 4, both corpora were sentence tokenized by CoreNLP (Manning et al., 2014) and then parsed by the Stanford NLP UD parser (Qi et al., 2018). The parsed sentences in CoNLL-U format were then converted to the UUD representation using a specialised tool[5] (Mille et al., 2018b). Utterances from the E2E corpus were delexicalised to anonymize restaurant names in both the *name* and *near* slots of the meaning representation. All tokens were lower cased before training.

Models As with the experiments in Chapter 4, to create the pipeline surface realization system, we train two separate seq2seq models using the neural machine translation framework, OpenNMT (Klein et al., 2017). Both the knowledge-base-entries-to-utterance-plan model and the sentence-plan-to-text model used the same hyperparameters. A single layer LSTM (Hochreiter and Schmidhuber, 1997) with RNN size 450 and word vector size 300 was used. The models were trained using Adam (Kingma and Ba, 2015) with a learning rate of 0.001. The only difference between the two models was that the sentence-plan-to-text model was trained with copy attention (Vinyals et al., 2015). We did not use copy attention with

[3]We choose 100%, as compared with 80% in Chapter 4, as the quality of the Trip Advisor corpus is lower than the other corpora and there was likely to be little benefit on test set performance to increasing vocabulary size.

[4]We chose the shorter length of 30 tokens, as compared with the 50 tokens in Chapter 4, to reflect the distribution of sentence lengths which was shorter in E2E than EWT.

[5]https://gitlab.com/talnupf/ud2deep

the knowledge-base-entries-to-utterance-plan model because source attributes in the E2E dataset are abstract concepts and not intended to be copied word-for-word into the target text.

To evaluate the quality of generation from the sentence plan, two different sentence-plan-to-text models were compared: one trained solely on sentences from the E2E corpus and another trained on a combined corpus of E2E and TripAdvisor sentences. For the full E2E task, a single knowledge-base-entries-to-utterance-plan model was trained. As baselines for the full E2E task, we compare our system against two seq2seq models which both use semantic rerankers on their generated utterances: TGen (Dušek and Jurcicek, 2016a), the baseline system for the E2E challenge and Slug2Slug (Juraska et al., 2018), the winning system of the E2E challenge.

Automated Evaluation For completeness, we report results from the E2E NLG Challenge's official scoring script, which consists of the following n-gram overlap metrics; BLEU (Papineni et al., 2002), NIST (Doddington, 2002), METEOR (Lavie and Agarwal, 2007), ROUGE (Lin, 2004), and CIDEr (Vedantam et al., 2015).

To evaluate the quality of their generated outputs, both sentence-plan-to-text models were evaluated using silver parse sentence plans from the E2E validation set. Sentences from the validation set were parsed to create silver parse sentence plans that could be used for automated evaluation. For the generated texts from the silver parse sentence plans, we report BLEU-4 scores.[6]

As previously noted in Section 3.3, both the E2E (Dušek et al., 2020) and WebNLG challenge (Shimorina, 2018) found that automated evaluation metrics did not correlate with the human evaluation. However, in the SRST shallow task, the correlation between BLEU score and human evaluation was found to be highly significant (Mille et al., 2018a, 2019, 2020). As we are using a similar sentence plan

[6]We input tokenized, lowercased and relexicalised sentences to the Moses multi-bleu perl script: https://github.com/OpenNMT/OpenNMT-py/blob/master/tools/multi-bleu.perl

to the SRST shallow task, we use BLEU score to get a reasonable estimation of the quality of the generated text from the sentence-plan-to-text model.

Manual Analysis Even given a high correlation between BLEU score and human evaluation on the sentence-plan-to-text task, we must still recognize the importance of using human evaluation. To get a more accurate understanding of the quality of the generated text, we perform human evaluation on the outputs of the sentence-plan-to-text model — where the source sequence is the silver parse of sentences from the validation set. We take a sequential approach to human evaluation; we first evaluate the outputs for meaning similarity and then for readability.

We define meaning similarity as whether or not the two sentences contain the same expressed attribute-value pairs. We treat this as a binary Yes / No decision. As the generated sentences are using a sentence plan as their source, the model ought to be able to reconstruct a sentence that, while possibly differently structured, expresses the same meaning. Before performing human evaluation of meaning similarity, we automatically filter out generated sentences with: no differences, only differences involving the presence or absence of hyphens, and only capitalization differences. We assume that in these cases the generated sentences have achieved almost perfect meaning similarity.

We manually analyze failure cases where semantic similarity is not achieved to discover where the issues arise. For example, when parsing and converting the original sentence into the sentence plan, some information may be misrepresented or lost. Alternatively, there may be an issue with the quality of the generation from the sentence-plan-to-text model.

We then pass on only those generated utterances deemed to have the same meaning with the reference utterance into the next stage of readability evaluation. To evaluate readability we perform pairwise comparisons between generated sentences and reference sentences. We randomize the order during evaluation so it is not clear

what the origin of a particular sentence is. We defined readability, sometimes called fluency, for the evaluator as how well a given utterance reads, *"is it fluent English or does it have grammatical errors, awkward constructions, etc"* (Mille et al., 2018a). By investigating readability of sentences with meaning similarity, we hope to see how the sentence-plan-to-text model performs compared with a human written sentence. The sentence-plan-to-text model is required to at least match human level performance in order to be usable. If it does not then we need to investigate where it fails and why. We used Prodigy (Montani and Honnibal, 2018) as our data annotation tool.

5.3 Results

5.3.1 Sentence-Plan-to-Text Analysis

	BLEU
E2E	0.8247
+ TripAdvisor	**0.8338**

Table 5.1: BLEU scores of the sentence-plan-to-text models evaluated on silver parse sentences from the validation set.

Automated Evaluation To establish if training with the TripAdvisor augmented dataset was beneficial, we performed automated evaluation of the sentence-plan-to-text models. Each sentence-plan-to-text model is provided with a sentence plan of the target sentence. When the model is trained with the augmented dataset, the BLEU score is slightly higher, see Table 5.1.

	E2E	E2E + TripAdvisor	Win?
BLEU	77.4194	78.2680	s2>s1
	[76.8013,78.0453]	[77.6815,78.8740]	p=0.0000

Table 5.2: Statistical significance tests of the sentence-plan-to-text model evaluation using compare-mt's BLEU score bootstrap resampling functionality with a sample size of 1,000 pairs.

	E2E	+ TripAdvisor
Total Validation Set Sentences	8024	8024
Exact matches	3807	3935
Punctuation and/or determiner differences	1242	1268
Remaining Sentences	2975	2821

Table 5.3: Filtering out generated sentences with exact or very close matches. Sentences are generated from a dataset of silver parse E2E validation set sentences.

In Table 5.2 we show a statistical significance test using compare-mt's BLEU score bootstrap resampling functionality with a sample size of 1,000 pairs (Neubig et al.). The BLEU scores reported by compare-mt differ from those of the Moses multi-bleu perl script, likely due to differences in tokenization used by each evaluation. However, we still see that the relative performance of each system is the same and that the difference between the two systems is highly statistically significant, $p < 0.0001$.

	E2E	+ TripAdvisor
Remaining Sentences analysed	325	325
Failed meaning similarities	76	45
Same readability as reference	198	208
Worse readability than reference	30	43
Better readability than reference	21	29

Table 5.4: Manual analyzing 325 of the *remaining sentences* generated from the silver parsed E2E validation set sentences

Manual Analysis Starting with generated sentences from the E2E validation set, we first filter out exact or very close matches to the reference sentences (see Table 5.3). Then taking a subset of the remaining generated sentences, we establish that they contain the same meaning as the reference sentence. Finally we compare the readability / naturalness of the generated text with the human reference sentences (see Table 5.4).

While the sentence-plan-to-text model trained on both E2E and TripAdvisor corpora generally outperforms the model trained on only E2E data, it has more sentences rated as *Worse readability than reference*. More detailed manual analysis

is required to tell whether this is a statistical anomaly or a true insight into how the silver data from the TripAdvisor corpora is affecting model performance.

Analysis of Failed Meaning Similarities Analysing examples where a generated sentence failed to correctly capture the meaning of the reference sentence we find the causes for this fall into a number of categories:

- Poor sentence tokenization
- Problems with the reference sentence
- Unusually phrased reference sentence
- Unknown words
- Generation model failures (repetition or missing words)

The model trained on the additional TripAdvisor corpus has a larger vocabulary and has seen a wider range of sentences, and thus fails less often. Most failures appear to be due to reference sentences containing unknown tokens or being phrased in a new or unusual way the model has not seen before. A smaller number of cases are attributable to issues directly with the generation model, namely repetition or absence of tokens from the sentence plan. Figure 5.2 contains three examples of failed generation.

5.3.2 End-to-End Analysis

We report results on the full E2E task in Table 5.5. Both our systems outperform the E2E challenge winning system Slug2Slug (Juraska et al., 2018), with the system using the sentence-plan-to-text model trained with the augmented dataset performing slightly better. Both sentence-plan-to-text models received the same set of sentence plans from the single knowledge-base-entries-to-utterance-plan model.

Ref: Do not go to The Punter near riverside.

SP: go _(not xname riverside)_

Gen: Not go to The Punter in riverside.

(a) Model generation failure

Ref: With only an average customer rating, and it being a no for families, it doesn't have much going for it.

SP: have _(rating _(only average customer no _(and it families)_)_it n't much _(going it)_)_

Gen: With a only average customer rating and its no families, it won't have much that going to it.

(b) Unusual phrasing in reference sentence

Ref: Have you heard of The Sorrento and The Wrestlers, they are the average friendly families.

SP: heard _(you xnear _(xname and)_families _(they average friendly)_)_

Gen: You can be heard near The Sorrento and The Wrestlers, they are average friendly families.

(c) Nonsensical reference sentence

Figure 5.2: Three different scenarios of failed generation from silver parse sentences. Each scenario includes the reference sentence *(Ref)*, sentence plan *(SP)* and generated text *(Gen)*.

As mentioned in Chapter 3, though these results on automated metrics are promising, automated metrics may not be an entirely reliable source of information on NLG tasks and further human evaluation is required to establish the meaningfulness of the automated results.

5.4 Discussion

The work most similar to this is Dušek and Jurcicek (2016a). It is also in the domain of task-oriented dialogue and they apply two-stage generation; first generating deep

	BLEU	NIST	METEOR	ROUGE_L	CIDEr
Validation					
TGen	0.6925	8.4781	0.4703	0.7257	2.3987
Slug2Slug	0.6576	8.0761	0.4675	0.7029	-
Pipeline	0.7271	8.5680	0.4874	0.7546	2.5481
+ TripAdvisor	**0.7298**	**8.5891**	**0.4875**	**0.7557**	**2.5507**
Test					
TGen	0.6593	8.6094	0.4483	0.6850	2.2338
Slug2Slug	0.6619	8.6130	0.4454	0.6772	2.2615
Pipeline	0.6705	8.6737	**0.4573**	0.7114	2.2940
+ TripAdvisor	**0.6738**	**8.7277**	0.4572	**0.7152**	**2.2995**

Table 5.5: Automated results on the E2E test set

syntax dependency trees using a seq2seq model and then generating the final utterance using a non-neural surface realizer. They found that while generation quality is initially higher from the two-stage model, it is outperformed by an end-to-end seq2seq model when combined with a semantic reranker.

Concurrent to this work was Moryossef et al. (2019). In their work they split apart the task of planning and surface realization. Conversely to Dušek and Jurcicek (2016a), Moryossef et al. employ a rule based utterance planner and a seq2seq surface realizer. They applied their system to the WebNLG corpus (Gardent et al., 2017b) and found that, compared with an end-to-end seq2seq model, it performed roughly equally at surface realization but exceeded the end-to-end seq2seq model at adequately including information in the generated utterance.

Other work has looked for innovative ways to separate planning and surface realization from the end-to-end neural systems, most notably Wiseman et al. (2018) which learns template generation also on the E2E task, but does not yet match baseline performance, and He et al. (2018) which has a dialogue manager control decision making and passes this information onto a secondary language generator. Other work has attempted either multi-stage semi-unconstrained language generation, such as in the domain of storytelling (Fan et al., 2019), or filling-in-the-blanks style sentence reconstruction (Fedus et al., 2018).

5.5 Conclusion

In this chapter, we introduced a new sentence plan and investigated whether it could work as an intermediate representation in a pipeline surface realization system. We found the generated sentences from the sentence plan to be of high quality, and that results improved further when trained on additional data. Automated evaluation results on the full task exceeded that of top performing systems from the E2E shared task, but further human evaluation and analysis is required, see below.

5.6 Retrospective

Chronologically, the paper this chapter is based on was written before the sentence-plan-to-text system improvements that we introduced in Chapter 4 were implemented (Elder et al., 2019). Thus, we did not use restricted decoding in the sentence-plan-to-text model. While the automated results in Table 5.5 look promising, in Chapter 6, Section 6.6, we introduce new analysis that highlights issues with using a seq2seq model for the first stage of the two stage approach.

Chapter 6

Reliable Neural Generation

In Chapters 4 and 5, we demonstrated that sentence-plan-to-text models can generate quality text when given the right kind of sentence plan and training data. In this chapter, we continue investigating the design of pipeline surface realization systems and focus more heavily on the adequacy of the generated text from sets of knowledge base entries.

Again using the E2E NLG Challenge dataset, we design an utterance plan that includes the surface form of each attribute-value pair. Surface forms are the exact tokens used in the target text to express information from the attribute-value pair. By including the surface form in the source sequence, we can use restricted decoding when generating an utterance. We achieve 100% reliability on the E2E dataset, as measured using semantic accuracy. By comparison, the best baseline seq2seq model gets 92% semantic accuracy (Juraska et al., 2018). Furthermore, we perform experiments to analyse the amount of diversity[1] lost from following this restricted approach, and find that the baseline systems exhibit minimal diversity — only using an average of 3 different surface forms to express an attribute-value pair. Given the apparent lack of diversity in baseline systems, our proposed approach incurs only a small sacrifice of unconstrained diversity in exchange for 100% reliability.

[1]We define diversity here as the number of distinct surface forms that a model uses to express a given attribute.

Our contribution in this chapter is demonstrating how an utterance plan containing the key content words of the target text can be used to train seq2seq models that reliably generate adequate text. Additionally, we perform an analysis of the diversity of text generated by baseline systems from the E2E NLG Challenge.

6.1 Method

How can a surface realization system generate text from a set of attribute-value pairs and ensure that they appear correctly in the generated text? As opposed to templates, which are static, seq2seq models are statistical generators and provide no inherent guarantees of accuracy — unless we can find an alternative solution. Thus, we propose creating an utterance plan which contains the surface forms of each attribute-value pair from the set of knowledge base entries in the E2E dataset. As previously mentioned, surface forms are the exact tokens used in the target text to express information from the attribute-value pair. Including surface forms in the source sequence enables us to restrict the text generated in a way that provides guaranteed reliability. We define reliability as generating an utterance which contains all, and only, the attribute-value pairs that appear in the source sequence. The process we outline for using the utterance plans is to; first find and extract the surface forms from the training data, then create an utterance plan for every example in the training data, and finally use restricted decoding at test time to guarantee reliability.

Finding Surface Forms The first step in this process is finding the surface forms in a given utterance; the content words used to express the attribute-value pairs in the human-authored target text. As discussed in Chapter 2, this provides a clear alignment between the source and target sequences. However, to find the content words we need to understand more about the dataset. Specially designed regular expressions (Dušek et al., 2019) or heuristics involving dependency relations

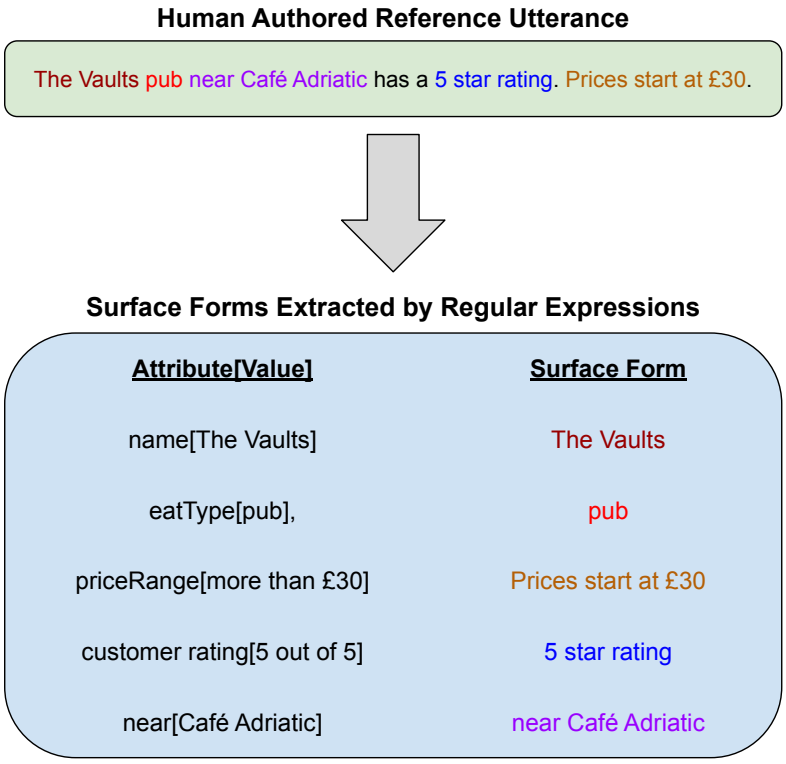


Figure 6.1: Extracting surface forms from an utterance. First, use regular expressions to find the surface forms in a target utterance. Then, use the surface forms to construct an augmented input sequence.

(Oraby et al., 2019) must be used. Our work was enabled by a set of regular expressions released by Dušek et al. (2019).[2] The regular expressions capture the entire phrase used to express an attribute-value pair, focusing on the content words and attempting as much as possible to leave out the function words, e.g.

```
(?:(?:price|range).*)?(?:inexpensive|cheap)(?:ly)?(?:.*(?:price\w|range))?
```

Augmented Input Sequence Once the surface forms of each attribute-value pair in a target utterance are found, we add them to the source sequence, as shown in Figure 6.1. During training, only the source sequence is altered, the target utterance remains the same. Our source sequence is an utterance plan in the format of a single token representing an attribute-value pair followed by multiple tokens for the

[2] https://github.com/tuetschek/e2e-cleaning

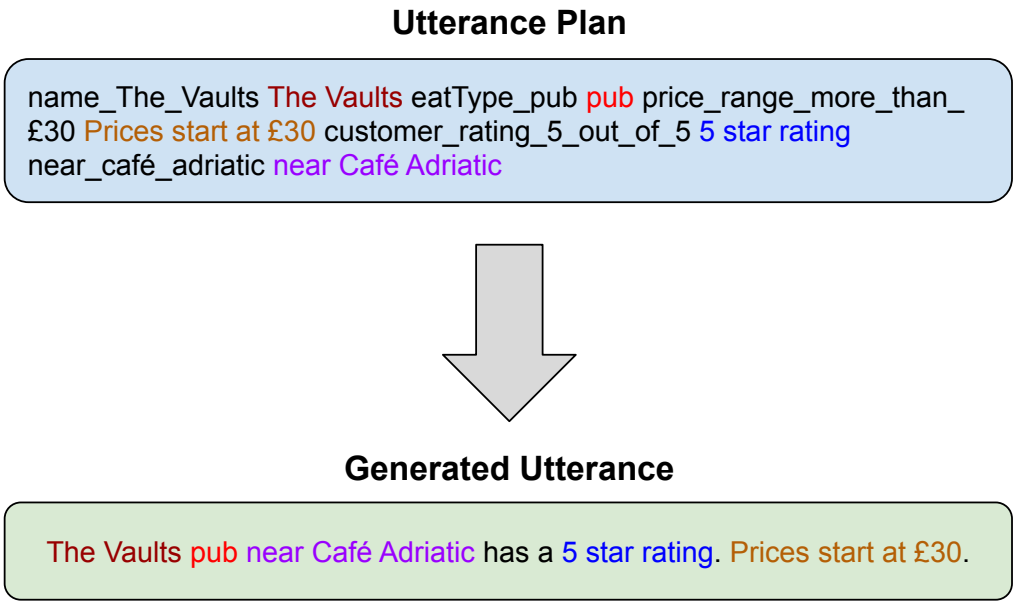


Figure 6.2: Generating text from an utterance plan containing surface forms for each attribute-value pair.

surface form, e.g. **eatType_pub** *pub* **customer_rating_5_out_of_5** *5 star rating*. The order of the attribute-value pairs in the source sequence remains the same as in the original dataset. If the surface form of an attribute-value pair was not found in the target utterance then a *missing* token was added instead. We thought that using a *missing* token would be preferable to omitting the attribute-value pair entirely because this would allow for cases where the rule based parser failed to detect an attribute-value pair that was actually present; this could help to minimise hallucination that occurs in cases where the model incorrectly associates unannotated text with an unrelated attribute-value pair. Any additional attribute-values, those that appeared in the target utterance but not in the input, were ignored.

An obvious challenge with using surface forms is deciding what surface forms to use in the source sequences of the validation and test sets as there are no reference target utterances available. Therefore, to avoid peeking at the target utterance, we used a simple heuristic: we choose the most common surface form for each attribute-value from the training set to add to the source sequences in the validation and test sets.

Restricted Decoding Why do we focus so much on surface forms? Because when surface forms are part of the source sequence, we can add restrictions to the generation strategy, e.g. beam search, which guarantee that all, and only, the surface forms provided have been expressed in the generated text (Zhong et al., 2017). Furthermore, by including all the necessary content words in the input sequence, it is possible to limit the vocabulary used during generation to only these content words and a couple of hundred function words. This would enable the use of a constrained softmax (Hu et al., 2015) – an optimization that can greatly speed up the decoding step. However, as decoding speed was not an important factor for the research, we did not implement the constrained softmax optimisation in this work. Compared with the sentence-plan-to-text task of Chapter 4, we observed that hallucination was less of an issue with this sentence-to-plan task, perhaps due to the smaller vocabulary and lower diversity of the training data. The task proved to be simple enough for the model so that only minimal restricted decoding was necessary. During experiments, our restricted decoding consisted of a single rule added to the beam search: if *"restaurant"* does not appear in the input then it should not appear in the output.[3] This is the only restriction we used.

6.2 Experimental Setup

6.2.1 Modelling

As with the systems presented in Chapters 4 and 5, our baseline is a seq2seq model with copy attention, trained on the E2E dataset, using the neural machine translation framework OpenNMT (Klein et al., 2017). It is referred to as *OpenNMT (Baseline)* in tables. To test our method, we trained the OpenNMT model with the same hyperparameters on a surface-form-augmented-version of the E2E dataset.

[3] https://github.com/Henry-E/surface_realization_opennmt-py/commit/12569e4f93e708dfbafaf892d93cfef49ec5bd9f

This is referred to as *OpenNMT + Surface Forms* in tables.

Details of the Python modules and bash scripts needed to run the experiments are available in our *main repository.*[4] Inside the *"scripts/"* folder of our main repository there are bash scripts for running the preprocessing required by OpenNMT and the actual training. We use our own fork of OpenNMT; the only changes made were to the beam search decoding code.[5] Full hyperparameter details are available in the main repository. Here is a short synopsis of the model: a seq2seq model with copy attention, using the Adam optimizer with learning rate 0.001, 2 layers, 300 dimension word vectors, 600 dimension LSTM cells, and shared embeddings between encoder and decoder.[6] We train for 20 epochs of the data, this takes 15 minutes using two Nvidia 1080 Ti GPUs. We then choose the checkpoint with the highest validation set accuracy. This is a checkpoint from before overfitting becomes noticeable and performance on the validation set declines, usually around the 15 epoch mark.

6.2.2 Reference Systems

The E2E NLG Challenge organisers released the generated outputs of all participant systems. In our analysis, we compare with three of these systems:

1. the E2E baseline, TGen (Dušek and Jurcicek, 2016b), a seq2seq model with a semantic reranker as a final step to improve accuracy
2. the overall winner of E2E, Slug2Slug (Juraska et al., 2018), a seq2seq model, also with a reranker, trained using an augmented dataset in which attribute-value pairs are aligned to individual sentences in the utterance
3. a template based-system, TUDA (Puzikov and Gurevych, 2018), which, by using a set of handwritten templates, was able to express attributes more

[4] https://github.com/Henry-E/reliable_neural_nlg

[5] https://github.com/Henry-E/surface_realization_opennmt-py

[6] Our choice of hyperparameters was based on values that had worked well for previous models we trained on this dataset

	OK	Added	Missing	A+M
♡ **TGen**	502 (80%)	14 (2%)	100 (16%)	14 (2%)
♡ **Slug2Slug**	582 (92%)	0	23 (4%)	25 (4%)
♡ **OpenNMT (Baseline)**	426 (68%)	13 (2%)	191 (30%)	0
♢ **OpenNMT + Surface Forms**	630 (100%)	0	0	0
♠ **TUDA**	630 (100%)	0	0	0

Table 6.1: Semantic accuracy on the test set. System architectures are coded with colours and symbols: ♡seq2seq, ♢augmented data ♠template-based

	OK	Added	Missing	A+M
♡ **OpenNMT (Baseline)**	59	2	39	0
♢ **OpenNMT + Surface Forms**	100	0	0	0

Table 6.2: Results of manual semantic accuracy analysis on 100 examples from the test set. System architectures are coded with colours and symbols: ♡seq2seq, ♢augmented data

reliably than all other systems and came in second place in the challenge's human evaluation.

6.2.3 Evaluation

To evaluate the reliability of our proposed approach we focus on semantic accuracy. Semantic accuracy scoring was also provided by Dušek et al. (2019). It reports the number of generated utterances that: correctly express all attribute-value pairs (OK), have additional pairs (Added), are missing pairs (Missing), have both missing and added pairs (A+M).

For completeness, we report results from the E2E NLG Challenge's official scoring script, which consists of the following n-gram overlap metrics; BLEU (Papineni et al., 2002), NIST (Doddington, 2002), METEOR (Lavie and Agarwal, 2007), ROUGE (Lin, 2004), and CIDEr (Vedantam et al., 2015).

	BLEU	NIST	METEOR	ROUGE_L	CIDEr
Validation					
♡ **TGen**	0.6925	8.4781	0.4703	0.7257	2.3987
♡ **Slug2Slug**	0.6576	8.0761	0.4675	0.7029	-
♡ **OpenNMT (Baseline)**	**0.7415**	**8.7010**	**0.4898**	**0.7663**	**2.5999**
♢ **OpenNMT + Surface Forms**	0.6589	8.4099	0.4372	0.6907	2.2848
♠ **TUDA**	0.6051	7.5257	0.4678	0.6890	1.6997
Test					
♡ **TGen**	0.6593	8.6094	0.4483	0.6850	2.2338
♡ **Slug2Slug**	0.6619	8.6130	0.4454	0.6772	**2.2615**
♡ **OpenNMT (Baseline)**	**0.6815**	**8.7481**	0.4452	**0.6904**	2.2391
♢ **OpenNMT + Surface Forms**	0.6283	8.3107	0.4277	0.6682	2.1465
♠ **TUDA**	0.5657	7.4544	**0.4529**	0.6614	1.8206

Table 6.3: Automated evaluation metrics on the E2E validation and test sets. System architectures are coded with colours and symbols: ♡seq2seq, ♢augmented data, ♠template-based

6.3 Results

6.3.1 Semantic Accuracy

Table 6.1 demonstrates that the semantic accuracy of our proposed method is on par with that of the template system; both achieve 100% accuracy. On the other hand, the baseline seq2seq models struggle, with the best system, Slug2Slug, only achieving 92%. Our baseline OpenNMT system performs particularly poorly as it does not use a semantic reranker, which both TGen and Slug2slug do.

In table 6.2, we present the results of a manual semantic accuracy analysis on 100 examples from the test set. We find that these numbers line up with automated semantic accuracy results.

6.3.2 N-gram Overlap Metrics

According to the automated results on the E2E validation and test sets, shown in Table 6.3, semantic accuracy and n-gram overlap metrics have little correlation. The highest scoring system in many of the n-gram metrics, the OpenNMT baseline, is the worst performing in semantic accuracy. On the other hand, the template system

♡Blue Spice is a coffee shop near Crowne Plaza Hotel with a customer rating of 5 out of 5. ♢Blue Spice is a coffee shop near Crowne Plaza Hotel with a customer rating of 5 out of 5. ♠Blue Spice is a coffee shop located near Crowne Plaza Hotel. It has a customer rating of 5 out of 5.
♡The Cricketers is a family friendly coffee shop near Avalon. It has a customer rating of 1 out of 5. ♢The Cricketers is a family friendly coffee shop near Avalon with a customer rating of 1 out of 5. ♠The Cricketers is a family-friendly coffee shop located near Avalon. It has a customer rating of 1 out of 5.
♡Blue Spice is a chinese pub located in the city centre near Rainbow Vegetarian Café. It is not family-friendly. ♢Blue Spice is a chinese pub near Rainbow Vegetarian Café in the city centre. It is not family-friendly. ♠Blue Spice is a pub which serves chinese food. It is located in the city centre area , near Rainbow Vegetarian Café. It is not family friendly.

Table 6.4: Generated text from three systems, for three different sets of attribute-value pairs. Systems are coded with the colours and symbols: ♡SLUG2SLUG, ♢OPENNMT + SURFACE FORMS, ♠TUDA

scores highest in METEOR but lowest in all other metrics. Overall, we infer that the n-gram metrics results are ambiguous, making it difficult to draw useful conclusions from these metrics.

Automated results from the system presented in Chapter 5 are not included in Table 6.3 as we felt this would distract from the comparison being made between systems here.

6.3.3 Generated examples

In Table 6.4, we compare randomly selected examples from Slug2Slug, our Surface Forms system and TUDA. In each of the examples, the systems appear to follow a very similar sentence structure to each other. In the E2E human evaluation for naturalness, Slug2Slug came second while TUDA came eighth, compared with the human evaluation for overall quality in which Slug2Slug came first and TUDA second. Dušek et al. (2020) hypothesised that the lower performance in naturalness

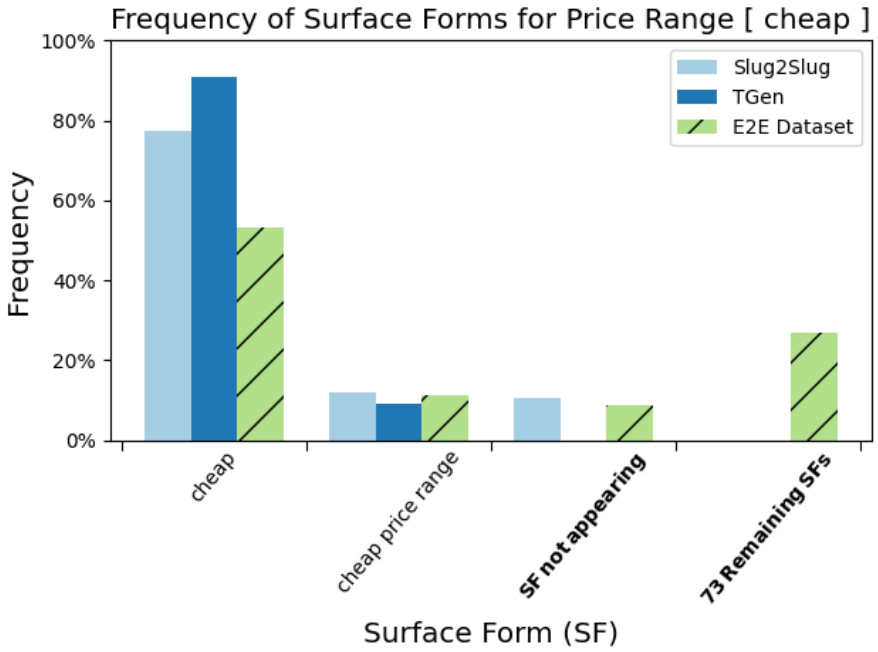


Figure 6.3: Surface forms used to express the attribute-value pair: PriceRange[Cheap]. **SF not appearing**: no surface form was found in the utterance, **73 Remaining SFs**: surface forms other than *cheap* and *cheap price range*

may be linked to sentence length; template systems tend to be slightly longer than neural ones. Slug2Slug has an average utterance length of 24 tokens, while TUDA has an average length of 32 tokens. Our system has an average length of 23, which is closer to that of Slug2Slug. This suggests that our approach has the potential to combine the reliability of template systems with the perceived naturalness of neural ones.

6.4 Frequency of Surface Forms

By restricting the generated text to a single surface form for each attribute-value pair, one could infer that we are losing a significant amount of diversity. To address this point, we performed analysis of surface form usage in the dataset and by the different seq2seq models. Each attribute-value pair has many possible surface forms that appear in the training split of the dataset. On average there are 133, and in

Figure 6.3, we show there are 75 different surface forms used for the attribute-value pair: "price range [cheap]". This is displayed in the green column with diagonal lines. In the two blue columns, we show the frequency of surface forms used by the two neural systems, Tgen and Slug2Slug, in their test set generated utterances. Intuitively, one might expect that a neural NLG system trained on a dataset with 75 different surface forms would use a wide variety of them. Instead, we see that Slug2Slug and Tgen only use the top two most common surface forms: "Cheap" and "Cheap price range". Across all the attribute-value pairs, the neural systems generated, on average, only the top 3 most common surface forms. Given that the neural systems are not very diverse and that our main goal is reliability, this seems like a small trade off to make. Diversity can always be introduced at a later stage, in future research.

6.5 Discussion

This is not the first data-focused approach to improving reliability; Balakrishnan et al. (2019) also proposed a constrained decoding strategy. The difference between our decoding strategies lies in the guarantees provided. Their approach focuses on an augmented target utterance — as opposed to our augmented source sequence — in which special bracket tokens are used to surround surface forms. e.g. [__**ARG_AREA_CITY_CENTRE**__ *city centre*]. Their constrained decoding strategy guarantees that when an opening bracket is generated, a closing bracket will also be generated. However, this provides no guarantee as to what will be contained within the brackets. What sets our method apart is that: we can guarantee the text will actually be generated as requested; we generate shorter sequences (no bracket tokens in the output); we have the option for a restricted vocabulary, which could speed up decoding.

The major weakness of both approaches, however, is the difficulty of extracting

surface forms from human-authored text. We were able to avail of the hand-crafted regular expressions of Dušek et al. (2019) in our E2E experiments, but moving to another dataset would entail a similar exercise. A method to do this automatically would be convenient. Some work has already been done by Oraby et al. (2019), in which dependency trees were used to find adjectives that describe a specific list of food related nouns. In the 2018 surface realization shared task (SRST) (Mille et al., 2018a), the deep task dataset was created by pruning function words from a dependency tree, leaving only content words remaining. For example, in Chapter 5, we created a dataset from the TripAdvisor Corpus using the UUD parser.

In our proposed method, surface forms still need to be joined together with function words. We believe seq2seq models are well suited for this task because they are good at generating natural sounding, though sometimes nonsensical, text. By combining seq2seq models with constraints based on content words included in the input sequence, we aim to achieve both reliability and naturalness.

An alternate approach, which we did not compare with, is automatic template generation (Biran et al., 2016; Wiseman et al., 2018). As with neural generation, when applied to the E2E task, it has issues with reliability. Mille and Dasiopoulou (2017) used an automated template generation approach on the E2E shared task and their accuracy score was similar to that of a neural system, 92% (Dušek et al., 2020), mostly due to missing attributes in templates.

However, the question remains: why pursue this approach when templates perform satisfactorily? We believe that seq2seq models are easier to maintain, generate more natural text, and, as surface form extraction improves, they also become more scalable: to new domains, languages, and possibly even personalization.

	OK	Added	Missing	A+M
E2E	413 (65.5%)	0 (0%)	205 (32.5%)	12 (2%)
+ TripAdvisor	413 (65.5%)	0 (0%)	205 (32.5%)	12 (2%)

Table 6.5: Semantic accuracy results on the E2E test set from our pipeline neural surface realization systems, introduced in Chapter 5.

	OK	Added	Missing	A+M
Generated Sentence Plans	50	0	46	4

Table 6.6: Manual semantic accuracy results on 100 of the unique sentence plans generated from the E2E test set from the content selection system, introduced in Chapter 5.

6.6 "Designing a Sentence Plan" Revisited

We apply semantic accuracy evaluation to the outputs from the pipeline surface realization approach in Chapter 5 and find the pipeline generated text performs poorly, 65.5%; a worse result than even this chapter's baseline OpenNMT-py seq2seq model, 68%. The evaluation in Chapter 5 largely focused on the quality of text generated by the sentence-plan-to-text model. In that evaluation we found that the sentence-plan-to-text model was able to reliably reconstruct sentences from silver parse trees. Given that the majority of errors in Table 6.5 are linked to missing attributes, this poor semantic accuracy is likely due to a failure of the knowledge-base-entries-to-utterance-plan model to include these attributes in the generated utterance plan. Similarly, the fact that both systems (with and without TripAdvisor data) have identical semantic accuracy results is an additional indicator that the problem lies with the knowledge-base-entries-to-utterance-plan model. We investigated this hypothesis by performing a manual semantic accuracy analysis[7] on the generated sentence plans, shown in Table 6.6. We found that of the 100 sentence plans generated for the test set we analysed, only 50 were rated as OK. This is a lower score than even the automated semantic accuracy results and could be a sampling issue.

Ultimately these experiments led us to determine that seq2seq models are too

[7]Full details of the analysis can be found at https://github.com/Henry-E/wp_neural_pipeline/tree/master/experiments/manual_analysis_content_selection

unreliable to use for utterance planning. It is difficult to collect additional data to train the knowledge-base-to-utterance-plan model and, unlike the sentence-plan-to-text model, there is no obvious way to restrict the generation to ensure accuracy.

6.7 Conclusion

In this chapter, we showed how by using the surface forms in the source sequence, we could train a model that reliably generated the text we needed. We believe this provides more evidence that seq2seq models are capable of reconstructing sentences from sentence plans containing content words but no function words. Furthermore, we addressed the possible issue with using surface forms others may have, namely a lack of diversity. We showed the neural system themselves, despite being trained on a wide range of surface forms, only used a limited set of them. This concludes our set of experiment chapters.

Chapter 7

Discussion

In this thesis, we have analysed the surface realization task in detail and proposed ways to train more reliable seq2seq models. Throughout the thesis, we have highlighted examples where seq2seq models generate text that is fluent but ultimately inadequate. We proposed that surface realization systems be built using a pipeline framework, with a sentence plan or utterance plan as an intermediate representation. In this chapter, we summarise the reasons and evidence in support of our case for using sentence plans in surface realization. We also look back at the research questions set out in the introduction and evaluate our contributions to each question. Finally, we highlight areas for further research.

7.1 The Case for Sentence Plans

We claim that sentence plans can be used to build surface realization systems that are reliable. Our reasons for claiming this are threefold:

1. Sentence plans use source representations that contain detailed information about the text to be generated. This extra detail provides additional implicit alignment to the seq2seq models, which helps to train better models and thus generate better quality text.

2. We propose to use sentence plans based on representations from parsing tasks. The availability of high quality parsers makes collecting silver data straightforward. Silver data can be used to create datasets for new domains or to augment existing datasets. We have shown that training on silver data provides significant benefit to model performance.

3. Sentence plans can contain the words or tokens intended to appear in the generated text. By including the key content words in the source sequence, we can restrict the generated text. These restrictions give us some guarantee of adequacy and reliability.

Detailed Source Representations In Chapter 2, we described the differences between sentence-plan-to-text tasks and knowledge-base-entries-to-text tasks. Sentence plans use more detailed source representations than sets of knowledge base entries. These additional details provide greater implicit alignment that the seq2seq model can use to train itself on. For instance, in our experiments on the SRST shallow task in Chapter 4, by simply adding permanently random linearizations and scoping brackets to the preprocessing of the text, the model gained an extra 12 BLEU score. This highlights the importance of understanding implicit alignments and designing a source representation that is well aligned with the target text.

Silver Data In Chapter 4, we demonstrated a significant increase in BLEU score when using silver data on the SRST shallow task. In Chapter 5, we created a new dataset of silver parse sentences from the E2E dataset, and also augmented the training data with silver parse sentences from the TripAdvisor corpus. We demonstrated that the seq2seq model was able to generate sentences of reasonable quality from our new sentence plan. These sentence plans were automatically parsed, first by the Stanford universal dependency parser and then by the underspecified universal dependency rule-based parser. Collecting this new dataset required no

additional human annotation, beyond what was needed to create both the parsers initially. Kedzie and McKeown (2019) successfully generated additional training data for the E2E task and demonstrated that it resulted in much more accurate models. However, this approach required Kedzie and McKeown to hand-craft a rule-based parser, specifically for the E2E data, that was used to filter generated sentences based on semantic accuracy. This is a time consuming approach, one that is unlikely to scale as it requires creating a new rule-based parser for each dataset. In Chapter 6, our method for creating utterance plans also suffers from this issue as it relies on a set of hand-crafted regular expressions. Other proposed methods for data efficient generation of text rely on human annotated representations of target sentences, a similarly burdensome approach (Balakrishnan et al., 2019; Arun et al., 2020). By comparison, parsing representations, such as universal dependency trees, are generic and able to parse text from a wide number of domains.

Restricted Decoding We demonstrated the benefits of restricted decoding in Chapters 4 and 6. In Chapter 4, the SRST shallow task's sentence plan contained all of the token lemmas required in the target text. This made it easy to restrict the generation and guarantee the correct tokens appear in the generated text. Restricted decoding increased performance by 3 BLEU score. In Chapter 6, generation was restricted using surface forms that were included in the source sequence. Restricted decoding ensured that only those surface forms appeared in the generated text. This enabled us to score 100% semantic accuracy.

7.2 Research Questions Revisited

In Chapter 1, we introduced the three research questions that arose out of our proposed focus on pipeline surface realization systems involving sentence plans. Here, we will evaluate how each was addressed in the thesis.

1. What is the best way to use seq2seq models for the sentence-plan-to-text task?

2. Can automatically parsed sentence plans significantly benefit generation?

3. How can a sentence plan be generated from a set of knowledge base entries?

RQ1 Sentence-Plan-to-Text The major contribution of this thesis is the exploration of implicit alignments and how to train models on the sentence-plan-to-text task. This analysis has directly affected our approach to the design and preprocessing of sentence plans in order to maximise performance. In Section 2.2, we introduced a framework for analysing source-target alignment in surface realization tasks, focusing primarily on the order of source tokens and what the source tokens represent in the target sequence. In Chapter 4, we described our system for the SRST shallow task. This system has achieved joint first place in the English language portion of all three shared tasks (Mille et al., 2018a, 2019, 2020). In Chapter 5, we used automated and human evaluation to determine the quality of generated text from our sentence plan. This evaluation helped to validate both our approach to designing preprocessing for sentence plans and our method for generating text from sentence plans using seq2seq models. In Chapter 6, our contribution to this research question was the design of a utterance plan with clear implicit alignments. We demonstrated how this utterance plan can be combined with restricted decoding to achieve reliable text generation from a seq2seq model.

RQ2 Silver Parse Data Although additional training data benefits performance on many other machine learning tasks, the benefit of automatically parsed data to surface realization tasks was not guaranteed and required validation. Our contribution to this research question lies in the analysis done in Chapters 4 and 5. In Chapter 4, we performed detailed analysis to better understand the effect of different corpora and the ways in which silver data was benefiting performance. In Chapter 5, our contribution was to use silver parse data in the creation of a new sentence

plan dataset. Furthermore, we analysed the use of additional in-domain silver parse data and its effect on generation.

RQ3 Knowledge-Base-Entries-to-Utterance-Plan As part of our proposed pipeline surface realization system, knowledge base entries are required to be processed into sentence plans. We investigated two possible approaches to this problem. In Chapter 5, we tested a model-based solution to generate utterance plans, a seq2seq model was trained on a knowledge-base-entries-to-utterance-plan dataset. Ultimately, we determined seq2seq models were not a good fit for this problem as the utterance plans generated may have been inadequate. In Chapter 6, we experimented with a rule-based heuristic. This approach was more successful, the rule-based heuristic generated utterance plans included all the correct information.

Manual analysis presented in Section 6.6 supports the hypothesis that Chapter 5's knowledge-base-entries-to-utterance-plan seq2seq model was largely responsible for the errors seen when applying semantic error analysis to the output of the full system in Chapter 6. Similarly in Section 6.3, we found that the manual semantic accuracy analysis performed supported the high automated semantic accuracy results for the proposed system.

7.3 Future Work

We see two main issues still to be resolved with our proposed pipeline surface realization system. The first is that we need a generic approach to converting knowledge base entries to sentence plans. The second is that while we advocate for the use of sentence plans with content words because of the ability to restrict decoding from them, we don't actually have guarantees of fluency in this new approach.

Generic Knowledge-Base-Entries-to-Utterance-Plan Systems Although Chapter 6 presented a reliable rule-based heuristic, due to the nature of the ut-

terance plan, this rule-based heuristic isn't as generalizable as we would like it to be. A better solution would be a rule-based system that can convert knowledge base entries into a sentence plan more like the underspecified universal dependency representation, for example the sentence plan used in Chapter 5. Moryossef et al. (2019) used a rule-based system to construct sentence plans for the WebNLG task, although we have yet to investigate the effort required to generalize their approach to new domains. Creating an accurate, generalizable approach for converting knowledge base entries into a sentence plan is quite an undertaking, which we suggest as an avenue for extending the work presented in this thesis. But it is an essential component to making a pipeline surface realization system.

Fluent Restricted Generation In both Chapters 5 and 6, the sentence plans we designed focused on the use of content words in the source sequence. Though we could use restricted decoding to ensure the content words appeared in the target sequence, this provides no guarantee of fluency. We are relying on the seq2seq model being well enough trained that it will generate fluent sentences in the context of these restrictions. This is based on the known ability of seq2seq models to generate fluent text, which we then try to constrain in a way that ensures the text is adequate. The fluency of generated sentences under these restrictions needs to be evaluated further, though preliminary evidence from Chapters 5 and 6 is promising. Similarly, output from massively pretrained models discussed briefly in Chapter 3 is well known to be fluent though lacking coherency. Fine-tuning these models for surface realization may contribute towards confidence in the fluency of outputs, though still without any concrete guarantees.

7.4 Conclusion

In this thesis we have presented a pipeline system for reliable generation from seq2seq models trained for surface realization that can be used in task-oriented dialogue.

This system relies heavily on sentence plans and we have presented analysis and evidence supporting our claim that sentence plans can be used to attain more reliable text generation from seq2seq models. We hope that this work contributes to the design and development of more generalizable surface realization systems which can enable adequate and fluent text generation in task oriented dialogue.

Bibliography

Roee Aharoni and Yoav Goldberg. 2018. Split and Rephrase: Better Evaluation and Stronger Baselines. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 719–724, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ankit Arun, Soumya Batra, Vikas Bhardwaj, Ashwini Challa, Pinar Donmez, Peyman Heidari, Hakan Inan, Shashank Jain, Anuj Kumar, Shawn Mei, Karthik Mohan, and Michael White. 2020. Best Practices for Data-Efficient Modeling in NLG:How to Train Production-Ready Neural Models with Less Data. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 64–77, Stroudsburg, PA, USA. International Committee on Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained Decoding for Neural NLG from Compositional Representations in Task-Oriented Dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844, Stroudsburg, PA, USA. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf

Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The First Surface Realisation Shared Task: Overview and Evaluation Results. In *Proceedings of the European Workshop on Natural Language Generation*, December, pages 217–226.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, 3:1137–1155.

Or Biran, Terra Blevins, and Kathleen McKeown. 2016. Mining Paraphrasal Typed Templates from a Plain Text Corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1913–1923, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*,

pages 169–214, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 Bilingual, Bi-Directional {W}eb{NLG}+ Shared Task: Overview and Evaluation Results ({W}eb{NLG}+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. 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 552–562, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A Survey on Dialogue Systems. *ACM SIGKDD Explorations Newsletter*, 19(2):25–35.

Mingjie Chen, Gerasimos Lampouras, and Andreas Vlachos. 2018. Sheffield at E2E:

structured prediction approaches to end-to-end language generation. In *E2E NLG Challenge System Descriptions*.

Robert Dale. 2020. Natural language generation: The commercial state of the art in 2020. *Natural Language Engineering*, 26(4):481–487.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and Improving Morphological Learning in the Neural Machine Translation Decoder. *IJCNLP*, pages 142–151.

Florian Daniel, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh. 2018. Quality Control in Crowdsourcing. *ACM Computing Surveys*, 51(1):1–40.

Andrea F Daniele, Matthew R Walter, Mohit Bansal, and Matthew R Walter. 2017. Navigational Instruction Generation as Inverse Reinforcement Learning with Neural Machine Translation. In *Proceedings of HRI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *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.

George Doddington. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145, San Diego, CA, USA.

Shibhansh Dohare, Vivek Gupta, and Harish Karnick. 2018. Unsupervised Semantic Abstractive Summarization. In *Proceedings of ACL 2018, Student Research Workshop*, pages 74–83. Association for Computational Linguistics.

Ondrej Dušek and Filip Jurcicek. 2016a. Sequence-to-Sequence Generation for Spoken Dialogue via Deep Syntax Trees and Strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51. Association for Computational Linguistics.

Ondřej Dušek, David M. Howcroft, and Verena Rieser. 2019. Semantic Noise Matters for Neural Natural Language Generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, October, pages 421–426, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ondřej Dušek and Filip Jurcicek. 2015. Training a Natural Language Generator From Unaligned Data. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 451–461, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ondřej Dušek and Filip Jurcicek. 2016b. A Context-aware Natural Language Generator for Dialogue Systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, September, pages 185–190, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of End-to-End Natural Language Generation: The E2E NLG challenge. *Computer Speech & Language*, 59:123–156.

Henry Elder. 2020. ADAPT at SR'20: How Preprocessing and Data Augmentation Help to Improve Surface Realization. In *Proceedings of the Third Workshop on Multilingual Surface Realisation*, pages 30–34, Barcelona, Spain (Online). Association for Computational Linguistics.

Henry Elder, Robert Burke, Alexander O'Connor, and Jennifer Foster. 2020a. Shape of Synth to Come: Why We Should Use Synthetic Data for English Surface Re-

alization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7465–7471, Stroudsburg, PA, USA. Association for Computational Linguistics.

Henry Elder, Jennifer Foster, James Barry, and Alexander O'Connor. 2019. Designing a Symbolic Intermediate Representation for Neural Surface Realization. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 65–73, Stroudsburg, PA, USA. Association for Computational Linguistics.

Henry Elder, Sebastian Gehrmann, Alexander O'Connor, and Qun Liu. 2018. E2E NLG Challenge Submission: Towards Controllable Generation of Diverse Natural Language. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 457–462. Association for Computational Linguistics.

Henry Elder and Chris Hokamp. 2018. Generating High-Quality Surface Realizations Using Data Augmentation and Factored Sequence Models. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, pages 49–53, Stroudsburg, PA, USA. Association for Computational Linguistics.

Henry Elder, Alexander O'Connor, and Jennifer Foster. 2020b. How to Make Neural Natural Language Generation as Reliable as Templates in Task-Oriented Dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2877–2888, Stroudsburg, PA, USA. Association for Computational Linguistics.

Angela Fan and Claire Gardent. 2020. Multilingual AMR-to-Text Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2889–2901, Stroudsburg, PA, USA. Association for Computational Linguistics.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable Abstractive Summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for Structuring Story Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660, Stroudsburg, PA, USA. Association for Computational Linguistics.

Farhood Farahnak, Laya Rafiee, Leila Kosseim, and Thomas Fevens. 2019. The concordia NLG surface realizer at SR'19. In *MSR@EMNLP-IJCNLP 2019 - 2nd Workshop on Multilingual Surface Realisation, Proceedings*, pages 63–67, Stroudsburg, PA, USA. Association for Computational Linguistics.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. MaskGAN: Better Text Generation via Filling in the_____. In *International Conference on Learning Representations*.

Thiago Castro Ferreira, Sander Wubben, Emiel Krahmer, Thiago Castro Ferreira, Sander Wubben, and Emiel Krahmer. 2018. Surface Realization Shared Task 2018 (SR18): The Tilburg University Approach. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, volume 2018, pages 35–38. Association for Computational Linguistics.

Michele Filannino and Marilena Di Bari. 2015. Gold standard vs. silver standard: the case of dependency parsing for Italian. In *Proceedings of the Second Italian Conference on Computational Linguistics CLiC-it 2015*, December 2015, pages 3–4. Accademia University Press.

Reid Fu and Michael White. 2018. LSTM Hypertagging. In *Proceedings of the*

11th International Conference on Natural Language Generation, pages 210–220, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural Approaches to Conversational AI. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating Training Corpora for Micro-Planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. Creating Training Corpora for NLG Micro-Planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, i, pages 179–188, Stroudsburg, PA, USA. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017c. The WebNLG Challenge: Generating Text from RDF Data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Stroudsburg, PA, USA. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61(c):65–170.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Anuoluwapo Aremu, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh Dhole, Wanyu Du, Esin Durmus, Ondřej Dušek, Chris Chinenye Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Mihir Kale, Dhruv Kumar, Faisal Ladhak, Aman Madaan,

Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Andre Niyongabo Rubungo, Salomey Osei, Ankur Parikh, Laura Perez-Beltrachini, Niranjan Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc, Thibault Sellam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabezudo, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. 2021. The GEM Benchmark: Natural Language Generation, its Evaluation and Metrics. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, volume abs/2102.0, pages 96–120, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. {C}ycle{GT}: Unsupervised Graph-to-Text and Text-to-Graph Generation via Cycle Training. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, December, pages 77–88, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Valerie Hajdik, Jan Buys, Michael Wayne Goodman, and Emily M Bender. 2019. Neural Text Generation from Rich Semantic Representations. In *Proceedings of the 2019 Conference of the North*, pages 2259–2266, Stroudsburg, PA, USA. Association for Computational Linguistics.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling Strategy and Generation in Negotiation Dialogues. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2333–2343.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and

Comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Chris Hokamp and Qun Liu. 2017. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Stroudsburg, PA, USA. Association for Computational Linguistics.

David M Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty Years of Confusion in Human Evaluation: {NLG} Needs Evaluation Sheets and Standardised Definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Xiaoguang Hu, Wei Li, Xiang Lan, Hua Wu, and Haifeng Wang. 2015. Improved beam search with constrained softmax for NMT. *Machine Translation Summit XV*, 1(2014):297–309.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. GenWiki: A Dataset of 1.3 Million Content-Sharing Text and Graphs for Unsupervised Graph-to-Text Generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2398–2409, Stroudsburg, PA, USA. International Committee on Computational Linguistics.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A Deep Ensemble Model with Slot Alignment for Sequence-to-Sequence Natural Language Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, Volume 1 (Long Papers), pages 152–162, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Kedzie and Kathleen McKeown. 2019. A Good Sample is Hard to Find: Noise Injection Sampling and Self-Training for Neural Language Generation Models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 584–593, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Kedzie and Kathleen McKeown. 2020. Controllable Meaning Representation to Text Generation: Linearization and Data Augmentation Strategies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5160–5185, Stroudsburg, PA, USA. Association for Computational Linguistics.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context. pages 284–294.

Seokhwan Kim, Michel Galley, Chulaka Gunasekara, Sungjin Lee, Adam Atkinson, Baolin Peng, Hannes Schulz, Jianfeng Gao, Jinchao Li, Mahmoud Adada, Minlie Huang, Luis Lastras, Jonathan K. Kummerfeld, Walter S. Lasecki, Chiori Hori, Anoop Cherian, Tim K. Marks, Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, and Raghav Gupta. 2021. Overview of the Eighth Dialog System Technology Challenge: DSTC8. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29(Section 5):2529–2540.

David King and Michael White. 2018. The OSU Realizer for SRST '18: Neural Sequence-to-Sequence Inflection and Incremental Locality-Based Linearization. In *Proceedings of the First Workshop on Multilingual Surface Realisation*, 2009, pages 39–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gerasimos Lampouras and Andreas Vlachos. 2016. Imitation learning for language generation from unaligned data. In *Proceedings of {COLING} 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1101–1112, Osaka, Japan. The COLING 2016 Organizing Committee.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA.

Remi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, pages 1203–1213.

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, and Emiel Krahmer. 2021. Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language*, 67:101151.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART:

Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Stroudsburg, PA, USA. Association for Computational Linguistics.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning Semantic Correspondences with Less Supervision. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, (August):91–99.

C Y Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the workshop on text summarization branches out (WAS 2004)*, 1, pages 25–26.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume abs/1708.0, pages 1116–1126, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The {Stanford} {CoreNLP} Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 Task 9: Abstract Meaning Representation Parsing and Generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 536–545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal Dependency Annotation for Multilingual Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the State of the Art of Evaluation in Neural Language Models. *International Conference on Learning Representations*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. In *5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation, Boston, MA, May 2004*, pages 24–31.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pages 1–12.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, Emily Pitler, and Leo Wanner. 2018a. The First Multilingual Surface Realisation Shared Task (SR'18): Overview and Evaluation Results. In *Proceedings of the 1st Workshop on Multilingual Surface Realisation (MSR), 56th Annual Meeting of the Association for Computational Linguistics*, pages 1–10, Melbourne, Australia.

Simon Mille, Anja Belz, Bernd Bohnet, Yvette Graham, and Leo Wanner. 2019. The Second Multilingual Surface Realisation Shared Task (SR'19): Overview and Evaluation Results. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, Msr, pages 1–17, Stroudsburg, PA, USA. Association for Computational Linguistics.

Simon Mille, Anja Belz, Bernd Bohnet, and Leo Wanner. 2018b. Underspecified Universal Dependency Structures as Inputs for Multilingual Surface Realisation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 199–209. Association for Computational Linguistics.

Simon Mille, Anya Belz, Bernd Bohnet, Thiago Castro Ferreira, Yvette Graham, and Leo Wanner. 2020. The Third Multilingual Surface Realisation Shared Task ({SR}{'}20): Overview and Evaluation Results. In *Proceedings of the 3nd Workshop on Multilingual Surface Realisation (MSR 2020)*, Dublin, Ireland. Association for Computational Linguistics.

Simon Mille and Stamatia Dasiopoulou. 2017. FORGe at E2E 2017. In *E2E NLG Challenge System Descriptions*.

Ines Montani and Matthew Honnibal. 2018. Prodigy: A new annotation tool for radically efficient machine teaching. *Artificial Intelligence*, to appear.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation. In *Proceedings*

of the 2019 Conference of the North, pages 2267–2277, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shashi Narayan and Claire Gardent. 2020. Deep Learning Approaches to Text Production. *Synthesis Lectures on Human Language Technologies*, 13(1):1–199.

Graham Neubig, Zi-yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. compare-mt : A Tool for Holistic Comparison of Language Generation Systems.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 2019. compare-mt: A Tool for Holistic Comparison of Language Generation Systems. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL) Demo Track*, Minneapolis, USA.

Jekaterina Novikova, OndDušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why We Need New Evaluation Metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252. Association for Computational Linguistics.

Chris Olah and Shan Carter. 2016. Attention and Augmented Recurrent Neural Networks. *Distill*.

Shereen Oraby, Vrindavan Harrison, Abteen Ebrahimi, and Marilyn Walker. 2019. Curate and Generate: A Corpus and Method for Joint Control of Semantics and Style in Neural NLG. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5938–5951, Stroudsburg, PA, USA. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-Text Generation with Content Selection and Planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6908–6915.

Yevgeniy Puzikov and Iryna Gurevych. 2018. E2E NLG Challenge: Neural Models vs. Templates. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 463–471, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D Manning. 2018. Universal Dependency Parsing from Scratch. In *Proceedings of the (CoNLL) 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2019. Universal Dependency Parsing from Scratch.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. Unsupervised Pretraining for Sequence to Sequence Learning. In *Proceedings of the 2017 Conference on*

Empirical Methods in Natural Language Processing, pages 383–391, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ehud Reiter. 2017. You Need to Understand your Corpora! The Weathergov Example.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning Robust Metrics for Text Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 83–91, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anastasia Shimorina. 2018. Human vs Automatic Metrics: on the Importance of Correlation Design.

Anastasia Shimorina and Claire Gardent. 2018. Handling Rare Items in Data-to-Text Generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 360–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anastasia Shimorina, Claire Gardent, Shashi Narayan, and Laura Perez-Beltrachini. 2018. WebNLG Challenge: Human Evaluation Results. Technical Report January.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. A Gold Standard Dependency Corpus for {E}nglish. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation ({LREC}'14)*, pages 2897–2904, Reykjavik, Iceland. European Language Resources Association (ELRA).

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Marco Antonio Sobrevilla Cabezudo and Thiago Pardo. 2020. {NILC} at {SR}{'}20: Exploring Pre-Trained Models in Surface Realisation. In *Proceedings of the Third Workshop on Multilingual Surface Realisation*, pages 50–56, Barcelona, Spain (Online). Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A Graph-to-Sequence Model for AMR-to-Text Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616–1626, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lucia Specia, Dhwaj Raj, and Marco Turchi. 2010. Machine translation evaluation versus quality estimation. *Machine Translation*, 24(1):39–50.

Symon Stevens-Guille, Aleksandre Maskharashvili, Amy Isard, Xintong Li, and Michael White. 2020. Neural {NLG} for Methodius: From {RST} Meaning Representations to Texts. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 306–315.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics*, 26(3):339–373.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 2015, pages 1556–1566, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:4566–4575.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *Advances in Neural Information Processing Systems 28*, pages 2692–2700.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*, pages 783–792, Washington, DC, USA. ACM Press.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, page 618, New York, New York, USA. ACM Press.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain Neural Network Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 120–129, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, September, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning Neural Templates for Text Generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiang Yu, Agnieszka Falenska, Marina Haid, Ngoc Thang Vu, and Jonas Kuhn. 2019a. IMSurReal: IMS at the Surface Realization Shared Task 2019. In *Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR 2019)*, Msr, pages 50–58, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiang Yu, Agnieszka Falenska, Ngoc Thang Vu, and Jonas Kuhn. 2019b. Head-First Linearization with Tree-Structured Representation. In *Proceedings of the 12th International Conference on Natural Language Generation*, 2018, pages 279–289, Tokyo, Japan. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria DePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Stroudsburg, PA, USA. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating

Structured Queries from Natural Language using Reinforcement Learning. *ArXiV*, abs/1709.0.