Enhancing Lifelog Question-Answering with Retrieval-Augmented Generation and Reasoning Large Language Models

Quang-Linh Tran, B.Sc

A Dissertation submitted in fulfillment of the

requirements for the award of

Doctor of Philosophy (Ph.D.)

to the



Dublin City University

School of Computing

Supervisors

Prof Cathal Gurrin

Prof Gareth J. F. Jones - Maynooth University

August 2026

Declaration

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

Signed: Quang-Linh Tran No.: 22270072 Date: 27/08/2026

Reporting of Gen AI Tool Use

To ensure the regulation of Generative AI use, I provide a report of Gen AI use in this research thesis. The use of Gen AI is as follows:

1. **Tools:** Google Gemini, Flash, used for drafting between 12/11/2025 to 28/02/2026. Google Gemini, 3.5 Flash, used for reviewing between 01/03/2026 to 25/05/2026.

2. **Outputs:** I used Gemini while writing the draft of the thesis to edit wording and fix grammar errors of some of the sentences and/or paragraphs. I write the original version of sentences first, then if it is not clear or the word use is not appropriate, I prompt Gemini to polish the sentences.
   In my second use, I give Gemini a section to check grammar errors and references.

3. **Prompts**: the first prompt I used is: "I am writing an academic report using formal language. Please help me to edit some word choices and fix grammar in the text to ensure it does not have any critical errors."
   The second prompt using when reviewing errors is: "Please help me to review this section by showing any grammar errors or references and suggest edition."

4. **Iterations:** I used the prompt only one time of interacting, use the suggestion from Gemini to edit the text, and do not perform any further iterations of chat.

5. **Critical Reflection:** I only used Gen AI to edit errors in my thesis. Gemini helped me to edit grammar errors, typos, and check references to ensure there is no errors in the thesis. I do not use Gen AI in literature review, result analysis or running experiment.

Acknowledgements

The completion of this dissertation would not have been possible without the guidance, support, and encouragement of numerous people, to whom I owe a deep debt of gratitude.

First and foremost, this journey would not have been possible without the love, patience, and support of my family. I want to thank my parents, my siblings and my partner, who have given me a strong foundation for this journey.

I am grateful to my supervisors, Prof. Cathal Gurrin and Prof. Gareth J. F. Jones, for their mentorship throughout this PhD journey. Their expert guidance, thoughtful feedback, and constant encouragement have been invaluable in shaping both this research and my growth as a researcher. I also want to thank Prof. Binh Nguyen, who guided me in the path of pursuing a PhD.

I would like to acknowledge Dublin City University, the ADAPT Centre (under funding 13/RC/21-06_P2), and the Faculty of Engineering and Computing Accommodation Scholarship for their financial support during the course of my PhD.

I am also deeply thankful to my co-authors and collaborators. Working alongside such talented and committed researchers has been both a privilege and an inspiration to me. To my friends and colleagues in the ADAPT Centre and Dublin city, I want to especially thank you for the countless conversations that made this journey so much more enjoyable.

This dissertation stands as a testament to the collective support and encouragement of all these remarkable individuals, and I am profoundly thankful for the role each of them has played in my academic journey.

List of Publications

During the course of my PhD, I have published several papers in workshops, conferences and journals, presenting my research on lifelog QA and lifelog retrieval. For each research question, I have published at least one paper to present the approach and experiment to address the question. In addition, I also contribute to the field of multimedia QA by co-organising a workshop of AI-powered Question Answering in Multimedia [103, 102]. The list below is my publications:

Research Question 1: Context Retrieval

- Quang-Linh Tran, Ly-Duyen Tran, Binh Nguyen, Gareth J. F. Jones, and Cathal Gurrin. 2025. Multi-modal Context Reranking for Lifelog Question Answering. In Proceedings of the 2025 International Conference on Content-Based Multimedia Indexing (CBMI 2025). IEEE.

- Quang-Linh Tran, Ly-Duyen Tran, Binh Nguyen, and Cathal Gurrin. 2023. MemoriEase: An Interactive Lifelog Retrieval System for LSC'23. In Proceedings of the 6th Annual ACM Lifelog Search Challenge (LSC '23). Association for Computing Machinery, New York, NY, USA, 30–35.

- Quang-Linh Tran, Binh Nguyen, Gareth J. F. Jones, and Cathal Gurrin. 2024. MemoriEase 2.0: A Conversational Lifelog Retrieval System for LSC'24. In Proceedings of the 7th Annual ACM Workshop on the Lifelog Search Challenge (LSC '24). Association for Computing Machinery, New York, NY, USA, 12–17.

- Quang-Linh Tran, Binh Nguyen, Gareth J F Jones, and Cathal Gurrin. 2025. MemoriEase 3.0: A RAG-Enhanced Conversational Lifelog Retrieval System

at LSC'25. In Proceedings of the 8th Annual ACM Workshop on the Lifelog Search Challenge (LSC'25). Association for Computing Machinery, New York, NY, USA, 52–57.

- Quang-Linh Tran, Binh Nguyen, Gareth Jones, Cathal Gurrin. MemoriEase at the NTCIR-17 Lifelog-5 Task. In Proceedings of the 17th NTCIR Conference Evaluation of Information Access Technologies.

- Quang-Linh Tran, Binh Nguyen, Gareth Jones, Cathal Gurrin. DCU MemoriEase at the NTCIR-18 Lifelog-6 Task. In Proceedings of the 18th NTCIR Conference Evaluation of Information Access Technologies.

Research Question 2: Baseline RAG

- Quang-Linh Tran, Ngo Ngoc Diep Pham, Quoc Trung Truong, Minh Hung Nguyen, Hong Cat Le, Dang Khoi Vu, Van Minh Thien Nguyen, Van Kinh Nguyen, Luu Phuong Ngoc Lam Nguyen, Tan Le, Minh Phuc Dang, Binh Nguyen, Gareth J. F. Jones, and Cathal Gurrin. 2025. A RAG Approach for Multi-Modal Open-ended Lifelog Question-Answering. In Proceedings of the 2025 International Conference on Multimedia Retrieval (ICMR '25). Association for Computing Machinery, New York, NY, USA, 1303–1312.

Other publications during the course of my PhD:

- Quang-Linh Tran, Binh Nguyen, Gareth Jones, Cathal Gurrin. (2024). ViEcomRec: A Dataset for Recommendation in Vietnamese E-Commerce. In: Hà, M.H., Zhu, X., Thai, M.T. (eds) Computational Data and Social Networks. CSoNet 2023. Lecture Notes in Computer Science, vol 14479. Springer, Singapore.

- Quang-Linh Tran, Binh Nguyen, Gareth J. F. Jones, and Cathal Gurrin. 2024. MemoriLens: a Low-Cost Lifelog Camera Using Raspberry Pi Zero. In Proceedings of the 2024 International Conference on Multimedia Retrieval

(ICMR '24). Association for Computing Machinery, New York, NY, USA, 1255–1259.592.

- Tai Tan Mai, Quang-Linh Tran, Ly-Duyen Tran, Tu Ninh, Duc-Tien Dang-Nguyen, and Cathal Gurrin. 2024. The First ACM Workshop on AI-Powered Question Answering Systems for Multimedia. In Proceedings of the 2024 International Conference on Multimedia Retrieval (ICMR '24). Association for Computing Machinery, New York, NY, USA, 1328–1329.

- Quang-Linh Tran, Binh Nguyen, Gareth J. F. Jones, and Cathal Gurrin. 2024. MemoriQA: A Question-Answering Lifelog Dataset. In Proceedings of the 1st ACM Workshop on AI-Powered Q&A Systems for Multimedia (AIQAM '24). Association for Computing Machinery, New York, NY, USA, 7–12.

- Thao-Nhu Nguyen, Quang-Linh Tran, Hoang-Bao Le, Binh T. Nguyen, Liting Zhou, Gareth J. F. Jones, Cathal Gurrin. 2024. VidBasys: A User-Friendly Interactive Video Retrieval System for Novice Users in IVR4B. In Proceedings of the 2024 International Conference on Content-Based Multimedia Indexing (CBMI), Reykjavik, Iceland, pp. 1-5.

- Quang-Linh Tran, Binh Nguyen, Gareth J. F. Jones, and Cathal Gurrin. 2025. VideoEase at VBS2025: An Interactive Video Retrieval System. In: Ide, I., et al. MultiMedia Modelling. MMM 2025. Lecture Notes in Computer Science, vol 15524. Springer, Singapore.

- Zhou Liting, Quang-Linh Tran, Hoang Bao Le, Duc-Tien Dang-Nguyen, Graham Healy, ChenYang Lyu, Tianbo Ji et al. 2025. Overview of the NTCIR-18 Lifelog-6 Task. In Proceedings of the 18th NTCIR Conference on Evaluation of Information Access Technologies (NTCIR-18). Tokyo, Japan.

- Luca Rossetto, Werner Bailer, Duc-Tien Dang-Nguyen, Graham Healy, Björn THór Jónsson, Onanong Kongmeesub, Hoang-Bao Le, Stevan Rudinac, Klaus Schöffmann, Florian Spiess, Allie Tran, Minh-Triet Tran, Quang-Linh Tran,

and Cathal Gurrin. 2025. The CASTLE 2024 Dataset: Advancing the Art of Multimodal Understanding. In Proceedings of the 33rd ACM International Conference on Multimedia (MM'25). Association for Computing Machinery, New York, NY, USA, 12629–12635.

- Tai Tan Mai, Allie Tran, Quang-Linh Tran, An Nguyen, Hoang Nguyen, Tho Quan, Duc-Tien Dang-Nguyen, and Cathal Gurrin. 2025. AIQAM'25: The 2nd ACM Workshop on AI-powered Question Answering Systems for Multimedia. In Proceedings of the 33rd ACM International Conference on Multimedia (MM'25). Association for Computing Machinery, New York, NY, USA, 14299–14301.

- Quang-Linh Tran, Hoang-Bao Le, Thang-Long Nguyen-Ho, Graham Healy, Liting Zhou, and Allie Tran. 2025. Extending Lifelog Retrieval to Multistream Video Retrieval at the CASTLE Challenge 2025. In Proceedings of the 33rd ACM International Conference on Multimedia (MM'25). Association for Computing Machinery, New York, NY, USA, 14280–14285.

- Quang-Linh Tran, Hoang-Bao Le, Diep Tuong Nghiem, Binh Nguyen, Gareth Jones, Cathal Gurrin. 2025. OpenLifelogQA: An Open-Ended Multimodal Lifelog Question-Answering Dataset. Information and Communication Technology. SOICT 2025. Communications in Computer and Information Science, vol 2912. Springer, Singapore.

Contents

Table of Contents **xi**

List of Figures **xiv**

List of Tables **xvi**

List of Abbreviations **xvii**

Abstract **xviii**

1 Introduction **1**

1.1 Background 1

1.1.1 Lifelogging 1

1.1.2 Question-Answering on Lifelog Data 5

1.1.3 Large Language Models and Retrieval-Augmented Generation 8

1.2 Motivation and Research Problem 11

1.3 Hypothesis and Research Questions 12

1.4 Contributions 14

1.5 Thesis Outline 15

2 Background and Related Work **17**

2.1 Lifelogging Background 17

2.1.1 Lifelog History 19

2.1.2 Lifelog Retrieval 22

2.1.3 Lifelog Search Challenge 24

2.1.4 NTCIR Lifelog Task 28
2.2 Lifelog Question-Answering 30
2.2.1 Task Definition 30
2.2.2 Benchmark Datasets 33
2.2.3 Methods for lifelog QA 36
2.3 Large Language Models and Retrieval-Augmented Generation 40
2.3.1 Large Language Models 41
2.3.2 Retrieval-Augmented Generation 46
2.3.3 Reasoning LLMs 50
2.4 Reasearch Gaps 54
2.5 Chapter Summary 55

3 Methodology **56**
3.1 Research Methodology 56
3.2 Datasets 59
3.2.1 OpenLifelogQA Dataset 59
3.2.2 DCU-TimelineQA Dataset 65
3.3 Operating Constraints 68
3.4 Evaluation Metrics 69
3.4.1 Retrieval Metrics 69
3.4.2 QA Metrics 70
3.5 Experimental Settings and Justification 73
3.6 Chapter Summary 75

4 Lifelog Context Retrieval **76**
4.1 Introduction and Problem Statement 76
4.2 MemoriEase Retrieval System 78
4.2.1 System Architecture 78
4.2.2 From Challenge Observations to System Updates 86
4.2.3 Performance at Interactive and Automatic Benchmarks 87

4.3 Lifelog Context Reranking 96
4.3.1 Introduction 96
4.3.2 Multimodal reranking model 98
4.3.3 Experiment 100
4.4 Chapter Summary 106

5 Retrieval-Augmented Generation for Lifelog QA 109
5.1 Introduction 109
5.2 RAG approach 112
5.2.1 Retrieval 113
5.2.2 Answer Generation 115
5.3 Experiment 117
5.3.1 Experimental Settings 117
5.3.2 Results 118
5.4 Analysis 121
5.4.1 Performance Across Different Question Types 121
5.4.2 Performance in Different Context Settings 122
5.4.3 Performance on Different Prompts 123
5.4.4 Qualitative Analysis 124
5.5 Chapter Summary 125

6 Reasoning Large Language Models for Lifelog QA 128
6.1 Introduction 129
6.2 Enhancing Reasoning for LifelogQA 130
6.2.1 LLMs 131
6.2.2 Prompt 134
6.2.3 Fine-tuning methods 136
6.3 Experiment 139
6.3.1 Settings 139
6.3.2 Results 140

6.4 Analysis . . . 143
6.4.1 Open-source vs Proprietary Models . . . 143
6.4.2 Fine-tuning Strategies . . . 144
6.4.3 Fine-grained Questions Analysis . . . 146
6.4.4 DCU-TimelineQA Evaluation . . . 147
6.4.5 Ablation Study . . . 149
6.4.6 Error Analysis . . . 150
6.4.7 Human Evaluation Comparison to LLM-as-a-judge . . . 152
6.5 Chapter Summary . . . 154

7 Conclusion and Future Work **156**
7.1 Summary . . . 156
7.2 Contributions . . . 159
7.3 Limitations . . . 160
7.4 Future Works . . . 162

A Additional Experimental Materials **165**
A.1 Prompt for reasoning-enhanced LLMs in RAG . . . 165
A.2 Prompt for LLM-as-a-judge . . . 167
A.3 User Interface of MemoriEase System . . . 168
A.4 Model Cards and Parameter Settings . . . 168

List of Figures

1.1 Development of lifelog cameras from 1998 to 2013. 3
1.2 Some examples of images from a lifelog collection 4

2.1 The MyLifeBits platform store and tools [46] 20
2.2 The SenseCam prototype V2.3 and its viewer application interface [64]. 20
2.3 A prototype retrieval system supports memory retrieval from Yi et al. [22] 22
2.4 A multifaceted interface for lifelog retrieval systems with several search filters [36]. 24
2.5 MyEachtra user interface with suggested answers for QA task [160]. . 28
2.6 An ad hoc topic in the LSAT sub-task and its corresponding answer [191]. 29
2.7 Transformer Encoder-Decoder Architecture [164] 42
2.8 Examples of RAG used in QA from [45] 47
2.9 PPO and GRPO architectures [138] 51

3.1 Diagram of research process, adapted from [81] 57
3.2 Annotation process of the OpenLifelogQA dataset 61
3.3 Annotation interface for OpenLifelogQA dataset 62
3.4 Analysis on questions in OpenLifelogQA dataset 65
3.5 Analysis of question words and types distribution of the DCU-TimelineQA test set 67

4.1 Development of MemoriEase from the first version to the 3.0 version. 77

4.2 MemoriEase overall architecture and flow 79
4.3 Comparison of low and high edge weight of blur and normal images . 79
4.4 RAG pipeline for MemoriEase 3.0 84
4.5 MemoriEase performance in LSC'23 expert track. 88
4.6 MemoriEase performance in LSC'23 QA sub-task. 89
4.7 MemoriEase 2.0 performance in LSC'24. 90
4.8 MemoriEase performance in LSC'25. 91
4.9 Comparing the best runs of the three automatic teams from NTCIR-17 Lifelog-5 Task [190] 93
4.10 Best recall results across all submissions at NTCIR-18 Lifelog6 Task [191] 94
4.11 A two-stage retrieval framework for the lifelog QA task. 97
4.12 The overall architecture of the cross-encoder multi-modal reranking model. 98

5.1 RAG pipeline for lifelog QA 112
5.2 A structure of a prompt for LLMs 115
5.3 Performance of Llama 3.1 (line) and Mistral (dashed line) on Open-LifelogQA under different context settings. 123
5.4 Performance of the Llama 3.1 8B model on OpenLifelogQA under different prompting instructions, measured using the scaled LLM score. 124

6.1 Pipeline for enhancing reasoning LLMs in lifelog QA. 132
6.2 A structure of a prompt for LLMs 135
6.3 Open-source vs Proprietary models in three types of questions. 144
6.4 Comparison of 4 models on specific question words. 148
6.5 LLMScore for 3 types of questions from GRPO fine-tuned Llama 3.1 8B model on different training data settings. 150

A.1 Conversational user interface of the MemoriEase system. 168
A.2 Temporal search interface. 169

A.3 Image-to-Image search interface. 169

List of Tables

2.1 Lifelog QA datasets statistics. 33

3.1 Statistics on three sets of OpenLifelogQA. 63

3.2 Some QA pairs and ground-truth context examples 64

3.3 Statistics on train and test sets of DCU-TimelineQA. 67

4.1 Properties of the Elasticsearch index 81

4.2 User needs and failures observed at each Lifelog Search Challenge, and the corresponding design responses in the following version of MemoriEase. 86

4.3 Performance of different approaches in ad hoc and KIS queries 95

4.4 Reranking experimental results on the OpenLifelogQA test set. 103

4.5 DCU-TimelineQA retrieval results from BM25 and Stella-1.5B. 105

4.6 Error examples of the reranking model 106

5.1 Context retrieval results on the OpenLifelogQA test set. 118

5.2 Experimental result for all-day contexts on the OpenLifelogQA test set 119

5.3 Experimental results of two LLMs for the DCU-TimelineQA test set . 119

5.4 Experimental results for different types of questions in the Llama 3.1 8B model for all-day contexts on OpenLifelogQA. 121

5.5 Some examples of Llama 3.1 predictions 125

6.1 Performance comparison of LLM models on the OpenLifelogQA test set. 141

6.2 Performance of fine-tuned models on different methods and prompt types . . . 145
6.3 Performance of LLMs on aggregation questions in the OpenLifelogQA test set. . . . 147
6.4 Model performance on the DCU-TimelineQA test set. . . . 149
6.5 LLaVA Next Interleave 7B performance on the OpenLifelogQA test set when fine-tuning on different modalities of input contexts . . . 150
6.6 Taxonomy of error types in the predictions of the fine-tuned model on the OpenLifelogQA test set ($n = 170$ incorrect answers). . . . 151
6.7 Retrieved contexts for the examples in Table 6.6 . . . 152
6.8 Agreement between the GPT-4o LLM-as-a-judge metric and two human raters (V1, V2) on $n = 100$ QA pairs per prediction model . . . 153

A.1 Model cards for all models used in this thesis. . . . 170
A.2 Training hyperparameters, with the provenance of each value. . . . 171

List of Abbreviations

Acronym	Description
AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
BLIP	Bootstrapping Language-Image Pre-training
CLIP	Contrastive Language-Image Pre-training
CoT	Chain-of-Thought
DRES	Distributed Retrieval Evaluation Server
GPS	Global Positioning System
GPT	Generative Pre-trained Transformer
GRPO	Group Relative Policy Optimization
KIS	Known-Item Search
LLMs	Large Language Models
LIT	Lifelog Insight Task
LSAT	Lifelog Semantic Access Task
LSC	Lifelog Search Challenge
LQAT	Lifelog Question-Answering Task
NLP	Natural Language Processing
NTCIR	NII Testbeds and Community for Information Access Research
PPO	Proximal Policy Optimization
PoV	Point-of-View
QA	Question-Answering
RLHF	Reinforcement Learning from Human Feedback
SFT	Supervised Fine-tuning
TF-IDF	Term Frequency-Inverse Document Frequency
UI/UX	User Interface/User Experience
ViT	Vision Transformer
VQA	Visual Question-Answering

Enhancing Lifelog Question-Answering with Retrieval-Augmented Generation and Reasoning Large Language Models

Quang-Linh Tran

Abstract

Lifelogging refers to the process of passively collecting, storing, and analysing an individual's daily life data from wearable cameras, sensors, and mobile devices. Question-answering (QA) on lifelog data involves posing questions on lifelog data and generating answers in natural language. Lifelog QA is a compelling yet challenging problem due to the extensive volume and the multimodal and temporal nature of lifelogging. Addressing this task requires a retriever to find relevant information within the lifelog collection before generating answers. With recent advancements in large language models (LLMs), their capabilities in QA tasks show potential to advance lifelog understanding. In addition, with their reasoning capability, LLMs can solve the aggregation question in lifelogging that requires contextual understanding and reasoning. This study investigates the lifelog QA task with Retrieval-Augmented Generation and reasoning LLMs by proposing an end-to-end lifelog QA system. To achieve this, I first introduce a lifelog retrieval system to retrieve relevant information necessary for answering questions. A multi-modal reranking model is also developed to further improve the retrieval performance for lifelog QA. Subsequently, a RAG pipeline is developed to analyse the context from the retriever and generate the answers to the questions. Finally, reasoning LLMs for lifelog QA are also investigated to address complex aggregation questions. In summary, this thesis primarily contributes to the field with a QA system using RAG and reasoning LLMs to address the lifelog QA task.

Chapter 1

Introduction

This chapter introduces the foundation of lifelogging, from concept to characteristics and tasks. Building on this, I present the lifelog QA task in Section 1.1. Then I describe the research problem and the motivation for this research in Section 1.2. The research hypothesis and research questions for the research are also introduced in Section 1.3. Section 1.4 describes some contributions of this research. Finally, the structure of this thesis is presented to provide a clear overview of the organisation and flow of the thesis.

1.1 Background

This section provides background information about this study, ranging from the definition of lifelog to the lifelog data and its tasks with applications.

1.1.1 Lifelogging

Reflective questions about one's own life may arise naturally with questions such as "How long did I exercise last week?" or "When was the last time I visited my mom?". These questions ask for information about past experiences and provide insights into life. Based on the question, one can determine how active they were in the last week and plan for more exercise in the next week. There are several ways to answer these questions, such as opening the health app on the phone to

check exercise details, or trying to recall all the exercise events and performing the summation. However, this is challenging for the brain to remember all these details [107]. One may have a faded memory of exercise sessions from yesterday or several days ago, but not very detailed information, such as the duration of each exercise session. From this limitation, the idea of logging daily life through digital technology to provide a digital memory has become a reality.

Lifelogging is the practice of automatically collecting, storing, and analysing personal daily life data from sensors and wearable devices [50]. The concept of lifelogging is not novel. It dates back to 1945, when Vannevar Bush described the "Memex", a device storing everything a person records and consults as a second brain [16], and was first realised at scale by the MyLifeBits project [47]. Section 2.1.1 provides more information about lifelog history and these early works.

Lifelog data

Lifelog data is multi-modal, which includes images, tabular data from Global Positioning System (GPS), and health metrics. To collect data for lifelogging, multiple sensors and devices can be used, such as wearable cameras, smartwatches, and phones. Wearable cameras such as SenseCam [64] and Memento Narrative Clip[1] worn on the chest can capture point-of-view (PoV) images automatically. In addition, a smartwatch can be used to track health-related metrics, and a phone can be used to collect location data through GPS. With the increasing popularity of the internet and social media, digital footprints can also be collected, including emails, songs, messages, and activities on social media. The person wearing these devices and capturing the data is known as a lifelogger.

Wearable cameras have evolved dramatically, from a fist-sized camera to a small, square, wristwatch-sized device. Figure 1.1 shows the size change of lifelog cameras from 1998 to 2013 to illustrate the development of these cameras over time. Wearable cameras can automatically capture images continuously with a short time gap to

[1] https://getnarrative.com/



Figure 1.1: Development of lifelog cameras from 1998 to 2013. From left to right: Mann (1998); Microsoft SenseCam (2004); Mann, Fung, and Lo (2006); Memoto (2013). Source: Wikiwand

balance accuracy and efficiency. The cameras capture images over time to save a life trace of what lifeloggers experience. However, when capturing at a high rate, they create a high burden for storing images and power consumption. To address this problem, capturing approximately 2 images per minute, it can reduce the cost of storage while still capturing the meaning of events in lifelogging and saving battery [64]. The PoV images are slightly different from normal images captured from phones. While lifelog images provide a trace of life through continuous capturing, in which they capture all events, from memorable to unnoticed ones, images captured from phones only save intended and important events. In addition, as lifelog images are automatically captured, the quality is not comparable to normal images [151]. When the lifelogger is in motion, the camera is not stable, resulting in blurry images. Although lifelog images provide valuable insights into life, the challenge of processing lifelog images effectively to make them usable still remains.

Smartwatches, such as the Apple Watch[2], can collect health metrics, including heart rate, saturation of peripheral oxygen (SpO2), sleep quality, and exercise, which enriches the lifelog data. Smartphones can be used to collect the location of lifeloggers through the GPS, the activity or online footprint from the internet and social media. All these data sources are time-synchronised and saved in digital storage to create a multimodal lifelog collection [50]. Figure 1.2 shows some PoV images

[2]https://www.apple.com/watch/



Figure 1.2: Some examples of images from a lifelog collection

captured from the camera.

Tasks and applications

From the collected lifelog data, several tasks and applications can be proposed to utilise lifelog data. Analysing daily activities from lifelog data provides insights into personal life [184, 63]. This task detects predefined activities from the lifelog images, such as eating, watching TV, and using a phone, similar to an action recognition task in the computer vision field. From the detected activity, a comprehensive view of the activity plot in a chart shows the habits and lifestyles. Another important task is lifelog image retrieval [56, 173, 179], which involves finding an event in a person's lifelog from a query. Lifeloggers may need to find something from the past, such as the location of a misplaced item. This task helps lifeloggers to find missing information and then enhance their memory [84, 77]. Lifeloggers do not need to remember an event in full; a few contextual hints are enough to guide the search. This task may have important applications for individuals with memory problems or those with Alzheimer's disease [163].

For efficient and accurate retrieval, several subtasks are required, including lifelog event segmentation and removing low-quality images. Lifelog event segmentation divides the stream of lifelog images into events that represent meaningful activities, such as eating, driving, or broader activities like a trip to Vietnam. Current methods [34, 49] combine both image content and tabular data, which includes location and time, to segment events. After segmentation, the number of retrievable images reduces, helping with fast retrieval and low storage requirements. Because lifelog

images are captured automatically, this can result in many blurred or degraded images when the user is in motion. Removing these images helps to reduce storage requirements. Beyond retrieval, other tasks like activity recommendation [82] suggest activities to do based on lifelog data. This task utilises lifeloggers' past behaviours and contextual signals, such as time, location, and social setting, to suggest meaningful and personalised actions that enhance daily productivity and a healthy lifestyle.

The most visible applications of lifelog in daily life are for memory enhancement [77, 84]. Lifelog data helps lifeloggers retrieve their memories [23]. Even if they do not know what to save now, they can use the lifelog data when they need it. This is helpful for people with memory problems [163]. It is different from photos captured intentionally with phones. Images from phones can capture more meaningful events by our intention, but some events that may have value later are still not intentionally captured. Sometimes, when we want to reminisce about past events, lifelog data is the most truthful source of information we can consult. Overall, lifelogging shows a reliable source for memory augmentation and enhancement.

For health, lifelog data can be used as a monitoring tool for doctors and patients. Meaningful activities, such as eating, drinking, walking, and exercising, which are extracted from the daily activity recognition task [6, 15, 92], will reflect the lifestyle and health conditions of patients. Lifelog data, including health metrics collected from smartwatches, can be used for health analysis and early detection of diseases [26]. Current AI models already analyse health metrics to suggest personalised recommendations for exercise, diet, and stress management [137]. From these applications, lifelog data serve as a powerful foundation and source for better lifestyles and higher well-being.

1.1.2 Question-Answering on Lifelog Data

A relatively new but important application of lifelog data is question-answering (QA). QA systems extend lifelog retrieval by not only finding past events but also

providing answers to specific questions. QA is a well-established problem in Natural Language Processing (NLP), though it still remains challenging [18, 5, 25]. The objective of QA is to generate answers to natural language questions. A newer branch of QA is Visual Question Answering (VQA) [7], which requires generating answers to questions related to images. Advances in multi-modal large language models (LLMs) have addressed this task [169]. Video Question Answering (VideoQA) [187] is a more advanced task compared to VQA, involving questions related to videos rather than a single image. This task requires the QA model to have both temporal and spatial understanding of the video to answer correctly. Beyond these general classifications of QA, there are also QA for specific applications such as medical, legal, business, etc., and lifelog is one of these applications.

Definition and classification

Lifelog QA is a QA task for the lifelog domain that involves asking questions about the life of a lifelogger and providing answers based on the lifelog data [156]. The questions and answers help lifeloggers learn about their life through the ask-answer paradigm. Lifelog QA can provide valuable insights into a lifelogger's daily life and habits. Several types of questions can be asked to get the information [146]. Factoid questions can verify the factual information of life events. A factoid question example is "Did I have dinner at Jackie's restaurant last week?", which confirms the event of having dinner at Jackie's restaurant. Other questions that only ask about one event in lifelog data can be classified as atomic questions [146]. The temporal nature of lifelog poses the need for temporal questions, in which lifeloggers want to know information about the next or the previous events of a known event. For example, "Where did I go after leaving Jackie's restaurant?" asks about the next place the lifelogger went after leaving a known event of leaving Jackie's restaurant. Aggregation questions, which relate to calculating the time duration or counting events, provide insights into the lifelogger's lifestyle and ad hoc information. An example question like "How many times did I go shopping last week?" can give

lifeloggers information about their shopping habits. These questions help to explore various aspects of a lifelogger's life through an ask-answer paradigm, but they are also difficult to address, especially the aggregation questions. Lifelog QA is related to Visual QA [7] and Video QA [187], but differs from both in its multi-modal data, its egocentric viewpoint, and its very low capture rate of roughly two images per minute. Section 2.2.1 discusses more on these differences.

Applications and challenges

Lifelog QA has a wide range of applications for lifeloggers, from answering questions about past events to providing insights into habits and lifestyle. Extending the lifelog applications to store and enhance memory with not just the retrieved content from the query in the lifelog retrieval task, lifelog QA provides more specific answers to questions on lifelog data. Depending on the questions, lifelog QA can provide a wide range of insights. With the classification of questions mentioned previously, atomic questions help lifeloggers recall specific events from daily life and monitor personal activities, such as drinking water and exercising. Temporal questions share many common applications, but in a more specific context, such as those involving previous and subsequent events, to ensure that the correct events are retrieved. Aggregation questions provide a deeper analysis of lifestyle and habits, help to analyse and monitor health and productivity.

Along with the huge potential application of lifelog QA, several challenges need to be addressed to create an effective lifelog QA system. Firstly, lifelog data is multi-modal, comprising not only lifelog images but also time, location, and other information, as seen in the Lifelog Search Challenge (LSC) dataset [56]. Combining them to provide a knowledge base for question-answering is challenging in ensuring a consistent and time-aligned data source. Secondly, with the vast amount of lifelog data, lifelog QA models need a retrieval model to efficiently find relevant information for answering questions. Thirdly, aggregating information from several events for aggregation questions requires complicated data manipulation and calculation with

tools or in a database, as in previous work [146]. Although lifelog QA shares some similar characteristics with VQA and VideoQA, especially in the context of egocentric videos, it differs in the long-range temporal nature of lifelogging and the distinct use cases of lifelog QA and VideoQA. These challenges inspire me to research an accurate and efficient lifelog QA model/system that can effectively integrate lifelog data and provide accurate, meaningful answers to lifelog questions.

1.1.3 Large Language Models and Retrieval-Augmented Generation

In recent years, the development of LLMs, especially several chatbots such as ChatGPT, Gemini, and Claude, has gained public attention and opened a new era of LLMs. These chatbots can understand users' input and generate human-like texts in response [113, 149]. This event has triggered a wave of LLM development and has achieved substantial success in many applications. Basically, LLMs are transformer-based neural network models that have billions of parameters and are trained on massive amounts of text data [186]. They receive a textual input, which is a prompt and generate textual output to address the request in the prompt. The rapid development with a large number of new training data, training recipes, and models released frequently is driven by both industry and research communities, such as GPT-4 from OpenAI [111], Gemini from Google [149], and DeepSeek [95]. Thanks to the vast amount of training data and huge number of parameters, LLMs can generate text to address numerous natural language problems, such as QA [132], language understanding [62], and code generation [21]. Beyond basic capabilities such as language comprehension, world knowledge, and multilingual capabilities, LLMs possess several emerging capabilities that are only observed once they cross specific parameter thresholds [167]. A key example is in-context learning, which helps models to perform novel tasks that were not present in their original training data with only a few examples in the prompt.

LLMs have proven their capability in text generation, but for logical tasks, such

as mathematics or coding, they need to have a step-by-step process to solve these tasks, and the term reasoning is used for this capability. Reasoning is an important emerging capability which helps LLMs to generate intermediate rationales before giving the final answer [166]. Reasoning LLMs are a more advanced version of standard LLMs because standard LLMs just directly generate the output, while reasoning LLMs generate intermediate rationales to decompose the problem into logical steps and solve it before aggregating information to generate the final answer. Reasoning LLMs also reflect the previously generated rationales, examine their correctness, and fix them. This capability helps reasoning LLMs to provide better output with more accurate results, especially on tasks that require complex logic, such as mathematics, coding, and date and time processing. An example of reasoning for date processing from [166] is: "**Q:** The concert was scheduled to be on 06/01/1943, but was delayed by one day to today. What is the date 10 days ago in MM/DD/YYYY? **A:** One day after 06/01/1943 is 06/02/1943, so today is 06/02/1943. 10 days before today is 05/23/1943. So the answer is 05/23/1943.". The intermediate text of solving steps of date subtraction helps produce more accurate answers. Advanced LLMs fundamentally attain this capability, but require instructions like Chain-of-Thought prompting [166], a way of prompting which adds "thinking step-by-step" to the prompt to foster LLMs to do reasoning. Later reasoning LLMs focus more on using reinforcement learning, such as GRPO [138], to improve the reasoning capability. Date and time processing is essential for answering lifelog questions such as "How long was my last exercise?". To solve this, the model must first identify the start and end times of the session and then compute the duration. Additionally, time operates on a base-60 system rather than base-100 like standard arithmetic, which requires the model to handle calculations with a different numerical structure. This is a simple example of cases when reasoning LLMs are needed for the lifelog QA task.

Despite many powerful capabilities, LLMs still possess several drawbacks. Firstly, the training data of LLMs is static, which hinders the update capability of LLMs.

For example, if asked about the latest stock prices or recent market trends in 2026, the model may provide outdated or inaccurate information because such data was not available during training. Secondly, although the training data covers a wide range of topics, the personal lifelog data is private, so it cannot be used to train the models. Because of this, when asking a lifelog question on LLMs without additional information, such as "How long was my last exercise?", the model will output "Sorry, but I cannot answer this question because I do not have your information.". Finally, when being asked about uncertain information, LLMs can generate hallucinated output, which is fluent, plausible-sounding, but factually incorrect or unsupported content [65]. To address these limitations, a retrieval approach to support LLMs is proposed, which is Retrieval-Augmented Generation.

Retrieval-Augmented Generation (RAG) [87] is a powerful technique that enhances LLMs' knowledge by incorporating external, up-to-date, and specialised knowledge sources such as lifelog, medical, and law. A basic RAG architecture includes a two-stage process. In the first stage, it retrieves relevant information for the question/query from the knowledge base, such as a database or the internet. In the second stage, the retrieved information is incorporated into the prompt, and LLMs generate answers based on that information. This approach enables LLMs to access relevant knowledge about the question, rather than relying on parametric knowledge and thereby avoiding hallucination. As lifelog data is personal data which is not used to train any LLM, RAG is a good candidate approach for lifelog QA to integrate lifelog data into the prompt to ask for an answer.

In this research, I examine the RAG method with novel LLMs, especially reasoning LLMs, to address lifelog-related questions and focus on aggregation questions, which are complicated and have the most useful use case. With the vast and diverse amount of lifelog data, retrieving the correct, relevant events for questions is also a challenge. I investigate both stages of RAG, from retrieving contexts for questions to generating correct answers.

1.2 Motivation and Research Problem

The development of digital devices and storage technology has made continuous self-documentation increasingly accessible, and lifelog applications show great potential to transform how people recall, reflect on, and reason about their daily lives. As wearable sensors, smartphones, and other devices become ubiquitous, the volume and richness of personal lifelog data will only grow, making intelligent lifelog applications an important and timely research direction.

Question-answering is a challenging task, especially when applied to the lifelog domain. With the vast amount of lifelog data, automatically retrieving correct events is already difficult. In addition, analysing the retrieved contexts to create answers for questions poses another challenge in the lifelog QA task. Although several previous studies have explored lifelog QA [146, 154], the previous research still exhibits notable limitations. They motivate me to advance the research of lifelog QA further. In this work, I aim to address the lifelog QA task by investigating the use of the RAG system in retrieving contexts and answering lifelog-related questions.

With the rapid growth of LLMs, they are used everywhere and transform our lives by helping us with various problems. Therefore, utilising LLM techniques for lifelog QA is a timely, robust, and largely unexplored topic that I aim to investigate in this research. Using a very large model to generate answers for simple, atomic questions with clear evidence seems like "using a sledgehammer to crack a nut", but lifelogging is challenging enough to explore how LLMs can work with this data. In addition, LLMs still exhibit some weaknesses in logic problems, such as performing mathematical operations or comparing floating-point numbers, which require external tools or additional reasoning to solve. Aggregation questions require reasoning on relevant events, calculation, and counting, which can pose challenges to LLMs. This problem motivates me to explore a reasoning LLM for lifelog QA. To address these challenges, the next section presents the research hypotheses and key questions that guide this study.

1.3 Hypothesis and Research Questions

In order to achieve the research goal of investigating an approach to address the lifelog QA task with the RAG system and reasoning LLMs, this dissertation aims to validate the following hypothesis:

Research Hypothesis: *Reasoning LLMs in a RAG framework improve the accuracy of the open-ended lifelog question answering task compared to standard LLMs.*

To make this hypothesis measurable, reasoning LLMs are models that generate intermediate reasoning steps before answering, either by post-training or through the chain-of-thought. Standard LLMs are instruction-tuned models of comparable size that answer directly. Both of them are in the RAG framework under identical retrieved contexts and prompts, so that any difference is attributable to generation rather than to retrieval or model scale. Accuracy is measured by conventional and novel metrics for open-ended QA, which will be presented in Section 3.4. The research hypothesis is supported only if reasoning models outperform their matched counterparts on all types of questions, especially aggregation questions, where multi-step counting and duration arithmetic are required. To examine the research hypothesis, I propose three research questions to guide the research process as follows:

Research Question 1: What is the most effective context retrieval method to support the LifelogQA task when compared to standard retrieval approaches?

This question aims to investigate the method of lifelog context retrievers to find information to answer a question. The lifelog contexts are textual information about time, location, and descriptions of lifelog events. A lifelog event is a sequence of lifelog images that indicate a single activity of the lifeloggers. To achieve this, I conduct an experiment on different approaches to retrieving lifelog data to find the context for lifelog questions. Several approaches are experimented with, including BM25, dense retrieval, and a cross-encoder for reranking the context. The accuracy of the retrieved context is measured by various metrics, such as Recall and Precision

at different K values. Based on the evaluation, I selected the most effective approach to establish context for lifelog QA.

Both interactive and automatic models are examined for retrieving context. I developed the MemoriEase retrieval system, an interactive system that retrieves lifelog events. The system develops from a normal text-to-image embedding-based search engine to a conversational and RAG-based retrieval system. I also propose a multimodal reranking model for lifelog context retrieval for QA. This model contributes to RAG to improve the accuracy of retrieved contexts.

Research Question 2: How effective is RAG when generating answers for LifelogQA from retrieved contexts?

This research question focuses on constructing a RAG system for lifelog QA. Integrating retrieved contexts and generating answers from them is investigated to examine how RAG works for lifelog QA. Firstly, a lifelog QA dataset is constructed for model development and evaluation purposes. After that, several lifelog QA models are developed from small and large language models. The accuracy is evaluated using conventional metrics, such as BERTScore, ROUGE-L, and LLM-as-a-judge. To further explore the topic of RAG for lifelog QA, I propose two sub-questions as follows:

RQ2.1: To what level of accuracy can LLMs in RAG answer open-ended lifelog questions from retrieved context? This question evaluates the overall performance of LLMs on lifelog QA. It compares small and large language models and examines correct and incorrect answers. I conduct an error analysis and investigation into the causes of the errors.

RQ2.2: How does that accuracy decompose across the atomic, temporal and aggregation question types of OpenLifelogQA? This question aims to evaluate the performance of RAG with different LLMs in different types of questions, including atomic, temporal, and aggregation questions. Through fine-grained analysis, the strengths and weaknesses of LLMs are examined for each type of question.

Research Question 3: Can LLMs with reasoning capability improve the accuracy of answers for aggregation questions in lifelog QA compared to standard LLM baselines in RQ2?

This research question focuses more deeply on the QA task, especially the aggregation question. This type of question is more difficult than atomic and temporal questions because it requires a reasoning capability to find the answer. To answer this research question, I aim to employ several techniques from both inference-based and training-based approaches, such as chain-of-thought prompting and reinforcement learning fine-tuning, to exploit and improve the reasoning capability of LLMs for lifelog QA. An evaluation of several metrics is carried out to measure the improvement. Several analyses on different aspects of reasoning LLMs for lifelog QA are also performed to confirm the answer to this research question.

1.4 Contributions

In this section, I list the contributions of my research on lifelog QA as follows:

- I developed a lifelog QA dataset called OpenLifelogQA[3], which contains 14,187 question–answer pairs generated from the 18-month LSC lifelog collection. This benchmark dataset opens up research on generative answers for lifelog questions, focusing on aggregation questions that present challenges to resolve.

- A RAG approach is proposed for lifelog QA that leverages semantic search and LLMs to answer lifelog-related questions. The experiment demonstrates a substantial improvement over keyword-based retrieval and small language models. The analysis also points out the weaknesses of standard LLMs in aggregation questions and opens opportunities for work on this type of question.

- To improve the retrieval engine in the RAG, I developed a multi-modal reranking model, which reranks the list of retrieved contexts to a better list with

[3]In collaboration with ASIA Lab, University of Science, Vietnam

relevant contexts on top of the list. The model combines both text and images from lifelog data to rerank, demonstrating an improvement over baseline models.

- I propose reasoning lifelog LLMs with advanced techniques on supervised and reinforcement learning. The lifelog LLM achieves a big improvement compared to previous baselines and a comparable performance to the state-of-the-art closed-source model from OpenAI. It can accurately generate answers for aggregation questions, which the vanilla LLMs from the RAG fail to answer. A detailed analysis and discussion are presented to demonstrate the capabilities of the lifelog LLM.

- MemoriEase, an interactive lifelog retrieval system, was built in the course of this PhD. It has developed from a standard embedding-based retrieval system to a conversational and RAG-enhanced system. I then integrated a lifelog LLM into this system and turn it into an assistant system that can not only find relevant lifelog events but also generate answers to questions and summarise lifelog data. It is evaluated on an interactive benchmark, the Lifelog Search Challenge, and achieved a remarkable performance with top-3 in the latest LSC 2025.

1.5 Thesis Outline

This thesis presents my research on an RAG approach for lifelog QA, from the retrieval model to answer generation. In this chapter, I have already described the concept of lifelogging and the lifelog QA tasks. I also present some previous works and the research motivation for me to do this research. The contribution and list of publications are shown in the previous sections. The structure of other chapters in this thesis is as follows:

- **Chapter 2: Background and Related Work** presents the background information on lifelogging, from its history to the present, including tasks

such as lifelog retrieval and lifelog search challenges. Related work on lifelog QA, LLMs and RAG is also discussed in the later sections of this chapter.

- **Chapter 3: Methodology** discusses the research methods I used to carry out this research. I also present the datasets used in the experiment for lifelog QA, DCU-TimelineQA and OpenLifelogQA. Some constraints and limitations are introduced in this chapter. Finally, evaluation metrics for lifelog QA are discussed in the final section.

- **Chapter 4: Lifelog Context Retrieval** describes the first stage of lifelog QA, in which it retrieves relevant information to answer questions. I present the MemoriEase system, an interactive lifelog retrieval system to find events from a query. A reranking model for automatic retrieval is also presented, which combines multi-modal lifelog data to rerank the list of lifelog events relevant to questions.

- **Chapter 5: RAG for lifelog QA** presents a vanilla RAG approach for lifelog QA. The proposed methods combine semantic search and baseline LLMs for answering lifelog questions. The analysis of performance by question types shows the drawbacks of LLMs in aggregation questions.

- **Chapter 6: Reasoning LLMs for lifelog QA** describes a fine-tuned, reasoning-enhanced LLM specialised for lifelog questions. The approach of fine-tuning and improving the LLMs is described in the methodology section. The experiment and analysis show superior performance compared to standard LLMs from Chapter 5.

- **Chapter 7: Conclusion and Future Work** concludes the thesis by examining the research hypothesis and answering research questions. The contribution and limitations of the research are also discussed. Future work on this topic is presented.

Chapter 2

Background and Related Work

In this chapter, I provide background on lifelogging and related work in lifelog retrieval and question-answering. Section 2.1 describes the concept of lifelogging, the characteristics of lifelog data, and some tasks and applications. Section 2.2 describes the question-answering task in lifelog, previous research on this topic, and discusses the gap in research for lifelog QA. Section 2.3 provides information on LLMs and RAG, especially on reasoning LLMs and their potential in addressing the lifelog QA problem.

2.1 Lifelogging Background

According to Gurrin et al. [50], lifelogging refers to the passive collection, processing, and reflection of an individual's daily life through the use of wearable sensing technologies. Although some researchers propose a stricter definition, most definitions emphasise the central role of wearable devices in continuously capturing and documenting aspects of a person's life [27, 13]. Wearable devices include cameras, smartwatches, and microphones, used to collect lifelog data; however, the most popular devices are wearable cameras. As introduced in previous sections, the data in a lifelog encompasses images/videos, health metrics, location, audio, and other relevant information. All of these data sources unify to create a lifelog collection.

The primary motivation for wearing these devices and collecting the data is to

augment memory. As the well-known Ebbinghaus forgetting curve [107] shows, our brain forgets quickly and follows an exponential decay pattern. It is essential to save memories by other means, rather than relying on the brain, and lifelogging is a good choice for that. Lifelog data is also used for personal analytics and well-being management by analysing the data.

Gurrin et al. [50] defined lifelogging as the large-scale collection of data, acquiring specific characteristics to form personal big data. There are five main characteristics of big data [67], known as the 5Vs, including volume, velocity, variety, veracity, and value. The SenseCam wearable camera [64] captures images every 5–30 seconds and can operate for 12–24 hours. In typical use (1 image every 30 seconds over 16 active hours), it produces about 2,000 images per day. Our MemoriLens camera [161] uses the same capture rate but records higher-resolution images (Full HD–4K), resulting in image sizes of 2–6 MB per image. This amounts to roughly 4 GB of visual data per day. When combining data with other wearable sensors, lifelog collection can require substantial data volumes. Lifelog data is captured daily, 16 hours a day, 7 days a week, and encompasses multi-modality data, including visual data from lifelog images, tabular data from health metrics, and audio and location information. These are the velocity and variety characteristics of big data. The veracity or reliability of data is a crucial characteristic that indicates the accuracy of the data. Lifelog data, being automatically captured in everyday life, provides an unbiased and dependable source of information. The value of lifelog data is enormous, and it has been applied in many domains. From a healthcare perspective, lifelog data is used to monitor lifestyle [79, 6], activity recommendation [82], and early detection of diseases [26, 83]. For memory augmentation, lifelog data helps users preserve memories and recall meaningful events [77].

The evolution of digital cameras to replace photographic film cameras has changed the way images are stored. Camera sensors can now be as small as a few centimetres, allowing an entire camera with a battery to be worn. Recent wearable cameras, such as those described in [161, 128], are typically built on compact platforms like the

ESP32-Cam or Raspberry Pi Zero[1]. These devices are paired with high-resolution camera modules (Omnivision sensors or Raspberry Pi camera module) to capture Full HD images while maintaining low power consumption. Thanks to wearable cameras, lifelogging has evolved from a text-based format, similar to a diary, to a visual-based approach with point-of-view (PoV) images [53].

2.1.1 Lifelog History

Early Works

The concept of lifelogging dates back to 1945, when Vannevar Bush described a personal information system called "Memex" in the article "As We May Think" [16]. He described the Memex in the terms "Consider a future device ... in which an individual stores all his books, records, and communications, and which is mechanised so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.". This system showed a vision for logging everything in our lives to digital devices and assisting us in memory augmentation.

However, the development of lifelog did not become an active area of research until Gordon Bell proposed a research project, MyLifeBits, in 2001 [47, 46]. The project aimed to create a comprehensive digital storage of all personal information, including documents, photos, and interactions, for an individual. Figure 2.1 shows the platform for storing data and tools to capture multiple data sources. The goal of the platform was to capture a lifetime's worth of data and provide retrieval and analysis capabilities, demonstrating the potential of digital memory archives.

Microsoft SenseCam [64] was one of the first wearable cameras, introduced in 2006. Figure 2.2 shows the prototype and application of SenseCam. The camera features a microcontroller with 128KB of flash memory, a 2GB SD Card for storage, and a VGA camera module, powered by a 980mAh battery. This allows for up to 24 hours of continuous capture, with one image captured every 30 seconds. The accompanying application provides users with an interface to view, playback, and

[1]https://www.raspberrypi.com/

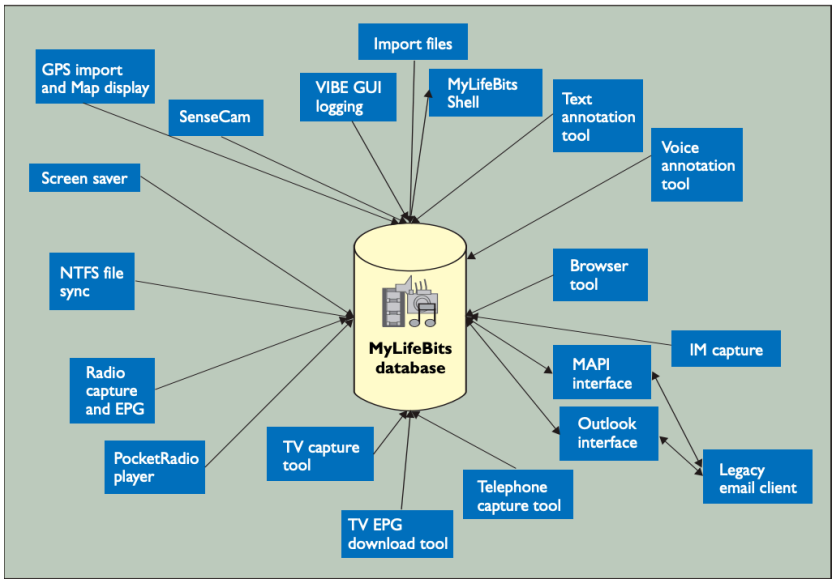


Figure 2.1: The MyLifeBits platform store and tools [46]

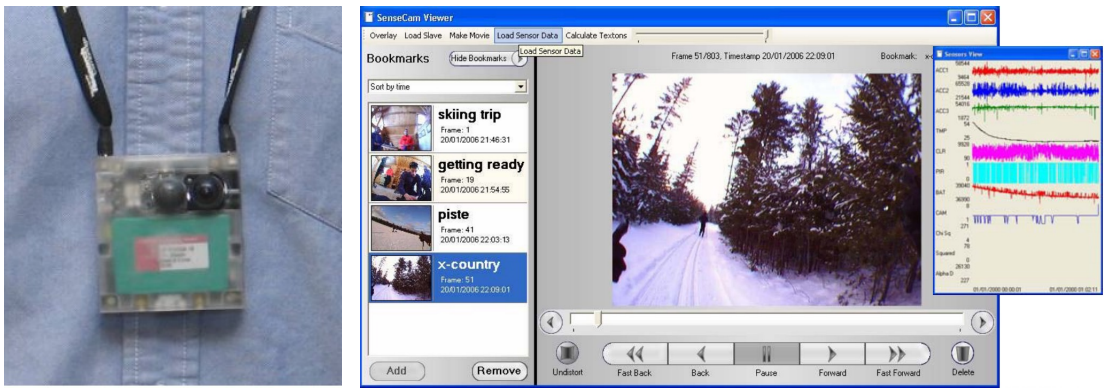


Figure 2.2: The SenseCam prototype V2.3 and its viewer application interface [64].

annotate captions for events in the captured images. This work demonstrates the feasibility of lifelogging with wearable cameras and the applications where lifelog data are being utilised.

With advancements in wearable devices and sensors, as well as the decreasing cost of storage, the development of lifelogging has experienced notable growth. Around 2010, researchers worldwide in the field of lifelogging applied various techniques from human-computer interaction, data mining, and other fields to explore the potential of lifelog data in everyday life. Zhou et al. [189] proposed a new lifelogging browser system for logging data and sharing with groups. To enhance the retrieval of lifelog data, Qiu et al. [119] mined user activity as a context source

to improve retrieval. Several problems like event segmentation, meaningful lifelog detection, content analysis, and activity recognition [118, 82, 34, 126, 43] have been investigated.

Lifelog Research in DCU

Lifelogging has also been an active research area at Dublin City University (DCU) for the past 20 years, with many researchers conducting research on most of the applications of lifelogging. Early works focused on lifelog collecting and organising data to build lifelog collections [52, 59]. Other foundational tasks like keyframe detection [12, 37], activity recognition [38, 17], lifelog visualization and interaction [175, 41], ethics and privacy consideration [51, 68] were also investigated. Several lifelog collections have been constructed, such as the LSC 2018 dataset with over 41K images spanning 27 days [52] or the first NTCIR Lifelog Test Collection [59] with over 88K images. Doherty et al. [37] investigated methods for keyframe selection using MPEG-7 descriptors to measure similarity between images and picking a threshold to discriminate events. From segmented events, recognition tasks can be performed, as seen in research by Byrne et al. In their studies [17], they employ a Support Vector Machine (SVM) model to classify each lifelog image into a concept, such as Indoors, Screen, Meeting, etc. The way of interacting with lifelog data was also studied with different visualisations for different platforms. In the study [175], Yang et al. sketched a series of interactive visualisations and their corresponding usage scenarios for three different interactive platforms, including smartphones, tablets, and desktops. Yi et al. [22] also studied lifelog retrieval based on episodic contexts and proposed a prototype of a retrieval system that adopts a memory-oriented and user-centred approach. The front-end of the system can be seen in Figure 2.3. These works laid the foundation for lifelog studies at DCU and in the community.

In recent years, the task of visual lifelog retrieval has gained popularity as a means to address the problem of finding lifelog events. One of the earliest studies for lifelog retrieval in DCU by Doherty et al. [35] proposed using content-based

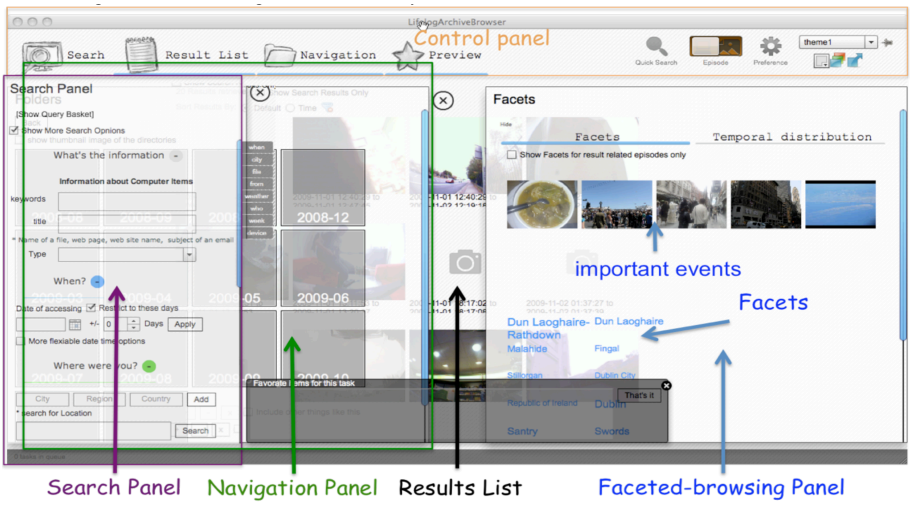


Figure 2.3: A prototype retrieval system supports memory retrieval from Yi et al. [22]

features from images to retrieve lifelog data. Especially after several challenges and workshops organised by researchers in DCU, such as NTCIR Lifelog [60], Lifelog Search Challenge [58], and ImageCLEF Lifelog [30], research on lifelog retrieval has become more and more popular. Details about the lifelog retrieval research are described in the following section.

2.1.2 Lifelog Retrieval

Lifelog retrieval is the task of searching lifelog data from a collection for a query. This task is an application of the information retrieval field, where the information to be found is lifelog data. With visual data now dominating lifelogging, the lifelog retrieval task can be classified as a text-to-image retrieval task. The input of this task can be a textual query to describe the information needs, such as "Find me drinking water" or "Find images when I eat a burger.". From the query, a search engine scans through the lifelog collection to find the most relevant image to the query.

Methods for lifelog retrieval

The technique for finding an image from a textual query has developed extensively, from concept-based retrieval to embedding-based retrieval. Concept-based retrieval

involves extracting visual concepts by object detection models or generating captions from lifelog images. Each image may contain several visual concepts, which are then matched with the query using various techniques, such as word matching methods like TF-IDF [125] or BM25 [127]. Myscéal [158] is a retrieval system employing this approach and improves the TF-IDF to aTF-IDF (area TF-IDF). One major drawback of this approach is that it depends on the pre-detected concepts to perform retrieval. If the visual concept is not detected, it leads to a mismatch between the query and concepts, resulting in failure to retrieve the correct images. A newer approach, embedding-based retrieval, has improved retrieval performance by utilising latent embeddings between text and images. A typical approach to embedding-based retrieval is to use a pre-trained vision-language model, such as CLIP [121] or BLIP [91], to encode both textual queries and images into vector embeddings. Thanks to the training strategy that minimises the difference between positive text-image pairs, the model can encode meaningful embeddings with high similarity for queries that match relevant images. Cosine similarity is commonly used to measure the relevance of the query and images with the following formula, where i is the embedding of an image and q is the embedding of a query:

$$\cos(i, q) = \frac{i \cdot q}{\|i\| \, \|q\|} = \frac{\sum_{j=1}^{n} i_j q_j}{\sqrt{\sum_{j=1}^{n} i_j^2} \sqrt{\sum_{j=1}^{n} q_j^2}} \tag{2.1}$$

This retrieval method has been employed in numerous lifelog retrieval systems [134, 108, 155] and has demonstrated its effectiveness in terms of performance. However, lifelog data is enormous, so using filters to narrow down the scope of the search helps to improve the search time. Several filters can be applied, including time, location, people, and activity. These filters may not be exact, such as the morning of November 26, 2025, but in relative terms, like Monday morning. The location can be the name of a place, such as Dublin Airport or a city. Filter terms can be extracted through a heuristic-based approach [158] from the query or input

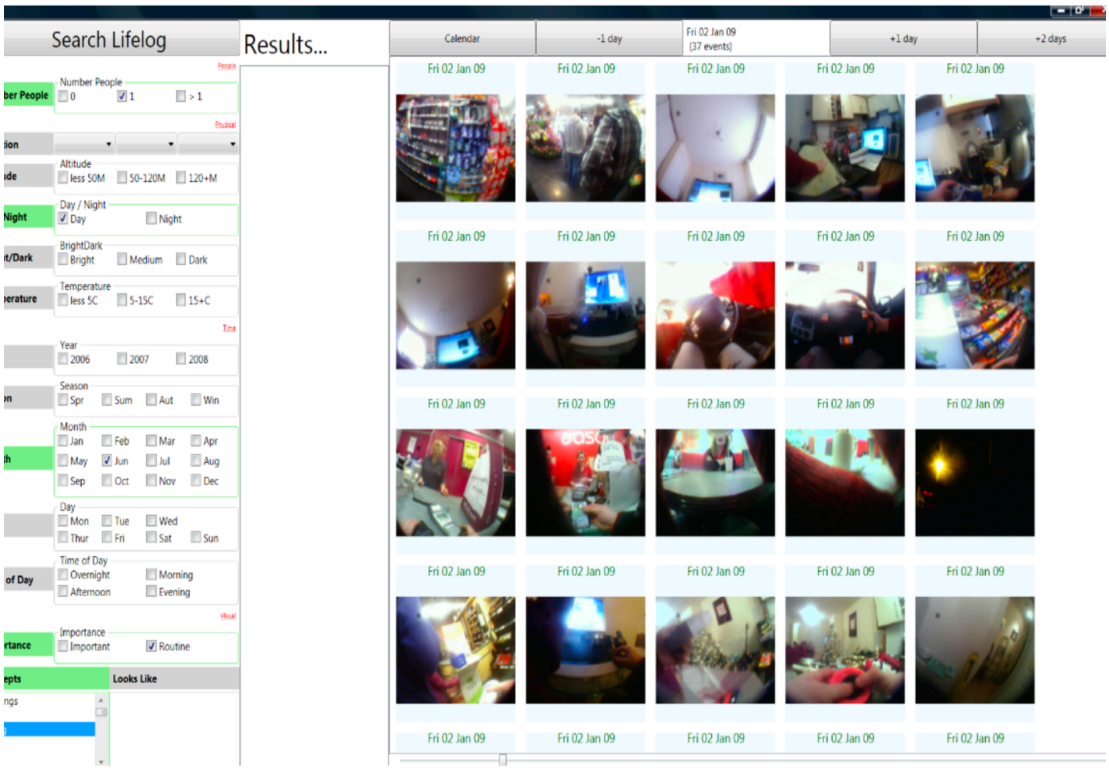


Figure 2.4: A multifaceted interface for lifelog retrieval systems with several search filters [36].

via a multifaceted interface, as shown in Figure 2.4.

2.1.3 Lifelog Search Challenge

Lifelog Search Challenge (LSC) [54] is one of the largest and longest challenges for interactive lifelog retrieval. It has been organised for 8 consecutive years, from 2018 to 2025 [58, 56, 54], and has attracted numerous researchers from the community. The challenge offers an interactive platform for systems to benchmark their performance on a diverse set of queries. Participant systems utilise the hints to identify the correct lifelog images in a lifelog dataset and submit them to the DRES evaluation system [130]. Datasets used for LSC have increased in both quality and quantity. The size of the dataset increases from a 27-day lifelog collection [52] with 41K lifelog images to 725K images covering 18 months. All images are captured from a Narrative Clip device with a resolution of 1024 x 768 pixels. All faces and most readable text are anonymised through a semi-automated process. Along with the images is

the metadata, such as timestamps, biometrics (heart rate, steps, and sleep statistics), and GPS coordinates of lifeloggers every minute. Visual concepts are also provided in the metadata, categorised into two types: detected objects and scenes. Objects are identified using a COCO-trained detector with 80 classes, while scenes are recognised using a Places365-trained model [188], yielding 102 scene labels such as "waiting in line" and "working". This dataset poses a challenge for participant systems to find correct answers in a large pool of images.

Tasks

There are three tasks in the LSC, including known-item search (KIS), ad hoc search (ad hoc), and question-answering (QA). In the KIS task, participants are given a textual query describing an event in the lifelog dataset and are required to find one image from a set of relevant images within a time limit of 300 seconds. The query is divided into six hints, which are released to participants every 30 seconds to provide additional information gradually. The scoring function for the KIS submission is defined by the following formula:

$$\text{score}_{KIS} = \max\left(0,\ 100 - 50 \cdot \frac{t}{T} - 10 \cdot w\right) \tag{2.2}$$

where t is the time taken, T is the time limit (300 seconds), and w is the number of wrong submissions. This formula takes into account both the time to submission and the penalty for incorrect submissions. Ad hoc search provides a query describing an activity or event that happens multiple times, such as "Find examples of wearing a yellow star t-shirt", and participants can submit as many images as possible. This type of query helps analyse the frequency of an event in the lifelog data. The scoring function for this task is as follows:

$$\text{score}_{Ad\text{-}hoc} = 100 \times \frac{\text{correct}}{\text{correct} + \text{incorrect}/2} \times \frac{\text{correct}}{\text{total}} \tag{2.3}$$

With "correct" is the number of relevant images submitted, "incorrect" is the

number of irrelevant images submitted, and "total" is the total number of relevant images aggregated over all submitting teams. Human judges are needed to evaluate the submissions, as there is no predefined ground truth for this task. QA is the newest task among the three, which was only introduced in LSC'22. It requires a textual answer for a given question. For instance, an actual question from LSC'24 is "How many times did I visit an outdoor/farmers market in February 2020?". The scoring function for this task is similar to that of the KIS task; however, human judges are required to evaluate variations in the submitted correct answers.

User Participation

To evaluate the system under realistic conditions, user participation in the LSC sessions is divided into two categories: expert users and novice users. Expert users are those who are either involved in the development of the system or have prior familiarity with it, as well as experience with lifelog image retrieval in general. Novice users, on the other hand, are participants who have no prior exposure to the system or to lifelog retrieval tasks. Both groups operate under identical conditions in terms of time limits and task assignments to ensure a fair basis for comparison across sessions.

This distinction helps to assess not only the system's retrieval performance in the hands of experienced users but also its usability for first-time users. As the results between the two groups can vary considerably due to differences in familiarity with the interface and search strategies, the performance results reported and analysed in this study are primarily drawn from the expert sessions, where users can fully leverage the system's capabilities and interact with it in the manner it was designed to be used.

Participant systems

The challenge has attracted numerous participant systems from all over the world to showcase their retrieval systems. I will list some of the notable participants'

systems that employ innovative approaches and achieve top performance. Myscéal [158] was first introduced in LSC'20 and has continued to participate in the challenge, showing a lot of improvement every year [155, 160, 153]. This system achieved top performance from LSC'20 to LSC'22, with implementation evolving from aTF-IDF concept-based retrieval [158] to embedding-based retrieval [155] and an event-based user interface [160]. Figure 2.5 shows a QA task interface with grouped images as an event and a simple search box without any filters for novice users. The accurate retrieval, thanks to the CLIP model, and the efficient user interface with event segmentation help the system maintain its top performance compared to other systems. Other participants, such as LifeSeeker [108] and LifeGraph [131], as well as T-EAGLE [110], also present their systems at LSC, featuring embedding-based retrieval, and demonstrate remarkable performance on all tasks.

One of the first and longest participants is lifeXplore [106, 134, 120] from 2018 to 2025. In the first participation [106], this system was adapted from a video retrieval system by converting lifelog images to videos. They performed keyframe selection and feature extraction before searching the videos of the lifelog. With the increasing popularity of CLIP, this system transitioned to using CLIP for embedding-based retrieval and achieved top scores in LSC'23 and LSC'24 [134]. In the latest version [120], they changed the method of retrieval by using text-to-text search from the generated captions from ChatGPT [2] and BLIP2 [90], but experimental results from past challenges showed that text-to-text search performs worse than embedding-based retrieval.

As LSC is an interactive challenge, the user interface (UI) and user experience (UX) are important for users to quickly submit correct answers. Although most systems employ a standard search interface with a search bar, filters, and a grid of result images, some unique interfaces are also presented. Vitrivr-VR [144] utilises a VR-based interface for retrieval, thereby fostering users' immersion in data exploration. Voice-based search from Voxento-Pro [4] to reduce the typing time. MyEachtraX

[2] https://chatgpt.com/

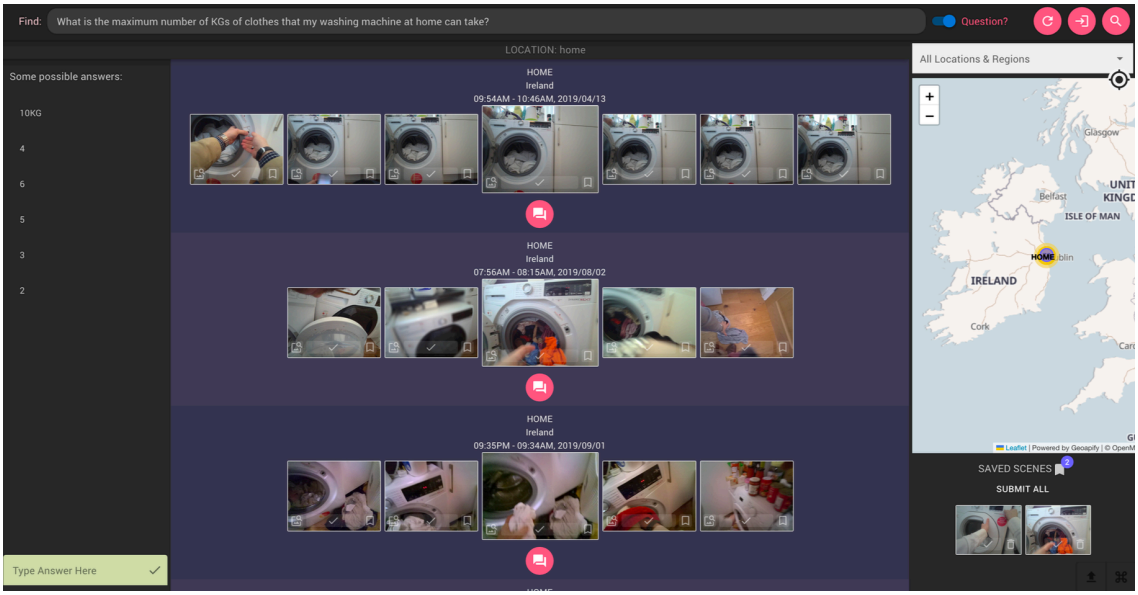


Figure 2.5: MyEachtra user interface with suggested answers for QA task [160].

[153] presented a mobile-friendly user interface to enhance the lifelog search in everyday settings. These works on UI/UX show the progress in developing interactive retrieval systems to support efficient image retrieval during the competition.

With the increasing popularity of LLMs, more and more systems in LSC employ them for conversational search and question-answering. Memento 4.0 [3] used the GPT-3.5 Turbo and Mistral 7B models for its QA and conversational search engine. Voxento-Pro [4] leverages OpenAI Assistant and the Whisper model for a voice-based conversational interface. Other systems, such as lifeXplore [120] and MyEachtraX [153], utilise Vision LLMs to extract information from lifelog data, thereby enhancing retrieval.

2.1.4 NTCIR Lifelog Task

The Lifelog Task at the NII Testbeds and Community for Information Access Research (NTCIR) conference has been the longest-running task, which began in 2016 [60]. This is a hybrid benchmark for evaluating both automatic and interactive lifelog retrieval systems, but the automatic one is more popular. The automatic task provides a lifelog dataset and a set of queries for participants to run automatically, without any human involvement, similar to the interactive one. It aims to test

Listing 1: Ad-hoc Topic AD005

```
<topic>
<id>AD005</id>
<type>adhoc</type>
<title>Watching sports in a pub</title>
<description>Find all the times I was
    watching sports in a pub</
    description>
<narrative>This query focuses on
    watching sports events in the
    setting of a pub. Home or other
    locations are not relevant.</
    narrative>
</topic>
```



Figure 2.6: An ad hoc topic in the LSAT sub-task and its corresponding answer [191].

the performance of the search engine only. In recent years [190, 191], this task has utilised the same dataset from the LSC challenge [56], but with different queries.

There are three subtasks in the NTCIR Lifelog Task: the Lifelog Semantic Access Task (LSAT), the Lifelog Insight Task (LIT), and the Lifelog Question-Answering Task (LQAT). The LSAT sub-task is similar to a combination of known-item and ad hoc search from LSC. Queries in this sub-task ask for specific events from the lifelog collection. The LIT sub-task asks participants to extract meaningful insights from the lifelog dataset and visualise them. The LQAT task presents a set of approximately 25 questions, as seen in the latest NTCIR18 Lifelog-6 Task [191], and requires a textual response. An example of an ad hoc topic in the LSAT task and one of its corresponding correct answers are depicted in Figure 2.6.

In the automatic setting, participants run the search with provided queries and extract the top-ranked list of relevant images, then submit it to the organisers. In the interactive setting, participants can interact with the search system and choose the relevant images to submit. The performance is evaluated through metrics such as Recall@K, Precision@K, and Mean Average Precision (MAP). Several systems participated in the latest NTCIR Lifelog Tasks, and I will summarise their systems as well as their performance. In the NTCIR18 Lifelog-6 Task [191], there are four systems that participated in the LSAT automatic sub-task. vitrivr-engine system [129] employed an LLM (Mistral 8B) for automatic query transformation to con-

vert queries to third-person descriptions. However, the low performance indicates the misalignment between the LLM-generated query and the CLIP training data. LifeIR team [20] proposed a novel post-processing approach with the Qwen2-VL model to check the result and show a small improvement. EAGLE [109] presented a hybrid approach that combines concept-based and embedding-based methods with the CLIP model. Overall, these systems demonstrate lifelog retrieval methods and showcase both the progress made and the remaining challenges in retrieval using real-world lifelog data.

2.2 Lifelog Question-Answering

In this section, I provide some background information on the Question-Answering (QA) task, followed by an overview of the QA task for lifelog data and its applications. Previous works on benchmarking datasets and methods are discussed to highlight the research gap in this task.

2.2.1 Task Definition

General QA

Question-Answering (QA) is a well-established task in Natural Language Processing (NLP) [18, 5, 25]. This task involves asking and providing an answer to a question about a specific topic. Depending on the knowledge required to answer the question, QA can be classified into two main categories. Closed-domain QA is similar to the Machine-Reading Comprehension task (MRC), where a provided paragraph of knowledge and a question are given, and the closed-domain QA model creates an answer for the question. Unlike closed-domain QA, Open-domain QA [192] only receives a question and a large knowledge corpus, which can comprise several documents or a database. Open-domain QA models then need to retrieve the relevant information to the question first before creating the answer. QA tasks can also be classified by the format of answers, such as multiple-choice QA, span-based QA, and

open-ended QA. Depending on the modality of the knowledge base, QA has several subtasks, like Visual QA and Video QA. Visual QA [7, 169] requires generating answers for questions based on one image or several images. Lifelog QA can be classified as Open-domain QA, as the lifelog collection can play the role of a knowledge corpus for answers to ground on.

With the multi-modal nature of lifelog data, which encompasses not only images but also tabular and time-series data, lifelog QA shares some commonalities with Visual QA, while also exhibiting some unique characteristics. When treating sequences of lifelog images as frames and converting them to video, question-answering on lifelog now becomes VideoQA. VideoQA is a task that generates an answer to a question for a video [145]. The input consists of a short video clip and a question that inquires about the clip. The output is a natural language answer to that question. The answer can be a choice in multiple-choice questions or a text in open-ended questions. Although the domain of vision-language understanding has developed in recent years [91, 90, 121], the VideoQA task remains a challenging task due to the need to comprehensively understand the video and question to generate the correct answer. Many VideoQA datasets are constructed across various domains, including movies [147, 101], TV shows [85], and traffic [172]. While most of the VideoQA datasets consist of third-person view videos, EgoVQA [42] is a first-person view video dataset, comprising 600 question-answer pairs from 16 videos. With a large number of VideoQA datasets, the methodology for finding the answer to the question is also explored. A common solution for this problem is an encoder-decoder network. In which the video and question are encoded using a video encoder and a question encoder. The video encoder is a module that extracts information in the form of vectors, such as those used in ResNet [61] and VGG [140] models. The question encoder extracts a token-level representation of questions using some models, like BERT [33]. A cross-model interaction module processes information from the encoders using several techniques, such as the Transformer [164], to derive useful information. Then the answer decoder generates the answers from

learned information. After the introduction of LLMs, the SOTA VideoQA model is Video-LLM, such as VideoLLaMa2 [24], VLM [171], with the advanced reasoning ability of LLMs, combined with spatial and temporal information, to enhance video understanding. These models, with appropriate prompts, can be accurate VideoQA models.

Lifelog QA task definition

Lifelog QA is the task of creating the answer to a question about a past moment or experience from a lifelogger's daily life [157]. This task requires a question and a lifelog collection as input, and a textual answer as output. The questions focus on the lifelogger's habits, events, and activities, but their nature varies depending on who is asking. When lifeloggers ask about their own lifelogs, the questions are often highly specific, such as inquiries about a particular event at a specific time. In contrast, questions from other people tend to be more general, such as "How many times did he exercise per week?". This variation highlights the wide and fine-grained nature of lifelog questions. A formal formulation of this task is: Given a lifelog collection L, with an input of a question q, the QA model f needs to generate the answer a to that question.

$$a = f(q, L) \tag{2.4}$$

From my observation, lifelog QA has three unique characteristics compared to Video QA. Firstly, while the video consists of clips, audio, subtitles, and captions, lifelog data is images associated with timestamps, locations, biometric data, and internet footprints. The increasing number of information channels and the relationship between lifelog data make it more complex. Secondly, the egocentric view of lifelog data differs from most third-person view VideoQA datasets, which makes it challenging to adapt VideoQA models to lifelog QA data, thereby reducing their performance. Finally, VideoQA uses high frame-rate videos, while lifelog data are images captured at very low frame rates. These differences pose a big challenge

Table 2.1: Lifelog QA datasets statistics.

Dataset	# QA	Lifelog data	Question format	# Event description	Image	Time & Location
LLQA [157]	15,065	85 days	Multiple-choice	None	✓	✓
Timeline QA [146]	600,000	Synthetic	Span-based	128,023,476		✓
MemexQA [70]	20,860	Personal images	Multiple-choice	None	✓	✓

for lifelog QA to adapt a VideoQA model, which can inherit some features from VideoQA but also possess some special characteristics unique to lifelog.

Besides these distinctions, lifelog QA also introduces challenges that are not commonly encountered in Visual QA, Video QA, or text-based QA. Lifelog questions often require long-term temporal reasoning, as lifeloggers may ask for events spread across days, weeks, or even months. In addition, lifelog data combined from various sources such as images, metadata, health metrics, and location makes the structure of lifelog complex. Finally, aggregation questions, such as counting occurrences and durations of events, are natural in lifelog QA. These differences emphasise the need for a QA model that can handle temporal dependency, multimodal data, and the aggregation focus of lifelog QA.

2.2.2 Benchmark Datasets

Benchmark datasets are essential for evaluating and comparing lifelog QA methods in a consistent and reproducible manner. Despite the growing interest in this area, only a small number of dedicated datasets have been released, largely due to the inherent difficulty of collecting and annotating long-term personal lifelog data. Table 2.1 summarises the three lifelog QA datasets, along with key statistics regarding their scale, data modalities, and question formats.

- **MemexQA:** Several works [146, 157, 156] have investigated the problem of lifelog QA, providing benchmarks and solutions to address this challenge. There are several notable benchmarks in the domain of lifelog QA. The first one is MemexQA [70], a personal image-based question answering dataset. The images in this dataset are 13,591 personal images collected from 101 Flickr

users. They are then annotated by crowdsourcing to create 20,860 questions and 417,200 multiple-choice answers. The human baseline performance on this dataset is 0.93 accuracy, thanks to the information provided by the images. This dataset is not a true lifelog dataset because only a fraction of life events are collected using smartphones. The images are not POV images, and only limited metadata is collected, such as the capture time. However, this work laid a foundation for question-answering on personal data.

- **LLQA:** Another multiple-choice lifelog QA dataset is LLQA [157], comprising over 15,000 questions. The lifelog data comes from a single lifelogger, spanning 85 days. The QA pairs are generated in a semi-automatic approach. From a given lifelog event description, the author utilises a Seq2Seq neural network to extract entities, including subjects, verbs, and objects. From the entities, they transform it into a yes/no question or Wh-word question generation with another Seq2Seq model. Finally, a distractor generator [44] generates false answers for multiple-choice questions. Annotators then only check for duplicated QA or balance the ratio of multiple-choice and yes/no questions. The multiple-choice questions are easy to evaluate in terms of accuracy, but they lack a real-world application. The human baseline for this dataset is 0.8417 accuracy for yes/no questions and 0.8625 accuracy for multiple-choice questions. The authors argue that human performance cannot reach 1.0 accuracy because volunteers were only shown the specific event segment relevant to each question, rather than the full day of lifelog data. As a result, the volunteers sometimes lacked the broader context needed to fully understand the events referenced in the questions. Data from a single lifelogger over a short period may not fully capture the diversity and complexity of the lifelogger's behaviours.

- **TimelineQA:** Another benchmark dataset is TimelineQA [146], a span-based QA dataset with synthetic lifelog data. The dataset is generated using templates that include both important and detailed life events, followed by QA

for each event. The persona is generated first with some important information about the subject, such as gender, birth date, name, education, family members, profession, preferences, and habits. From those skeletal details, episodes (lifelog event descriptions) are generated, ranging from annual to monthly to daily events. For each episode, they generate a set of QA pairs using various templates, ranging from yes/no questions to Wh-questions. The answers are extracted from the episodes, making this dataset a span-based QA dataset. The authors classify lifelog questions into several types, including atomic, complex multi-hop, complex aggregation, and temporal questions. I adopt this classification and make a small modification for the construction of my dataset, which will be described in the following section. Thanks to the generative content, this dataset can be expanded to include data from many lifeloggers, with thousands of QA questions per lifelogger.

- **PerLTQA:** is also a synthetic lifelog QA dataset, generated in a similar process but in Chinese [40]. This dataset comprises 141 imaginary people, with 4,501 event descriptions and 8,593 QA pairs. The answer is span-based, which can be extracted from textual descriptions of events, with an average of around 27.4 words per answer. However, since the lifelog data is generated from templates, it reduces the diversity of content and cannot fully replicate a real person's lifelog data. In addition, span-based QA in the TimelineQA dataset can only extract contiguous text spans from the events, which is not suitable for questions that require reasoning and combining information from multiple events of the lifelog.

According to related research in the field of lifelog QA, there is a lack of realistic, open-ended lifelog QA systems. The TimelineQA and PerLTQA datasets provide a detailed lifelog episode, along with pairs of questions and answers. However, the data is not from realistic individuals but from synthetic lifelog episodes generated for research purposes, which limits the diversity and the challenge presented by realistic lifelog data. The LLQA and MemexQA datasets generate question-answer

pairs from realistic lifelog data, but only during a short period of logging or from phone-captured images, and the multiple-choice questions lack practical use. From these previous weaknesses, I aim to construct a dataset with realistic lifelog data from a real person and open-ended questions for practical use.

2.2.3 Methods for lifelog QA

Several methods have been proposed to address the lifelog QA task, ranging from treating sequences of lifelog images as a video and applying a VideoQA model [157, 85], to converting lifelog data into textual descriptions of events and utilising Retrieval-Augmented Generation (RAG) [87] and ExtractiveQA models [73] for atomic questions, and TableQA models [8] for aggregation questions.

VideoQA models for lifelog QA

On the LLQA dataset [157], a pilot experiment used VideoQA models, such as TVQA [85], a state-of-the-art system in VideoQA at the time of publication, and Sequence to Sequence (S2S) [11] to choose answers for multiple-choice questions. The accuracy for TVQA is 63.38% on Yes/No questions and 61.36% on multiple-choice questions. The performance of the S2S model is lower, at 50.66% for Yes/No questions and 36.26% for multiple-choice questions. The low performance indicates the challenge of lifelog QA for these models, and the gap compared to the human baseline suggests a potential for improvement. In addition, the multiple-choice format is not suitable for natural interactions and constrains responses to predefined options. The difference between VideoQA and lifelog QA, especially in the fundamental purpose of the two problems, makes the use of VideoQA models for lifelog QA unsuitable, and the result from previous research also shows limited performance. For these reasons, I do not further investigate VideoQA models for lifelog QA.

ExtractiveQA and TableQA models for lifelog QA

A baseline experiment on a TimelineQA dataset of 128 million synthetic lifelog episodes is carried out on three approaches: RAG [87] and ExtractiveQA [73] for atomic questions, and TableQA with Tapex [8] and BART [86] variants for multi-hop and aggregation questions. For the atomic questions, they use RAG and ExtractiveQA on three retrieval settings (zero-shot (ZS), fine-tuned (FT), and oracle (OR, the ground-truth episode associated with the question)). The retriever is a dense passage retriever on FAISS with top-5 episodes. The reader for RAG is RAG-Token[3] and for ExtractiveQA is the RoBERTa model [96] fine-tuned on the SQuAD dataset. For the multi-hop and aggregation questions, they apply the TableQA approach by constructing relational tables and utilising Tapex and BART to answer the questions based on the generated Structured Query Language (SQL) queries. The experimental results show that for atomic queries, the ExtractiveQA model outperforms the RAG QA model, achieving 94.8% F1 compared to 84.4% F1. For multi-hop queries involving aggregates, the results show that the best result is achieved with a TableQA technique using a fine-tuned Tapex-large model. The TableQA technique constructs topic-based event tables (medical care, chat, exercise) before generating Structured Query Language (SQL) from a natural language query to perform a search. The fine-tuned Tapex-large model used to generate SQL queries achieved an F1-score of only 59%. The results demonstrate that the multi-hop and aggregation questions pose a challenge for QA models to answer correctly. However, the data is only synthetic lifelog data, so it can be more diverse and challenging to QA a model in real lifelog datasets.

Nevertheless, the method still has some drawbacks. On the TimelineQA benchmark, ExtractiveQA outperforms RAG due to the span-based nature of the answers. However, the RAG method used in the research is not optimised for lifelog, and the language model fails to generate span-based answers from the provided context. This gap underscores the need for lifelog-oriented retrieval and reasoning methods

[3]https://haystack.deepset.ai/tutorials/07_rag_generator

to generate accurate answers, particularly for aggregation questions.

Other QA Paradigms and Their Limitations for lifelog QA

Beyond the methods reviewed above, the broader QA literature includes several paradigms that are worth considering for lifelog QA: knowledge graph QA (KGQA) and SQL-based TableQA.

A knowledge graph is a directed graph of knowledge entities as vertices and their relations as edges [94]. KGQA tries to answer questions by traversing or reasoning over this graph. This approach is well-suited for factoid questions about well-defined, relatively static entities. However, lifelog data lacks a pre-existing, curated knowledge graph. Constructing one from raw lifelog collections would require entity extraction, relation linking, and schema alignment across heterogeneous modalities (images, GPS, health metrics). Moreover, lifelog questions [146, 157] frequently involve temporal aggregation and personal context rather than factoid entity lookups, making graph traversal less appropriate [133]. For these reasons, KGQA methods are not adopted in this work.

SQL-based TableQA, such as Tapex [8], converts natural language questions into SQL queries executed against relational tables. As discussed in Section 2.2.3, this approach has been explored as a baseline for aggregation questions in TimelineQA [146] by constructing topic-specific relational tables from lifelog episodes via information extraction. While this is effective for synthetic data with regular schemas, real-world lifelog data presents several challenges. First, constructing reliable relational tables from noisy, unstructured lifelog descriptions requires a robust extraction pipeline, including entity recognition and schema setting. Second, the fixed schema assumptions of SQL are not suitable for the diversity of personal lifelog data and for open-ended QA. Third, SQL-based models are trained on structured data and do not handle free-text context or multi-hop temporal reasoning across episodes naturally. These limitations make SQL-based TableQA less suitable as a general approach for lifelog QA, particularly for complex temporal and aggregation questions over real

lifelog collections.

Interactive approaches

For an interactive approach to lifelog QA, Tran et al. [156] conducted a user study involving 235 questions, where 10 participants were asked to perform an interactive search and answer the questions. They proposed an Event Retriever-Event Reader pipeline with CLIP embedding-based retrieval as Event Retriever and FrozenBiLM [174] as Event Reader to generate several suggested answers. They compared the performance of participants when using a baseline search-only system versus a QA-enhanced system. The results showed that participants using the QA-enhanced system performed better than the other group, but the difference was not by a large margin, with only a 4.82-point higher score on a scale of 100 points. Time to first correct submission is 61.84 seconds in the search-only group and 63.82 seconds in the QA-enhanced group. The results indicate that participants had very little experience with lifelogging and QA. Although they did not measure whether the suggested answers from the QA pipeline are correct, the small difference highlights the slight improvement of the QA-enhanced system compared to the baseline.

Another user-study focused research is a system called OmniQuery [89]. This is an interactive system doing QA on past memories from personal photos. Thirty-two participants logged their lives by capturing images with their mobile phones and created 299 queries. The authors proposed a memory augmentation process by structuring, merging, and organising photos into contexts that comprise images, time, descriptions, and other relevant elements. They then perform a pipeline of query augmentation, retrieval, and answer generation with LLMs to answer questions. Based on user evaluation with 10 participants, OmniQuery outperformed the baseline RAG system (which lacks memory augmentation, merged context, and semantic knowledge), achieving an overall accuracy of 71.5% (where accuracy is defined as User Perceived Accuracy >= 4 on a scale of 5). In direct comparisons against the conventional RAG baseline, OmniQuery demonstrated its effectiveness

by winning or tying in 74.5% of cases. The study is primarily focused on a user study, rather than a large-scale evaluation of pipelines. Both studies using the interactive system employ 10 participants to evaluate the system and justify that fixed benchmark datasets for personal data are challenging due to the difficulties in collecting long-term lifelog data.

Overall, most of the reviewed methods employ a two-stage process, involving retrieval and reader or answer generation. This is also typical for solving an Open-domain QA problem. However, lifelog QA presents additional challenges that current pipelines have not addressed, including long-term temporal dependencies, multimodal and often noisy visual data, and queries that require aggregation or reasoning across multiple events. Existing retrieval modules are not optimised for lifelog structures such as daily routines, temporal locality, or event continuity, and current reader models struggle to generate precise, reasoning-enhanced answers from the retrieved context. There is considerable room for improvement and optimisation to make RAG a robust QA system for lifelog QA, both in the retrieval and reader stages.

2.3 Large Language Models and Retrieval-Augmented Generation

In this section, I will provide background information on LLMs, RAG and Reasoning. Transformers, the core architecture of most LLMs now, were first introduced with detailed information on the attention mechanism and the encoder-decoder architecture. The process of training and fine-tuning to create an LLM is described next. The history, method, and applications of RAG are also presented in the following section. Finally, reasoning LLMs, with details on CoT and reinforcement learning training, are described last.

2.3.1 Large Language Models

LLMs are transformer-based neural networks designed to generate text [186]. Scaling laws [72] show that performance predictably increases as model size, dataset size, and compute increase. LLMs are trained on vast and diverse datasets, including webpages such as Wikipedia [4], books [10], news articles [182], and code repositories [80], and consist of billions of parameters. LLMs, after training, can perform various downstream tasks via zero-shot or in-context learning without requiring additional fine-tuning.

Transformers

The transformer model was introduced by Google in 2017 [164] and was a breakthrough for many language processing tasks. Figure 2.7 depicts an encoder-decoder architecture of a Transformer model originally introduced by Google. Inputs are embedded to vectors and added with positional encoding before going through N identical layers, each consisting of a Multi-Head Attention sublayer and a Feed-Forward Network (FFN). In the Multi-Head Attention layer, the query (Q), key (K), and value (V) vectors for each input token are produced by applying linear projections to the positional-encoded input embeddings. These vectors are then passed into the scaled dot-product attention mechanism, which is computed using the following formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.5}$$

This helps to create an attention output that contains contextualised information of all words in the inputs from key K and query Q. They then stack H heads of the scaled dot-product attention function to create a multi-head attention. Multi-head attention lets the model attend to multiple types of representations at once and across different positions. The value V is then added to the original vector after positional encoding with a residual connection, and normalised.

[4] https://www.wikipedia.org/

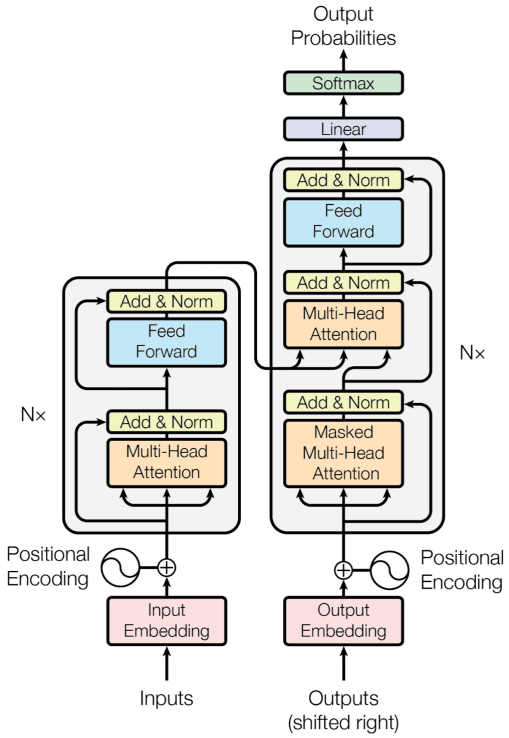


Figure 2.7: Transformer Encoder-Decoder Architecture [164]

The decoder architecture is nearly the same, except for Masked Multi-head attention, which only attends to previous tokens (and itself). It aims to prevent rightward information flow in the decoder to preserve the autoregressive property. Then a Multi-head attention combines key and value from the encoder and the query from the decoder to create the final embedding. The encoder-decoder architecture is mainly used for the translation task, where inputs are the text in one language and outputs are the translated text in another language.

An encoder-only model like BERT [33] only use the block of the encoder to encode sentences to a meaningful representation of words. This embedding can be used in tasks like retrieval and text classification. Current LLMs are mainly built on a decoder-only Transformer architecture [111], in which many attention layers are stacked to create a very deep neural network [186]. Thanks to the attention mechanism, LLMs can learn the relationships between words in sentences and predict the next word from previous words in an autoregressive way.

Training phases

In training LLMs, there are two primary phases: pre-training and post-training (preference or alignment tuning). In the pre-training phase, the model is trained on massive amounts of unlabeled data for the next-word prediction problem. After the pre-training phase, the model has acquired general knowledge and can generate meaningful text. However, it does not produce the desired output for humans, so it requires a preference tuning phase. In the second stage, the model undergoes supervised fine-tuning on labelled data, utilising sample input and output. The goal is to minimise the difference between the generated output and the labelled output for a given input.

Besides supervised fine-tuning, reinforcement learning is also used for preference tuning to reduce the cost of constructing labelled data. OpenAI is a pioneer in this field, utilising Reinforcement Learning from Human Feedback (RLHF) [114] to mitigate untruthful, toxic, or harmful output from LLMs. In this method, human annotators label a rank list of different outputs for an input, and LLMs learn to generate high-rank outputs more often than low-rank outputs. Many other reinforcement learning-based preference tuning algorithms [122, 138] have been developed to enhance the performance of the outputs. Details about these algorithms will be discussed in Section 2.3.3. After this stage, an LLM can generate meaningful, safe, and aligned output in response to the provided prompt.

Capabilities

Since the release of OpenAI's ChatGPT [113], the development of LLMs has advanced rapidly in both academia and industry [152, 111, 181]. These models have demonstrated remarkable performance across various NLP tasks, such as QA, summarisation, and translation [132, 62, 149]. The expected capabilities of LLMs, including simple arithmetic, basic linguistic competence, and general world knowledge, have been observed in various tasks. Furthermore, some emergent capabilities are also observed when increasing the size, data, and compute resources of LLMs [167].

These capabilities include in-context learning [14], instruction following, chain-of-thought reasoning [166].

In-context learning, or few-shot learning, is a capability of LLMs that allows them to perform a new task with provided input-output examples in the prompt. This capability replaces the typical approach of adapting language models to new tasks by fine-tuning. Brown et al. [14] demonstrate that few-shot learning can even reach similar competitiveness to fine-tuning. This capability introduces a new prompting technique, called few-shot prompting, in which a subset of examples with input and desired output is provided to help models generate an aligned output. Instruction following is another emergent capability of LLMs. The models can follow the instructions in the prompt to generate the desired output. For example, a prompt with the instruction "Generate a 200-word summary of this article, with an opening sentence and a conclusion sentence." can guide the models to generate approximately 200 words for a summary. This capability promotes a prompting technique called a zero-shot prompt, in which the model is only instructed to generate outputs with long instructions without any examples. Combining instruction-following and in-context learning, along with examples and detailed task descriptions, helps guide the model and improve the output of LLMs.

Reasoning is an emergent capability of LLMs with the ability to generate step-by-step intermediate rationales on complex, logical problems, such as mathematics and coding. Wei et al. [166] showed that LLMs with a large number of parameters (over 100 billion) can generate intermediate rationales via a simple prompt "think step-by-step to solve the problem" and some examples. Other methods, such as the tree-of-thought [177], propose a reasoning tree to represent the structure of complex questions and use LLMs to provide answers based on each branch. Overall, combining these emergent capabilities through prompting techniques such as zero-shot prompting, few-shot prompting, and chain-of-thought brings the best performance for LLMs.

Applications

LLMs have been used in a wide range of applications across natural language understanding and generation. Instead of specific models for specific tasks such as sentiment analysis, question-answering, and translation, LLMs can do all these tasks in an ask-answer manner. LLMs are now used in question-answering and conversational agents to solve various problems for users, with applications such as ChatGPT [113] and Gemini [149]. LLMs are also applied in code generation and software engineering tasks with Codex [21], Copilot [5], and Claude Code[6]. They help programmers write code more efficiently by suggesting code or even autonomously generating code from requirements. LLMs also aid in research, assisting with literature analysis and experimental design. Galactica [148] is an LLM from Meta AI that can write LaTeX code and literature reviews with citations. These applications demonstrate the application of LLMs in various domains and yield remarkable results. Using LLMs for lifelog data is promising, especially in the question-answering task.

LLMs have proven effective in addressing QA tasks that require reasoning across various domains, from medical [141, 100] to finance [116] and law [98, 66]. For instance, Lucas et al. [100] utilised LLMs with chain-of-thought prompts to answer complex multiple-choice medical questions, showcasing their potential in challenging QA scenarios. In the scientific domain, Lu et al. [99] proposed a multimodal reasoning approach using reasoning chains to resolve science QA tasks. Aggregation questions in lifelog QA require multiple steps of information extraction and reasoning before producing a final aggregated answer. Leveraging the reasoning capabilities of LLMs can address these questions, which have not been the focus of prior research.

Limitations

Although LLMs demonstrate strong capabilities and are applied to various applications, they still have some drawbacks. As LLMs are trained on fixed datasets and

[5]https://github.com/features/copilot
[6]https://claude.com/product/claude-code

typically general data from various domains, they often lack up-to-date data and specific, in-depth domain knowledge. For example, the knowledge cut-off date for the GPT-5 model is September 30, 2024, so every event after that date is not included in the parametric knowledge of GPT-5 models. In addition, LLMs lack deep, specialised, or internal knowledge, such as specific laws or legal documents, proprietary company documents, or personal lifelog data. These drawbacks necessitate the incorporation of new, specific knowledge into LLMs, and RAG was published in 2020 to address this issue.

2.3.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) [87, 45] is a powerful technique that enhances LLMs by incorporating external, up-to-date, and specialised knowledge sources. Instead of relying solely on LLMs' fixed training data, RAG enables the model to first retrieve the most relevant information from a knowledge base (such as a database or document collection) and then use this context to prompt LLMs to ground their generated answer. The incorporated knowledge in the prompt helps LLMs generate relevant answers and avoid hallucination. The retrieved data from external knowledge bases improve the accuracy, relevance, and currency of LLMs' responses, addressing the limitation of their fixed knowledge cutoff. Figure 2.8 illustrates an example of RAG for QA from [45].

RAG architecture

The key idea of retrieval and generation is not new; it has been used in Open-domain QA as a Retriever-Reader method [192]. Early Open-domain QA systems, such as DrQA [19], employ sparse retrieval with TF-IDF [125] and a span-based QA model to generate answers for questions based on the Wikipedia source. However, with the emergence of semantic retrieval based on embeddings from language models and the advancement of LLMs, RAG has become a popular technique for improving LLMs. Lewis et al. [87] created RAG with a dense retriever, DPR [74] and a generator, the

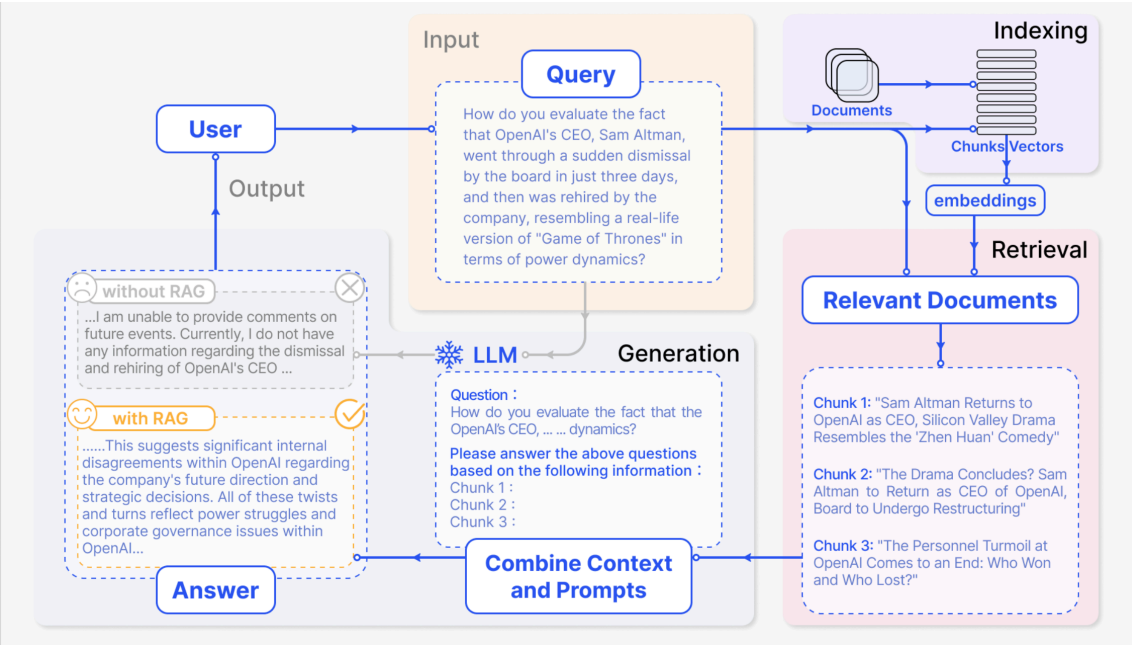


Figure 2.8: Examples of RAG used in QA from [45]

BART model [86] in 2020. The DPR retriever encodes documents and a query to a same-dimension embedding using a language model like BERT [33]. They index the embedding to a database and refer to it as non-parametric memory. The list of top k documents relevant to the query is calculated via the dot product between embeddings. The input of the query and retrieved documents is concatenated before prompting the generator to generate the answer. In the original paper, the author proposed fine-tuning the query encoder BERT and the BART generator in an end-to-end training setting. As BART has around 400 million parameters, it is not considered an LLM, so it needs to be fine-tuned to perform in a downstream task. With LLMs now with billions of parameters, the RAG does not need fine-tuning.

Retrieval phase

In the retrieval stage, relevant information is sourced from external knowledge bases using dense vector retrieval, which improves accuracy and efficiency compared to sparse retrieval. Sparse retrieval methods, such as TF-IDF [125] and BM25 [127], utilise the matching of words in documents and queries to rank relevance. It is precise if the requirement needs exact word matching, and it is fast and efficient in

computation. On the other hand, dense retrieval methods, such as DPR [74] and ColBERT [76], extract the embeddings of queries and documents using a language model and then perform similarity calculations, such as cosine similarity, to calculate the relevant score and rank the list of documents. This method improves accuracy by comparing the semantics of documents and queries, rather than just the words, but the cost of embedding is high. A hybrid approach, which combines sparse and dense retrieval, can accommodate both word and semantic matching in retrieval, but the cost is even higher. Dense retrieval is used in the original RAG study and preferred by many use cases, showing substantial improvement compared to sparse retrieval.

To improve the retrieval process, several vector databases and chunking strategies have been introduced. Several vector databases, such as FAISS[7], Elasticsearch[8], and Milvus[9], help the retrieval process run faster thanks to the Hierarchical Navigable Small World (HNSW) indexing and Approximate Nearest Neighbour (ANN) search algorithm. HNSW uses a hierarchical structure to organise vectors into layers, enabling ANN search to perform quickly by approximating from the top to the bottom layers. Chunking strategies involve splitting long documents into manageable chunks for indexing in databases. They have also developed from a fixed chunk to a semantic chunk and a sliding window chunk. Metadata can also be indexed to facilitate retrieval through filters. It is helpful to reduce the search space in a large knowledge base.

Answer generation stage

From the retrieved information, the second stage of RAG relates to generating the answers. The original paper of RAG just concatenates the question and retrieved documents to form a prompt for LLMs to generate answers. Other prompting techniques, such as few-shot and chain-of-thought [14, 166], are also used to form a

[7]https://faiss.ai/index.html
[8]https://www.elastic.co/
[9]https://milvus.io/

prompt. An important instruction is "Only generate an answer grounded on provided contexts" which helps the answer not be hallucinated from parametric knowledge in LLMs.

Research in RAG

The RAG framework has been refined to further enhance both retrieval and generation accuracy in numerous works. For instance, InstructRAG [168] proposes explicit denoising through self-synthesised rationales to enhance generation accuracy by instructing models to explain how answers are derived from retrieved documents. DuetRAG [71] introduces a collaborative framework that integrates domain fine-tuning with RAG models to enhance knowledge retrieval quality, particularly in complex domains such as HotPot QA [176]. Several studies have also explored RAG's potential in various domains, such as open-domain QA [142], medical QA [170], and financial QA [180]. In lifelogging, RAG-based models have been explored to generate answers to questions based on personal data [146], but they underperform the extractive approach due to the extractive nature of the answers in the TimelineQA dataset.

Limitations

Although RAG solves the static knowledge problem of LLMs, it still has some limitations. Firstly, the quality of retrieved information affects the output. If irrelevant content is retrieved and LLMs are used to generate answers, the resulting answers can be incorrect. To mitigate this problem, evaluating the quality of retrieval using metrics such as Recall@K and Precision@K is crucial. In addition, the limit of context length, the maximum number of tokens that LLMs can receive as input, can reduce the amount of information provided from RAG. Increasing the context length and reducing the size of retrieved documents are proposed as solutions to this problem.

RAG is a strong fit for lifelog QA thanks to its retrieval and generation parts.

RAG can retrieve the most relevant lifelog events from a big collection and feed them to LLMs to ask for an answer. As lifelog data is personal and too specific to be captured in LLMs' parametric knowledge, using retrieval helps to address this problem. With dense retrieval and powerful LLM generators, RAG can provide accurate answers for questions over a large personal lifelog. It is well-suited for open-ended and aggregation questions.

2.3.3 Reasoning LLMs

Reasoning is an emerging capability of LLMs and plays a crucial role in solving various real-world tasks, such as coding and planning. Reasoning is a process of analysing evidence and applying logic to form conclusions [75]. In LLMs, reasoning refers to the ability to generate a logical analytic process before deriving the final answer. Intermediate thinking helps formulate a path to a solution and reduces the likelihood of hallucination. In the early days of LLMs, they often failed to answer basic math (e.g. from [166] paper: the cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have? Model's answer: 27), and later, there are more complex problems that require LLMs to think carefully before answer such as coding, planning, etc.

Methods for reasoning

Scaling the model weights is not a practical solution for improving the reasoning capability of LLMs [143]. A more intuitive approach is the chain-of-thought (CoT) [166], which involves simply modifying the prompt by adding "Let us think step by step" or some in-context samples of reasoning. This prompt improves LLMs' reasoning accuracy and does not require any additional training, just more time for inference. Other prompting methods to foster the reasoning of LLMs are also proposed, such as Tree-of-thought [177].

There are two main scaling methods that enhance the reasoning capability of LLMs: train-time scaling and test-time (or inference-time) scaling. Test-time scal-

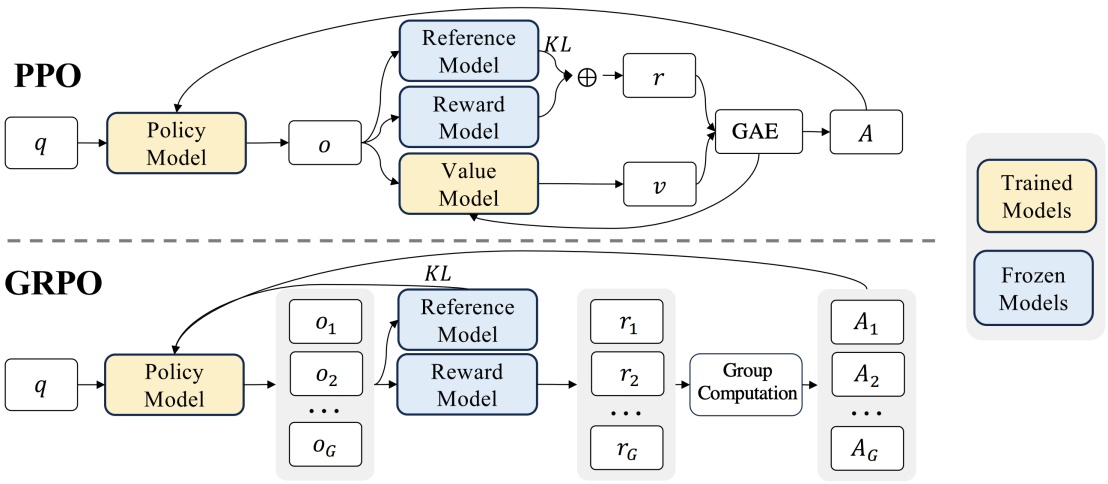


Figure 2.9: PPO and GRPO architectures [138]

ing focuses on allowing additional test-time computation for LLMs to use before generating the final answer. CoT and tree-of-thought are examples of test-time scaling, where the model is given computational resources to generate intermediate reasoning steps before producing the answer. On the other hand, train-time scaling aims to enhance the reasoning capability of LLMs during training by fine-tuning, particularly in the context of reinforcement learning. Due to the high cost of creating chain-of-thought data for training LLMs, reinforcement learning is an alternative yet efficient approach, with numerous methods for improving the reasoning of LLMs, such as PPO and GRPO.

PPO

In reinforcement learning–based fine-tuning of LLMs, the language model is treated as the policy, the prompt or input sequence serves as the environment state, and the model's token-by-token text generation is its actions. Proximal Policy Optimisation (PPO) [136] is an actor-critic RL algorithm that tries to optimise the policy (LLMs) to maximise the advantage intake from generating outputs for a question.

As can be seen in Figure 2.9, PPO contains two trained models: a policy model π_θ as an LLM generator to generate output o from question q, and a value model to estimate the expected return value v. The reference model $\pi_{\theta_{ref}}$ and reward model are frozen to estimate the reward r from that output. The advantage is

computed by applying Generalised Advantage Estimation (GAE) [135] based on the value v and reward r. At each training step, PPO computes a probability ratio between the updated policy and the old policy for each generated token. This ratio measures how much the new policy deviates from the previous one. The policy is then updated to maximise the advantage weighted by this ratio, but the ratio is clipped to a narrow range to prevent excessively large updates that could destabilise training. The advantage at each token reflects how much better the generated output is compared to the baseline value estimate, so the policy is tuned towards outputs that yield higher rewards while staying close to the previous policy. One problem with PPO is that the value model needs to be trained alongside the policy model, and it typically requires comparable resources to train the policy model.

GRPO

To address this problem, Group Relative Policy Optimisation (GRPO) [138] proposes to replace the value model with multiple generations and group relative advantage computation to compute advantage. GRPO is proposed by the DeepSeek team to improve mathematical reasoning for the DeepSeekMath model. For each input question, the policy samples G outputs $\{o_1, o_2, ..., o_G\}$ based on sampling-based decoding, where words are picked from a probability distribution rather than always selecting the maximum-probability token. The reward model then scores the outputs by verifiable reward functions to produce a list of rewards $r = \{r_1, r_2, ..., r_G\}$. The advantage is computed as $\hat{A}_{i,t} = \frac{r_i - mean(r)}{std(r)}$, then use this advantage to optimise the policy by maximising the objective function:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \\ \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\Big[r_{i,t}(\theta)\,\hat{A}_{i,t},\ \text{clip}\big(r_{i,t}(\theta),\,1-\varepsilon,\,1+\varepsilon\big)\,\hat{A}_{i,t}\Big] \\ -\ \beta\,\mathbb{D}_{\text{KL}}(\pi_\theta\,\|\,\pi_{\text{ref}})\,,\end{aligned} \tag{2.6}$$

where the probability ratio is

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}.$$

This group-relative advantage replaces the learned value model, unlike PPO. The policy is then updated using the same clipped probability-ratio objective as PPO, applied over every token of every output in the group and averaged across the group. An additional KL divergence penalty between the updated policy and a frozen reference model is included to prevent the policy from drifting too far from the original pre-trained behaviour, further stabilising training. The elimination of the value model helps reduce the resources needed for training the model. The authors of the DeepSeekMath model also notice "Aha" moments when the model exhibits behaviours such as reflection (where the model revisits and reevaluates its previous reasoning steps) and the exploration of alternative approaches to problem-solving arises spontaneously [31]. These behaviours are not explicitly programmed or in the training data.

Reasoning LLMs have demonstrated strong performance on logical tasks such as coding, planning, and mathematics, and are increasingly deployed in applications ranging from coding assistants [21] to autonomous agents. Modern coding assistants, including GitHub Copilot[10] and Qwen3 Coder [150], can interpret large code bases, provide in-line suggestions, and create test cases. In mathematics, the DeepSeek-Math model [138] achieves state-of-the-art results on multiple English and Chinese benchmarks, thanks to GRPO fine-tuning presented earlier. Lifelog QA similarly requires multi-step reasoning to answer aggregation questions involving temporal duration ("How long"), event ordering ("What happened before or after"), and event counting ("How many times"). Whether the strong performance of reasoning LLMs on these general tasks extends to lifelog QA remains an open question in research, and this study aims to investigate that question by using reasoning LLMs to answer aggregation questions in lifelog QA.

[10]https://github.com/copilot

2.4 Reasearch Gaps

The literature reviewed above shows lifelog QA as an active but immature task. In this section, several research gaps are stated, which motivate this thesis. There are four gaps, from the benchmark shortage for open-ended lifelog QA to the adaptation of reasoning LLMs to lifelog data. Each of these gaps is as follows:

Benchmark gap (G1). Existing lifelog QA resources are closed-form, span-extractive, or synthetic, such as LLQA [157] and MemexQA [70], which are multiple-choice, while TimelineQA [146] and PerLTQA [40] are span-based over template-generated episodes rather than a real collection. No existing benchmark provides open-ended, human-verified answers over a real, long-duration lifelog with a focus on aggregation questions. Consequently, open-ended answer quality on real lifelog data cannot currently be measured at all. This gap motivates the construction of the OpenLifelogQA dataset (Section 3.2) and underpins RQ2.

Retrieval gap (G2). Lifelog retrieval has been optimised exclusively for interactive image finding under LSC and NTCIR conditions (Sections 2.1.3 and 2.1.4), where the output is a ranked image list presented to a human for examination. Retrieval in a RAG framework as the evidence-supply stage of a generative pipeline imposes different requirements, from high recall over a compact context set, to improving the textual event descriptions for providing information for answers. These requirements have not been studied for lifelog data. This gap motivates RQ1 and the retrieval work in Chapter 4.

Aggregation-reasoning gap (G3). The lifelog QA methods reviewed in Section 2.2, drawn from video QA [85] and extractive QA [73], can only surface an answer that appears literally within a single retrieved segment. Questions requiring counting, summation, or duration arithmetic over multiple events fall outside their formulation, and the one baseline that addresses them through TableQA [146] reaches only 59.0% F1 on synthetic data. This type of question is important for habit and lifestyle analytics but remains underexplored. This gap motivates RQ2.2 and Chapter 5.

Adaptation for LLMs to lifelog data gap (G4). Reasoning-oriented post-training, including chain-of-thought prompting [166] and fine-tuning [138, 31], has been developed and validated principally on mathematics and code, where correctness is programmatically verifiable. Lifelog QA does not satisfy the condition, as the target is an open-ended natural-language answer and any reward must be constructed from imperfect similarity signals over free text. In addition, none of the lifelog QA methods reviewed in Section 2.2 adapts a language model to the lifelog domain at all, since the models are either used off-the-shelf or fine-tuned on synthetic episodes. Whether reasoning-oriented adaptation can bring open-source models, small enough to be run locally, to competitive accuracy on personal temporal data is therefore untested. This gap motivates RQ3 and Chapter 6.

2.5 Chapter Summary

In this chapter, I have presented the background information on lifelogging, from the history to current research on the lifelog retrieval task. From the basic need of finding events in lifelog data, the lifelog QA task resolves a more sophisticated problem with finding textual answers for specific lifelog-related questions. The task of lifelog QA, with previous research on benchmarks and methods to solve it, has been described in this chapter. The literature review on previous works for lifelog QA shows the limitations of benchmarks on multiple-choice QA or unrealistic lifelog data for QA. In addition, methods for lifelog QA, like Video QA models and ExtractiveQA, do not fully address the task, leaving room for improvement.

LLMs and RAG are also presented in this chapter as promising candidates to solve the lifelog QA task. LLMs with their generative capability have demonstrated their superior performance in various tasks. Incorporating them into the RAG framework can help LLMs answer lifelog questions from retrieved information. To address the challenge of lifelog QA, I propose a RAG framework with LLMs to formulate a new state-of-the-art lifelog QA system. Detailed exploration and implementation of this framework are presented in the following chapters.

Chapter 3

Methodology

In this chapter, I present the methodology I employed to conduct the research on lifelog QA. I begin with the details of the research stages, from hypothesis formulation to result analysis and report, which is described in Section 3.1. Section 3.2 provides information about the newly constructed OpenLifelogQA dataset and the DCU-TimelineQA dataset used for the experiment. Several operating constraints in this research are discussed in Section 3.3. Finally, several evaluation metrics used in this research are presented in Section 3.4.

3.1 Research Methodology

Research is a rigorous process of systematic investigation aimed at providing answers to specific questions on a particular topic [124]. Depending on the data and subjects being studied, research can be classified into two primary approaches: qualitative and quantitative. Qualitative research focuses on understanding the behaviours and sentiments of people in specific events, which is carried out by collecting non-numeric data through interviews or polls. On the other hand, quantitative research is conducted using a large amount of numerical data to measure and analyse patterns and relationships through statistical methods. Research can also be classified as exploratory or explanatory, depending on the novelty of topics.

This study is classified as exploratory research due to the novelty of the lifelog

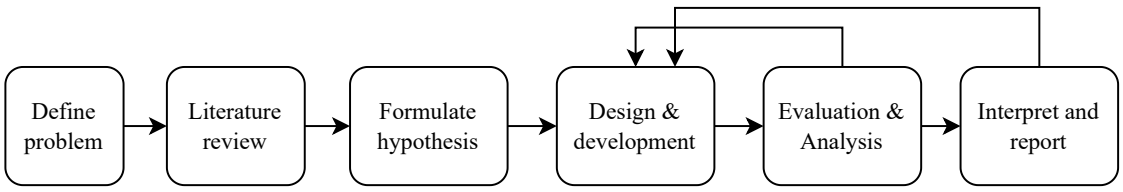


Figure 3.1: Diagram of research process, adapted from [81]

question-answering (QA) problem. A quantitative approach is adopted, in which systematic evaluations of QA models are conducted on benchmark lifelog datasets. Adapted from the research process outlined in [81], I present the research process applied to this study in Figure 3.1.

I start the process by defining the problem in lifelog QA. Lifelog QA presents unique challenges that raise questions about events in lifelog data and provides answers based on that data. Lifelog QA presents unique challenges due to the unstructured, multimodal, and temporally fragmented nature of lifelog data, as well as the presence of complex aggregation questions requiring reasoning across multiple events. With the development of LLMs and RAG, utilising them for lifelog QA is a promising approach. A literature review is then conducted to examine previous works on the problem, identifying the research gap and recent advancements in LLMs to inform a solution.

From the identified gap and background review on lifelog QA, lifelog retrieval, and LLMs, I propose a research hypothesis: *Reasoning LLMs in a RAG setting/framework improve the accuracy of the open-ended lifelog question answering task compared to standard LLMs.* The research hypothesis investigates whether LLMs, when combined with an advanced lifelog retrieval engine, can improve QA performance through enhanced reasoning over retrieved context. To validate this research hypothesis, three research questions are formulated as mentioned in Section 1.3. Each research question is systematically explored through extensive experiments, starting with the design and development of models/systems to address the questions, followed by rigorous evaluation and detailed analysis, and culminating in final interpretation and reporting. The results from experiments are used to answer the

research question, and from that, can be used to validate the research hypothesis. If the answer cannot be reached from the results, the design and development of the experiment are revised with modified research questions.

Specifically, I conduct the following experiments to address the research questions:

- To address RQ1: *What is the most effective context retrieval method to support the LifelogQA task when compared to standard retrieval approaches?*, a lifelog retrieval system is constructed as a baseline for interactive evaluation. The system is evaluated across three interactive lifelog search challenges. Additionally, experiments are conducted on multimodal lifelog reranking models for the automatic retrieval and reranking of candidate contexts for QA. The retrieval system design, reranking models, experimental setup, and evaluation results are presented in Chapter 4.

- To answer the RQ2 about RAG for lifelog QA: *How effective is RAG when generating answers for LifelogQA from retrieved contexts?*, a proposed RAG approach is built and evaluated through rigorous experiments. A two-stage pipeline of retrieval and answer generation based on LLMs is constructed to answer lifelog questions. A comparative analysis is performed between LLMs and SLMs to demonstrate the superiority of LLMs in lifelog QA and provide an answer for RQ 2.1. A fine-grained analysis of the performance of LLMs for different question types is also presented and discussed in the results of RQ 2.2. Details about the approach and experiment for RQ2 are described in Chapter 5.

- Aggregation questions pose a challenge for QA models to answer, so I propose reasoning-enhanced LLMs for lifelog QA to answer RQ3: *Can LLMs with reasoning capability improve the accuracy of answers for aggregation questions in lifelog QA compared to standard LLM baselines in RQ2?*. Experiments are designed to enhance the reasoning capability of LLMs via both chain-of-thought

[166] and fine-tuning. Various fine-tuning techniques, ranging from supervised learning to reinforcement learning, have been developed to enhance the reasoning capabilities of LLMs. A detailed analysis and evaluation are performed to demonstrate the superiority of the reasoning models over the standard model in the RAG baseline. Details about the implementation, experiment, and results are described in Chapter 6. These results help me to address RQ3 and support the evaluation of the main hypothesis.

3.2 Datasets

In this section, I present the datasets used for the experiment in this research, the OpenLifelogQA and DCU-TimelineQA datasets. The OpenLifelogQA dataset was constructed in collaboration with ASIA Lab from the University of Science, Vietnam National University. I led this project with a group of 10 volunteers from the ASIA Lab to generate open-ended QA for lifelog data. The dataset aims to serve as a benchmark for evaluating open-ended, aggregation-focused questions for realistic lifelog data. Details about the construction process, the statistics, and the analysis of the OpenLifelogQA dataset will be presented in the following sections. The DCU-TimelineQA dataset is another lifelog QA dataset used to further evaluate the accuracy of lifelog QA models proposed in this research. This dataset ensures no bias in training for OpenLifelogQA and evaluates the broad applicability of lifelog QA models across various lifelog datasets.

3.2.1 OpenLifelogQA Dataset

Dataset Construction

The lifelog dataset from the Lifelog Search Challenge 2024 (LSC'24) [56] serves as the starting point for creating the QA dataset. It contains 722,606 PoV images from a single anonymous lifelogger, spanning a period of 18 months from January

2019 to June 2020. The images are captured from a PoV camera[1] worn on the chest of the lifelogger. The images have a resolution of 1,024 x 768 pixels. All faces and readable text have been removed, and certain scenes and activities have been manually filtered out to respect local privacy requirements. In addition to the images, the metadata of the lifelog collection is also used. This includes the capture time and location of each image, as well as whether an image was captured every minute if no image was captured at that time, along with related biometrics from a smartwatch.

From the lifelog data source, and in line with prior lifelog QA datasets [157, 146], event descriptions are first generated to serve as the data source. To support this process, 10 undergraduate volunteers are recruited, majoring in Data Science and Information Technology. Their English proficiency ranged from good to proficient (TOEIC $\geq$ 650)[2], and they possessed basic knowledge of QA tasks. The volunteers participated in two training sessions covering lifelog concepts, lifelog QA, and the annotation workflow. In addition, a comprehensive guideline was provided to describe the overall procedure and address common annotation issues.

The QA generation is performed month by month, beginning with January 2019. Each volunteer was assigned four days of lifelog data per month, with one day overlapping among all volunteers to assess the consistency of event description generation. The process consisted of four stages, as illustrated in Figure 3.2. The QA generation process is similar to the LLQA dataset generation [157] in the description generation stages. However, for the open-ended QA generation, to save the high cost of manual annotation, and inspired by previous works on using LLMs for generating data [117], I apply a semi-automatic approach to generate QA. An annotation platform in Figure 3.3 was developed to support these tasks.

- **Stage 1: Event Review.** Volunteers reviewed all lifelog images and associated metadata for a given day to obtain an overview of the lifelogger's

[1] https://getnarrative.com/

[2] TOEIC are an English test with scores ranging from 10–990; $\geq$ 650 corresponds to an upper-intermediate level.

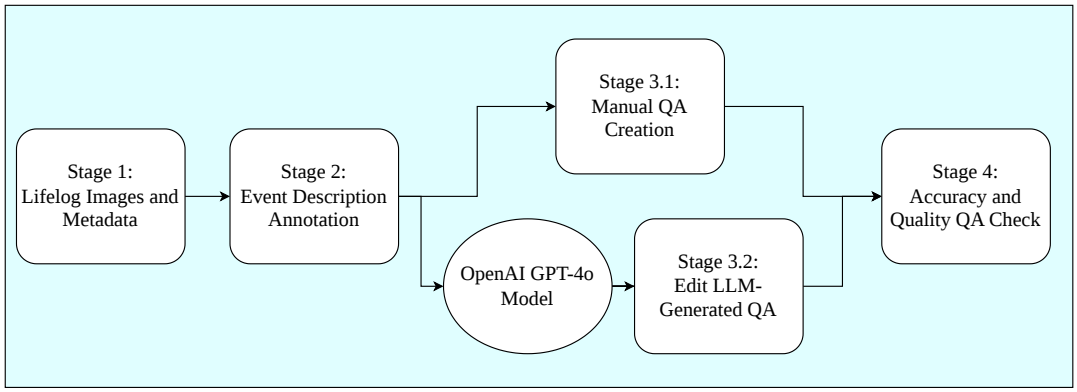


Figure 3.2: Annotation process of the OpenLifelogQA dataset

activities.

- **Stage 2: Event Description.** Volunteers segmented the data into events and wrote first-person descriptions for each. An event was defined as a sequence of images representing a single activity (e.g., eating, watching TV, or working on a laptop). Each event was associated with a start time and a representative image, while the end time was defined by the start of the next event. For example: *"I am working on my laptop"* at 2019-01-01 12:23:42, with ImageID `20190101_122342_000`.

- **Stage 3: QA Generation.** Both volunteers and the GPT-4o model [112] generated QA pairs from event descriptions. Each volunteer created 10 QA pairs per day, while GPT-4o generated 20 QA pairs, balancing quality, scalability, and human effort. Every QA pair was linked to one or more events from Stage 2, with corresponding event IDs (the ImageID of the event's start time) recorded as ground-truth contexts for retrieval tasks. The QA format was deliberately unconstrained to encourage diversity and practicality. Complex aggregation questions are particularly emphasised that require reasoning or calculation (e.g., counting or measuring duration). For instance: *"How long did I drive from my house to the church on the morning of January 21, 2020?"*.

- **Stage 4: Quality Control.** Volunteers conducted a final review to verify the accuracy of QA pairs and event descriptions. One day per month was

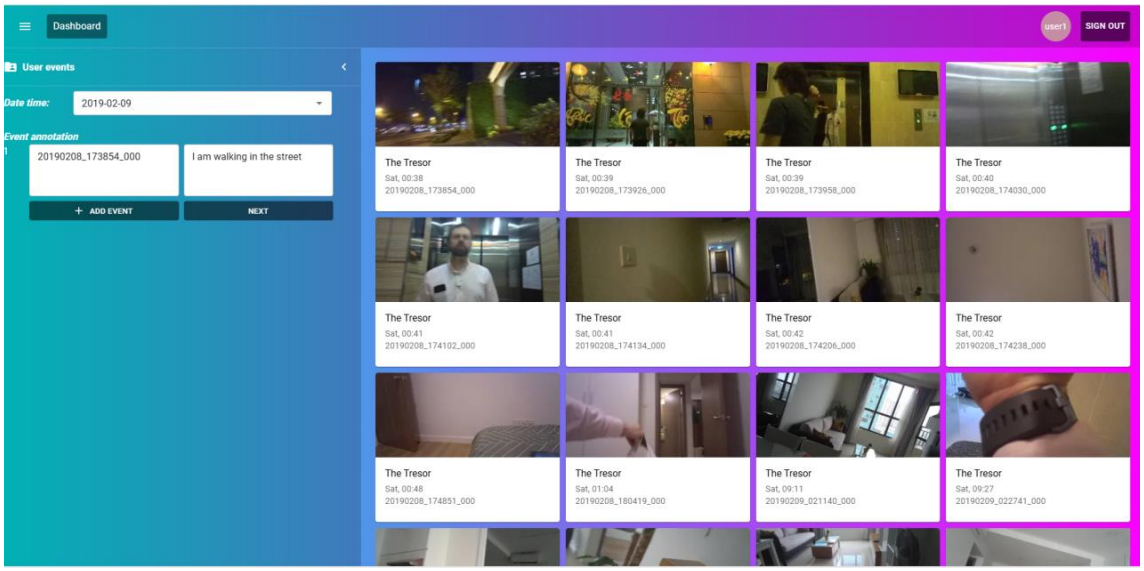


Figure 3.3: Annotation interface for OpenLifelogQA dataset

annotated by all volunteers to measure consistency in event description creation. The BERTScore [185] between event descriptions generated by annotators is measured to calculate the semantic similarity. A threshold of 0.85 was adopted as the acceptance criterion. This value was determined empirically through the distribution of pairwise BERTScore over all overlapping days, and description pairs drawn from across that distribution were then manually inspected to establish the level at which two descriptions convey substantially the same meaning. Pairs scoring at or above 0.85 were consistently judged to be semantically equivalent, differing only in wording, and the threshold was therefore fixed at this value. Since every annotator is freely generating QA, it is infeasible to measure the inner annotator agreement. Instead, volunteers cross-reviewed and corrected GPT-4o outputs, resolving errors, duplicates, and inconsistencies before finalising them as official QA. There are about 200 QA pairs that are corrected after the cross-check.

Dataset Information

After the QA generation process, the OpenLifelogQA dataset [162] contains 27,705 event descriptions and 14,187 QA pairs. This data spans 514 days of lifelog data,

Table 3.1: Statistics on three sets of OpenLifelogQA.

Set	Question Type			Average Length	
	Atomic	**Temporal**	**Aggregation**	**Question**	**Answer**
Training	6578	946	3653	12.74	6.67
Validation	895	131	476	11.99	5.74
Testing	882	156	470	12.23	5.73

which means an average of 54 events and 28 questions per day. The events and QA pairs are independent, as a QA pair can involve between 1 and several events, and some events can relate to no QA. The average number of events for each QA pair is 1.64 events. Table 3.2 shows some examples of QA pairs and ground-truth contexts of the OpenLifelogQA dataset.

The dataset is split into three sets for model development and evaluation, including training, validation, and test sets. The splits were determined by the month of lifelog data; with all lifelog data and QA from January 2019 to February 2020 for training, April and May 2020 for validation, and March and June 2020 for testing. This splitting strategy aims to avoid data leakage in model development and evaluation. Statistics of the three datasets are shown in Table 3.1. The training set contains 11,177 QA pairs, while the validation and testing sets each include approximately 1,500 QA pairs. An average question has about 12 words, and answers have about 6 words. This length suggests concise, direct questions and answers in real-world applications.

The questions are divided into three categories: atomic, temporal, and aggregation, following classifications used in prior studies [146].

- **Atomic questions** can be answered using a single context, such as "*What time did I have lunch yesterday?*".
- **Temporal questions** involve reasoning about the sequence of events in lifelog data, asking about what happened before or after a known event, such as "*What did I do after having lunch yesterday?*".
- **Aggregation questions** require synthesising information across multiple events

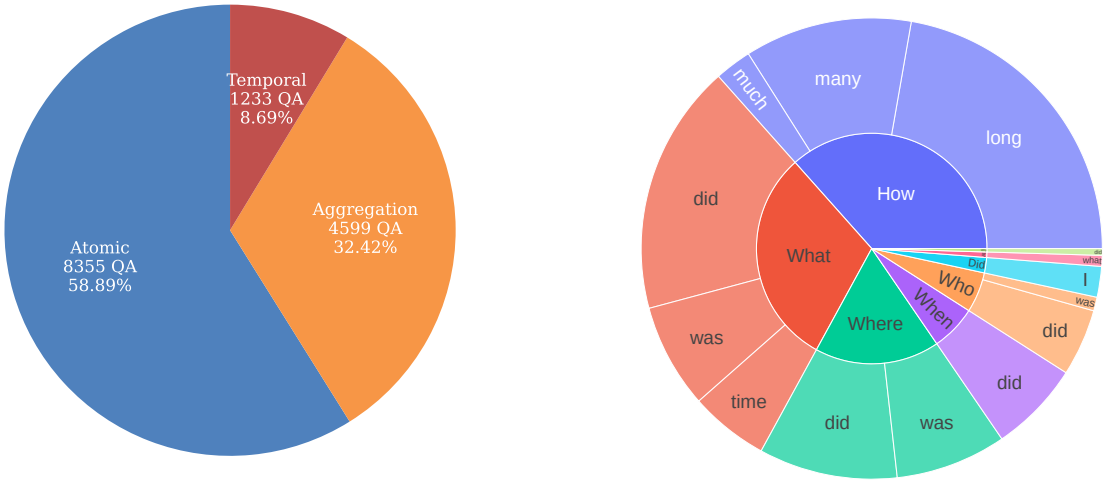
Table 3.2: Some QA pairs and ground-truth context examples

Question	Answer	Context	Type
How long did I spend at the store on the evening of June 26, 2020?	I spent 23 minutes at the store.	20200626_203417_000: I am at a store, 20200626_205727_000: I am driving away from the store	Aggregation
How many times did I watch TV on June 1, 2020?	I watched TV 3 times.	20200601_115751_000: I am watching TV. 20200601_165938_000: After dinner, I started watching TV. 20200601_182032_000: I stopped working and started watching TV again.	Aggregation
When did I go to the kitchen to make a smoothie on June 21, 2020?	I went at 11:02 AM.	20200621_110203_000: I go to the kitchen to make a smoothie	Atomic
What did I do before preparing to leave the house on June 21, 2020?	I watched TV.	20200621_061246_000: I continue to sit and watch TV. 20200621_062502_000: I have left the house and am moving by car	Temporal

to derive an answer, such as calculating a duration, count, or total. An example is: "*How long did I have lunch yesterday?*".

The distribution of these types of questions is illustrated in Figure 3.4a. An analysis of the distribution of question words in Figure 3.4b. The distribution of question words shows that the dataset reflects realistic lifelog use cases, with a strong focus on "what", "where", "when" and "how" questions, and less attention to more exploratory or uncommon queries.

One important limitation of the OpenLifelogQA dataset is that all data were collected from a single lifelogger. This problem hinders event diversity and may introduce bias in QA model evaluation. Additionally, since no two individuals live identical lives or experience the same events, incorporating data from a diverse range of people is essential for effectively evaluating the generalizability of QA models. This reason motivates me to use DCU-TimelineQA, which is discussed in the



(a) Question type distribution

(b) Question word distribution

Figure 3.4: Analysis on questions in OpenLifelogQA dataset

following section.

3.2.2 DCU-TimelineQA Dataset

TimelineQA [146] is a synthetic dataset for question-answering spanning long-term lifelog data from imaginary people. The authors proposed an automatic generation process to generate a dataset of 600,000 QA pairs. The use of synthetic data for lifelog QA is for controlled, scalable, and privacy-safe evaluation of QA models over personal data. The three-stage generation framework ensures the quality, correctness, and answerability of generated question–answer pairs. In the first stage, it begins with creating a persona for the subject. The persona includes important information such as date and place of birth, gender, educational and professional records, family members, preferences, and hobbies. The age of the subject ranges from 18 to 75 years old, with names and genders randomly selected from a dictionary of names and binary genders. Other information is generated from a model, depending on the probability distribution of predefined templates. In the second stage, episodes in life are created via template-based selection. The episodes are descriptions of events that happen in life, such as “I have lunch with my friend Lysa" or “I graduated from Dublin City University on 12/10/2020." The episodes

are generated at different levels of granularity, based on timescales such as annual, monthly, weekly, and daily. Several templates are predefined and generated for each granularity, such as tourism for annual, shopping for weekly, and eating for daily use. There are also some super episodes, which include several sub-episodes to provide finer granularity. An example of this is a trip with multiple destinations and an itinerary for each day. The third stage is QA generation, in which the authors attempt to convert every episode into questions of what, where, when, and who. By identifying the subject, objects, time, and location, they convert an episode to a question by adding question words and extracting answers. For example, with the episode "I went to a Japanese restaurant with Sarah on October 7th and ate sushi", the questions can be "when did I have Japanese food?, when did I meet Sarah?, and where did I eat on October 7th?". The density of episodes can be configured with three options (dense, medium, and sparse), in which the dense setting generates more weekly and daily episodes, while the sparse setting generates fewer.

However, the 600,000 QA sets in the original paper are not available, so I created a new TimelineQA dataset from the same framework used in the paper [146], with a single person's data using the three-stage process described earlier. I call it a DCU-TimelineQA set. The subject in this dataset is a 39-year-old extroverted woman born in Sheffield, United Kingdom. She studied Chemical Engineering at KU Leuven, Belgium, graduating in 2005, and later earned a graduate degree in Labour/Industrial Relations from the Chinese University of Hong Kong in 2010. She works in public relations and has held two jobs. Her hobbies include writing, gardening, yoga, and listening to audiobooks and podcasts. With a medium density setting, the dataset comprises 51,883 QA pairs and 20,224 events. Around 14,784 yes/no questions are discarded as they are too simple and not suitable for a generative QA task. The dataset is split into training and test sets with a ratio of 90:10, resulting in 3554 QA pairs for the test set. The answers in this dataset are span-based, which can be extracted directly from the contexts.

Table 3.3 shows some statistics on the training and testing sets of the DCU-

Table 3.3: Statistics on train and test sets of DCU-TimelineQA.

Set	# atomic question	# aggregation question	Avg context per question	Avg question length	Avg answer length
Training	24335	7651	1	8.65	2.23
Testing	2729	825	1	8.64	2.29

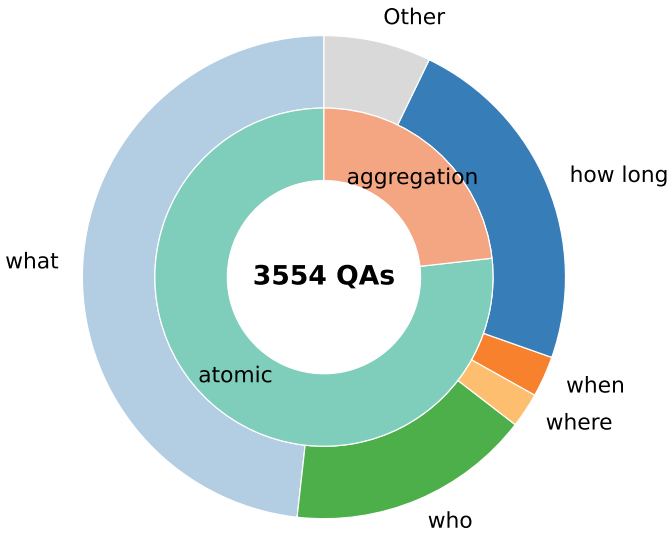


Figure 3.5: Analysis of question words and types distribution of the DCU-TimelineQA test set

TimelineQA dataset. All the questions in the two sets have only one ground-truth episode that contains the answer to the question, so the average context per question is 1. However, there are some "how long" questions in the datasets, so I still classify them as aggregation questions. The average lengths of questions and answers in the dataset are approximately 8.65 and 2.23, respectively. These numbers are smaller than the OpenLifelogQA dataset due to the template-based nature of QA. Figure 3.5 shows pie charts depicting the distribution of question words and types. The number of atomic questions is three times more than the aggregation questions. "What" questions account for nearly 50% of questions, follow by "who" and "how long". This dataset is synthetic, but it provides a good benchmark for evaluating lifelog QA models on a wide range of questions.

3.3 Operating Constraints

As is typical for emerging research areas, I clearly specified the boundaries within which this study was conducted. These constraints are identified as follows:

- This research is limited to a predefined range of multimodal lifelog data (18 months) obtained from an existing and available collection [56]. To avoid ethical concerns associated with personal data collection, no new lifelog data is collected for this study. Furthermore, the existing dataset has undergone systematic redaction, including the blurring of faces, readable text, and identifiable objects, to protect the privacy of individuals captured in the recordings. The redaction can reduce the richness of the visual information available, potentially limiting the model's ability to answer certain questions that rely on such redacted details. Only additional annotations on QA pairs are performed through a collaboration with the University of Science, Vietnam National University. Additionally, a synthetic lifelog dataset (DCU-TimelineQA) is utilised, as it raises no personal privacy concerns and allows for the generation of a wide and diverse range of questions for comprehensive evaluation.

- The newly annotated lifelog QA dataset is limited in size due to the substantial cost associated with annotating question–answer pairs for lifelog data. As a result, the dataset will not reach the scale of hundreds of thousands of questions commonly found in large-scale QA benchmarks. Nevertheless, its size is sufficient for the purposes of this research: at 14,187 QA pairs, it is comparable to the largest existing realistic lifelog QA dataset, LLQA [157], while covering a substantially longer period. The held-out test set of 1,508 questions spans all three question types and is large for showing the differences observed between models. The training split of 11,177 pairs is used to adapt already pre-trained LLMs rather than to train them from scratch, which requires far less data, and the consistent gains reported in Chapter 6 confirm that this is adequate for the adaptation performed.

- The use of closed-source LLMs is limited due to computational and financial resources. The study focuses on developing and experimenting with open-source models. However, the GPT-4o model is employed in a limited number of experiments solely as a comparative baseline to evaluate the performance of the proposed approach.
- The evaluation scope of this study is limited, with primary emphasis placed on accuracy-based metrics. Although the QA model and retrieval system are intended to support comprehensive evaluation, the evaluation focuses only on accuracy metrics rather than time or computational requirements.

These constraints are maintained in this PhD research and serve as limiting factors for concentrating the research effort.

3.4 Evaluation Metrics

This section describes the Recall@K and Precision@K for retrieval evaluation and several metrics, ranging from lexical-matching, semantic-matching and LLM-as-a-judge, for QA to assess the performance of the proposed study.

3.4.1 Retrieval Metrics

For interactive lifelog retrieval, the system is evaluated in the LSC challenge, with metrics for the KIS and QA tasks defined in (2.2), and the ad hoc task defined in (2.3). For automatic lifelog retrieval, I utilise conventional metrics in the retrieval field, such as Recall@K and Precision@K, to measure the accuracy with which the retrieval system identifies the context. These metrics are calculated by comparing a list of top K retrieved lifelog contexts to a list of ground-truth contexts. Other ranking-based metrics, such as Mean Reciprocal Rank (MRR) and Normalised Discounted Cumulative Gain (NDCG), are not considered in this work. These metrics emphasise the relative ranking positions of relevant contexts within the retrieved list. However, in the lifelog context retrieval for question-answering, the exact rank

order of relevant contexts is not critical. As long as the correct contexts appear within the top-K retrieved results (e.g., all three relevant contexts are retrieved in the top three positions), their internal ordering is not important. Let $\mathcal{G}$ denote the set of relevant ground-truth contexts and $\mathcal{R}_K$ denote the set of top-K retrieved contexts.

The precision at K is defined as:

$$\text{Precision@}K = \frac{|\mathcal{R}_K \cap \mathcal{G}|}{K} \tag{3.1}$$

The recall at K is defined as:

$$\text{Recall@}K = \frac{|\mathcal{R}_K \cap \mathcal{G}|}{|\mathcal{G}|} \tag{3.2}$$

3.4.2 QA Metrics

When evaluating the performance of lifelog QA models, I primarily focus on accuracy. Accuracy measures how closely the model's answer aligns with the ground truth answer, and it can be evaluated through lexical matching, semantic matching, and LLM-as-a-judge approaches. Lexical matching involves comparing words between the predicted and ground truth answers using metrics such as Exact Match, F1 score, BLEU score [115], and ROUGE score [93]. The ROUGE scores are particularly useful in this field due to their prevalence. I use the ROUGE-L version, where L stands for longest common subsequence, which measures the longest common subsequences of the predicted answer $\hat{a}$ and the ground-truth answer a. β controls the balance between recall and precision, and I choose the value of β to balance both recall and precision. The formula of ROUGE-L is as follows:

$$\text{ROUGE-L} = \frac{(1+\beta^2)\, P_{\text{LCS}}\, R_{\text{LCS}}}{R_{\text{LCS}} + \beta^2\, P_{\text{LCS}}} \tag{3.3}$$

with P_{LCS} and R_{LCS} are as follows, with $LSC(\hat{a}, a)$ the length of the longest common subsequence:

$$P_{\text{LCS}} = \frac{\text{LCS}(\hat{a}, a)}{|\hat{a}|}, \quad R_{\text{LCS}} = \frac{\text{LCS}(\hat{a}, a)}{|a|} \tag{3.4}$$

However, lexical matching can sometimes fail to capture the accuracy of an answer when there is lexical variation but semantic equivalence. To address this, semantic matching is employed to evaluate accuracy by calculating the semantic similarity between the predicted and ground-truth answers. BERTScore [185] is a well-known metric for this purpose, using RoBERTa [96] models to assess semantic similarity before aggregating the results into a final score. The RoBERTa model first generates semantic embeddings of the ground-truth and predicted answers. These two embeddings are same-space vectors; then a cosine similarity of the two vectors is measured to produce BERTScore. The detailed formulation of BERTScore is given as follows. Let $a = (w_1, \ldots, w_n)$ denote the ground-truth answer and $\hat{a} = (\hat{w}_1, \ldots, \hat{w}_m)$ denote the predicted answer.

Each token is mapped to a contextualised embedding using RoBERTa:

$$\mathbf{h}_{\hat{w}_j} = \text{RoBERTa}(\hat{w}_j), \quad \mathbf{h}_{w_i} = \text{RoBERTa}(w_i) \tag{3.5}$$

The precision of BERTScore is computed as:

$$P = \frac{1}{m} \sum_{j=1}^{m} \max_{i \in \{1,\ldots,n\}} \cos\big(\mathbf{h}_{\hat{w}_j}, \mathbf{h}_{w_i}\big) \tag{3.6}$$

The recall is defined as:

$$R = \frac{1}{n} \sum_{i=1}^{n} \max_{j \in \{1,\ldots,m\}} \cos\big(\mathbf{h}_{w_i}, \mathbf{h}_{\hat{w}_j}\big) \tag{3.7}$$

Finally, the F1 score of BERTScore is computed as:

$$\text{BERTScore}_{F1} = \frac{2PR}{P + R} \tag{3.8}$$

LLM-based evaluation [32, 48] is an emerging method for assessing QA accuracy,

where an LLM is prompted to rate the accuracy of predicted answers in comparison to the ground truth. This method requires a clear definition of accuracy in the prompt to guide the LLM's evaluation process. State-of-the-art LLMs, such as GPT-4o [112], can be utilised for this task due to their capabilities across various domains. A prompt containing the instruction, ground-truth answer, and predicted answer is provided to the GPT-4o model, which rates the accuracy of the predicted answer on a scale from 1 to 5. The resulting score is then normalised by dividing it by 5, which maps the five rating levels onto the range $[0.2, 1]$ rather than $[0, 1]$. Consequently, LLM Score values are not directly comparable with metrics that possess a true zero, such as exact match or ROUGE. I retain the division by 5 for consistency with [48], but emphasise that the scores reported here should be compared across systems within this work, or converted to range $[0, 1]$ before comparing results reported elsewhere. The prompt is provided in the Appendix A.2. Let $\{(\hat{a}_i, a_i)\}_{i=1}^{N}$ denote a set of predicted and ground-truth answer pairs, where $\text{LLM}(\hat{a}_i, a_i)$ represents the score produced by an LLM-based evaluator. The average LLM score over the dataset is computed as:

$$\text{LLM Score} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{LLM}(\hat{a}_i, a_i)}{5} \tag{3.9}$$

In the original TimelineQA paper [146], the authors used the F1-score to evaluate the accuracy of answers. I also use the same metric for evaluating generated answers in the DCU-TimelineQA test set to have the same way to compare with the original work's result. The number of overlapping tokens between the predicted and ground truth answers is $|\hat{a} \cap a|$. Token-level Precision and Recall are computed as:

$$P = \frac{|\hat{a} \cap a|}{|\hat{a}|}, \qquad R = \frac{|\hat{a} \cap a|}{|a|} \tag{3.10}$$

The F1-score is then the harmonic mean of Precision and Recall:

$$\text{F1} = \frac{2 \cdot P \cdot R}{P + R} \tag{3.11}$$

A predicted answer that exactly matches the ground truth yields an F1-score of

1, while a completely non-overlapping answer yields 0. This token-level F1-score is tolerant of partial matches, making it more suitable than an exact match (EM) for evaluating answers that may differ in phrasing but share the same content words.

3.5 Experimental Settings and Justification

Every model choice, prompt, decoding parameter and training hyperparameter in this thesis comes from one of four sources: published prior work or vendor guidance; the hardware constraints set out in Section 3.3; empirical selection on the validation split of OpenLifelogQA; or informal refinement by inspecting model outputs during development. This section states the principle applied in each case, and Appendix A.4 tags every individual setting with its source.

Model selection. Models were selected to support the comparisons the RQs require rather than to maximise absolute performance, along three axes: open-weight against proprietary, reasoning-oriented against instruction-tuned, and small against large within one family. This yields the matched pairs used in Chapters 5 and 6: Qwen3 8B against Llama 3.1 8B Instruct at equal parameter count, Qwen3 8B against Qwen3 14B within one family, and Qwen3 against proprietary GPT-4o. Selection was based on the most current published benchmark results when each experiment was run; because the field moved quickly, the models in Chapter 5 predate those in Chapter 6 and the two chapters are not directly comparable.

Prompt design. The prompt structure in Chapter 5 follows published prompt-engineering guidance[3] in assigning an explicit role, a single objective and a fixed answer delimiter, and follows the chain-of-thought literature [166] in the form of its worked examples; the six-section prompt in Chapter 6 adds a question-category section and a specified output format. Beyond these inherited elements, the prompts were refined empirically by running candidate wordings on the validation split and revising them by inspection. This refinement was iterative and informal, with no systematic search over prompt variants carried out, and the final prompts are a

[3]https://cloud.google.com/discover/what-is-prompt-engineering

working configuration arrived at by trial rather than an optimum.

Decoding settings. Decoding parameters differ across chapters because the experiments differ in purpose. Chapter 5 uses a temperature and top-p of 0.5 for all generators, a moderate-sampling configuration adopted from common practice in the RAG literature and held fixed so that comparisons across models, prompts and context configurations are not confounded by decoding. In Chapter 6, non-fine-tuned models run at a temperature and top-p of 1 in line with vendor guidance for reasoning-oriented models; the temperature is lowered to 0.1 after supervised fine-tuning, where the model must reproduce a fixed answer format and sampling diversity is unhelpful; and a temperature of 1 is retained during GRPO, which requires diverse samples within each group by construction.

Training hyperparameters. Fine-tuning followed the reference configurations of the Unsloth framework [4] for supervised fine-tuning and GRPO. Batch sizes and gradient-accumulation steps were fixed by the memory of a single GPU rather than chosen for their effect on accuracy, while learning rate, weight decay, LoRA rank and the number of epochs were varied on the validation split and the best-scoring configuration retained.

Prompt stability and run-to-run variation. This is one limitation of this design. All reported results come from a single generation run per configuration: seeds were not fixed or varied systematically, no configuration was repeated, and no paraphrase variants of the final prompts were evaluated, so no measurement of intra-prompt stability (variation of one prompt across repeated runs) or inter-prompt stability (variation across equivalent rephrasings) is available. No reported difference can be attributed to the manipulated variable with confidence unless it is large relative to the sampling variation of a decoder run at non-zero temperature; and the prompt comparison in Section 5.4 is not evidence that the strategies perform equivalently, only that a single run of each did not separate them. Claims are phrased accordingly throughout, and this limitation is revisited in Section 7.3.

[4] http://unsloth.ai/

3.6 Chapter Summary

This chapter has presented the research methodology, datasets used in experiments, research operating constraints and all of the metrics that are used for evaluating the lifelog context retrieval and QA. Two datasets were introduced, including OpenLifelogQA and DCU-TimelineQA. They range from realistic to synthetic lifelog data with a diverse set of questions about personal lifelog. The operating constraints were also discussed, including the use of available and redacted lifelog data, the limitations of data construction, and the evaluation. The evaluation framework spans both retrieval and QA tasks, employing standard information retrieval metrics and lexical, semantic, and LLM-as-a-judge for answer generation evaluation to provide a comprehensive assessment of the approach's performance.

Building on this foundation, the following chapters present the experimental development and evaluation of models for each research question. The next chapter addresses RQ1 by describing the lifelog context retrieval problem and the proposed solution for identifying relevant contexts to support lifelog question answering, while Chapter 5 and 6 provide details of the fine-tuning and reasoning experiments conducted on OpenLifelogQA and DCU-TimelineQA for RQ2 and RQ3.

Chapter 4

Lifelog Context Retrieval

This chapter aims to resolve RQ1 *What is the most effective context retrieval method to support the LifelogQA task when compared to standard retrieval approaches?*. I will first introduce the lifelog context retrieval task in Section 4.1, and then discuss an interactive lifelog retrieval system in Section 4.2. An automatic lifelog context retrieval and reranking model is proposed to enhance the accuracy of context retrieval, as described in Section 4.3. Finally, the discussion and conclusion for the answer to RQ1 are presented in Section 4.4.

4.1 Introduction and Problem Statement

Lifelog retrieval is an application of information retrieval in which the retrieval system assists the user in finding the lifelog data related to the query. In lifelog retrieval, the required information is an event, or more specifically, a context within a lifelog. While an event is simply an activity of lifelog data, such as an image of a lifelogger having lunch, the context contains more information, including time, location, and so on. I introduce the MemoriEase system for interactive and automatic lifelog retrieval tasks. This system develops from a standard retrieval system to a conversational search and RAG-enhanced one, as illustrated in Figure 4.1. The development is based on users' needs in live challenges. The notes from users' needs from each challenge and the development based on that are described in Section

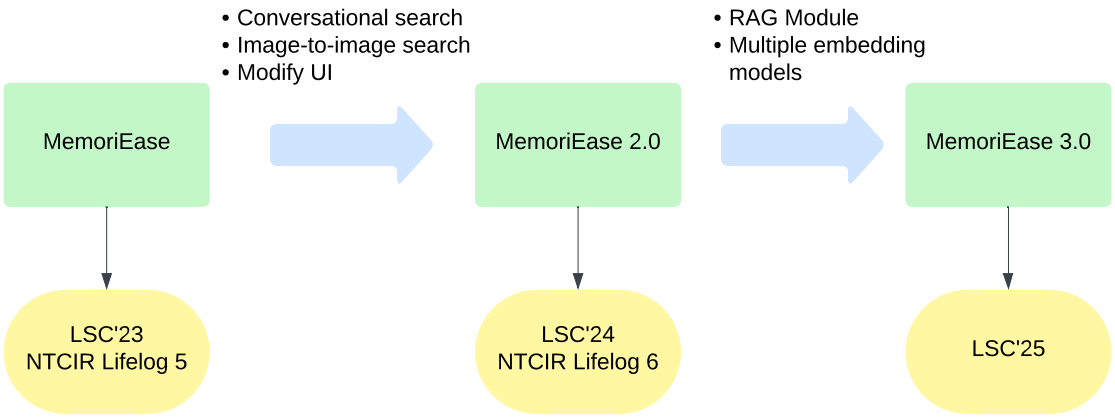


Figure 4.1: Development of MemoriEase from the first version to the 3.0 version.

4.2.2. This system serves as a baseline for the interactive evaluation of methods for lifelog retrieval and a foundation for a retrieval engine in the lifelog QA task.

For automatic retrieval in the lifelog QA task, I present the typical framework [78, 28], which consists of two stages: retrieval and reranking. While the retrieval stage is efficient for finding relevant contexts, the reranking stage enhances the accuracy of the ranked list from the retrieval stage by prioritising more relevant contexts at the top of the list. The formulation of this problem is for a set of N questions $Q = \{q_1, q_2, ..., q_n\}$ and a set of M contexts $C = \{c_1, c_2, ..., c_m\}$ with each context $c_i = (T_i, I_i, M_i)$ where I_i is the image, T_i is the text description, and M_i is the metadata (timestamp/location), the goal of context retrieval and reranking is to retrieve the top relevant contexts to the questions. Given the question $q_i \in$ Q, the goal is to retrieve a subset of contexts $C_{retrieved} \subseteq$ C. An embedding-based approach is used to retrieve the ranked list of top-K contexts $C_{retrieved}$.

The context reranking problem can be formulated as learning a ranking function $f_R(q_i, C_{\text{retrieved}})$ that assigns a relevance score to each context $c_j \in C_{\text{retrieved}}$. Given a question q_i and the initially retrieved contexts $C_{\text{retrieved}} = \{c_1, c_2, \ldots, c_K\}$, the model M computes a new relevance score s_j for each context c_j as:

$$s_j = f_R(q_i, c_j) \tag{4.1}$$

where: f_R is the reranking model, and $s_j \in \mathbb{R}$ represents the relevance score

of context c_j with respect to question q_i. The final output is a reranked list of $C_{\text{retrieved}}$ based on descending s_j values. In the following sections, I present details on MemoriEase interactive retrieval, followed by two-stage automatic retrieval for lifelog QA.

4.2 MemoriEase Retrieval System

In this section, I present the MemoriEase lifelog retrieval system, from the data processing and indexing step to the retrieval modules, QA modules, and user interface. This system aims to find relevant information from queries interactively. Users can interact with the system to explore lifelog data and refine the search to find the most relevant data. For interactive challenges, it also needs to be fast to quickly locate the correct items for submission. I also present the results of this system at various challenges, including interactive ones like LSC and automatic ones like the NTCIR Lifelog task.

4.2.1 System Architecture

The architecture of the system is built upon the embedding-based retrieval method with the use of vision-language models to encode queries and images into the same embedding space for similarity calculation. Figure 4.2 provides an overall architecture of the system, from the data processing flow in the offline stage to user retrieval in the online stage. In the offline phase, metadata, including time and location, image descriptions and captions, and embeddings extracted from images, are indexed into Elasticsearch. During the online phase, when a user initiates a chat with the conversational user interface, the system determines whether it is a question or a query. If it is a query, it will be encoded into the embedding by the same embedding model that encodes images. If it is a question, it will go through a RAG module to be retrieved by description. The retrieved data then passes through an LLM to generate a summary or an answer to the question before being returned to the user

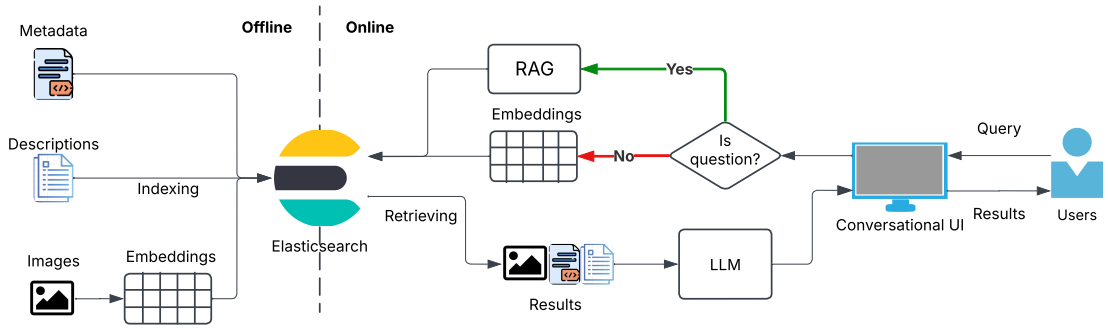


Figure 4.2: MemoriEase overall architecture and flow

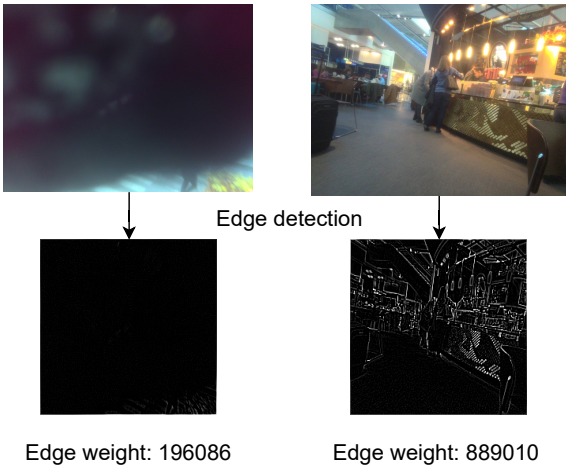


Figure 4.3: Comparison of low and high edge weight of blur and normal images

interface for display. More details about every step of the processing are described in the following subsections.

Data Processing and Indexing

The lifelog dataset used for LSC from 2022 to 2025 is an 18-month lifelog dataset comprising 725,000 PoV images, metadata for images including time and location, and visual concepts that describe the visual content of the images. The PoV images are captured from a Narrative Clip[1] wearable camera with a 1024x768 pixel resolution. All images are fully redacted and anonymised by removing faces and readable text, as well as manually filtering out specific scenes and activities to comply with local privacy requirements.

[1]https://getnarrative.com/

From the original lifelog images, I use the OpenCV package [2] in Python to calculate the edge weight of objects in the images. The edge weight is the numerical value assigned to an edge of objects in the image. A low edge weight indicates that the image has no distinct objects or appears blurry. A demonstration of this approach can be seen in Figure 4.3. I chose the threshold of quantile 0.13 with the weight value of 350,000 to remove 94,450 images. This weight value results in the removal of a large proportion of blurred or covered images through random examination. In the first version of MemoriEase, I implemented an event segmentation algorithm inspired by [34, 49]. This algorithm groups images into events, a sequence of images illustrating an activity, such as eating or watching TV. I use BLIP embedding [91] of images to calculate the cosine similarity between images and their next or previous images in the timeline. If the similarity is higher than a predefined α (0.7 on the scale of 1), I group them in a single event. The α is chosen to balance between precision and recall of similarity-based grouping. It is high enough to ensure only semantically coherent events are merged, while low enough to capture similar images that may differ in views but still be in the same event. The result of this segmentation is 173,269 events. I then choose an image in the middle of the sequence of images of events to represent the event. However, in the competition, I observed that the representative image used for retrieval can lead to failed retrieval due to the slight difference between the desired and representative images. In the latest version, I discard event segmentation to search on images, thereby avoiding this problem. The images are then encoded into a 256-dimensional vector embedding by the BLIP2 model [90]. This model is a state-of-the-art visual-language model with high performance on zero-shot text-to-image retrieval.

From the latitude and longitude metadata for images, I obtain information about the city and country. I also use the semantic name provided in the dataset, which is the name of the location the lifelogger visited, such as “home”, “Dublin City University”, “Dublin Airport”, etc. Three attributes, city, country, and semantic

[2] https://opencv.org/

Table 4.1: Properties of the Elasticsearch index

Attribute	Description	Data type	Example
ImageID	Unique identifier for each lifelog image	text	20190101_103717
OCR	Text detected in the image	text	Cocacola
semantic name	Name of the current place	text	Home, Dublin airport
city	Name of the city and country	text	Dublin, Ireland
local time	Date and time at the location	datetime	2019-01-01 10:37:17
hour	Hour extracted from local time	integer	10
date of week	Day name extracted from local time	text	Tuesday
is weekend	1 if the day is Saturday or Sunday, else 0	binary	0
time period	Part of the day (e.g., Morning, Evening)	text	Morning
description	Text description of what the user sees	text	A room with curtains
BLIP2 vector	Vector representing the image from BLIP2	array	[0.2, 0.4, ..., 0,4]

name, are used as filters for the retrieval module.

From the UTC time provided in the dataset, I convert it to local time using the city's time zone. It matches better with the LSC query because most queries use the local time zone. I also extract several other time-related attributes for filtering, such as hour (in 24-hour format), day of the week, whether it is a weekend, and time period (morning, afternoon, etc.). These filters are helpful for time-constrained queries, such as shopping on a weekend afternoon. All the extracted data is then formed into a table of 11 attributes to index into an Elasticsearch[3] index. Table 4.1 provides the details of attributes in the Elasticsearch index for indexing.

Retrieval Modules

The retrieval modules serve as the model for retrieving lifelog images from both text and image inputs. MemoriEase has two retrieval modules: text-to-image retrieval

[3]https://www.elastic.co/

and image-to-image retrieval. For text-to-image retrieval, when users input a textual query, the query is processed to extract relevant filters. MemoriEase employs a heuristic approach to extract time-related filters, including date, hour, and time period. For location-related filters, MemoriEase compares the semantic name and city dictionary in the Elasticsearch index, and if the query matches the words in these fields, MemoriEase uses them as filters. In addition, several advanced filters are available for expert users, allowing them to use the @ symbol before specific keywords, such as weekend, start_hour, end_hour, OCR, and location, to indicate the desired value for the filters. These filters are applied before the cosine similarity calculation in Elasticsearch to reduce the number of candidate images.

For text-to-image search, all the filter-related words in the original query are removed to get the processed query and are encoded into a vector embedding by the BLIP2 embedding model. Elasticsearch calculates the cosine similarity between K candidate image embeddings and the query embedding. I use K around 100 to balance the time of calculating similarity and the accuracy of retrieval. The result of this is a ranked list of 100 images that are relevant to the query. The text-to-image search is useful for the KIS sub-task in LSC, where the query requires several filters and complicated semantics.

With the temporal nature of lifelog data, when an event happens before or after another event, such as "I am watching TV after having dinner.", a temporal search function is needed to support this case. There are three text-free search boxes for users to input the main event, the preceding event, and the following event, along with a time gap constraint to indicate the maximum time difference between the main event and the temporal events. The preceding and following events are optional, allowing users to input either both or only one of them. The system will first perform a text-to-image search for the main query and then use the datetime of each retrieved image and the time gap constraint as a filter to search for temporal events. The final ranked list is based on the weighted average score of the cosine similarity between the main event and the temporal events, ranking the top 100

relevant images.

Conversational search is proposed in the second version of MemoriEase for LSC'24 to support natural and engaging search for lifelog. In addition, the multi-turn search in conversation mimics the process of memory recall, with new information emerging after every search and review of the images. Thanks to the development of LLMs, they can aggregate information from multiple turns of conversation and produce human-like responses for users' queries or questions. The GPT-4o-mini model [112] is used as a conversational agent to aggregate information from multi-turn conversations and generate a summary of the query's results. It is prompted to generate a response with constructive information and suggest further directions for users' search.

For queries in the ad hoc subtask, I propose an image-to-image retrieval approach. This type of retrieval also starts with a query, and the same process as text-to-image is applied. However, after the returned results are displayed on the interface for users, they can choose images from the results as the input for the next iteration of the search. They can iteratively search until no more matching images are found. This search is implemented through the semantic matching of image embeddings. Image embeddings are extracted from the visual input and perform a cosine similarity search in the Elasticsearch index, returning a ranked list of new results. This search proves effective in finding specific objects, accounting for a small fraction of the image, such as "stop before a red traffic light" or "shopping on an empty shelf". The text-to-image retrieval only finds several relevant images of "red traffic light", but when using these images to search again, more similar images are returned.

RAG for QA Module

RAG is introduced in the third version of MemoriEase, with the aim of supporting participants in solving the QA subtask. A RAG pipeline includes two stages: a retrieve information stage and a generate answers to questions stage, which is depicted

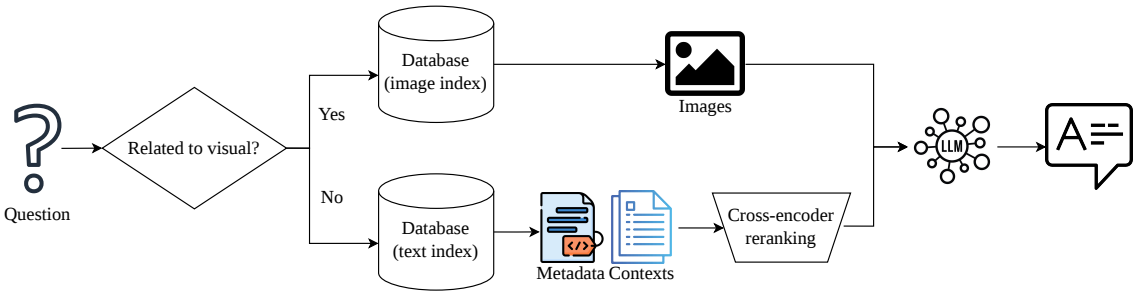


Figure 4.4: RAG pipeline for MemoriEase 3.0

in Figure 4.4.

Depending on the required information for the questions, such as those related to the visual content of lifelog images (e.g., the location where the key is left) or the metadata of the lifelog (e.g., the city of travel), MemoriEase employs different strategies to solve the questions. A rule-based approach based on question words is used to classify the type of question. If it is a visual-related question, the question is encoded into an embedding and searched for the top K relevant images (K from 10 to 20). All the images and the question are prompts for an LLM model to generate an answer. GPT-4o-mini [112] is chosen as the LLM thanks to its capability to read images and reasonable cost.

For metadata-related questions, the retrieval stage differs in that MemoriEase combines the weighted average of the cosine similarity between the query image embedding and the query description embedding. The output is a ranked list of image descriptions and their corresponding metadata. I further refine the list by using a cross-encoder reranking model, introduced in the next section, to enhance the ranking of relevant descriptions. The reranked list is passed to LLMs with the question to ask for an answer. The answer is displayed on the user interface, accompanied by corresponding images for further verification.

User Interface

MemoriEase 3.0 provides a multi-modal search interface to support the diverse retrieval scenarios encountered in interactive lifelog search challenges. The system provides three interface modes, each tailored to a distinct retrieval subtask pre-

sented in Section 2.1.3, including KIS, QA and ad hoc. The conversational search interface supports natural language querying for the KIS and QA subtasks, presenting results alongside AI-generated summaries in a chat layout. The temporal search interface extends this capability with structured triplet-based querying, allowing users to anchor a target event between its preceding and following context within a configurable time window. Finally, the image-to-image retrieval interface supports ad hoc search, where users select one or more images as visual anchors and iteratively refine results based on visual similarity. Together, these interfaces form a unified but flexible front-end that accommodates both textual and visual retrieval strategies during the competitive challenge.

MemoriEase introduces a conversational search interface in the second version to adapt to LSC 6-hint search in the KIS subtask and textual answer requirement in the QA subtask. When a user inputs a query into the search bar, the result page is displayed, showing the user's chat on the left side and the results with a grid of images on the right side, as shown in Figure A.1. The system will provide users with answers to their questions or summaries of results for queries. On the results side, each image is accompanied by its semantic name, indicating the location where the image was captured, along with the date and time in a human-readable format. The underline in the datetime implies that it can be clickable to show lifelog images and a highlighted clicked image.

For advanced search with filters and temporal search for the KIS sub-task, users can input three queries for the main event, the preceding event, and the following event, with a time gap constraint to facilitate the search. The result will display on a triplet of 3 images, with the large image in the middle showing the main event, while the left and right images show the previous and next events. An example of a temporal search interface is depicted in Figure A.2.

When users click on the ad hoc task, an image-to-image retrieval page will pop up with several random images for users to choose from. If you want to search for a specific query, the results will be displayed on the right side with a familiar grid

format. Users can click on several images to select as input and then click search again. The new result will be displayed with more visually similar content to the input images. An example of this is shown in Figure A.3. Overall, these interfaces support users in their search for interactive challenges. The performance of the system in these challenges will be presented in the following section.

4.2.2 From Challenge Observations to System Updates

Table 4.2: User needs and failures observed at each Lifelog Search Challenge, and the corresponding design responses in the following version of MemoriEase.

Version	**Observed need/failure**	**Design response**
1.0 (initial)	Retrieval engine matching free-text queries against 725K images for fast visual scanning	BLIP2 text-to-image retrieval in Elasticsearch; heuristic time/location filters; grid results
2.0 (LSC'23)	Text-to-image search returns diverse near-misses, as BLIP favours global scene over small objects ("red traffic light")	Image-to-image retrieval with iterative relevance feedback; retrieved image reused as next query
2.0 (LSC'23)	Event-level representative images miss when target differs (grey t-shirt query)	Event segmentation dropped; indexing and search at image level
2.0 (LSC'23)	Progressive KIS hints and QA subtask need incremental query refinement, not single-shot queries	Conversational search via GPT-4o-mini; aggregates turns, summarises results
2.0 (LSC'23)	Users need surrounding images to confirm activity context	Clickable timestamp opens image sequence, selected image highlighted
3.0 (LSC'24)	Manual image inspection too slow for QA under time pressure	RAG QA module to suggest answers for questions
3.0 (LSC'24)	Generated answers can't be trusted without verification	Answers displayed alongside supporting evidence images

The three versions of MemoriEase were not designed in advance and then evaluated, but each was built in response to needs and failures observed during the preceding challenge. This subsection describes that chain of development, so that every feature described in Section 4.2.1 can be traced to the observation that motivated it. Two sources of evidence informed the design of each iteration. The first is my own use of the system as its expert operator during the live LSC sessions, which exposed the operations that were slow or impossible under time pressure. The second is a post-hoc analysis of failed queries after each challenge, which identified the

retrieval or interface behaviour responsible for each miss. Table 4.2 summarises the resulting requirements and the design response to each.

4.2.3 Performance at Interactive and Automatic Benchmarks

In this section, I provide information about the performance of the MemoriEase system in three LSC challenges and two NTCIR Lifelog tasks. LSC is an interactive retrieval challenge in a live competition, as presented in Section 2.1.3, while the NTCIR Lifelog task is a benchmark for evaluating lifelog retrieval automatically, as presented in Section 2.1.4. An analysis of the strengths and drawbacks is also discussed to draw lessons learned from each challenge. For the LSC challenges, the reported results are from the expert user only (referred to in Section 2.1.3) for a fair comparison, as the LSC'25 is only for expert users due to the restriction of travelling to the ICMR 2025 Conference in the United States of America. The scores are computed by metrics defined in Formula (2.2) and (2.3).

Results at LSC'23

LSC'23 [57] was the first time the MemoriEase system participated. There were 13 teams in LSC'23, excluding the baseline from LSC'22, and the MemoriEase system ranked in 8th position, tied with the Memoria team. MemoriEase found 7 out of 10 correct images for KIS queries and provided 8 out of 10 correct textual answers for the QA sub-task. However, it only scored 33/100 on the ad hoc sub-task.

Figure 4.5 illustrates the performance of systems in LSC'23. MemoriEase showed average performance on the KIS sub-task compared to other systems, but it poorly found many images in the ad hoc task. I found that the BLIP embedding model [91] tends to focus on global visual features rather than specific characteristics of objects, such as blue balls or red traffic lights. For example, a query like "stopping in front of a red traffic light" was found with a lot of images before the traffic light, but not always the red light. This was a big drawback that I aimed to improve in the next version of MemoriEase.

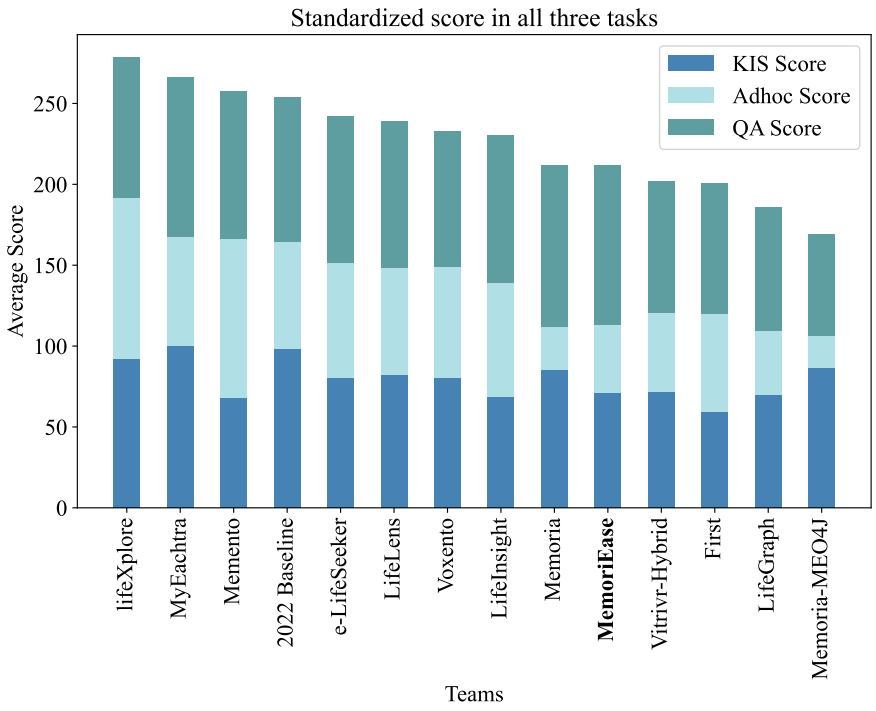


Figure 4.5: MemoriEase performance in LSC'23 expert track.

On the QA sub-task, the MemoriEase system performed very well compared to other systems, ranking in third position, as shown in Figure 4.6. The system typically achieved the highest score for answering the questions, except for failing to provide an answer for two questions. The question "I normally wear shirts, but what is the brand of the grey t-shirt that I wore at the start of COVID-19 time?" can be solved by finding the grey t-shirt from early 2020. However, the image was not found because it was segmented into an event. The representative image of the event differs from the t-shirt image, so MemoriEase was unable to find it. This is the reason I removed event segmentation in the following version of MemoriEase. This was the first time MemoriEase participated in the LSC challenge, so there were still many things that were new and needed to be adapted to the system accordingly.

Results at LSC'24

LSC'24 attracted 21 systems and 35 participants in total (one system can have multiple users), resulting in a highly competitive benchmark among participants. Overall, the MemoriEase system only achieved the 21st position out of 35 partici-

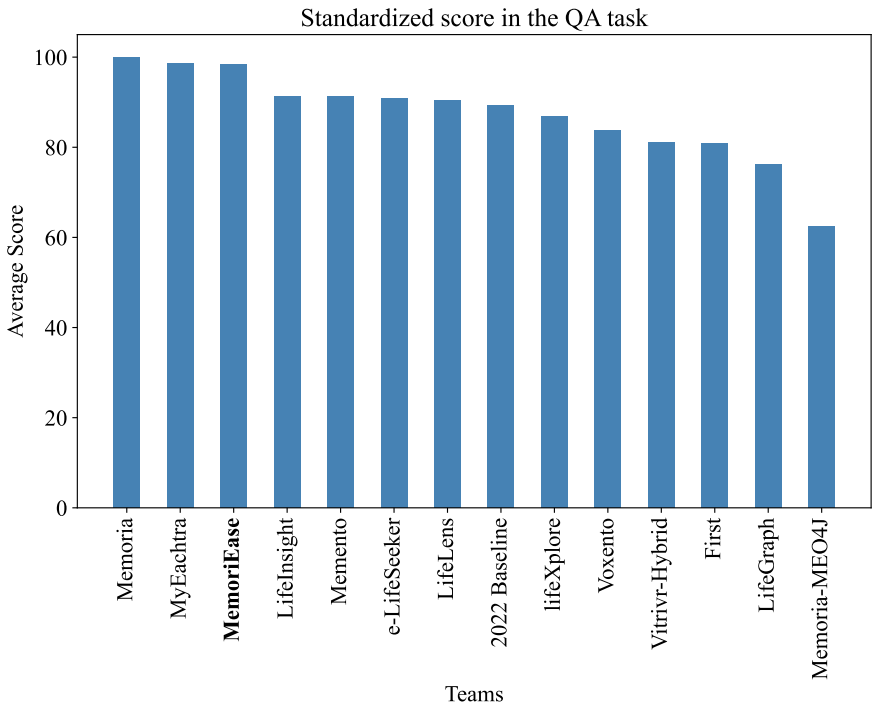


Figure 4.6: MemoriEase performance in LSC'23 QA sub-task.

pants and ranked 13th out of 21 systems. MemoriEase performed best on the KIS task, achieving a score of 65 out of 100, followed by the QA task with a score of 55 out of 100. It achieved the lowest score on the ad hoc task, with only 53. This performance is comparable to that of other systems. Figure 4.7 depicts the scores of teams in LSC'24.

Although the ad hoc score was the lowest, it was an improvement from previous results in LSC'23. This is thanks to the image-to-image retrieval described in Section 4.2.1. Users used the textual query to find one correct image, and continued to use that image to search for more. This approach proved effective for queries with fewer than 10 correct answers, like "Find examples of when I was waiting for a security guard to let me into my office building". For other queries with more than 10 correct answers, the time and number of submitted answers played a big role in achieving a high score, and the participant was not as fast as other competitors. While the participant mainly looks at one-by-one images and clicks submit sequentially, other competitors can submit a pool of multiple images and achieve a higher score.

On the two other sub-tasks, the MemoriEase system found the correct answers

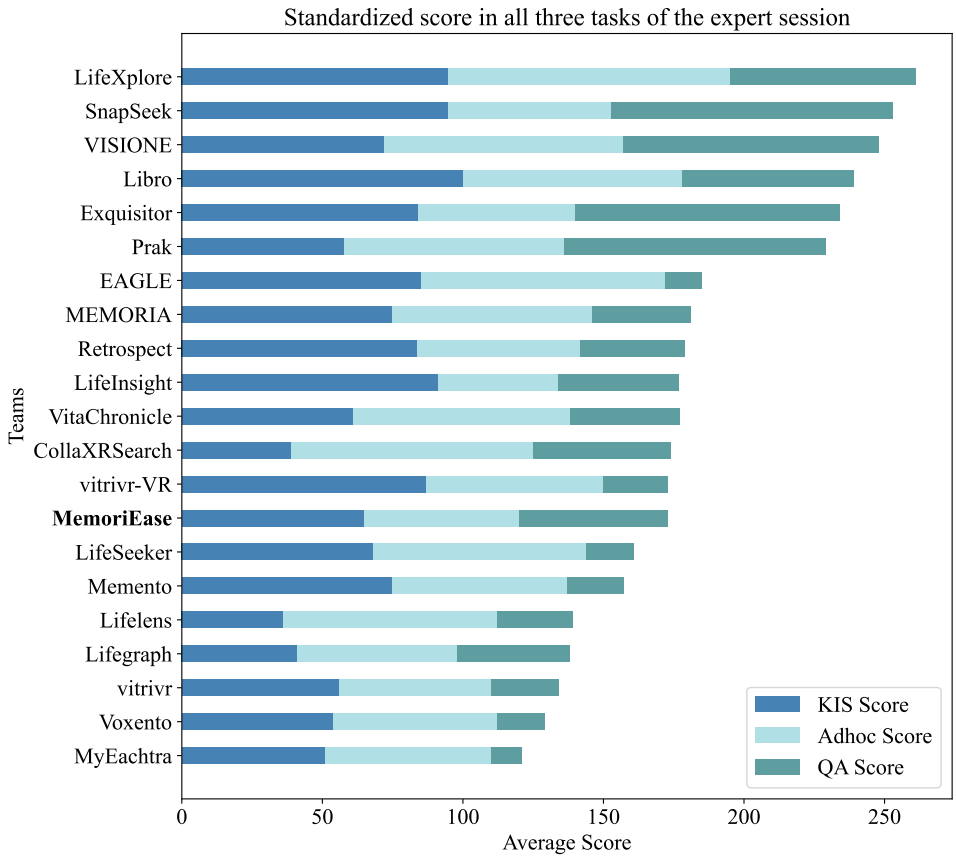


Figure 4.7: MemoriEase 2.0 performance in LSC'24.

for 4 out of 5 in KIS queries and 3 out of 5 QA queries. Although the system found the answer in the top 100 retrieved images, the user still could not submit the correct answer for queries like "It was as if the meeting took place on the Starship Enterprise. I remember the wall had yellow lighted shapes and designs". The event segmentation also caused a failure on the query "I remember she was wearing a hat and taking a photo of a lake.". In this case, the correct image was grouped into an event whose representative image differed from the target image, so I only identified it at the last moment, ultimately failing to submit it in time.

Some QA queries were too challenging to find the answer, which required both reasoning and quick action, like "A,B,C,D,E,F, Y or Z? I can't recall which zone of the car park in the airport I parked in when I was going to France in 2019". This question requires finding the car park at the airport in the home country, rather than in France, so most of the team fails to find the answer.

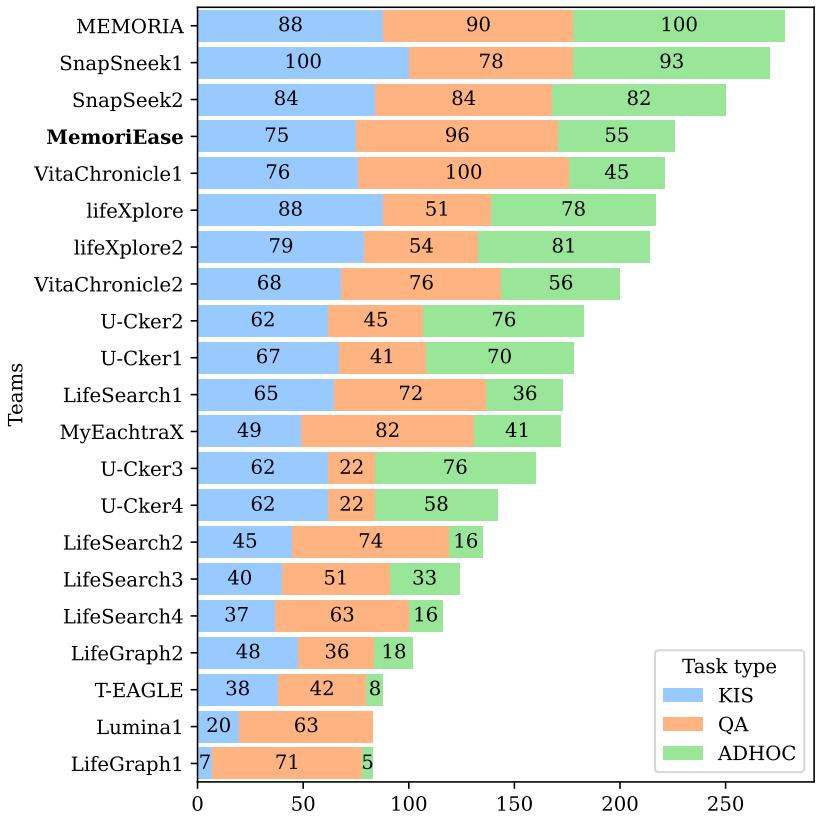


Figure 4.8: MemoriEase performance in LSC'25.

Results at LSC'25

With 21 participants from 11 teams in LSC'25, there is a decrease compared to LSC'24. In addition, LSC'25 is also a hybrid competition, where most participants join online due to travel restrictions to the USA for the ICMR 2025 conference. The MemoriEase system participated remotely in the expert session only. Remarkably, it achieved the third position in the leaderboard, following the Memoria and SnapSeek teams. MemoriEase also achieved the second-highest position in the QA sub-task, scoring 96 out of 100 thanks to the RAG component. The score of KIS queries also increased by 10 points, but the performance on the ad hoc sub-task remained the same compared to last year's competition. Figure 4.8 illustrates the leaderboard of LSC'25.

Specifically, MemoriEase resolved 4 out of 6 QA queries, which is equal to top teams like MEMORIA, SnapSeek, and VitaChronicle. However, thanks to the fast retrieval and less time to correct the submission, MemoriEase ranked second place

in the QA task, following the VitaChronicle team. There was a query that no other team found the correct answer to, except MemoriEase, so it helped boost the score. Some answers for queries are not explicitly found on the first search, but through refinement, the answer can be found.

On the KIS task, MemoriEase also found 4 out of 6 correct answers, but this is far below the 6 out of 6 correct answers achieved by top teams like SnapSeek or lifeXplore. It is an experience from challenges that answers are hardly found in the first three hints; it is difficult to find the correct answers even when the time and location of queries are revealed in the last hint. The two wrong queries are "I was in the kitchen of a restaurant" and "I was shopping for a rat trap with my brother". The system failed to find the answer because it only identified a location like a kitchen, but in a shop with numerous products, the correct answer is nearly an empty kitchen with numerous pipes. In the second query, the system found the mousetrap labelled "Mouse trap" instead of the rat trap. The correct answer showed the small rat trap, which may lead to the embedding model failing to encode it.

The ad hoc task in LSC'25 this year is challenging with 6 queries, but 5 of them have fewer than 10 correct images. Only the query "Drinking beer other than Guinness with other people" has more than 10 correct images. Due to this scenario, the relevance feedback for visual similarity search in our system did not work well to find as many visually similar images. MemoriEase relied mostly on text-to-image search to find the answers and failed to find the answer for the query "Menu in a picture frame in a restaurant". The object in the picture frame is unique and unusual to the embedding model, so it only found images of normal menus or large menus hanging on the wall. If MemoriEase could find one single correct image of the picture frame menu, it could find more similar images, but it failed in the first steps, so users cannot submit any correct images for this query. MemoriEase achieved an average score of 55 in this task due to slow submission times and incorrect submissions.

Overall, MemoriEase performed quite well in the LSC'25, securing third place. Thanks to the RAG for QA, MemoriEase can quickly find the answer. Queries in

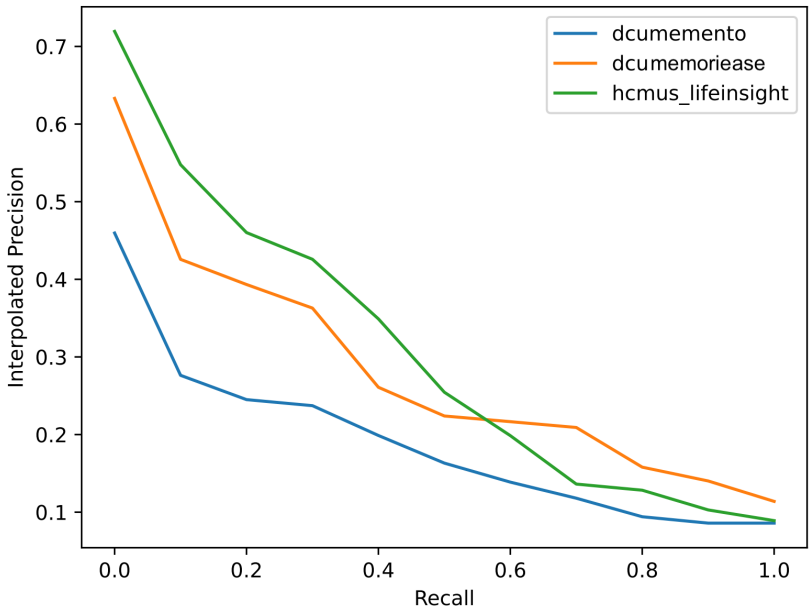


Figure 4.9: Comparing the best runs of the three automatic teams from NTCIR-17 Lifelog-5 Task [190]

LSC are becoming increasingly challenging, requiring not only effective systems but also prompt action and logical thinking. Some components in this system can be improved to address weaknesses during competition.

Results at NTCIR-17 Lifelog-5 Task

NTCIR Lifelog Task is an automatic task for a lifelog retrieval system, as presented in Section 2.1.4. This is the first time MemoriEase has participated in this task and competed with two other teams. Overall, MemoriEase achieved second place on several metrics. With 41 queries and 7494 ground-truth relevant images for these queries, MemoriEase found a total of 651 relevant images, accounting for approximately 8.68% of the ground-truth relevant images. Figure 4.9 shows a comparison in Recall versus interpolated precision of the best run of three automatic teams in the NTCIR-17 Lifelog-5 task.

In addition, the MemoriEase system achieved a MAP of 0.2713, which is relatively low because of the small number of relevant retrieved images. The precision at 5 (P@5) is 0.3707, indicating that approximately 37.07% of the top 5 retrieved images are relevant. However, when increasing the cutoff K, the precision decreases

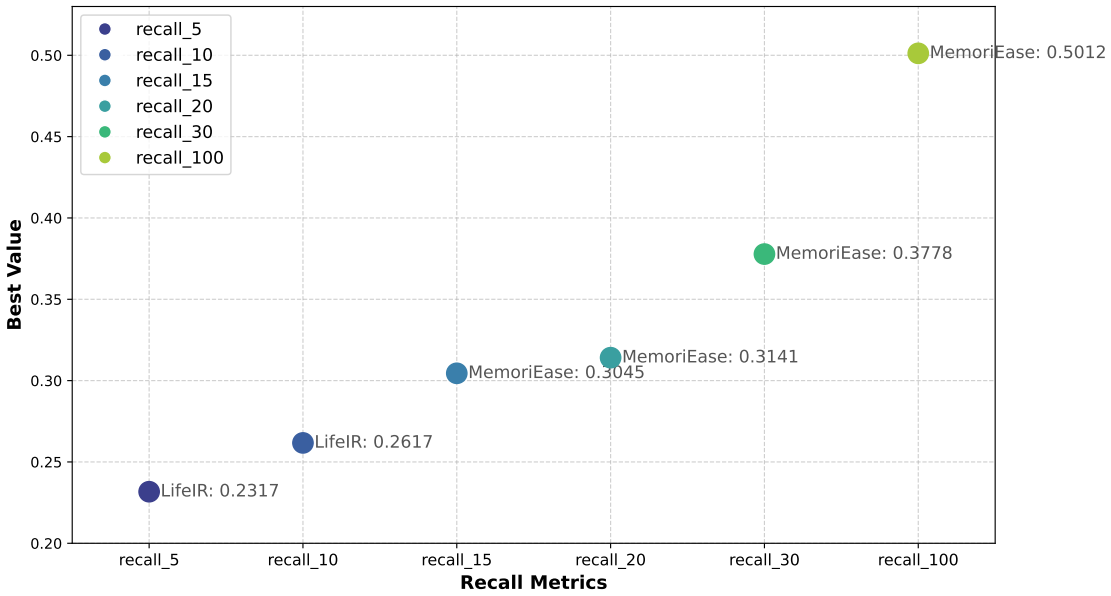


Figure 4.10: Best recall results across all submissions at NTCIR-18 Lifelog6 Task [191]

.

slowly, with values of 0.2878 P@20, 0.2496 P@30, and 0.1587 P@100. This slight decrease indicates that the most relevant images are at the top of the retrieved list. The result from NTCIR-17 Lifelog-5 Task for MemoriEase confirms the system's ability to deliver valuable results, even within the first few search results, thereby highlighting its practical utility.

Results at NTCIR-18 Lifelog-6 Task

With one more team participating in the NTCIR-18 Lifelog-6 Task in the automatic track, MemoriEase still secured second place. There are 13 topics for the KIS task, 13 topics for the ad hoc task, and 24 topics for the LQAT sub-task. Figure 4.10 shows a scatter chart depicting the recall by different K across all submissions. MemoriEase achieved the highest Recall for K from 15 to 100. However, the top 5 and top 10 ranked results are lower than those of the LifeIR team.

I performed numerous comparisons between the embedding models of BLIP2 and CLIP, as well as query rewriting and relevance feedback, for image-to-image search in the ad hoc task. This comparison helps to achieve the highest score in the final evaluation. Results of this experiment are described in Table 4.3. The

Table 4.3: Performance of different approaches in ad hoc and KIS queries. CLIP/BLIP: embedding models, Q/CQ: original queries vs concise, LLM-summarised queries, NoRF/RF: No relevance feedback vs pseudo relevance feedback.

(a) Ad hoc Task

Ad hoc Queries	R@5	R@20	R@100	P@5	P@20	P@100
CLIP-Q-NoRF	0.015	0.1223	0.2426	0.2615	0.2462	0.21
BLIP-Q-RF	0.0136	0.0382	0.132	0.2462	0.2385	0.22
BLIP-Q-NoRF	0.055	0.1518	0.3802	0.3692	0.2923	0.1815
BLIP-CQ-NoRF	**0.0598**	**0.167**	**0.4406**	**0.4308**	**0.3423**	**0.2192**

(b) KIS Task

KIS Queries	R@5	R@20	R@100	P@5	P@20	P@100
CLIP-Q-NoRF	0.191	0.4357	0.494	0.2	0.1346	**0.0638**
BLIP-Q-NoRF	**0.2822**	**0.5035**	**0.6349**	**0.2462**	**0.1462**	0.0554
BLIP-CQ-NoRF	0.2698	0.3847	0.5348	0.2	0.1192	0.0554

best combination is BLIP-Q-NoRF with 28.22% R@5, and 63.49% R@100 for the KIS task. The CLIP model finds more correct images, but the BLIP model achieves better performance due to the higher rank of correct images in CLIP. The concise queries do not enhance the performance of KIS queries compared to the original queries. This suggests that KIS queries require more detailed information to locate the correct images. In contrast, concise queries are more effective for ad hoc tasks, resulting in the highest performance. Additionally, relevance feedback for the ad hoc task did not perform as expected and yielded worse results than no relevance feedback.

MemoriEase also participated in the QA task in the NTCIR-18 Lifelog-6 task, but achieved modest performance. Overall, it predicted 8 correct answers in 24 questions. The correct answers are mainly from the visual-related questions that are answered in the lifelog images. With 6 correct answers from visual-related questions and 2 correct answers from metadata-related questions, it highlights the drawback of the system for metadata-related questions and the challenge posed by that type of question. The primary reason for incorrect answers in metadata-related questions is the lack of or incorrect input descriptions to the LLMs. There are 5 questions for

which the LLMs cannot determine the answer because the prompt does not provide any information. This highlights the importance of the retrieval model in the lifelog QA task.

4.3 Lifelog Context Reranking

In this section, I present the reranking model, which aims to further improve the ranked list of contexts for lifelog QA through additional reranking. This model aims to utilise multi-modal lifelog data to explore the relationship between modalities to the question and rerank the retrieved list of contexts to make it better. The following sections present the problem, model design and experiments.

4.3.1 Introduction

Finding the correct context to answer a question is challenging due to the vast and diverse nature of lifelog data. The multi-modal aspect of lifelog (images, metadata, and textual event descriptions) complicates context retrieval. A single-modal retrieval model often fails due to semantic mismatches and variations in how time, location, and visual semantics are represented in questions and contexts. These are the reasons why I propose a multi-modal reranking model for lifelog context retrieval.

To address this problem of context retrieval, I propose a two-stage retrieval framework for lifelog QA. Figure 4.11 depicts the framework. In the offline phase, multimodal lifelog data (metadata, contexts, and images) are indexed into a database, and the reranking model is trained on question–context pairs with binary relevance labels. In the online phase, the question retrieves the top-K candidate contexts by embedding similarity, which are then reranked by the trained model to produce a refined relevance ordering. The retrieval stage uses the Stella-1.5B model [183] to encode questions and contexts into embeddings, and then performs cosine similarity ranking. This model is chosen due to its superior performance in the experiment.

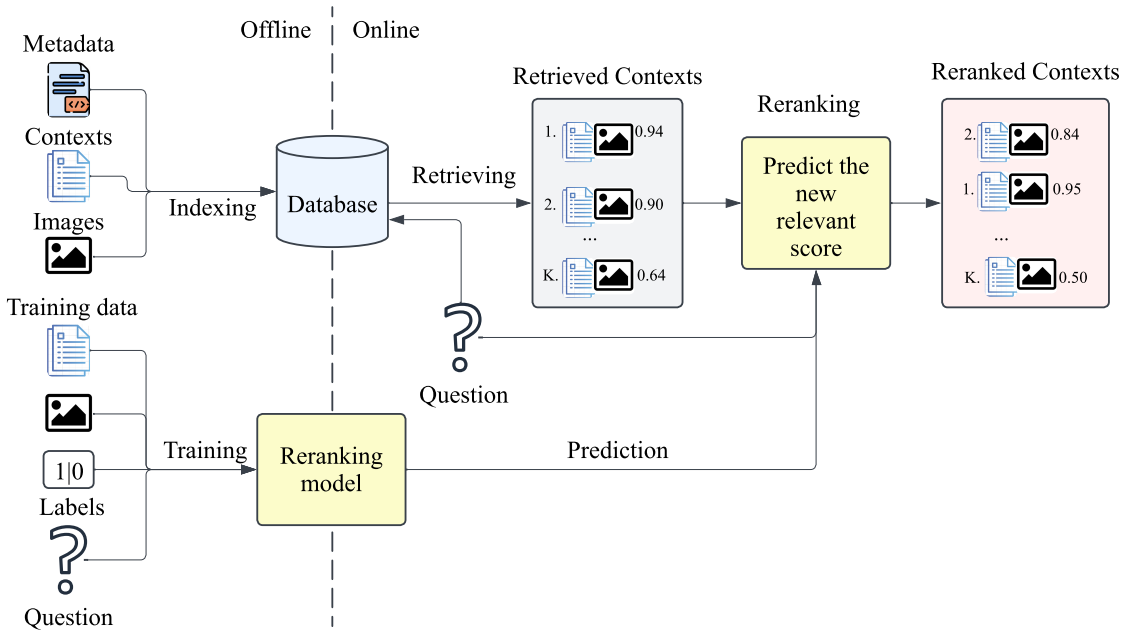


Figure 4.11: A two-stage retrieval framework for the lifelog QA task.

Reranking models can improve retrieval accuracy by prioritising the most relevant contexts while filtering out noise [78, 28]. It is a particularly effective approach in multimodal datasets, such as lifelogging, where different data types need to be integrated for better context selection, which poses a challenge for traditional retrieval models.

Lifelog data with a multimodal nature also differs from normal passage reranking, which largely relies on text [178]. I propose a cross-encoder reranking model that evaluates retrieved results and predicts their relevance to the question. By processing both lifelog images and textual descriptions, the model mitigates semantic mismatches and addresses the multimodal challenges inherent in lifelog data. Although LLMs show promising results on the reranking task [1, 104], they are too costly to implement for this reranking. In addition, the speed at which LLMs generate the ranking can slow down the whole process of interactive retrieval and QA when applied to interactive systems, so I decided to just experiment on cross-encoder reranking models using small language models and vision encoder models. Details about the architecture and implementation are discussed in the following sections.

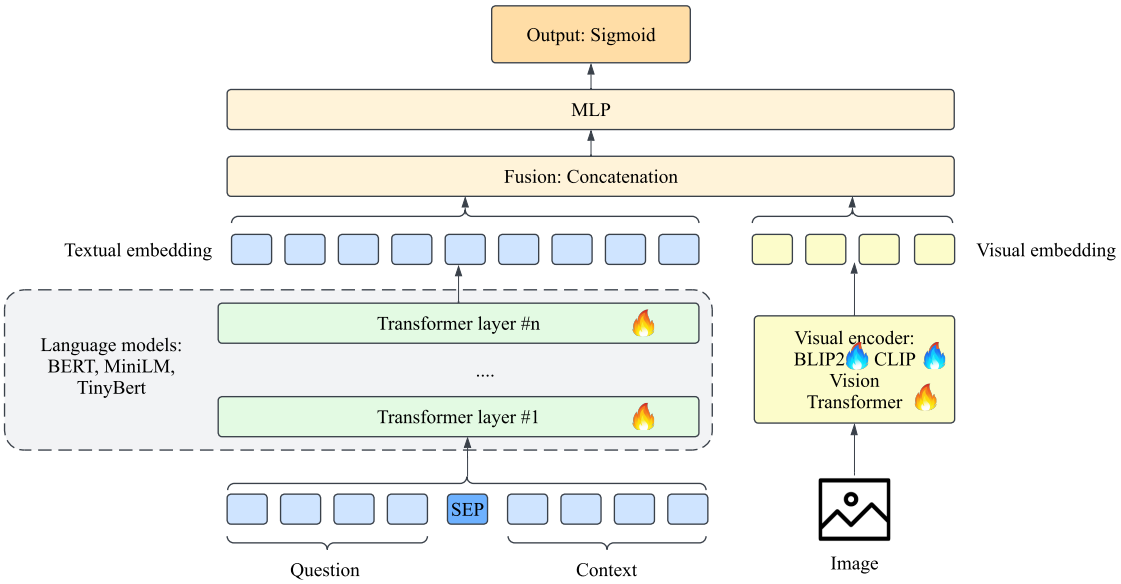


Figure 4.12: The overall architecture of the cross-encoder multi-modal reranking model. Red flame indicates a fine-tuned model, while blue flame indicates a frozen model.

4.3.2 Multimodal reranking model

Figure 4.12 illustrates the architecture of the proposed multimodal reranking model, which processes textual and visual inputs through two parallel branches before fusing them for relevance prediction.

In the textual branch, the question and context are concatenated into a single sequence separated by a special SEP token, following the standard cross-encoder input format. This allows the language model to jointly attend to both the question and the context, capturing their interactions directly through self-attention rather than encoding them independently. The token sequence is fed through a language model, which contains a stack of transformer layers to produce a contextualised textual embedding. Three language models are evaluated for this branch: BERT-base-uncased, MiniLM, and ModernBERT-base. More details about these three language models are presented in the following section.

In the visual branch, the lifelog image associated with the retrieved context is encoded independently by a visual encoder to extract a visual embedding. Three visual encoders are evaluated: BLIP2, CLIP, and Vision Transformer (ViT). While the BLIP2 and CLIP models have already been pre-trained on a large amount of

data, I froze their weights and used them as visual embedding extractors. Vision Transformer is freshly initialised with no pre-training, so its weights are trained during the reranking training phase.

The two embeddings are then fused via concatenation and passed through a multi-layer perceptron (MLP). The output layer produces a single logit, which is normalised by a sigmoid activation function to yield a relevance score in the range $[0, 1]$, indicating the degree to which the context is relevant to the question. The model is trained with binary labels, where 1 denotes a relevant context, and 0 denotes an irrelevant one. Details about the language and vision models are presented as follows:

- BERT-base-uncased [33]: Bidirectional Encoder Representations from Transformers (BERT) is a powerful language model designed to understand the meaning and context of words in a sentence by leveraging deep bidirectional encoding. Thanks to its encoder-only architecture, BERT is highly effective in various tasks related to natural language understanding, such as sentiment analysis.

- MiniLM[4]: This model is a lightweight yet efficient cross-encoder model designed for passage ranking. This model is a distilled version of BERT with 6 layers of Transformers [164] and fine-tuned on the MS Marco dataset [9]. Unlike bi-encoder models that independently encode queries and documents, the cross-encoder processes both together, enabling deeper interaction and improved ranking accuracy.

- ModernBERT-base[5][165] is a powerful, modernised BERT-style encoder-only Transformer model designed for long-context understanding and efficient processing. It integrates several advancements and is pre-trained on 2 trillion tokens of English and code, making it ideal for tasks involving long documents, such as retrieval, classification, and semantic search.

[4]https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2
[5]answerdotai/ModernBERT-base

- BLIP2 [90]: Bootstrapped Language-Image Pretraining is a vision-language model designed for efficient and unified language-image pre-training. It leverages a two-stage learning framework with Q-Former to bridge vision and language models. BLIP-2 employs a pre-trained vision encoder to extract visual features, which are then aligned with a language model through a lightweight Q-Former. The embedding of images from BLIP-2 provides a semantic embedding to represent the information in the image.

- CLIP [121]: Contrastive Language-Image Pretraining is a powerful vision-language model developed by OpenAI that encodes images and text into a shared embedding space. It learns by training on large-scale image-text pairs using a contrastive objective, making it highly effective at aligning visual and semantic representations. In this work, I leverage CLIP's visual encoder to extract image embeddings that are semantically grounded in natural language, which complements the textual branch of the reranking model.

- Vision Transformer (ViT) [39] is a deep learning model that applies transformer architecture to image processing. Unlike traditional convolutional neural networks (CNNs), ViT splits an image into fixed-size patches, linearly embeds them, and processes them through a self-attention-based transformer encoder. This patch-based attention mechanism allows the model to capture long-range spatial dependencies across the image, which is particularly beneficial for lifelog images where relevant visual cues may be distributed across different regions of the scene. In this work, I use ViT as the visual encoder to extract image embeddings, which are subsequently fused with the textual embeddings for relevance prediction.

4.3.3 Experiment

This Section presents experiments evaluating the proposed multimodal reranking model for lifelog QA. I first describe the experimental setup, covering dataset details,

hyperparameter configurations, and evaluation metrics, followed by several baselines for comparison. After that, a quantitative analysis of results across retrieval and reranking models is presented. Finally, a qualitative error analysis examines failure cases to identify the remaining limitations of the model.

Settings

I use the multimodal OpenLifelogQA dataset [162] for this experiment. OpenLifelogQA is an open-ended lifelog QA dataset comprising 14,187 QA pairs and 27,705 contexts, as described earlier in Section 3.2.1. Each QA pair has between 1 and 27 ground-truth contexts, with an average of 1.64 per question. A ground-truth context is defined as a context that contains sufficient information to answer the given question. The DCU-TimelineQA dataset is not used for the experiment because it does not contain images to satisfy the requirement of the multimodal reranking model. In addition, the retrieval result for this dataset is already high, which does not require reranking.

I employ a hard negative sampling method to generate negative samples for the training and validation datasets, using BM25 to retrieve the top 10 most relevant contexts, excluding the ground-truth contexts. The maximum length for concatenated contexts and questions is set to 128. The batch size is 32, and the loss function is Binary Cross-Entropy loss. I use AdamW [97] optimizer with a 1×10^{-5} learning rate and weight decay 1×10^{-4}. The model is run for 20 epochs and applies early stopping with a patience of 2 epochs on the validation set. All parameters here are tuned on the validation set of the OpenLifelogQA dataset and run on a single RTX 4090 GPU.

In the evaluation step, I first retrieve the top 100 relevant contexts to a question using cosine similarity between Stella 1.5B embeddings of questions and contexts. A date filter is extracted from the questions and applied to reduce the retrieval space. After that, the trained reranking model reranks the top 100 contexts and evaluates the metrics. Precision@K (P@K) and Recall@K (R@K), as defined in Formula (3.1)

and (3.2), are used to evaluate the accuracy of the top K contexts.

Baselines

I use four retrieval models to serve as the baseline in the retrieval-only approach, including BM25 [127], Stella-1.5B [183], CLIP [121], and a hybrid of Stella and CLIP. While BM25 calculates the relevant score between questions and contexts using lexical matching, three other models calculate the cosine similarity of embeddings. Stella-1.5B is used to embed contexts and questions. CLIP embeds the images and questions into embeddings. A hybrid approach combines the cosine similarity of CLIP and Stella by weighted average to produce the final relevant score. BM25 serves as a strong lexical retrieval baseline, while Stella-1.5B and CLIP enable semantic matching in textual and multimodal contexts, respectively. In addition, the retrieved list from the Stella-1.5B model will serve as the input list for reranking.

I also use MiniLM and the Electra-base reranking model fine-tuned on MS Marco data [9] to rerank the retrieved contexts as baselines. After receiving a retrieved list of contexts, these two models perform reranking on the list of contexts, with the relevant score being the model's final logit. I cannot run the BERT model without fine-tuning because BERT is not designed for reranking, so the final layer has more than one logit. All the models' weights are from the HuggingFace repository [6]. Using MiniLM and Electra-base as rerankers for baselines allows us to assess whether cross-encoder architectures can further improve retrieval quality in a zero-shot setting.

Results

[6]https://huggingface.co/cross-encoder

Table 4.4: Reranking experimental results on the OpenLifelogQA test set. Bold numbers indicate the highest result, while underlined numbers indicate the second-highest result. T denotes text, and I denotes images.

			Recall@K					Precision@K				
Model	Type	Modality	R@1	R@5	R@10	R@20	R@50	P@1	P@5	P@10	P@20	P@50
BM25	Retrieval	T	0.2299	0.6829	0.8171	0.9138	0.9859	0.3090	0.1999	0.1273	0.0795	0.0518
Stella-1.5B	Retrieval	T	0.3765	0.7767	0.8801	0.9435	0.9918	0.5265	0.2479	0.1476	0.0854	0.0524
CLIP	Retrieval	I	0.1369	0.3602	0.5305	0.7296	0.9506	0.2241	0.1296	0.0950	0.0684	0.0510
Hybrid (Stella+CLIP)	Retrieval	T+I	0.2949	0.6693	0.8029	0.9012	0.9800	0.4350	0.2245	0.1377	0.0822	0.0520
MiniLM	Reranking	T	0.3811	0.7674	0.8843	0.9570	0.9919	0.5398	0.2500	0.1496	0.0868	0.0524
Electra-base	Reranking	T	0.4012	0.8288	0.9208	0.9707	0.9971	0.5630	0.2613	0.1534	0.0877	0.0526
Fine-tuned BERT	Reranking	T	0.6342	0.9125	0.9569	0.9801	0.9976	0.8263	0.2926	0.1609	0.0886	0.0526
Fine-tuned MiniLM	Reranking	T	0.6408	0.9203	0.9629	0.9838	0.9978	0.8342	0.2954	0.1626	0.0890	**0.0527**
Fine-tuned ModernBERT	Reranking	T	0.6541	0.9236	0.9638	0.9833	0.9969	0.8541	0.2968	0.1626	0.0891	0.0526
ModernBERT+CLIP	Reranking	T+I	0.6330	0.9118	0.9606	0.9846	0.9968	0.8276	0.2875	0.1592	0.0881	0.0526
ModernBERT+BLIP2	Reranking	T+I	0.6532	0.9234	**0.9679**	0.9846	**0.9979**	0.8548	0.2968	**0.1637**	0.0892	**0.0527**
ModernBERT+ViT	Reranking	T+I	**0.6557**	**0.9304**	0.9676	**0.9866**	0.9963	**0.8588**	**0.2983**	0.1636	**0.0894**	0.0526

The results from Table 4.4 show that baseline retrieval models, such as BM25 and CLIP, perform poorly on the OpenLifelogQA test set. BM25 cannot capture semantic relationships due to its lexical matching method, whereas CLIP, which is based on image retrieval, lacks sufficient information. The Stella-1.5B model achieves better performance with 88.01% R@10 and 14.76% P@10, thanks to its semantic embedding. The hybrid approach provides poorer performance due to the combination of CLIP. Using a reranking model without fine-tuning does not improve the performance. The performance of MiniLM and Electra-base shows a small improvement in Stella-1.5B retrieval performance after reranking in the OpenLifelogQA test set.

Fine-tuning language models for reranking based on textual questions and contexts improves performance. To be specific, on the OpenLifelogQA dataset, performance increases by approximately 28% in R@1, 8% in R@10, and 33% in P@1 compared to the retrieval performance of the Stella-1.5B model. Fine-tuned ModernBERT achieves the highest performance among fine-tuned models. In general, the results suggest that a fine-tuned reranking model improves retrieval accuracy. With a 96.38% R@10, this provides a good list of context for the OpenLifelogQA dataset.

Although incorporating images into the reranking model does not enhance the ranking performance by a large margin, it still achieves the highest performance in some metrics. The multimodal reranking models leverage textual and visual features, allowing a more comprehensive representation of textual contexts and related images. While fine-tuning ModernBERT alone improves results, integrating visual information provides further enhancements, particularly in Recall at higher ranks (K is 20 and 50). However, the improvement from visual fine-tuning is modest compared to the fine-tuned language model. This suggests that textual context remains the primary contribution to context information, with images providing a supplementary contribution.

I run a retrieval experiment on a test set of the DCU-TimelineQA dataset and

Table 4.5: DCU-TimelineQA retrieval results from BM25 and Stella-1.5B.

Model	@1	@5	@10	@20	@50
Recall@K					
BM25	0.6933	0.9949	1.0000	1.0000	1.0000
Stella-1.5B	0.9291	0.9983	1.0000	1.0000	1.0000
Precision@K					
BM25	0.6933	0.2821	0.2711	0.2711	0.2711
Stella-1.5B	0.9291	0.2792	0.2674	0.2674	0.2674

present the results in Table 4.5. The performance is already high with the Stella-1.5B embedding-based retrieval model, achieving 92.91% Recall@1 and 99.83% Recall@5. Moreover, due to the limited number of events per day, the model achieves a recall of 1 at K when K is greater than 10. The results demonstrate good performance, as the synthetic data from DCU-TimelineQA is not challenging and does not require additional reranking.

In summary, the experiments across both datasets demonstrate that fine-tuned cross-encoder reranking models outperform retrieval-only baselines, with multimodal models providing further gains over text-only rerankers. On the OpenLifelogQA dataset, fine-tuning alone achieves the most improvement, while the addition of visual features offers consistent but modest gains, particularly at higher recall ranks. Based on these findings, the reranking model of ModernBERT with Vision Transformer (ViT) is selected as the reranking model for the full system, as it achieves the highest performance across the majority of metrics on the OpenLifelogQA test set. To further understand the strengths and limitations of this model, the following Section presents a qualitative error analysis of its retrieval outputs.

Qualitative Error Analysis

I analyse some errors in the context reranking list and provide examples in Table 4.6 below. The reranking model struggles to find events for temporal questions that require information about events that occurred after or before a known event. For example, the question: "What did I do after cycling on March 31, 2020?" asks for an event after cycling, but the ranking contexts only rank cycling activities at

Table 4.6: Error examples of the reranking model

Question	Ground-truth Contexts	Top 5 Ranking Contexts
What did I do after cycling on March 31, 2020?	1. From 04:30 to 05:25 PM on March 31, 2020, I start riding my bike 2. From 05:27 to 06:06 PM on March 31, 2020, I am preparing dinner with a woman	1. From 04:30 PM to 05:25 PM on March 31, 2020, I start riding my bike 2. From 05:25 PM to 05:27 PM on March 31, 2020, I have returned home 3. From 06:53 PM to 06:54 PM on March 31, 2020, I am eating and watching 4. From 04:09 PM to 04:30 PM on March 31, 2020, I am in the kitchen 5. From 01:14 PM to 01:33 PM on March 31, 2020, I am having lunch and drinking water
How did I get to DCU on March 25, 2020?	1. From 11:35 AM to 12:16 PM on March 25, 2020, I left home and am biking to go to the workplace	1. From 05:32 PM to 06:09 PM on March 25, 2020, I have left DCU and started biking home 2. At 2020-03-25 18:09:11, I have returned home 3. From 05:15 PM to 05:18 PM on March 25, 2020, I have reached where I parked my bike at DCU 4. From 12:16 PM to 12:20 PM on March 25, 2020, I have arrived at Tesco by bike and am moving to eat 5. From 05:25 PM to 05:32 PM on March 25, 2020, I finished working, left my room with my bike

the top of the list. To address this issue, I suggest incorporating more information from previous and subsequent events into the context to help the model identify the correct contexts. Another problem is the ambiguity in the data. For example, the context for the question, "How did I get to DCU on March 25, 2020?" is leaving home and riding a bike to the workplace; however, the model does not understand the workplace and DCU, so it fails to rank this context at the top. It can be addressed by providing more information, such as that DCU is a workplace.

4.4 Chapter Summary

Initially, I present an interactive solution for lifelog retrieval using the MemoriEase system. This system demonstrates competitive performance on the LSC and NT-

CIR Lifelog Task benchmarks. The development through each version improves the search capability and performance of the system. The RAG pipeline in MemoriEase 3.0 demonstrates a potential application of LLMs for the lifelog QA task, contributing to a second-place finish in the QA task at LSC'25. Although MemoriEase shows strengths in search flexibility across KIS, ad hoc, and QA tasks, conversational search, and the RAG pipeline, it still has notable drawbacks. The reliance on BLIP2 embeddings, while effective for general semantic similarity, fails to distinguish fine-grained visual details in the ad hoc task. Moreover, in live competition settings, processing speed remains a critical bottleneck. These weaknesses motivate future development directions for the system.

In addition to the interactive solution, I propose a multimodal reranking model for automatic context retrieval in lifelog QA, accompanied by systematic experiments on retrieval and reranking. The results demonstrate that embedding-based semantic retrieval using a language model such as Stella provides a strong baseline, but can be substantially improved through a fine-tuned cross-encoder reranking stage. Specifically, fine-tuning a reranking model on question-context pairs achieves approximately 28% improvement in R@1 and 33% improvement in P@1 over the retrieval baseline alone. Further incorporating lifelog images into the reranking model achieves the highest overall performance, with ModernBERT+ViT obtaining the best results across the majority of metrics, confirming that visual context provides a complementary signal to textual features.

These findings collectively address RQ1: *What is the most effective context retrieval method to support the LifelogQA task when compared to standard retrieval approaches?*. The answer is a two-stage pipeline combining dense semantic retrieval with a fine-tuned multimodal cross-encoder reranking model that can effectively retrieve relevant contexts for lifelog QA and outperform standard methods such as sparse retrieval (BM25) or dense retrieval only. The first stage efficiently narrows the candidate pool using embedding similarity, while the second stage refines the ranking by jointly reasoning over the question, its associated textual context, and

the corresponding lifelog image. This combination addresses the core challenges of lifelog QA: the semantic gap between natural language questions and lifelog data, the multimodal nature of lifelog contexts, and the need for precise relevance judgements beyond surface-level similarity. The retrieval and reranking outputs from this experiment are subsequently used as the context source for the RAG pipeline in the Chapter 6 experiments.

Chapter 5

Retrieval-Augmented Generation for Lifelog QA

In this chapter, I propose an RAG approach for lifelog QA to solve RQ2: *How effective is RAG when generating answers for LifelogQA from retrieved contexts?*. This RAG approach combines retrieval and generation to generate answers to lifelog questions from a vast volume of lifelog data. Firstly, I will introduce the problem, challenges, and the approach in Section 5.1. Then I describe the methodology of the RAG approach in Section 5.2. Experimental results and analysis are discussed in Section 5.3 and 5.4. Finally, a discussion section with a conclusion of the chapter and the answer to RQ2 is presented in Section 5.5.

5.1 Introduction

Lifelog QA extends the retrieval task by requiring the system to both identify relevant events and generate natural-language answers to questions about them. Lifelog QA offers personal insights into lifestyle and memorable events in an ask-and-answer manner. Although the applications of lifelog QA are numerous, two challenges remain in the QA task for lifelog data. Firstly, due to the large volume of lifelog data, it is necessary to find relevant information for the question before deriving answers. The accuracy of retrieval models affects the correctness of the answer. Secondly,

the temporal and multi-modal lifelog data challenge the QA model's reasoning for complicated questions. For example, "Who did I talk with at the ICMR conference last year?". The QA model first identifies events from last year's ICMR conference and then analyses them before providing an answer. The QA task in lifelog has been introduced in the LSC challenge since 2022 [55] and remains a challenging task with no specialised QA models.

Although some previous works have already applied the Retriever-Reader for Open-domain QA [156] or RAG [146], the performance of these methods is not higher than that of other methods. Several factors contribute to the results, as stated in their original paper, including the ExtractiveQA model's superiority for span-based QA in the DCU-TimelineQA dataset over RAG. However, for an open-ended QA problem like the one in the OpenLifelogQA dataset, can RAG work effectively to retrieve lifelog contexts and answer questions in a more unrestricted form of answers? This chapter aims to resolve this research question by breaking it down into two sub-research questions as follows.

In the first sub-research question: *To what level of accuracy can LLMs in RAG answer open-ended lifelog questions from retrieved context?*, I propose a RAG-based approach to lifelog QA that combines retrieval and LLM generation. Several LLMs are evaluated on lifelogQA when provided with retrieved lifelog contexts to check the performance in terms of accuracy of these models. I also conduct error analysis and investigate the causes of the models' failure to answer questions correctly.

The second sub-research question focuses on the fine-grained performance of LLMs on different question types as mentioned in Section 1.1.2. The question *How does that accuracy decompose across the atomic, temporal and aggregation question types of OpenLifelogQA?* aims to analyse the performance of LLMs in the lifelog QA task in different types of questions, including atomic, temporal, and aggregation questions. Through fine-grained analysis, the strengths and weaknesses of LLMs are examined for each question type. From the results for the two sub-research questions, the answer to RQ2 can be drawn.

I apply a standard RAG pipeline, without additional enhancements such as context reranking, advanced prompting strategies, or fine-tuning LLMs, to answer this RQ. The main reason is to measure how RAG solves the lifelog QA problem when lifelog data are provided via retrieval. In addition, the RAG approach in this RQ also serves as a baseline for enhanced RAG with reranking and reasoning-focused fine-tuned LLMs for the next RQ. This aims to see the difference between standard RAG and enhanced RAG for the lifelog QA problem.

Formally, in the RAG-based lifelog QA setting, let $Q = \{q_1, q_2, ..., q_n\}$ denote a set of questions and $A = \{a_1, a_2, ..., a_n\}$ be their corresponding answers. The task aims to generate the correct answer a_i for an input question q_i. The knowledge for finding answers is a lifelog collection segmented into events (a sequence of images illustrating a single activity, such as eating or walking). Each event description is added to the time and location in the metadata to form a lifelog context, such as "From 12:20 to 12:35 PM 01/01/2020, I had lunch at home with my family" to form a context, and let $C = \{c_1, c_2, ..., c_m\}$ denote a list of contexts.

For a given question q, the RAG model first retrieves all contexts relevant to the question C_q. The relevance is measured using metrics based on the retrieval model, such as BM25 or cosine similarity.

$$C_q = Retrieval(q)$$

From the set of contexts C_q and the question q, the RAG model formulates a prompt and provides it to an LLM to generate the answer a.

$$a = LLM(prompt(C_q, q))$$

From the problem formulation, the RAG approach involves two tasks: context retrieval and answer generation. The goal of context retrieval is to find the accurate context needed for a question, while the goal of answer generation is to generate the correct answer to the question. Details about the methodology of the two tasks,

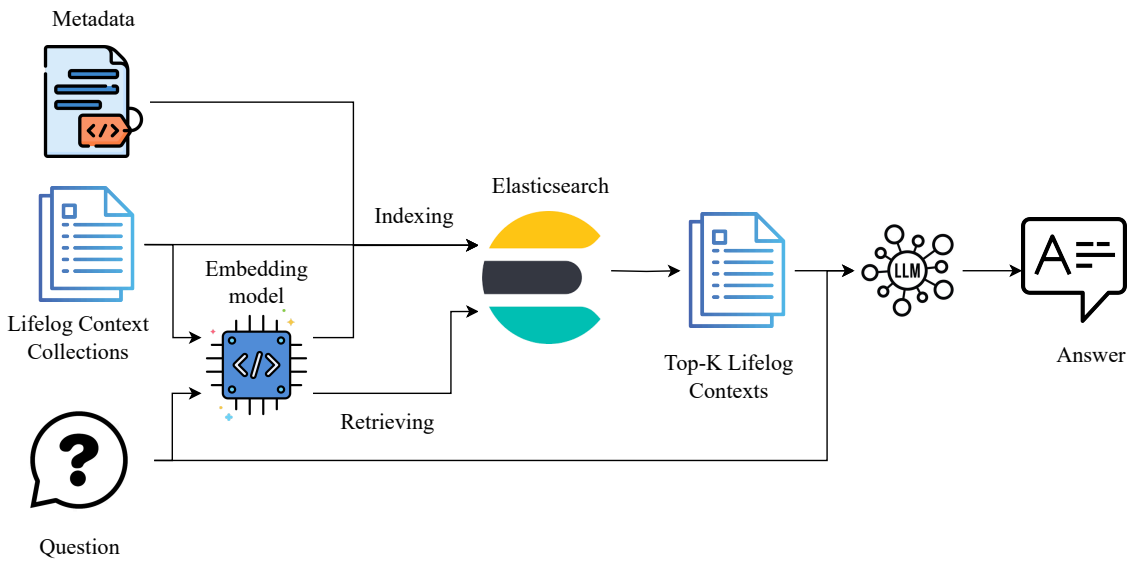


Figure 5.1: RAG pipeline for lifelog QA

including retrieval models, LLM details, and analysis, are provided in the following sections.

5.2 RAG approach

Figure 5.1 presents the RAG pipeline with three main stages: indexing, retrieving, and question-answering. In the indexing stage, lifelog contexts, along with other metadata, are indexed into Elasticsearch. The context descriptions, which describe what the lifelogger is doing in that event, are embedded into embedding vectors by an embedding model described in Section 5.2.1. Chunking is the process of splitting long documents into smaller chunks to ensure that each retrieved unit contains a coherent piece of information that fits within the context window of the language model. For the chunking strategy in this RAG approach, I treat each context as a chunk because it contains independent information about the life event. Each chunk with metadata, including time and location, event description, a keyframe image, and an embedding vector, is then indexed to the database.

In the retrieval stage, the question is processed into a query by removing the question word and mark. After that, it is also embedded by the same embedding model before using its embedding vector for retrieving relevant contexts. I used

BM25 as a traditional, standard method for retrieving texts as a baseline, and a novel embedding model, called Stella-en-1.5B-v5, served for semantic search to compare the two methods' performance. Information about these two models is presented in Section 5.2.1. After receiving a ranked list of contexts from stage 2, it is used in the answer generation stage.

In the answer generation stage, the question and a ranked list of lifelog contexts are formulated into a prompt. This prompt is provided to the LLMs to generate the answer to the question. Details about LLMs and the generation stage are described in Section 5.2.2.

5.2.1 Retrieval

Retrieval models play a crucial role in RAG systems, which retrieve information from the lifelog context collection to identify relevant contexts for questions. The model first transforms a question into a query by removing question words and punctuation marks. The metadata-related information in the question, such as time and location, is extracted for filtering. The retrieval model then retrieves the most relevant lifelog contexts by using BM25 or vector-based approaches.

BM25 [127] is a classic lexical retrieval algorithm that ranks documents based on term frequency and inverse document frequency, weighting the frequency with which query terms appear in a document relative to their rarity across the collection. Although BM25 is good at exact keyword matching and computationally efficient, it falls short when handling synonyms or conceptually related terms. In this RAG approach, BM25 compares transformed questions and the textual description of contexts to produce a ranked list of relevant contexts. Elasticsearch natively support BM25 search on textual columns in the indexed table, so I use the question to search on the context column. This approach serves as a baseline for comparison with the other vector-based approaches.

Stella-en-1.5B-v5[1] is an embedding model that transforms the textual query

[1]https://huggingface.co/dunzhang/stella_en_1.5B_v5

into an embedding vector. This vector, along with the embedding vectors of lifelog contexts indexed in Elasticsearch, is used to compute the cosine similarity between the query embedding and context embeddings. The ranked list is retrieved by sorting the cosine similarity in descending order. This approach captures deeper semantic relationships and better handles vocabulary mismatch, allowing it to find relevant documents even when they use different but related terms. I chose this embedding model because it is a general embedding model that ranks No. 5 on the Massive Text Embedding Benchmark (MTEB)(as of November 30, 2024) [105] with a score of 71.19 across all 56 text retrieval tasks. In addition, the computational resources required for this model are suitable for the current available hardware. Elasticsearch supports both keyword-based search (e.g., BM25) and embedding-based search[2]. For vector search, it uses the Hierarchical Navigable Small World (HNSW) graph-based indexing algorithm, which organises embedding vectors into a hierarchical structure and enables efficient Approximate Nearest Neighbour (ANN) search for fast and accurate retrieval. Cosine similarity is used to approximately estimate the similarity of questions and contexts. Let q denote a question, c denote a lifelog context description, and f_{stella} denote the Stella-en-1.5B-v5 embedding model. The embedding vectors of the question and context descriptions are computed as:

$$\mathbf{e}_q = f_{stella}(q), \quad \mathbf{e}_c = f_{stella}(c)$$

The relevance score between the query and a context is computed using cosine similarity as defined in Formula (2.1). Finally, contexts are ranked by sorting the cosine similarity scores in descending order:

$$\mathcal{D}_q = \text{rank}_{d \in \mathcal{D}} \left(\text{score}(q, d) \right) \tag{5.1}$$

[2]https://www.elastic.co/docs/solutions/search/vector

Prompt Structure

Instruction
Please act as a life assistant to answer questions based on the given contexts. You are given a question about life and a context describing time, location and description of an event of a person. Your goal is to provide an answer to that question based on deriving information from the context. Provide only the answer from the given context and no explanation.

Examples (Optional)
<few shot QA examples/>

CoT (Optional)
<few shot QA examples with CoT/>

Question
<question/>

Retrieved contexts
<contexts/>

Please provide the answer for question with only information from retrieved contexts.
Answer:

Figure 5.2: A structure of a prompt for LLMs

5.2.2 Answer Generation

In the RAG approach to lifelog QA, LLMs receive a prompt, retrieved contexts, and a question as input and generate an answer. A prompt for RAG includes several sections, from instruction to examples, optional CoT, retrieved contexts, and questions. The structure of the prompt provided to LLMs can be seen in Figure 5.2. For the instruction, I refer to some prompt engineering guidelines[3] to construct a role-played, clear goal and objectives for LLMs. I experiment with different types of examples in the prompt, from few-shot (5 examples of Q and A) to Chain-of-thought (CoT) [166] (several examples of intermediate explanations leading to the answer). The question and retrieved contexts are appended after examples. Finally, the word "Answer:" is placed at the end to prompt LLMs to directly generate an answer without any other supplementary text. The prompt is then fed to LLMs to generate the answer.

I use two LLMs for the experiment, Mistral 7B Instruct V0.3 [69] and Llama 3.1

[3]https://cloud.google.com/discover/what-is-prompt-engineering

8B Instruct [2], and two other smaller models, BART [86] and T5 [123] for comparison. I selected Mistral 7B and Llama 3.1 due to their strong performance across a wide range of benchmarks (MMLU [62], GSM8K [29], HumanEval [21]), where they were among the leading open-source models at the time of experimentation. Furthermore, both Mistral AI and Meta are recognised for their important contributions and leadership in large language model research and development, with numerous models released. Meanwhile, BART and T5 are two state-of-the-art language models from before the time of LLMs. They can solve various tasks in NLP and deliver high performance [86, 123]. I use these two models to provide a comparison with LLMs to highlight the challenge of the lifelog QA task and the need for an LLM-based RAG approach to address it. Details about these models are as follows:

Mistral 7B Instruct V0.3 [69] is a language model introduced by the Mistral.AI Research team. It has 7 billion parameters and is fine-tuned for instruction-following tasks to deliver precise, context-aware responses. Its performance across benchmarks is superior to that of Llama2 7B and Llama2 13B. Additionally, this model demonstrates reasoning capability in coding problems. Hence, I chose the Mistral 7B model for the generation part in RAG to solve lifelog QA questions, especially aggregation questions that require reasoning.

Llama 3.1 8B Instruct [2] developed by Meta is a state-of-the-art instruction-tuned language model optimized for multilingual conversational use. This model is trained on a massive dataset of over 15 trillion tokens, including both publicly available content and synthetic instruction examples. This model was released in July 2024 and achieved strong performance on benchmarks such as MMLU and HumanEval. I use this model to compare Mistral 7B Instruct V0.3 to comprehensively view LLMs in lifelog QA.

BART [86] and **T5** [123] models serve as baseline language models for comparison with LLMs. BART is a denoising autoencoder that combines bidirectional and autoregressive transformers for pretraining, making it effective across a range of generative tasks. T5, on the other hand, frames all NLP tasks as text-to-text problems

and is pretrained on a large corpus using a span-corruption objective. Both models are strong, pretrained sequence-to-sequence architectures, yet they are considered relatively small compared to modern LLMs like GPT-4 or Llama.

5.3 Experiment

This section presents the experimental evaluation of the proposed RAG pipeline for lifelog QA. I first describe the experimental settings, including dataset configuration, retrieval and generation hyperparameters, and evaluation metrics. I then report and analyse results across two stages: context retrieval, comparing BM25 and Stella-en-1.5B-v5 as retrieval models, and answer generation, comparing small language models (BART and T5) against LLMs (Mistral-7B and Llama-3.1-8B).

5.3.1 Experimental Settings

I used the OpenLifelogQA evaluation set to tune the retrieval and generation hyperparameters before running prediction and reporting the results on the test sets of OpenLifelogQA and DCU-TimelineQA. Information about these datasets has been presented in Section 3.2. As each question in the dataset has a specific date, for example, "How long did I have lunch on January 01, 2020?", there is a filter for retrieval on a date. For the retrieval setting, I set the top K retrieved lifelog contexts, with K from 1 to 100, as the maximum context per day is around 100. The vector embeddings for queries and lifelog contexts are 1024. I set up experiments on different retrieved contexts, including the top 20, the top 50, the all-day context (all contexts on the date of the questions), and the oracle (the ground-truth context of QA). For the LLMs, I set the temperature to 0.5, top P to 0.5, and the maximum number of generated tokens to 256. Details about setting up RAG for lifelog QA and reproducing experiments are published on GitHub [4]. All experiments are conducted on an NVIDIA A100 GPU with 80 GB of memory. For evaluation metrics,

[4]https://github.com/linh222/rag_openlifelog_qa

Table 5.1: Context retrieval results on the OpenLifelogQA test set.

Metrics	BM25	Stella-en-1.5B-v5
R@1	0.2299	0.3765
R@5	0.6829	0.7767
R@10	0.8171	0.8801
R@20	0.9138	0.9435
R@50	0.9859	0.9918
P@1	0.309	0.5265
P@5	0.1999	0.2479
P@10	0.1273	0.1476
P@20	0.0795	0.0854
P@50	0.0518	0.0524

the Recall@K and Precision@K as defined in Formula (3.2) and (3.1) are used for retrieval evaluation. BERTScore, ROUGE-L, and LLM-as-a-judge with the GPT-4o model, as defined in Formula (3.8), (3.3), and (3.9), are used as evaluation metrics for answer generation. Details about these metrics are already presented in Section 3.4.

5.3.2 Results

Table 5.1 provides the retrieval performance of two approaches for the context retrieval task on the OpenLifelogQA test set. This result has been presented in the previous chapter in Section 4.3.3, but I show it here again to illustrate the performance of retrieval in the RAG setting and point out how it affects the overall QA performance. On the OpenLifelogQA test set, BM25 performs worse than the Stella-en-1.5B-v5 model across all metrics. BM25 only achieves 22.99% Recall@1 and 91.38% Recall@20, compared to 37.65% Recall@1 and 94.35% Recall@20 in the Stella-en-1.5B-v5 model. The results indicate that the lexical-matching approach finds contexts for lifelog QA less accurately. This is due to a mismatch between the formats of temporal information in lifelog contexts and in queries. The diversity of date and time formats, such as 9:24 PM vs. 21:24, January 1, 2020 vs. 01/01/2020, makes it challenging for BM25 to find the correct context. Meanwhile, as the Stella-en-1.5B-v5 model is trained on massive data with various date and time formats, it

Table 5.2: Experimental result of two LLMs and baseline models for all-day contexts on the OpenLifelogQA test set (all lifelog contexts on the day of questions).

Models	BERT Score	ROUGE-L	LLM Score
BART	0.8131	0.0529	0.4077
T5	0.8356	0.0977	0.4936
Mistral-7B	0.9161	0.3954	0.6950
Llama-3.1-8B	0.9138	0.3838	0.7536

can better retrieve this information. Another reason for the low performance is the lack of information in the question. For example, questions like "What did I do at 3 PM on January 01, 2020?" do not contain any information about specific activity but only the time at 3 PM. If the retrieval model fails to understand time, it cannot find the correct context.

The retrieval results on the DCU-TimelineQA test set are presented in Table 4.5. It achieves a very high performance with approximately 93% Recall@1 and 99% Recall@5 from the Stella-en-1.5B-v5 model. Due to the synthetic nature of DCU-TimelineQA and the small number of candidates after performing date filtering, it is easier to find the correct context for the question than in the OpenLifelogQA dataset. The retrieved contexts are then used to prompt LLMs to generate answers with the results in the next paragraphs.

Table 5.3: Experimental results of two LLMs for the DCU-TimelineQA test set

Models	BERT Score	ROUGE-L	F1-Score
Mistral-7B	0.9748	0.8596	0.7614
Llama-3.1-8B	0.9812	0.9036	0.8298

Although the R@50 are approximately 99.18% from the Stella-en-1.5B-v5 model, I report the QA result from all-day contexts in Table 5.2 to show the performance of LLMs when provided with all necessary contexts. Both small language models perform poorly on this task. T5 is better than BART with 83.56% BERT Score and 0.4936 LLM Score. However, both models achieved an ROUGE-L score below 10%, indicating low performance in generating answers that match the format of the ground-truth answers. On the other hand, the Mistral 7B model outperforms the Llama 3.1 8B model in BERT Score and ROUGE-L metrics, but the Llama 3.1

8B model achieves a higher LLM Score. Llama 3.1 8B achieves a 91.38% BERT Score, 38.38% ROUGE-L Score, and 0.7536 LLM Score. This score indicates that the Llama model follows the instructions to generate accurate answers, but does not align the format of the ground-truth answers. In addition, although the BERT Score is high, the ROUGE-L and LLM Scores are moderate. This indicates that the semantics of predicted and ground-truth answers are highly correlated; however, the accuracy of the predicted answers is not high.

Table 5.3 reports the QA performance of two LLMs when provided with all-day lifelog contexts on the DCU-TimelineQA test set. I do not report the results from two SLMs (BART and T5) because the two models cannot generate the desired answer, resulting in low performance. Overall, Llama-3.1-8B outperforms Mistral-7B across all three metrics, achieving a BERT Score of 98.12%, a ROUGE-L of 90.36%, and an F1 score of 0.8298, compared to 97.48%, 85.96%, and 0.7614 for Mistral-7B, respectively. The high BERT Score of both models indicates that their generated answers are semantically close to the ground-truth answers. The F1 scores further confirm this trend, with Llama-3.1 8B achieving a notably higher F1 of 0.8298. These results suggest that when provided with relevant contexts, both LLMs can effectively leverage the contextual information to generate accurate and well-formed answers to lifelog questions.

The result here provides an answer for RQ2.1: *To what level of accuracy can LLMs in RAG answer open-ended lifelog questions from retrieved context?*. Overall, the results demonstrate that LLMs can leverage retrieved lifelog contexts to generate relevant, context-aware answers, substantially outperforming smaller language models. When provided with retrieved contexts, both Mistral 7B and Llama 3.1 8B achieve over 90% BERTScore and over 70% LLM Score. This shows that LLMs can understand lifelog contexts and capture the core meaning needed to answer open-ended lifelog questions.

However, the moderate ROUGE-L demonstrates that accurate semantic understanding does not always translate into fully correct or precise answers. In particu-

Table 5.4: Experimental results for different types of questions in the Llama 3.1 8B model for all-day contexts on OpenLifelogQA.

Question types	BERT Score	ROUGE-L	LLM Score
Atomic	0.9140	0.3616	0.7490
Temporal	0.9293	0.4344	0.8256
Aggregation	0.9085	0.4086	0.7383

lar, LLMs often struggle with numerical reasoning and fail to produce the expected answer format. Even when relevant contexts are available, the models may produce plausible but partially incorrect responses. Therefore, while LLMs show clear potential for lifelog QA within a RAG framework, their answer accuracy remains constrained by limitations in reasoning and precise information extraction. To gain a deeper understanding of the results, I perform several analyses, which are included in the following section.

5.4 Analysis

To further investigate the RAG approach in lifelog QA, I perform some analysis on various aspects, including performance on different question types, context settings, and prompts. This section presents these analyses and the qualitative error analysis in the predictions.

5.4.1 Performance Across Different Question Types

To examine LLMs' performance across different question types, I aggregate metrics by question type. Table 5.4 shows the performance of the Llama-3.1 8B model in all-day context settings for different types of questions. This model achieves the highest scores on temporal questions, with 43.44% ROUGE-L and 82.56% LLM Score, followed by Atomic with 36.16% ROUGE-L and 74.90% LLM Score. Aggregation questions are the most challenging type of question, and the Llama 3.1 8B model achieves the worst performance on this type, with an LLM Score of only 73.83%. Although the ROUGE-L of aggregation questions is higher than the figure

for atomic questions, it only indicates the lexical matching of predicted and ground-truth answers, not the accuracy of meaning. This result is expected because the aggregation question is difficult due to the complex reasoning required to calculate the performance from a lifelog context.

From the result of this analysis, the answer to sub-research question 2.2, *How does that accuracy decompose across the atomic, temporal and aggregation question types of OpenLifelogQA?* can be drawn. A baseline Llama 3.1 8B model in a RAG setting performs well across various question types, achieving strong results on temporal and atomic questions but showing low performance on aggregation questions. With an 82.56% LLM Score for the temporal question, this LLM demonstrates the ability to understand temporal contexts in lifelogging, including time dependencies between contexts such as preceding and following events, to generate correct answers. Atomic questions are simple and can be answered with information from a single event, which explains the LLM's good performance. In contrast, the aggregation question poses a greater challenge when it requires aggregation information from multiple events, requiring analysis or calculation before generating answers. There remains room for improvement in LLM reasoning in lifelog QA aggregation questions.

5.4.2 Performance in Different Context Settings

To examine the influence of context on answer generation, I analyse the performance of LLMs across different context settings, ranging from top-20 and top-50 retrieved contexts to all-day contexts (all contexts on the date of that question) to oracle contexts (ground-truth contexts for that question). Figure 5.3 illustrates the performance of two LLMs in lifelog QA tasks when different contexts are provided in the prompt. It can be seen that, as the number of contexts increases, this increase is observed in the BERT Score but not in the two other metrics. This is expected because the Recall@20 and Recall@50 are not very different, but the longer context length at 50 contexts makes LLMs fail to identify the correct contexts from which to derive the answer. All-day context performs the worst, as all contexts on the

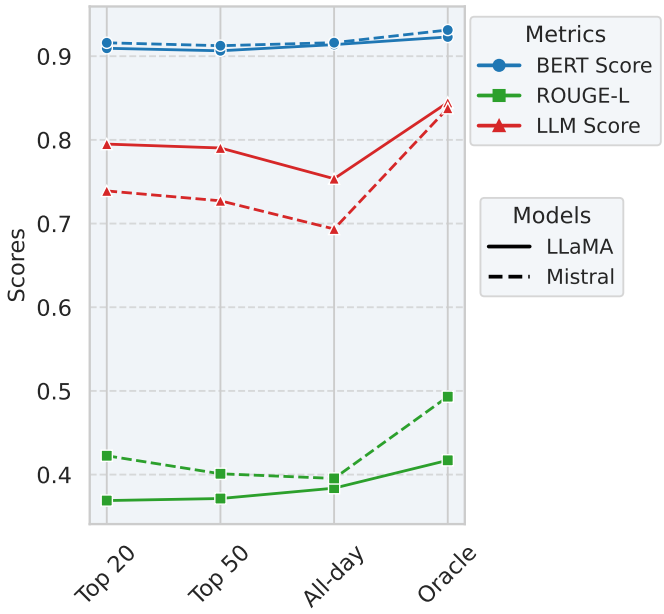


Figure 5.3: Performance of Llama 3.1 (line) and Mistral (dashed line) on OpenLifelogQA under different context settings.

question date are long and unranked, so LLMs fail hard in this context. When provided with ground-truth context, the performance boosts. However, when provided with oracle contexts, LLMs do not always generate correct answers. It indicates that LLMs cannot fully understand the context to generate correct answers, or the ground-truth contexts still lack information to generate an answer. For this reason, I perform error analysis to identify LLM failure modes.

5.4.3 Performance on Different Prompts

As discussed in Section 5.2.2 with three different types of prompts, including zero-shot learning, few-shot learning, and CoT, I examine the influence of prompts on answer generation by experimenting with different prompts. Figure 5.4 illustrates the performance of the Llama 3.1 model across different types of prompts. Notably, the prompt has no big impact on the model's performance. Few-shot learning and CoT are emerging capabilities of LLMs, as seen in models with hundreds of billions of parameters [167], suggesting that the 7B and 8B models cannot learn from examples in the prompt. The CoT prompt provides slightly better performance than the other two prompts. However, few-shot learning is expected to improve LLM performance,

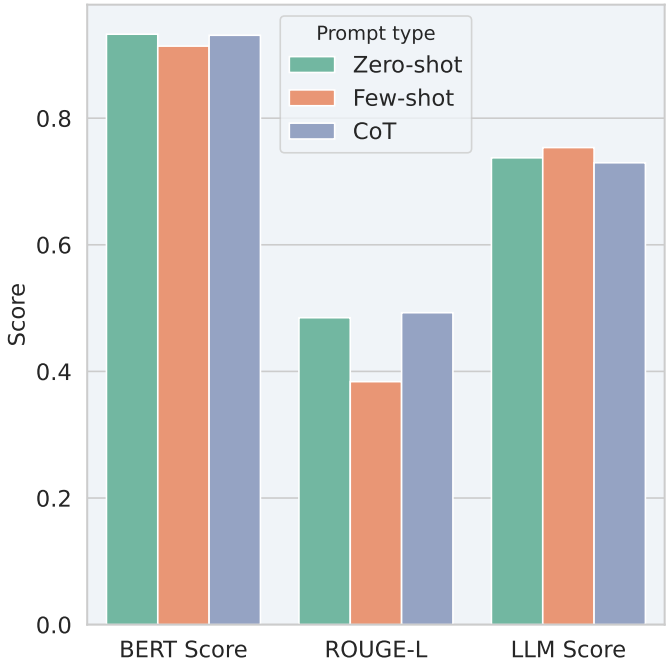


Figure 5.4: Performance of the Llama 3.1 8B model on OpenLifelogQA under different prompting instructions, measured using the scaled LLM score.

but it actually has the opposite effect. This may indicate that the LLMs are not large enough to follow the instruction examples. Further experiments are necessary to evaluate the effectiveness of prompts for lifelog QA in larger LLMs.

5.4.4 Qualitative Analysis

Table 5.5 shows some examples of correct and wrong predictions from Llama 3.1 models on oracle contexts. In the first example, the model correctly inferred the event duration, with only a slight discrepancy in phrasing, but the ground-truth answer is slightly incorrect, resulting in a 0.8 LLM score. However, in the second example, the model failed to capture the full semantic meaning of the event, leading to context confusion and an incomplete answer, resulting in a lower score. It misunderstands the intention of where to go to provide a correct answer. These examples highlight the strengths and weaknesses of LLMs in the lifelog QA task, revealing the room for improvement.

Table 5.5: Some examples of Llama 3.1 predictions

Question	**Oracle Contexts**	**Ground-truth Answer**	**Predicted Answer**	**LLM Score**
How long did I prepare dinner on March 15, 2020?	From 05:01 PM to 05:54 PM on March 15, 2020, I finish checking and prepare dinner at Home.	I prepared dinner for about 52 minutes.	I prepared dinner in 53 minutes.	0.8
Where did I go at 7:08 AM on June 4, 2020?	From 07:08 AM to 07:09 AM on June 04, 2020, I went to the gate to receive a courier package at Home.	I went to the gate to receive a courier package.	Home	0.2

5.5 Chapter Summary

This chapter investigates a solution for lifelog QA through the RAG approach and aims to find an answer to RQ2: *How effective is RAG when generating answers for LifelogQA from retrieved contexts?* by systematically evaluating different retrieval models and LLMs in RAG for lifelog QA. Through extensive experiments and detailed analysis across various aspects of RAG, several important findings have been made regarding its effectiveness and limitations for lifelog QA. Overall, there are 3 important points to point out:

Firstly, the experimental results demonstrate that retrieval performance affects overall RAG performance and is a critical prerequisite for successful lifelog QA. As the analysis depicted in Figure 5.3 shows, longer context does not always bring better performance, but a short and well-ranked list of contexts can do that. Compared to the lexical BM25, the semantic retrieval approach based on Stella-en-1.5B-v5 consistently achieves higher Recall@K and Precision@K. This highlights a better approach to retrieval in lifelog QA: an embedding-based approach. This approach can capture and present temporal expressions, activity descriptions, and contextual details that exhibit high variability in surface form. Semantic embeddings are better suited to bridge this mismatch and retrieve relevant contexts even when queries lack explicit activity keywords. Nevertheless, retrieval remains imperfect, especially for

questions that rely heavily on implicit temporal cues or sparse query information. This highlights that effective RAG architectures for lifelog QA must incorporate retrieval models that are robust enough to capture temporal order and heterogeneous lifelog descriptions.

Secondly, the generation results demonstrate LLMs' superior performance compared to small language models. Both Mistral 7B and Llama 3.1 8B show strong semantic alignment between generated and ground-truth answers, as indicated by high BERTScore values. However, the discrepancies between BERTScore, ROUGE-L, and LLM Score imply that semantic similarity does not always imply correctness. ROUGE-L is below an average point of 50%, while LLM Score is around 75% to 80% accuracy. This is particularly evident in aggregation questions, where models often generate plausible but incorrect answers, such as a wrong number. Therefore, an effective RAG architecture for lifelog QA must go beyond surface-level generation and support reliable reasoning over multiple events.

Third, fine-grained analysis across question types shows that baseline LLMs within a RAG framework perform unevenly. Temporal and atomic questions are handled relatively well, as they typically require identifying one or a small number of relevant events and extracting explicit information. In contrast, aggregation questions remain challenging due to the need for multi-step reasoning, counting, or duration calculation across multiple contexts. This finding suggests that while the current RAG approach is effective for retrieval-augmented recall, they are less effective for retrieval-augmented reasoning. This problem leaves room for improvement in aggregation questions in lifelog QA.

Other than those 3 key points, RAG for lifelog QA also demonstrates scalability. To address the challenge of the growing velocity of lifelog data, the RAG approach is highly scalable. As new lifelog data is generated, automatic event segmentation and description creation can be done using advanced image-to-text models, such as BLIP2 [90]. These descriptions can be processed and embedded into vector embeddings before indexing them in databases. Furthermore, this approach supports

incremental updates, requiring only new data to be indexed. The RAG approach allows retrieval models and LLMs to handle new data without additional fine-tuning. However, the use of LLMs for lifelog data needs consideration for privacy. The deployment of lifelog QA systems introduces privacy concerns due to the sensitive nature of personal data involved. Lifelogs often include intimate details of an individual's daily life, such as social interactions and location history. Ensuring the security and confidentiality of this data requires robust mechanisms for encryption, access control, and secure storage. Furthermore, since LLMs are not trained on personalised data, their responses may reflect generic biases or incorrect assumptions, particularly when questions lack sufficient context.

There are some limitations in this RAG approach. The performance of different prompt types shows a drawback of LLMs for lifelog QA. The evaluation indicates differences in the effectiveness of zero-shot, few-shot, and CoT prompts. This evaluation suggests that a small-parameter LLM may lack the capacity to exploit prompt engineering techniques. Larger models or fine-tuned variants could improve performance, particularly for complex reasoning tasks such as aggregation and temporal questions. Additionally, the evaluation is limited to a single lifelog dataset, and the findings for other lifelog collections or real-world deployments remain to be validated.

In conclusion, this chapter demonstrates that RAG architectures are a viable and effective approach for lifelog QA, particularly for handling large-scale lifelog data and open-ended questions. Effective integration of retrieved lifelog context requires robust semantic retrieval tailored to temporal and personal data, as well as LLMs capable of contextual understanding. While current RAG designs show strong performance on atomic and temporal questions, aggregation questions in lifelog QA remain an open challenge. These findings answer RQ2 by showing that RAG architectures can effectively support context-aware lifelog QA, but further enhancements are necessary to fully address the complex reasoning demands inherent in lifelog data. The next chapter focuses on addressing this problem by proposing reasoning LLMs for lifelog QA.

Chapter 6

Reasoning Large Language Models for Lifelog QA

In this chapter, I present reasoning-enhanced lifelog LLMs for lifelog QA, specialised for LLMs that can generate reasoning before answering, as mentioned in Section 2.3.3. The work in this chapter aims to address RQ3: *Can LLMs with reasoning capability improve the accuracy of answers for aggregation questions in lifelog QA compared to standard LLM baselines in RQ2?*. This question focuses on aggregation questions, a type of question requiring aggregating information from multiple event information to answer in lifelog QA, as mentioned in Section 1.1.2. This type of question poses challenges for standard LLMs to find the answer, as the result from the last experiment in Section 5.3 shows. To find the answer to this RQ, I first introduce reasoning LLMs in Section 6.1 and explain why lifelog QA needs them for aggregation questions. Section 6.2 introduces an approach to enhance reasoning in lifelogQA using different prompt strategies and fine-tuning methods. The experiment section 6.3 provides information on the experiment setup and results. Analysis and discussion are presented in Section 6.4. Finally, the conclusion and answer for RQ3 are discussed in Section 6.5.

6.1 Introduction

Aggregation questions such as "What is my favourite airline in 2020?" or "How long did I exercise last week?" pose a challenge for the lifelog QA model, as shown in the experiment in Section 5.3. Unlike atomic questions that can be answered from a single context, aggregation questions require the model to reason across multiple contexts of lifelog data. This makes them particularly demanding, as the model must not only retrieve all relevant contexts but also perform numerical or comparative reasoning over them, such as summing durations, counting occurrences, or ranking frequencies. These questions are also among the most practically valuable in a personal lifelog setting, as they reflect the kind of reflective, long-term queries that users are most likely to ask about their own lives, such as tracking health habits, travel patterns, or daily routines over time. This motivates further investigation into dedicated strategies for handling aggregation queries. To address the challenge posed by aggregation questions, both the retrieval and generation stages of RAG should be robust enough to identify relevant contexts and analyse them effectively to generate the answer. Because aggregating and calculating capabilities are required to generate correct answers, reasoning LLMs are a potential model for this type of question.

Reasoning LLMs are generative LLMs that have also acquired reasoning capabilities. Reasoning capability allows models to generate intermediate rationales before deriving the final answer. While general reasoning capability focuses on solving mathematical and code-generation problems, reasoning for lifelog aggregation questions stands out for several key differences. Firstly, aggregation questions require LLMs to analyse the context, determining whether it is relevant before generating answers. Some contexts can confuse LLMs, such as for a question "How long did I stay at the workplace last Friday?" and contexts "At 12:14 PM on Friday, I left my office for lunch at the canteen." LLMs can misunderstand the context of "left office" and choose it as the end time of the question. Secondly, although some LLMs are very good at reasoning, some mathematical problems, especially time-related ones,

such as calculating the difference between two timestamps, still trick the models.

To solve RQ3, I begin by extending a lifelog-specific RAG pipeline in previous chapter with context reranking from Section 4.3, and benchmark a diverse set of LLMs, including open-source LLMs (Llama 3.1 8B Instruct [2], Qwen3 8B, Qwen3 14B [150], GPT-oss-20B), proprietary LLMs (GPT-4o [112], GPT-o4-mini), and vision-language models (LLaVA Next Interleave 7B [88]) in the RAG setting. The model choice has been discussed in Section 3.5. A fine-grained analysis reveals that while current models can handle simple atomic questions, small open-source models consistently struggle with reasoning-intensive aggregation queries. Although the calculations required for aggregation questions are not as complex as those involved in coding or high-level mathematics, open-source LLMs struggle to reason about temporal context and perform accurate calculations. To address this gap, I propose a fine-tuning approach that combines supervised and reinforcement learning, aiming to strengthen temporal and aggregation reasoning in LLMs for lifelog QA.

6.2 Enhancing Reasoning for LifelogQA

Figure 6.1 illustrates the full pipeline of enhancing reasoning LLMs in RAG for lifelog QA. It is divided into two phases, offline and online. In the offline phase, the data for contexts, embeddings, and metadata is indexed in the Elasticsearch database. In addition, reranking model training and reasoning-enhanced LLM fine-tuning are also performed in this step. In the online phase, the question is processed, retrieved contexts are formulated into a prompt, and the answer is generated. As the retrieval result has been presented in Chapter 4, in this chapter, I mainly focus on the answer generation part with LLM fine-tuning and prompt formulation.

There are several key differences and enhancements in this pipeline compared to the RAG pipeline in Section 5.2. Firstly, I apply a reranker from Section 4.3 to improve the ranking quality of retrieved contexts. This aims to compare with the baseline RAG in RQ2 without reranking. Secondly, the prompt formulation is enhanced with dynamic prompt selection, which will be presented in the following

section. A more tailored prompt for each question word is designed to provide LLMs with clear, specific examples of how to approach and solve them. Finally, fine-tuned LLMs with reasoning enhancements are the main difference compared to the previous RAG pipeline and are expected to solve the aggregation questions effectively. Details about reasoning LLMs are provided in the following sections.

6.2.1 LLMs

Lifelog data is personal and private, so LLMs are not directly trained on it to answer questions related to lifelogging. However, thanks to the vast amount of trained data and strong capabilities in in-context learning, LLMs can refer to examples of lifelog QA pairs to answer new lifelog questions. I aim to explore the use of LLMs in processing retrieved contexts and generating the answer in lifelog QA, so I experiment on a wide range of LLMs, from proprietary to open-source models of various sizes. Proprietary LLMs are closed-source models that need to be paid to use, and I chose two models from the OpenAI provider. Open-source models are chosen from Llama from Meta and Qwen from Alibaba.

Although I primarily focus on the textual contexts of lifelog events to answer questions, as they convey most of the meaning of events, lifelog images also contain information that the contexts may miss. For example, a text description might record that a person "had lunch at a café", but the accompanying image could reveal the specific food on the plate, the brand of drink, or even a companion sitting across the table, and the question asks for the food. Therefore, I also evaluate a VLLM that incorporates both contexts and images in a single prompt for answer prediction.

Details about these models are provided as follows:

- **GPT-4o** [112]: is a state-of-the-art omni model from OpenAI. It is trained on a massive amount of data from websites, code & math, and multi-modal data. Although the model can be prompted to generate step-by-step reasoning thanks to training on code and math data, I classify it as a non-reasoning LLM because it requires explicit prompting to produce step-by-step reasoning,

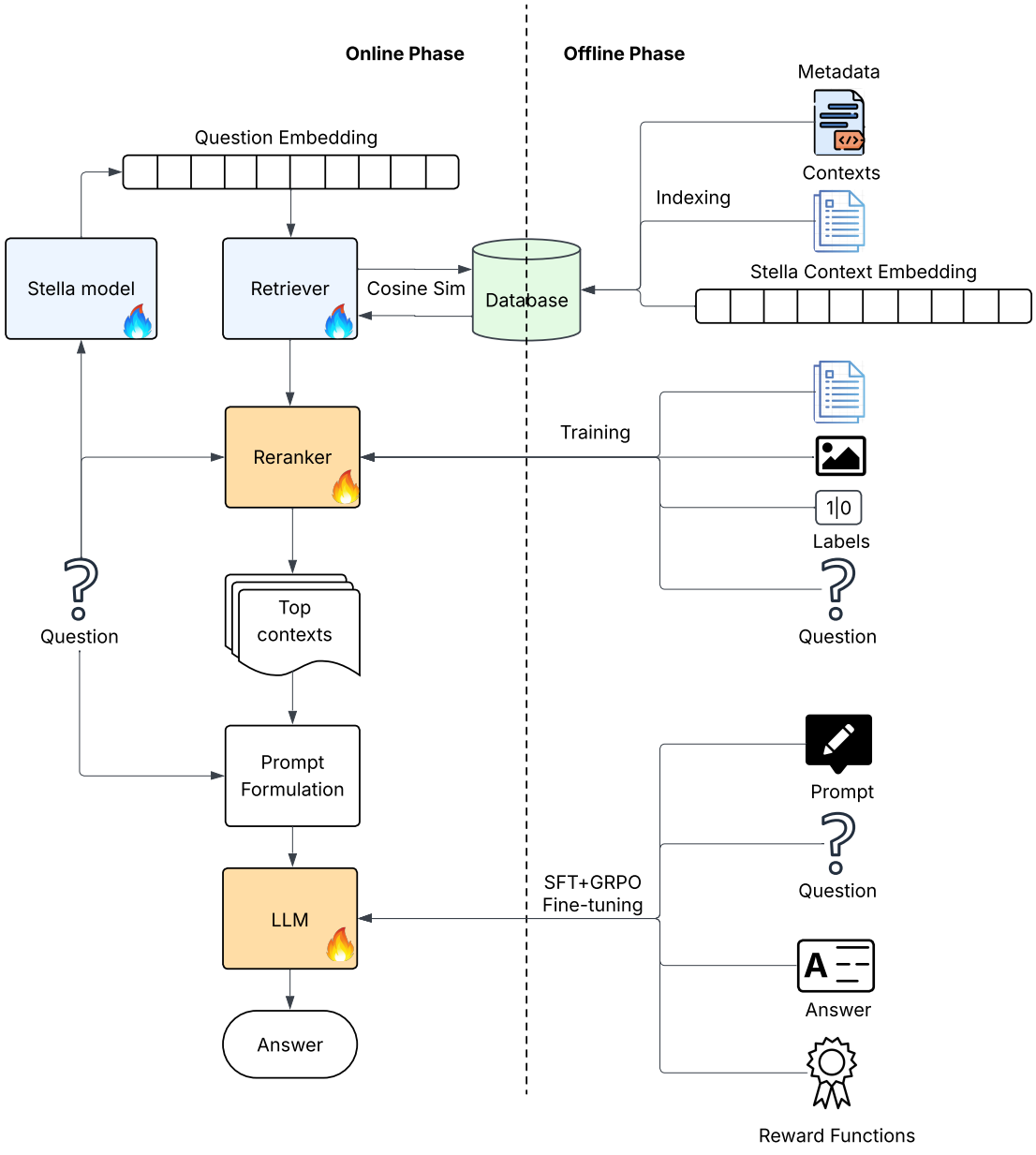


Figure 6.1: Pipeline for enhancing reasoning LLMs in lifelog QA.

rather than doing so inherently. GPT-4o matches GPT-4-Turbo in English text and code generation while surpassing it in non-English text generation. However, with 50% cheaper API cost than GPT-4-Turbo, I choose it to play as a non-reasoning proprietary model for lifelog answer generation.

- **GPT-o4-mini** [1]: is a reasoning model trained to think longer before responding. It is optimised for fast and cost-efficient reasoning while achieving SOTA performance in AIME 2024 and 2025 benchmarks. I chose it as a proprietary model due to its high performance and low cost.

- **GPT-oss-20B** [2]**:** is an open-sourced reasoning model from OpenAI. GPT-oss-20B is a transformer-based model leveraging Mixture-of-Experts (MoE) [139] with 3.6B active parameters in a total of 20B parameters. Thanks to training on the chain-of-thought prompt and reinforcement learning for reasoning, this model demonstrates strong reasoning capabilities. I use this model to compare with reasoning models from Qwen3 and closed-source GPT-4o and o4-mini models.

- **Qwen3 14B** [150]: is a member of Qwen3 family models with 14 billion parameters. It is a reasoning model, thanks to the Strong-to-Weak Distillation pipeline, which enables learning of reasoning capability from a larger model, such as Qwen3 235B. Qwen3 14B Base achieves a higher performance than Phi-4 and Qwen2.5-32B-Instruct on various benchmarks from general tasks, math & STEM tasks, to coding tasks. I use this model to compare it with the Qwen 8B model and evaluate the improved performance that results from increasing LLM size.

- **Qwen3 8B** [150]: is in the same family with Qwen3 14B but only has 8 billion parameters. It is also a reasoning model trained in the Strong-to-Weak Distillation pipeline. It outperforms other models of the same size, such as

[1] https://openai.com/index/introducing-o3-and-o4-mini/
[2] https://openai.com/index/introducing-gpt-oss/

Llama-3.1-8B-Instruct, Gemma-3-12B-IT, or even Qwen2.5-14B-Instruct, in various benchmarks. I use this model to compare with Qwen 14B for performance differences in size and Llama 3.1 8B Instruct for non-reasoning versus reasoning in lifelog QA.

- **Llama 3.1 8B Instruct** [2]: is an instruction-tuned with 8 billion parameters and a 128k-token context window, introduced in July 2024. It is a non-reasoning model, but it can generate step-by-step rationales through guidance in a prompt. I observe that the model is prone to hallucinating thoughts when the question is challenging, making it less well-suited for reasoning tasks. I use this model to compare it with Qwen3 8B in terms of its ability to generate answers in lifelog QA and to improve its reasoning capabilities in temporal and aggregation reasoning.

- **LLaVA Next Interleave 7B** [88]: addresses the challenge of single-image task visual instruction tuning for LLMs by introducing multi-frame/image integration into the LLM prompt. I use the Qwen 1.5 7B version as the LLM backbone to enable direct comparison with other LLMs of similar size. The model achieves state-of-the-art performance in various multi-image and video understanding tasks, demonstrating superior capability in reasoning over temporally and spatially related visual inputs. It is a suitable model for lifelog QA testing, where temporal reasoning and multiple-image inputs are crucial.

6.2.2 Prompt

I designed a dedicated prompt for lifelog QA, with details organised into sections inspired by a prompt engineering guideline[3]. There are seven sections in the prompt, but some of them are optional for experimental purposes, including Role& Objective, Instruction, Reasoning (Chain-of-thought), Examples, Output Format, Question, and Retrieved Context. This prompt structure is inherited from the prompt struc-

[3]https://cloud.google.com/discover/what-is-prompt-engineering

Prompt Structure

Role and Objective
You are a life assistant AI designed to answer life-related questions based on contextual event data from a person's lifelog.

Instruction
You are given a question and a context that describes the time, location, and activity of a person. Your task is to answer the question based solely on the context by deriving information carefully and logically. Each question word has a different way to find the answer; for example, for a "how long" question: You find the start time and end time of an event in the context, then calculate the difference to find out how long the event lasted.

Reasoning steps (Optional)
<Steps of performing reasoning to find the answer/>

Examples (Optional)
<few-shot QA examples/>

Output Format
Provide the answer as a short sentence starting with "I . . . ". Answer must be under 10 words. No extra explanation is allowed.

Question
<question/>

Retrieved contexts
<contexts/>

Please provide the answer for the question with only information from the retrieved contexts.
Answer:

Figure 6.2: A structure of a prompt for LLMs. In certain experiment settings (as seen in later sections), optional sections are excluded. For example, the Examples can be omitted to test the models' one-shot capabilities.

ture in Section 5.2.2, but it also presents clearer sections and detailed instructions. The full prompt structure can be seen in Figure 6.2.

The Role&Objective section provides short guidance on the role of LLMs in this problem. The Instruction section guides the model in the task of lifelog QA that it needs to perform. Examples are selected dynamically based on the question word (What, Where, How) to provide specific examples for each question word. Three examples (3-shot) are chosen for the experiments. The Reasoning section provides instructions for the chain-of-thought [166] strategy to find the answer. The Output format section describes the guidance for the desired output format of the answer,

to ensure a high match between predicted and ground-truth answers. Finally, the question and the list of retrieved contexts are included in the prompt. For VLLMs (i.e., LLaVA Next), the image for each context is inserted after the textual description in an interleaved manner, as suggested by the authors of LLaVA Next. An example of a prompt main structure can be found in the Appendix Section A.1.

6.2.3 Fine-tuning methods

To enhance LLM performance on lifelog QA, particularly in temporal reasoning and information aggregation, I propose a hybrid fine-tuning approach. I combine both supervised fine-tuning and reinforcement learning through Group Relative Policy Optimisation (GRPO) [138]. The supervised fine-tuning phase on atomic questions (simple, need only one event to answer) aims to provide the LLM with personalised lifelog contexts and question-answer pairs to help it familiarise itself with lifelog data. The GRPO fine-tuning phase on temporal and aggregation questions (related to time constraints and information aggregation, like count, sum) forces the model to think step-by-step to derive accurate answers. This dynamic thinking strategy optimises the efficiency and accuracy of lifelog QA by mimicking how humans answer questions. For easy questions, such as atomic questions, whose answers can be derived from a single context, LLMs can quickly generate the answers without effort. For more challenging questions, such as temporal and aggregation, LLMs are forced to think and reflect before providing final answers. Details about two fine-tuning methods are provided as follows:

Supervised Fine-tuning (SFT): For each QA pair in the training dataset, I formulate an input of a prompt, retrieved contexts, and a question. The output is the answer to that question. The optimisation minimises the standard token-level cross-entropy loss between the model's generated output and the ground-truth answer. LLM learns to locate specific contexts and generate answers, improving its ability to handle factual recall and straightforward atomic questions. Formally, the

loss is defined as:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i}\log P_\theta\left(y_t^{(i)} \mid x^{(i)}, y_{<t}^{(i)}\right) \tag{6.1}$$

where N is the number of training examples, T_i is the length of the target answer in the i-th example, $x^{(i)}$ represents the input sequence, $y_t^{(i)}$ is the target token at position t, and P_θ denotes the probability predicted by the model with parameters θ.

This phase ensures that LLMs already understand the lifelog domain before tackling more complex temporal and aggregation reasoning. In addition, supervised fine-tuning helps the model generate the desired answer format, which increases the lexical matching score, such as ROUGE-L.

Reinforcement Learning Fine-tuning: I adopt Group Relative Policy Optimisation (GRPO) to fine-tune LLMs for temporal and aggregation questions. GRPO, originally introduced in the DeepSeekMath model [138], is an efficient variant of Proximal Policy Optimisation (PPO) designed for LLM training. Unlike PPO, which relies on a separate value model to compute advantages, GRPO eliminates the need for value function approximation by using the average reward of multiple sampled outputs for the same question as the baseline. Specifically, for each question, the old policy samples several responses, each scored by a reward model, and the relative advantage of each output is computed with respect to the group average. This design reduces memory and computational overhead, stabilises training, and aligns well with the comparative nature of reward models. GRPO then optimises the policy to maximise these relative advantages while applying a KL-divergence regularisation term between the policy and a reference model to prevent over-optimisation. The GRPO objective has been defined in (2.6). I propose several reward functions to optimise the reasoning capability of LLMs in lifelog QA and the accuracy of the answer. These functions span from matching format and reasoning effort to the accuracy of the answer. Details about these reward functions are as follows:

- Matching format functions: I developed two functions to verify the output format from the LLM. One of which is to ensure the LLM generates 2 thinking tokens (<think>and </think>), the thinking content between the tokens, and the final answer after the </think>token. The second function ensures that only 1 pair of </think>is generated; any more will incur a penalty.

- Reasoning effort function: I design a reward function that evaluates the length of the model's generated reasoning. Since temporal and aggregation questions are generally less complex than mathematical or coding problems, I limit the model's reasoning to fewer than 1000 words. I found that extended reasoning often results from repetitive self-reflection, even when the answers are already correct, unnecessarily slowing the generation process. Therefore, I set a 1000-word cap to improve efficiency.

- Answer accuracy functions: I use Exact Match, BERT Score [185], and ROUGE-L Score [93] to measure the accuracy of the final answer, compared to the ground-truth answer. These metrics aim to optimise the format and style of open-ended answers to follow the ground-truth answer. LLM-as-a-judge [48] can be used to evaluate the accuracy of the answer; however, due to its high cost, I only utilise the automatic metrics above.

Due to limited computing resources, I fine-tune only 8B-sized LLMs. I chose Llama 3.1 8B Instruct and Qwen3 8B. While the Qwen3 8B model already possesses thinking capability, the Llama 3.1 8B Instruct model does not natively support thinking tokens or extended reasoning chains. Therefore, I acquire the Llama 3.1 model with reasoning capability and enhance Qwen3 8B's reasoning capability using lifelog data. Combining two fine-tuning approaches for Lifelog LLMs improves not only the alignment of answers but also semantic accuracy through enhanced reasoning capabilities.

6.3 Experiment

This section provides information on the experimental setting for evaluating and fine-tuning LLMs for lifelog QA. Additionally, the results are discussed, with several analyses of aspects of LLMs for lifelog QA.

6.3.1 Settings

For retrieval, I employ the Stella model [183] to encode textual contexts into dense embeddings, which are indexed in Elasticsearch alongside metadata such as time and location. This is the same model and method described in Section 4.3. For each question, the top 100 contexts are retrieved based on cosine similarity. This choice is based on evaluation on the validation set, with high recall at 100, ensuring that most or all relevant contexts are included within this set, but reducing the number of retrieved contexts lowers recall, meaning some relevant contexts may be excluded or ranked too low to be effectively utilised. Each retrieved event is paired with its representative image and passed to the reranking model to rerank the list. I select the top 20 contexts for the final input, as the performance difference between the top 20 and top 50 is marginal, while the shorter list reduces context length and speeds up prediction.

I use the training set of the OpenLifelogQA dataset to fine-tune LLMs, with the hyperparameters tuned through the evaluation set. The results reported in this section are predictions on the OpenLifelogQA test set. In addition, the DCU-TimelineQA test set is also used for evaluating the models trained on the Open-LifelogQA dataset to measure the transfer ability of different lifelog datasets. The evaluation metrics used to evaluate the QA models are BERTScore, ROUGE-L, and LLM Score, which have been described in Section 3.4.

For non-fine-tuned LLMs, I set both temperature and top_p to 1, with no limit on max_completion_tokens to allow the model to use the maximum available context window for reasoning. For supervised fine-tuning, I use the Unsloth framework with the following settings. Llama 3.1 8B Instruct and Qwen3 8B are fully fine-

tuned on all weights using a batch size of 4, gradient accumulation steps of 8, a learning rate of 1×10^{-5}, weight decay of 0.01, and the AdamW optimiser. Generation parameters are set to a temperature of 0.1 and top_p of 1. For Qwen3 14B, I perform 8-bit fine-tuning with LoRA adaptation (rank 128, alpha 256), a maximum sequence length of 8096, a batch size of 4, 16 gradient accumulation steps, and 5 training epochs, with early stopping after 60 evaluation steps. The learning rate is set to 3×10^{-5}, weight decay to 0.05, and AdamW is used as the optimiser. Generation parameters are a temperature of 0.1 and top_p of 1. All of these parameters are tuned on the evaluation set to pick the combination of hyperparameters that brings the best results.

For GRPO fine-tuning, I follow the Unsloth RL guide. Both Llama 3.1 8B Instruct and Qwen3 8B are fine-tuned with LoRA adaptation (rank 8, alpha 16), a maximum of 4048 new tokens, sampling parameters including temperature of 1, and top_p of 1, generating 4 outputs per sample. Training is conducted for 2 epochs with a batch size of 4, gradient accumulation steps of 16, a learning rate of 5×10^{-6}, and the AdamW optimiser. All experiments are conducted on a single NVIDIA A100 GPU with 80GB VRAM. The source code is available at: `https://github.com/linh222/reasoning_lifelog_llm`. In this experiment, I employ ROUGE, BERTScore, and LLM-based evaluation as defined in (3.8), (3.3), and (3.9) to comprehensively measure both lexical and semantic performance of lifelog QA models.

6.3.2 Results

Table 6.1 shows the performance of LLMs on the OpenLifelogQA test set. I compare the non-fine-tuned models, supervised fine-tuning, supervised & GRPO fine-tuning, to the proprietary model on reranked contexts with a chain-of-thought prompt. Without fine-tuning LLMs on lifelog data, the 8B-sized model fails to follow the prompt's example and generates answers that are poorly aligned with the expected output format, resulting in low Exact Match, BERTScore, and ROUGE-L scores.

Table 6.1: Performance comparison of LLM models on the OpenLifelogQA test set. The highest scores are highlighted in bold, while the second-highest are underlined.

Model	Exact Match	BERTScore	ROUGE-L	LLMScore
Non-Fine-tuned LLM				
LLaVA Next Interleave 7B	0.0312	0.9095	0.3646	0.6125
Llama 3.1 8B Instruct	0.0186	0.9261	0.4495	0.7565
Qwen3 8B (Non-thinking)	0.0597	0.9312	0.5041	0.8147
Qwen3 8B (Thinking)	0.0802	0.9273	0.4995	0.8610
Qwen3 14B (Non-thinking)	0.1041	0.9367	0.5377	0.8447
Qwen3 14B (Thinking)	0.1141	0.9366	0.5420	0.8740
GPT-oss-20B	0.1340	0.9428	0.5758	0.8849
SFT Fine-tuned LLM				
LLaVA Next Interleave 7B	0.2075	0.9492	0.6207	0.8259
Llama 3.1 8B Instruct	0.1300	0.9408	0.5497	0.8469
Qwen3 8B	0.1592	0.9495	0.6154	0.8772
Qwen3 14B	**0.2301**	<u>0.9522</u>	<u>0.6353</u>	0.8922
SFT+GRPO Finetuned LLM				
Llama 3.1 8B Instruct	0.1061	0.9389	0.5452	0.8461
Qwen3 8B	<u>0.2261</u>	**0.9542**	**0.6468**	<u>0.8992</u>
Proprietary LLM				
GPT-4o	0.1618	0.9475	0.5877	0.8902
GPT-o4-mini	0.1605	0.9499	0.5994	**0.8997**

LLaVA Next Interleave 7B performs worst with only 3.12% Exact Match, 90.95% BERTScore, 36.46% ROUGE-L, and 61.25% LLM Score. Llama 3.1 8B Instruct performs better with 75.65% LLMScore and 44.95% ROUGE-L. Qwen3 models, including the 8B and 14B versions, achieve higher scores than previous models, and, as expected, the larger version performs better than the smaller one. Additionally, generating chain-of-thought text between thinking tokens during the generation process boosts performance compared to not generating rationales (Non-thinking mode). GPT-oss-20B, with the highest number of parameters, outperforms other non-fine-tuning open-source models, achieving 13.40% Exact Match, 57.58% ROUGE-L, and 88.49% LLMScore. Overall, it can be seen that larger models achieve higher scores due to their instruction-following capabilities. In addition, thinking models like Qwen3 and GPT-oss-20B also address aggregation questions better than non-thinking models. However, due to a lack of prior knowledge of lifelog data and the question-answering task, these models still fail to follow the answer format, resulting in low Exact Match and ROUGE-L scores. Fine-tuning can improve these

drawbacks, and the results are presented in the following paragraph.

After supervised fine-tuning, all models show improvement in answer format alignment, as evidenced by Exact Match, BertScore, and ROUGE-L scores. In the LLaVA Next Interleave 7B model, Exact Match improves from 3.12% to 20.75%, ROUGE-L gains 25.61% to 62.07%, and LLM Score also shows a 21.34% improvement. The same trend can be observed in the Llama 3.1 8B Instruct model, with an increase of approximately 11% in exact match, 10% in ROUGE-L, and 9% in LLM Score. Qwen3 models also improve performance on these metrics, but to a lesser extent. After fine-tuning, Qwen3 14B achieves the highest Exact Match score of 23.01%, the second-highest BERTScore of 95.22%, and the second-highest ROUGE-L score of 63.53%. However, supervised fine-tuning does not improve accuracy on aggregation questions, which require higher reasoning to derive the answer. Analysis in section 6.4 shows more details on this problem. From this improvement, I conclude that fine-tuning via a supervised approach enhances the capability of LLMs to align with ground-truth answers and improves the performance of LLMs that do not exhibit high instruction-following and general knowledge understanding, such as LlaVA Next Interleave 7B and Llama 3.1 8B Instruct. However, for higher-reasoning tasks, such as answering aggregation questions, supervised fine-tuning does not help the model accurately answer them. I then move on to combining supervised and GRPO fine-tuning to address this problem.

After GRPO fine-tuning, the Llama 3.1 model decreases the Exact Match, BERTScore, and ROUGE-L scores by a small margin, and the LLM Score remains nearly unchanged. However, upon deeper investigation, the LLM Score for aggregation questions has increased. More details about this behaviour can be seen in Section 6.4.3. The performance of the Qwen3 8B model improves, achieving the highest BERTScore and ROUGE-L, and the second-highest Exact Match and LLM Score. This result shows the improvement of GRPO fine-tuning on LLMs for lifelog QA.

For proprietary LLMs, both GPT-4o and GPT-o4-mini show remarkable performance on the OpenLifelogQA test set, with the highest LLM Score for GPT-o4-mini.

These models achieve good Exact Match (16%), BERTScore (95%), and ROUGE-L (59%) even without fine-tuning. These models outperform all non-fine-tuned models, demonstrating strong instruction-following capabilities and generating answer-like examples in response to the prompt. The LLM Scores of the two models exceed 89%, which is higher than those of most non-fine-tuned and fine-tuned models in the experiment. These results show the robustness of state-of-the-art LLMs in a task like LifelogQA.

6.4 Analysis

This section presents a comprehensive analysis of the experimental results from multiple perspectives. I first compare open-source and proprietary models to assess whether specialised fine-tuning can close the performance gap with closed-source models. Then, an analysis of the effectiveness of different fine-tuning strategies, including supervised fine-tuning and GRPO, across various prompt templates. A fine-grained breakdown by question type is presented to highlight the distinct challenges posed by atomic, temporal, and aggregation questions. I further evaluate model performance on the DCU-TimelineQA dataset and conduct an ablation study to examine the contribution of different input modalities and training data configurations. Finally, an error analysis is presented to identify the key failures of the best-performing model to provide insights for future improvement.

6.4.1 Open-source vs Proprietary Models

Proprietary models like GPT-4o or GPT-o4-mini outperform other models in LLM-Score, but achieve lower scores than fine-tuned open-source models in Exact Match, BERTScore, and ROUGE-L. With a large number of parameters and extensive training data, this result indicates the general knowledge and instruction that these proprietary models impart. However, a specialised, focused model for lifelog data, with a smaller size and suitable for commodity hardware, can achieve comparable

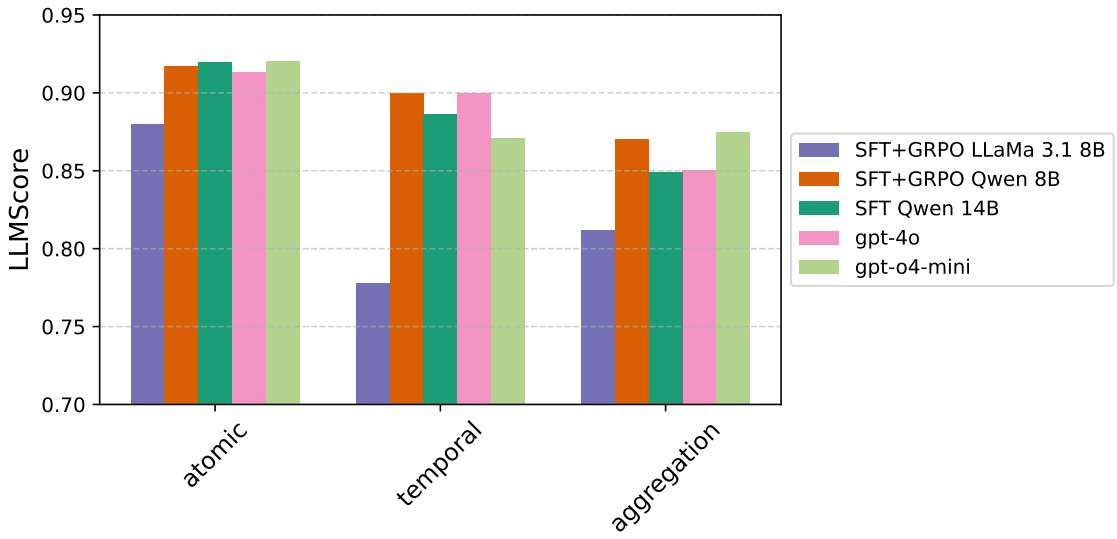


Figure 6.3: Open-source vs Proprietary models in three types of questions.

performance. After fine-tuning, the Qwen3 8B model achieves higher scores than GPT-o4-mini on Exact Match, BERTScore, and ROUGE-L. It also surpasses the LLMScore of the GPT-4o model, reaching the second-highest LLMScore of 89.92%.

Figure 6.3 below illustrates the LLMScore of 3 open-source and 2 proprietary models on three types of questions in the OpenLifelogQA test set. In atomic questions, supervised fine-tuned Qwen3 14B surpasses GPT-4o and achieves equal performance to GPT-o4-mini. In temporal questions, Qwen3 8B achieves a higher score than GPT-o4-mini and ties with GPT-4o. In aggregation questions, Qwen3 8B achieves a comparable performance to the GPT-o4-mini model. Qwen3 8B achieves a high score thanks to enhanced reasoning capability from GRPO fine-tuning, but is still lower than GPT-o4-mini. From the analysis result, it is clear that proprietary models are not much better than open-source models. However, a specialised model for lifelog data with fewer parameters is more efficient while achieving comparable performance to proprietary models.

6.4.2 Fine-tuning Strategies

I analyse the effectiveness of supervised fine-tuning and GRPO fine-tuning on three prompt templates for four different models, and the results are presented in Table

Table 6.2: Performance of fine-tuned models on different methods and prompt types

Model	FT Type	Prompt	Modal	BERTScore	ROUGE-L	LLMScore
LLaVA 7B	SFT	Zeroshot	T+I	0.9527	0.6285	0.8306
LLaVA 7B	SFT	Fewshot	T+I	0.9519	0.6220	0.8302
LLaVA 7B	SFT	CoT	T+I	0.9492	0.6207	0.8259
Llama 3.1 8B	SFT	Zeroshot	T	0.9497	0.6096	0.8424
Llama 3.1 8B	SFT	Fewshot	T	0.9409	0.5581	0.8288
Llama 3.1 8B	SFT	CoT	T	0.9408	0.5497	0.8469
Qwen3 8B	SFT	Zeroshot	T	0.9495	0.6158	0.8757
Qwen3 8B	SFT	Fewshot	T	0.9477	0.5951	0.8675
Qwen3 8B	SFT	CoT	T	0.9495	0.6154	0.8772
Qwen3 14B	SFT	Zeroshot	T	0.9515	0.6297	0.8814
Qwen3 14B	SFT	Fewshot	T	0.9454	0.5841	0.8622
Qwen3 14B	SFT	CoT	T	0.9522	0.6353	0.8922
Llama 3.1 8B	GRPO	CoT	T	0.9429	0.5916	0.8741
Qwen3 8B	GRPO	CoT	T	0.9294	0.5050	0.8553
Llama 3.1 8B	SFT+GRPO	CoT	T	0.9389	0.5452	0.8461
Qwen3 8B	SFT+GRPO	CoT	T	**0.9542**	**0.6468**	**0.8992**

6.2. Supervised fine-tuning has improved performance across all models, particularly in Exact match, BERTScore, and ROUGE-L scores. Exact match increases from 3-10% in the base model to 15-20% in fine-tuned models. The same trend is seen in ROUGE-L, with an increase from 40-50% to 55-64%. BERTScore increases by a smaller margin, with only 1-2% as it already achieves 94% in the base model. LLMScore increased to approximately 83% in the LLaVA Next Interleave model and reached 89.92% in the Qwen3 8B model. Supervised fine-tuning demonstrates good performance in enhancing answer alignment with the desired output, but it primarily improves for atomic questions. GRPO improves accuracy on temporal and aggregation questions, as evidenced by the LLMScore of the GRPO-fine-tuned Qwen3 8B model. GRPO fine-tuning enhances the performance of the Qwen3 8B model, which outperforms the supervised fine-tuned Qwen3 14B model.

Regarding the prompt template, differences in fine-tuning performance across templates are small. The CoT prompt achieves a higher LLM Score for several models, while the few-shot prompt is usually the weakest. Given the narrow margins, no single template is best across the board, and the most effective template depends on the type of question being asked rather than on the particular set of questions sampled. Aggregation questions, for example, require multi-step reasoning over

several retrieved events, so for GRPO fine-tuning I specify CoT in the prompt to ensure the model reasons explicitly before producing an answer.

6.4.3 Fine-grained Questions Analysis

A question word is a word used to ask a question, which can be generally classified into the 5 W's (what, where, when, why, and who) and 1 H (how). It is a finer classification of question types; for example, "How long" question words belong to the aggregation question type. I performed a fine-grained analysis on question words, and especially an analysis on aggregation questions, to see how LLMs generate answers for different questions. I analysed the LLM-Judge scores of QA responses broken down by question word from 4 LLMs, including supervised and GRPO fine-tuned LLama 3.1 8B Instruct, Qwen3 8B, supervised fine-tuned Qwen3 14B, and the GPT-o4-mini model. Figure 6.4 depicts the score in a bar chart. I highlight the Qwen3 8B and Qwen3 14B to show their superior performance in some question words. It can be clearly seen that the difference in the LLM Score between aggregation question words like how long, how many, and how much, and atomic question words like what, when, and where. In aggregation questions, the Qwen3 8B model beats the Qwen3 14B model. This higher performance indicates the benefit of GRPO fine-tuning on lifelog data. In W's question, Qwen3 8B and Qwen3 14B share dominance over these questions and usually surpass GPT-o4-mini's performance. In the case of "why" and other question words, GPT-o4-mini achieves the highest score and demonstrates its superior performance compared to other models.

I further investigate the performance of various models on aggregation questions, with detailed results presented in Table 6.3. This table provides a comparison of non-fine-tuned, supervised fine-tuned, GRPO fine-tuned, and supervised + GRPO fine-tuned versions of Llama 3.1 8B Instruct, Qwen3 8B, and GPT models. The "Change" column indicates the performance difference relative to the non-fine-tuned model in the few-shot prompt. All comparisons are within the same base model. The analysis shows that the chain-of-thought prompt substantially improves perfor-

Model	FT Type	Prompt	Inference time	BERTScore	ROUGE-L	LLMScore	Change
LLama 3.1 8B	None	Fewshot	5 min	0.9432	0.5172	0.7372	0
LLama 3.1 8B	None	CoT	30 min	0.9278	0.4794	0.7446	+0.74%
LLama 3.1 8B	SFT	CoT	5 min	0.9408	0.5204	0.7624	+2.52%
LLama 3.1 8B	GRPO	CoT	40 min	0.9464	0.5756	0.8236	+8.64%
LLama 3.1 8B	SFT+GRPO	CoT	40 min	0.9452	0.5494	0.8116	+7.44%
Qwen3 8B	None	Fewshot	5 min	0.9404	0.4973	0.7388	0
Qwen3 8B	None	CoT	7 h	0.9290	0.4901	0.8589	+12.01%
Qwen3 8B	SFT	CoT	5 min	0.9424	0.5389	0.7682	+2.94%
Qwen3 8B	GRPO	CoT	3 h	0.9335	0.5065	0.8686	+12.98%
Qwen3 8B	SFT+GRPO	CoT	3 h	0.9493	0.6029	0.8702	+13.14%
GPT-4o	None	CoT	5 min	0.9477	0.5618	0.8500	0
GPT-o4-mini	None	CoT	1 h	0.9473	0.5499	0.8748	0

Table 6.3: Performance of LLMs on aggregation questions in the OpenLifelogQA test set.

mance for Qwen3 8B, whereas it has a limited effect on the Llama 3.1 8B Instruct model. This difference can be rooted in Llama 3.1's limited inherent reasoning ability, which fails to generate meaningful intermediate rationales. However, after supervised and GRPO fine-tuning, Llama 3.1 shows enhanced reasoning capabilities, achieving a 7.44% increase in LLM Score. Qwen3 8B also benefits from fine-tuning, though to a lesser extent, due to its pre-existing reasoning capabilities acquired during pre-training. Notably, the inference time for Qwen3 8B is reduced by four hours relative to the base model, thanks to the reward function of generating fewer than 1,000 words. Notably, this reduction in reasoning length does not compromise performance but rather enhances computational efficiency. While the fine-tuned Qwen3 8B demonstrates LLM Scores comparable to those of the GPT-o4-mini model for aggregation questions, it achieves this performance with a substantially smaller size. These findings suggest that smaller-scale LLMs with task-specific knowledge can effectively address moderately complex tasks such as LifelogQA, offering a cost-efficient alternative to large, closed-source models like GPT-o4-mini.

6.4.4 DCU-TimelineQA Evaluation

Table 6.4 shows the performance of several fine-tuned and non-fine-tuned models in the test set of the DCU-TimelineQA dataset. Overall, GPT-o4-mini outperforms other models in all metrics. This indicates a strong capability of GPT-o4-mini mod-

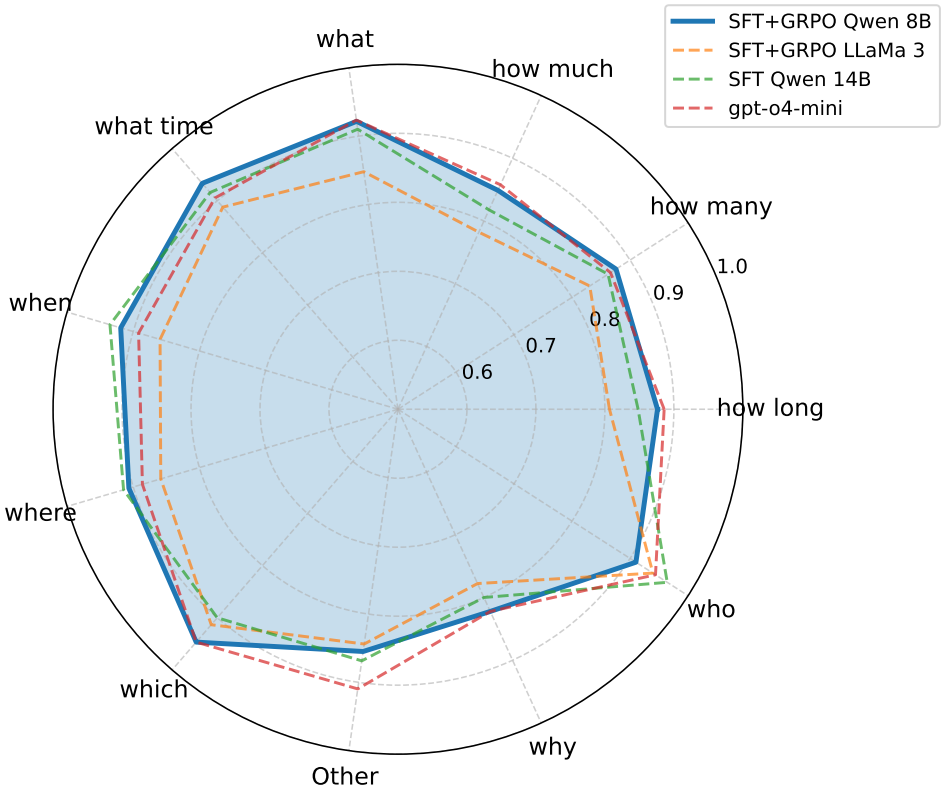


Figure 6.4: Comparison of LLM scores across four models on the OpenLifelog QA test set. The SFT+GRPO Qwen 8B (solid, shaded) is contrasted against SFT+GRPO LLaMA 3, SFT Qwen 14B, and gpt-o4-mini (dashed).

els in answering lifelog-related questions. Another insight from the result is the improved performance of open-source models after fine-tuning on lifelog data. Due to the extractive nature of answers in the DCU-TimelineQA dataset, F1-score is proposed to evaluate the performance on this dataset [146]. All models perform well, achieving a high F1-score. GPT-o4-mini still outperforms non-fine-tuned models, but after fine-tuning, the gap is reduced. From these evaluations, proprietary models outperform open-source models due to their larger size and broader knowledge base. Fine-tuning models on lifelog data can enhance the accuracy of answers to lifelog-related questions. Notably, the supervised and GRPO fine-tuned Qwen3 8B achieves a higher performance than the supervised fine-tuned Qwen3 14B, resulting in the second-highest performance. Although they cannot keep up with proprietary models, the gap is closing, and the resources needed are reduced.

Table 6.4: Model performance on the DCU-TimelineQA test set.

Model	Fine-tuning	BERT Score	ROUGE-L	F1-score
LLama 3.1 8B	None	0.9796	0.904	0.9133
Qwen3 8B	None	0.9809	0.8986	0.8989
Qwen3 14B	None	0.9793	0.8974	0.9069
GPT-o4-mini	None	**0.9905**	**0.9500**	**0.9512**
LLama 3.1 8B	SFT+GRPO	0.9799	0.9009	0.9113
Qwen3 8B	SFT+GRPO	0.9850	0.9210	0.9327
Qwen3 14B	SFT	0.9836	0.9153	0.9304

6.4.5 Ablation Study

In this ablation study, I examine the effect of data on the fine-tuning of Vision LLMs (LLaVA Next Interleave 7B) and LLMs (Llama 3.1 and Qwen3 8B) on answer performance. Table 6.5 shows the performance of LLaVA Next Interleave 7B fine-tuned on three data settings: only images + metadata (time and location), only textual context (event description and metadata), and both image and textual contexts formed in an interleaved way. Results show an interesting pattern: textual-only data achieves the best score, while metadata and image data show the worst performance. LLaVA Next Interleave 7B fine-tuned on text-only data achieves 86.59% LLM Score and 64.52% ROUGE-L, a little bit higher than the model fine-tuned on combined text and image by 4% LLM Score and 2.45% ROUGE-L. However, there is a bigger gap between image-only and text-only fine-tuned models. The difference is 7.31% LLM Score and 13.36% ROUGE-L. This gap suggests that textual contexts generated from images already provide sufficient information for LLMs to create answers. It still depends on the information asked in the question, but for the OpenLifelog dataset, the answers are found in the textual contexts. Combining text and image information in the prompt reduces performance, though not by a large margin, suggesting that images can interfere with the provided data. As an image occupies hundreds of tokens in LLaVA Next Interleave 7B [88], it is many times more than a word in the context, so it noisily perturbs the provided information and reduces performance.

For LLMs like Llama 3.1 and Qwen3 8B, I experimented on the data by question

Table 6.5: LLaVA Next Interleave 7B performance on the OpenLifelogQA test set when fine-tuning on different modalities of input contexts

Modality	BERT Score	ROUGE-L	LLMScore
Image + Metadata	0.9357	0.5116	0.7928
Context (Description + Metadata)	**0.9546**	**0.6452**	**0.8659**
Context + Image	0.9492	0.6207	0.8259

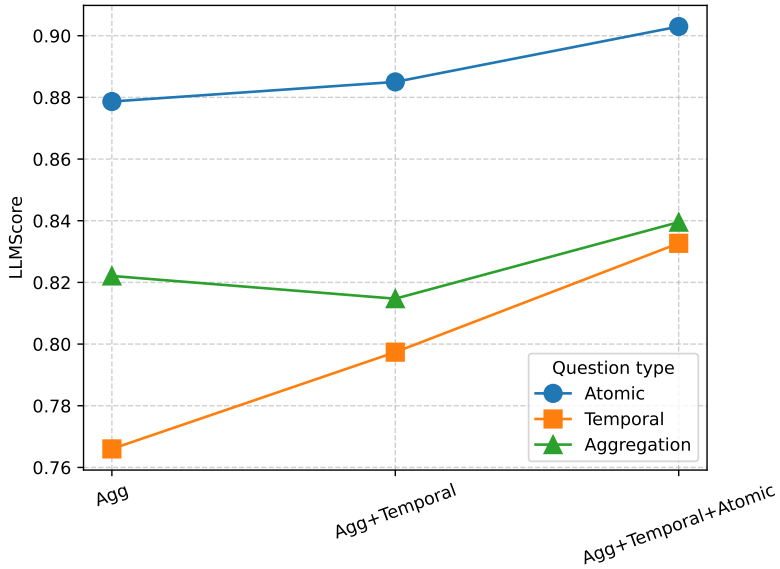


Figure 6.5: LLMScore for 3 types of questions from GRPO fine-tuned Llama 3.1 8B model on different training data settings.

types used for GRPO fine-tuning. Figure 6.5 shows the LLM Score by question types trained on aggregation questions only, aggregation + temporal, and all question types. There is an overall increase in the LLM Score as the amount of training data increases. This is expected: the more data used for fine-tuning, the more the model will learn to answer and provide more accurate responses.

6.4.6 Error Analysis

I manually analysed 170 low-LLMScore QA predictions (LLM Score smaller than 0.65) from the fine-tuned Qwen3 8B on the OpenLifelogQA test set and identified some notable examples, as shown in Tables 6.6 and 6.7. There are 5 main error types of LLMs in failing to generate the answer, with one example for each type and the justification of the cause. After analysis, there are two main causes why the model cannot generate the correct answers. Firstly, there is a failure to retrieve

Table 6.6: Taxonomy of error types in the predictions of the fine-tuned model on the OpenLifelogQA test set ($n = 170$ incorrect answers).

Error type	**Count (%)**	**Example**	**Cause**
Wrong activity at the queried timestamp	67 (39%)	*Q:* What was I doing at 8 AM on March 7, 2020? *Pred:* I was drinking coffee *GT:* I was driving away from home	The correct event is retrieved but not ranked first; the model treats the top-ranked context as the knowledge source and answers from it. A stronger reranker would address this.
Numeric reasoning error	46 (27%)	*Q:* How long was I at my workplace on June 12, 2020? *Pred:* I was there for 32 minutes *GT:* 7 hours and 39 minutes	The model fails to identify the end of the activity among several candidate events and therefore computes the duration over the wrong interval.
Wrong location name	39 (23%)	*Q:* Where was the last place I went on March 5, 2020? *Pred:* I went to the Spa *GT:* Home	The correct context is not retrieved at all, so the model answers from the most relevant of the available contexts.
Empty generation	10 (6%)	*Q:* What did I do at 4:47 PM on March 4, 2020? *Pred:* (empty) *GT:* I continued working on the computer	The relevant context is present in the retrieved data, but the model produces no output.
Hallucination	8 (5%)	*Q:* Who did I meet at the convenience store on March 12, 2020? *Pred:* I met a man *GT:* I did not meet anyone; I was shopping	The answer is not grounded in any provided context; the model introduces an entity that appears only in an unrelated retrieved event.
Total	**170 (100%)**		

sufficient data to derive answers, leading to wrong location names and failed numeric reasoning. As the Recall@20 is only around 97-98% after retrieval and reranking, there are several questions for which the retrieved contexts do not contain enough information. The top K can increase from 20 to 50, but it will increase the context length. There is a trade-off between accuracy and efficiency here, and the increase in Recall@50 is only around 2%, so I choose efficiency for the top 20 events. A dynamic context-length selection could be explored in future work, based on question type, to facilitate length selection.

Another reason for generating incorrect answers is that the model fails to identify

Table 6.7: Retrieved contexts for the examples in Table 6.6. The context containing the ground-truth answer is shown in bold, and the wrong contexts are in red.

Error type	Retrieved contexts
Wrong activity at the queried timestamp	08:16–08:42, 07 Mar 2020: I am drinking coffee and talking with a man at a coffee shop. 08:48–09:07: I am looking for things to buy at a Retail Park. 08:09–08:16: I am at a coffee shop. **07:44–08:09: I drove away from home in the car.** 09:38–09:42: I am returning home from aShopping Centre.
Numeric reasoning error	**15:30–15:39, 12 Jun 2020: I move out to the parking lot, in the car.** 08:10–08:11: I stop working and take off my glasses at my workplace. 08:32–08:35: I leave the office at my workplace. **08:00–08:05: I start working on the computer at my workplace.** 15:21–15:29: I leave the office at my workplace.
Wrong location name	20:19–21:01: I have arrived and am sitting working on my laptop at a Spa. (No context covering the return home was retrieved.)
Empty generation	The event covering 16:47 on 04 Mar 2020 was present among the retrieved contexts.
Hallucination	07:28–07:29, 12 Mar 2020: I arrive at a convenience store, in the car. **07:29–07:31: I am shopping at the convenience store, in the car.** 07:31–07:32: I put things in the back of the car. 10:48–10:50: I stop working to talk to a man at my workplace.

correct events and instead generates answers based on incorrect ones. The incorrect events are highlighted in red in Table 6.7. The model selects the top 1 event in the context to answer two atomic questions (where and what). For the other two aggregation questions, the model fails to consider broader contexts to select the correct events and generates incorrect rationales based on incorrect events, leading to incorrect answers. These 4 examples account for most of the incorrect predictions. This error can be addressed by improving the LLM's intelligence, reasoning, and familiarity with this lifelog data. Other subtle reasons, such as empty generation or hallucination, also contribute to the LLM's failure but in a smaller number of QAs.

6.4.7 Human Evaluation Comparison to LLM-as-a-judge

Because the LLM Score defined in Equation 3.9 is used as the primary accuracy metric throughout this thesis, it is necessary to establish that the automatic judge behaves comparably to a human rater applying the same criteria. I therefore conduct an evaluation study on a sample of 100 QA examples from the predictions of the

Model	Comparison	Pearson r	95% CI	p_{adj}	MAD
GPT-4o	Judge vs. V1	0.8060	[0.7242, 0.8654]	$< .001$	0.63
	Judge vs. V2	0.8920	[0.8433, 0.9262]	$< .001$	0.37
	Judge vs. mean(V1, V2)	0.8700	[0.8124, 0.9108]	$< .001$	0.49
	V1 vs. V2	0.9010	[0.8561, 0.9324]	$< .001$	0.35
Qwen 8B	Judge vs. V1	0.8590	[0.7971, 0.9031]	$< .001$	0.57
	Judge vs. V2	0.9050	[0.8618, 0.9352]	$< .001$	0.41
	Judge vs. mean(V1, V2)	0.9020	[0.8575, 0.9331]	$< .001$	0.47
	V1 vs. V2	0.9090	[0.8675, 0.9380]	$< .001$	0.28

Table 6.8: Agreement between the GPT-4o LLM-as-a-judge metric and two human raters (V1, V2) on $n = 100$ QA pairs per prediction model. p_{adj} values are two-tailed and Bonferroni-corrected for the eight correlations reported here ($\alpha = .05/8 = .00625$)

GPT-4o and Qwen 8B SFT+GRPO fine-tuned models. The volunteers receive the same guide as the prompt for LLM-as-a-judge in A.2 and work independently and are not shown one another's ratings, nor the score assigned by the LLM judge, so that agreement is not inflated by anchoring. The Pearson's correlation of GPT-4o score and the 2 experts' scores is in Table 6.8. The Average score column is calculated as the correlation between LLMScore and the average score from two volunteers. Mean absolute distance (MAD) measures how far apart the scores are in absolute terms on the raw 1–5 scale.

Table 6.8 shows that the LLM judge correlates strongly with both human raters, with r ranging from 0.806 to 0.905 across the two systems. The correlation is consistently higher for the Qwen 8B predictions than for the GPT-4o predictions, which is expected because GPT-4o can favour its own score for its prediction. The MAD figures follow the same pattern and quantify the practical size of the disagreement. The LLM judge differs from the mean human rating by 0.47–0.49 points on the 1–5 scale, against a human-human floor of 0.28–0.35 points. All correlations are tested against the null hypothesis $\rho = 0$ using a two-tailed t test with $n - 2 = 98$ degrees of freedom. Because eight correlations are reported from the same evaluation sample, I apply a Bonferroni correction to control the family-wise error rate, giving a per-test threshold of $\alpha = 0.05/8 = 0.00625$. Every correlation remains significant under this threshold by a wide margin. I also report 95% confidence intervals obtained via

the Fisher z transformation, since with a sample of this size the interval is more informative than the significance test: rejecting $\rho = 0$ is a weak claim, whereas the intervals show that the true agreement is unlikely to fall below $r \approx 0.72$ for the weakest pairing and $r \approx 0.80$ for the rest. Overall, this analysis shows high correlation of LLM-as-a-judge to human evaluation, but the bias in judging is still exhibited in GPT-4o predictions. Further limitations will be discussed in the limitations section 7.3.

6.5 Chapter Summary

This chapter investigates the RQ3: *Can LLMs with reasoning capability improve the accuracy of answers for aggregation questions in lifelog QA compared to standard LLM baselines in RQ2?* and, through extensive experiments on reasoning LLMs, the answer can be drawn that reasoning LLMs can improve the accuracy of aggregation questions. Compared to the results of Llama 3.1 8B and Mistral 7B in the experiment for RQ2, results from reasoning-enhanced Llama 3.1 8B and other reasoning models in the experiment for RQ3 show a notable improvement, with over a 9% increase in LLM Score. While general LLMs struggle with aggregation queries despite strong performance on atomic questions, the reasoning-enhanced models introduced in this chapter show consistent improvements. In particular, models that explicitly generate intermediate reasoning steps, especially after reinforcement learning fine-tuning, achieve substantially higher LLM Scores and improved semantic correctness on aggregation questions.

Another important contribution of this chapter is the demonstration that task-specific reasoning can be learned efficiently in relatively small models. While proprietary models such as GPT-o4-mini remain strong performers, the gap between closed-source and open-source models narrows after supervised and reinforcement learning fine-tuning on lifelog data. The SFT + GRPO fine-tuned Qwen3 8B achieves performance comparable to GPT-o4-mini, despite having far fewer parameters and operating on commodity hardware.

However, the experiment still has some limitations. Firstly, due to computing resource constraints, I only fine-tuned 8B-parameter models, especially for GRPO fine-tuning. While these models achieve competitive results, larger open-source models may further enhance aggregation and temporal reasoning, but their training cost exceeds our current capacity. Secondly, the evaluation primarily relies on two benchmark datasets, including OpenLifelogQA and DCU-TimelineQA [146]. Although these datasets cover diverse lifelog scenarios, they may not fully capture the variability of real-world lifelog data, particularly with multimodal inputs that combine long image streams or health sensor data. Extending evaluation to richer, more naturalistic datasets is the next step; however, there are currently no such datasets, so constructing one may require enormous resources. Thirdly, while reinforcement learning fine-tuning improves reasoning on aggregation and temporal questions, I observe that performance still degrades when relevant contexts are not retrieved or when noisy contexts dominate the prompt. This highlights the need for more adaptive retrieval strategies that can dynamically adjust context length and filtering. Ultimately, the framework prioritises reasoning capability over efficiency. Although I have designed the pipeline to be lightweight enough for personal use, the inference cost of fine-tuned LLMs may still be prohibitive for long queries or large-scale lifelog collections.

In conclusion, this chapter provides strong empirical evidence that LLMs with explicit reasoning capability improve accuracy on aggregation questions in lifelog QA compared to non-reasoning LLMs, especially after reinforcement learning fine-tuning. This answers RQ3 and extends the findings of RQ2 by showing that reasoning LLMs are necessary for robust lifelog QA. The results of this chapter pave the way for practical and trustworthy lifelog QA systems.

Chapter 7

Conclusion and Future Work

In this chapter, I summarise the work presented in this thesis with three main research questions to validate the research hypothesis. In addition, I describe several research contributions made in this study, as well as several limitations that remain. Finally, several future directions for this research to further explore this topic are discussed in the last section.

7.1 Summary

Lifelog QA is a compelling yet challenging task in the field of lifelogging. Through the literature review, the research gap for open-ended lifelog QA is identified. Although open-ended QA is hard to evaluate, it brings the task closer to real-world use cases and aligns with the trend of LLM open-ended generation. To address this gap, I built an open-ended lifelog QA dataset from 18 months of existing lifelog data. In addition, the method for solving open-ended lifelog QA has not been fully explored, so I have proposed an RAG-based approach to the task. It includes both a retrieval engine to find relevant lifelog contexts for questions and a generation part by LLMs to generate correct answers.

The work aims to validate the research hypothesis: *Reasoning LLMs in a RAG framework improve the accuracy of the open-ended lifelog question answering task compared to standard LLMs.* To validate this research hypothesis, I propose three

research questions, ranging from the retrieval system to RAG and reasoning LLMs for lifelog QA. The first research question discussed in Chapter 4 focuses on investigating approaches for context retrieval for lifelog QA. This task aims to find relevant contexts (events in a lifelog with contextual data such as time and location). To find the answer to this question, I first start by building an interactive lifelog retrieval system called MemoriEase. This system is evaluated using several interactive and automated benchmarks, such as LSC and the NTCIR Lifelog Task. The system performance results show that the embedding-based retrieval approach yields the highest results, but additional processing, including query filtering and post-processing, is needed to improve performance further. Based on the method for building a retrieval system, I adapt it for automatic context retrieval and further improve it by introducing a reranking method. The reranking model is a cross-encoder, multimodal model with a language model on one branch and a visual encoder on another. The model aims to rerank the retrieved contexts to further improve the ranking of relevant contexts for questions in lifelog QA. The best combination is ModernBERT as a language model to encode event descriptions, and Vision Transformer as a visual encoder to encode lifelog images. This combination achieves over 98% Recall@20 and 85% Precision@1. The experiment's results answer the first research question by showing that a two-stage retrieval and reranking combination outperforms other tested methods from sparse retrieval to dense retrieval. This reranking model paves a strong foundation for RAG by providing accurate contexts to derive answers.

The second research question in Chapter 5 inherits the retrieval engine from the first research question to construct a RAG pipeline for lifelog QA. This question investigates the practical approach to incorporating retrieved contexts and prompting LLMs to answer lifelog questions. Through experimentation and analysis, LLMs combined with retrieved contexts in a RAG setting achieve strong results, proving a viable and effective approach. LLMs outperform small language models and perform well on atomic and temporal questions, but their accuracy on aggregation questions

still lags behind. In addition, analysis of different retrieved context settings and prompts shows the effect of these factors on the RAG. Finally, the answer to the second research question is that RAG can effectively generate answers for open-ended lifelog QA, but further enhancements are needed to address the complex reasoning demands inherent in lifelog QA.

The third research question discussed in Chapter 6 aims to mitigate the gap in reasoning for aggregation questions in lifelog QA. To enhance LLMs' reasoning capabilities for lifelog QA, I propose a hybrid, comprehensive approach that combines dynamic prompting and different fine-tuning methods. The experiment shows a substantial improvement in accuracy on aggregation questions through reinforcement learning fine-tuning of two LLMs. These 7-8B LLMs can achieve comparable performance on aggregation questions to proprietary LLMs such as GPT-4o after fine-tuning. Several analyses confirm the capability of reasoning LLMs across various types of questions and question words, as well as their transferability to other lifelog QA datasets. From the results, the answer to the third research question is that reasoning capability in LLMs can improve the accuracy of answers to aggregation questions in lifelog QA. Furthermore, these reasoning models in an RAG setting provide a way to integrate LLMs into a lifelog search and QA system without leaking personal data to proprietary LLMs.

In conclusion, this thesis provides an investigation into open-ended lifelog QA with a RAG approach. By constructing a large-scale open-ended lifelog QA dataset, developing an effective two-stage retrieval and reranking framework, and demonstrating the effectiveness of RAG in lifelog QA with reasoning-enhanced LLMs, this work validates the research hypothesis that reasoning LLMs in an RAG framework improve open-ended lifelog QA accuracy compared to standard LLMs without reasoning capability. In addition, the results demonstrate that compact, open-source LLMs can achieve competitive performance through effective fine-tuning to allow a privacy-preserving and deployable LLM for a lifelog QA system. Collectively, this work advances lifelog QA from a retrieval-centric task toward a richer question-

answering paradigm that requires both contextual understanding and multi-event reasoning. The resources and methods introduced here, including an open-ended dataset, a competitive retrieval baseline, and an effective fine-tuning strategy, provide the community with the tools to build systems that bring us closer to the longstanding vision of a searchable, queryable personal memory.

7.2 Contributions

This section summarises the main contributions of this thesis to the field of lifelog question answering. This work has focused on multimedia retrieval and question answering, with particular emphasis on lifelog data and open-ended QA. The key contributions are as follows:

- I developed *OpenLifelogQA*, a large-scale open-ended lifelog QA dataset comprising 14,187 question–answer pairs derived from an existing 18-month LSC lifelog collection [1]. This dataset enables research on generative lifelog QA and places particular emphasis on aggregation questions, which represent a major unresolved challenge in existing lifelog QA benchmarks.

- I proposed an RAG approach for lifelog QA that integrates semantic retrieval with LLMs to generate context-aware answers. Experimental results show substantial improvements over keyword-based retrieval and small language model baselines. Further analysis reveals the limitations of standard LLMs when handling aggregation questions, motivating the need for enhanced reasoning capabilities.

- To strengthen the retrieval component of the RAG pipeline, I designed a multimodal reranking model that reranks retrieved lifelog contexts by jointly leveraging textual descriptions and visual information from lifelog images. This reranking approach improves retrieval accuracy over baseline methods and provides relevant contexts as input for LLMs in the RAG approach.

[1] In collaboration with researchers from the University of Science, Vietnam National University

- I developed and fine-tuned a reasoning-enhanced lifelog LLM using a combination of supervised fine-tuning and reinforcement learning. The resulting model achieves substantial performance gains over prior baselines and attains accuracy on aggregation questions comparable to state-of-the-art closed-source models from OpenAI. Detailed analyses are provided to demonstrate the model's reasoning capabilities across different question types and its robustness in lifelog QA scenarios.

- During this PhD, I developed MemoriEase, an interactive lifelog retrieval system that evolved from a basic embedding-based search engine into a conversational, RAG-enhanced lifelog assistant. By integrating the reasoning lifelog LLM, MemoriEase supports lifelog search, question answering, and summarisation. The system was evaluated in the Lifelog Search Challenge and achieved a top-3 ranking in LSC 2025, demonstrating its effectiveness in real-world interactive lifelog retrieval settings.

7.3 Limitations

There are still some limitations in this research, and this section lists several limitations as follows:

- The data generation process of OpenLifelogQA also has limitations. Event descriptions and part of the QA pairs were written by 10 undergraduate volunteers on a research internship, who were neither professional annotators nor native English speakers. Quality control relied on a final cross-check pass and one overlapping day per month, with agreement measured by BERTScore (≥ 0.85) — a similarity measure rather than true inter-annotator agreement, but the most practical option given the small annotator pool and the volume of data. Since GPT-4o generated 20 QA pairs per day against 10 per volunteer, most of the benchmark is model-authored and inherits GPT-4o's question framing and phrasing. Future lifelog QA datasets would benefit from

a larger, more diverse annotator pool, an owner-in-the-loop verification stage, and a clearly labelled split between human- and model-authored items.

- The use of LLM-as-a-judge to rate other LLMs' output may introduce some bias and uncertainty. A single judge model, GPT-4o, was used with one fixed prompt, and each answer was judged once, so no estimate of the judge's variance across repeated runs or alternative prompts is available. LLM judges are known to favour fluent, well-structured responses [48] and to be inconsistent on partially correct answers, which are common in aggregation questions where a count or duration is close to but not exactly the ground truth. GPT-family models also generated parts of the OpenLifelogQA data, so the judge is not fully independent of what it scores, and because ratings are integers on a 1–5 scale that are then normalised, the effective range is bounded below by 0.2 rather than 0, making the values less comparative to other works.

- The Lifelog Search Challenge (LSC) results provide real-world evidence for the MemoriEase retrieval approach, but they are a limited form of evaluation. LSC is one of the small number of benchmarks for evaluating retrieval systems interactively. Each topic is time-boxed to 300 seconds with hints released every 30 seconds, measuring convergence under pressure rather than best achievable accuracy, and each session is run once, so no variance estimate is available. Finally, the topics are predefined by the organisers and number only around ten per task, so rankings are indicative rather than statistically meaningful, and since the queries and participant pool changed between LSC'23, LSC'24 and LSC'25, the three participations are not controlled comparisons. The substantive claims of this thesis therefore rest on the offline, repeatable evaluations, with LSC used as complementary evidence.

- Due to the limit on resources, only around 7-8B models are fine-tuned, and only 2 proprietary LLMs are evaluated in the experiment for the third research question. This may not fully represent the performance of other LLMs in this

task. But the use of 7-8B models is already yielding substantial improvements after fine-tuning, making them comparable to proprietary LLMs. These models may also limit accessibility for researchers or practitioners with constrained resources, especially when scaling to larger models or longer lifelog histories.

- As the focus of this research is on RAG for lifelog QA, privacy considerations are not fully considered. Although the proposed RAG system emphasises using open-source models to avoid sharing personal data with proprietary services, there is still sensitive lifelog data to process before deployment. The Open-LifelogQA dataset is built on a privacy-protected lifelog dataset, with face and other identifiable information blurred in the images, but some other information remains, such as location. This thesis does not fully explore long-term data governance or user control strategies required for real-world deployment.

7.4 Future Works

Before concluding the thesis, I present some future directions for this research, ranging from improving retrieval and QA models for lifelog QA to building a lifelog assistant system using LLMs.

As retrieval plays an essential role in a RAG pipeline for lifelog QA, its performance directly affects the QA results. Improving the retrieval is a potential direction for future work. Although a two-stage framework with retrieval and reranking for context retrieval in lifelog QA offers some improvement, there is still room for improvement in this direction. Firstly, incorporating temporal and sequential modelling to capture event dependencies enables the retrieval model to provide more coherent, temporally aware retrieved contexts. In addition, a dynamic retrieval approach, in which each question type will have a different context set length. This approach focuses on the problem of varying information needed for different types of questions. While atomic questions require only one event, aggregation questions may require several, up to hundreds. This approach helps save input tokens for

LLMs, but the retrieval engine's accuracy must be high to ensure a correct list of events. Finally, improving multimodal alignment between textual descriptions and visual images may further refine retrieval quality and robustness in lifelog QA.

With the rapid development of LLMs, the ability of text generation and reasoning of these models has improved. This development helps to address tasks like open-ended QA effectively. However, there is considerable scope for future work in LLMs for lifelog QA. Future research can explore using LLMs to refine the list of retrieved contexts to improve retrieval accuracy. In addition, LLMs can actively guide context selection through iterative retrieval by refining queries rather than passively consuming retrieved contexts. This future work on LLMs for enhancing retrieval can improve retrieval accuracy in lifelog QA. Another important direction for improving LLMs in lifelog QA is to increase their long-context reasoning capabilities to handle questions spanning extended lifelog data, potentially through hierarchical reasoning or memory-based representations. Furthermore, improving the model's ability to handle uncertainty and avoid hallucination when answering questions with no answer opens a direction for future work. This helps LLMs to say no to unanswerable questions instead of making up a wrong answer. Privacy for lifelog data in the age of LLMs remains a pressing issue to investigate carefully. Future work on LLMs for lifelog QA should consider these aspects to improve and build a robust QA model.

Finally, an important future direction for this research is to improve the current retrieval and QA models into a fully agentic lifelog assistant system that can address a wide range of tasks in lifelogging, such as retrieval, question answering, summarisation, and activity recommendation. Building upon the reasoning-enhanced LLMs from this research, agentic AI can be employed to orchestrate complex workflows from process queries to extract key information and filter terms, invoke retrieval or analysis tools, analyse the results, and iterate the search if necessary before delivering the final answer to users. Future research could explore a multi-agent architecture [159] in which specialised agents handle specific subtasks, such as temporal filtering, visual understanding, aggregation reasoning, summarisation, and recommendation.

These agents could collaborate through a shared memory to handle complex queries that span long time periods or involve multiple constraints. With the combination of agentic AI, reasoning LLMs, and multimodal lifelog understanding, future lifelog assistant systems have the potential to move beyond retrieval and QA tasks toward intelligent, trustworthy companions that meaningfully support everyday life.

Appendix A

Additional Experimental Materials

A.1 Prompt for reasoning-enhanced LLMs in RAG

Role and Objective

You are a life assistant AI designed to answer life-related questions based on contextual event data from a person's lifelog.

Instructions

You are given a question and a context that describes the time, location, and activity of a person. Your task is to answer the question based solely on the context by deriving information carefully and logically.

There are several types of questions:

- Time-based questions (e.g., when did the person eat, arrive, or leave)
- Location-based questions (e.g., where an event occurred)
- Activity-based questions (e.g., what the person was doing)
- Aggregation questions (e.g., how many times an event occurred or how long an event lasted)
- Temporal questions (e.g., what happened before or after a specific event)

Reasoning Steps

1. Identify the relevant timestamp and event in the context.
2. Understand what action occurred at that time.
3. Convert the time to 12-hour format if needed.

4. Synthesize a short answer beginning with "I ..." that directly answers the question.

5. Limit the answer to under 10 words.

6. Do not provide any explanation—just the answer. # Output Format

Provide the answer as a short sentence starting with "I . . . ". Answer must be under 10 words. No extra explanation is allowed.

Examples

Specific examples for each type of question words, here are some CoT examples for how long and when questions:

CoT Instruction: You find the start time and end time of an event in the context, then calculate the difference to find out how long the event lasted.

CoT Example: Context: At 2019-07-10 14:55:12, I am at the church at the College. At 2019-07-10 15:22:55, I have left the church at Car.

Question: How long did I stay at the church on July 10, 2019?

Thoughts: I arrived at 14:55. I left at 15:22. The difference is (60-55)+22=27. That makes 27 minutes.

Answer: I stayed for 27 minutes.

CoT Instruction: You find the start time of an event in the context, then convert it to a 12-hour format to answer the question." CoT Example = Context: At 2019-02-20 13:18:58, I am getting food at Dublin City University (DCU). At 2019-02-20 13:29:38, I finished eating and left at Dublin City University (DCU).

Question: When did I have lunch on 20th February 2019?

Thoughts: I started getting food at 13:18 and left at 13:29, so lunch happened between those times.

Answer: I had lunch at 01:18 PM.

Question-Answer

Please act as a life assistant to answer questions based on the given context. Use the examples above as a guide. Think step by step and derive your answer directly from the context, following the format: "I . . . ". Keep your answers concise (under

10 words), without explanation.

A.2 Prompt for LLM-as-a-judge

You are an intelligent chatbot designed for evaluating the correctness of generative outputs for question-answer pairs. You are tasked with evaluating how well the predicted answers match the ground truth answers. Focus on the meaning and intent of the answer rather than the exact wording. A match should be considered acceptable if: Semantic Equivalence: The predicted answer conveys the same overall meaning as the ground truth answer, even if the wording or phrasing is different.

Contextual Match: Minor differences in description or detail (e.g., specific object characteristics like size, position, or color) should be considered a match if they still refer to the same object or event in context. Synonyms and Paraphrases: If the predicted answer uses a synonym, paraphrase, or a more general/specific term than the ground truth, it can be considered a match as long as it describes the same concept or entity.

Partial Matches: If the predicted answer captures a significant part of the ground truth but omits or changes some details, it may still be considered a match if the core meaning is preserved in the context of the question. Examples: "one" and "one large spoon": Consider these a match if “one” captures the essence of quantity. "30 minutes" and "29 minutes" or "31 minutes": Although slightly different, if they refer to a similar time duration, consider them a match. “blue” and “blue and black”: If “blue” is a dominant color, consider them a match. Provide your evaluation; only a score is an integer value between 1 and 5, with 5 indicating the highest meaningful match.

DO NOT PROVIDE ANY OTHER OUTPUT TEXT OR EXPLANATION. Only provide a syntactically valid Python dictionary string. For example, your response should look like this: score: 4

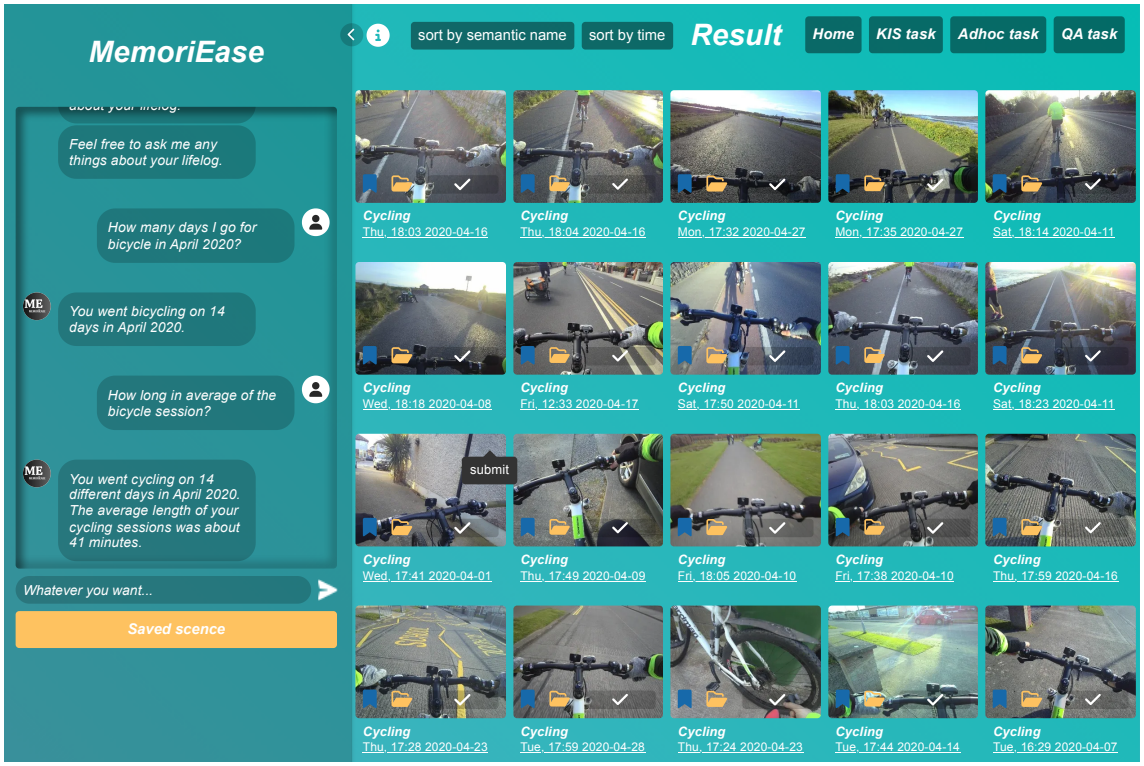


Figure A.1: Conversational user interface of the MemoriEase system.

A.3 User Interface of MemoriEase System

Several figures below show the user interface of the MemoriEase system. Figure A.1 presents the conversational user interface, which allows users to interact with the system through natural language queries for lifelog retrieval and question answering. Figure A.2 shows the temporal search interface, which helps users to browse and filter lifelog data based on time-related information. Finally, Figure A.3 shows the image-to-image search interface, where users can retrieve visually similar moments by providing images as queries.

A.4 Model Cards and Parameter Settings

This appendix consolidates the models and training hyperparameters used across Chapters 4–6. The Source column in Table A.2 records the provenance of each value using the following tags: **PW**: taken from published prior work, cited in the relevant chapter; **VD**: vendor or framework default; **VAL**: selected empirically on

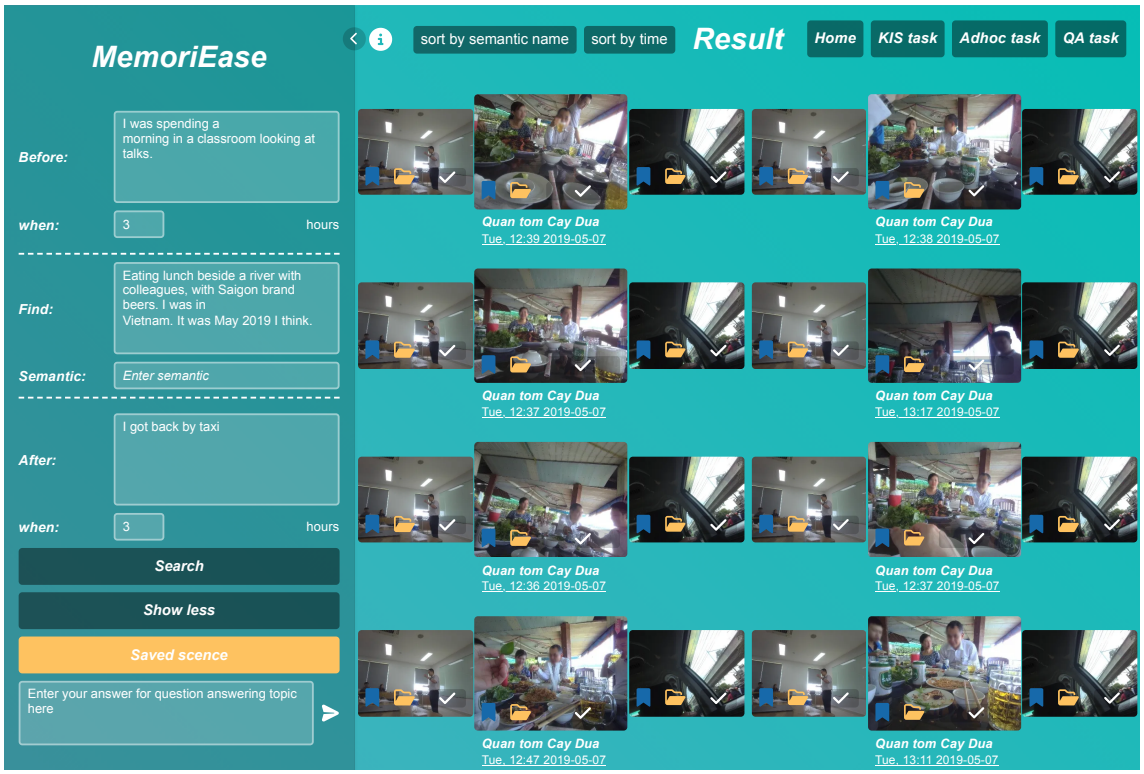


Figure A.2: Temporal search interface.

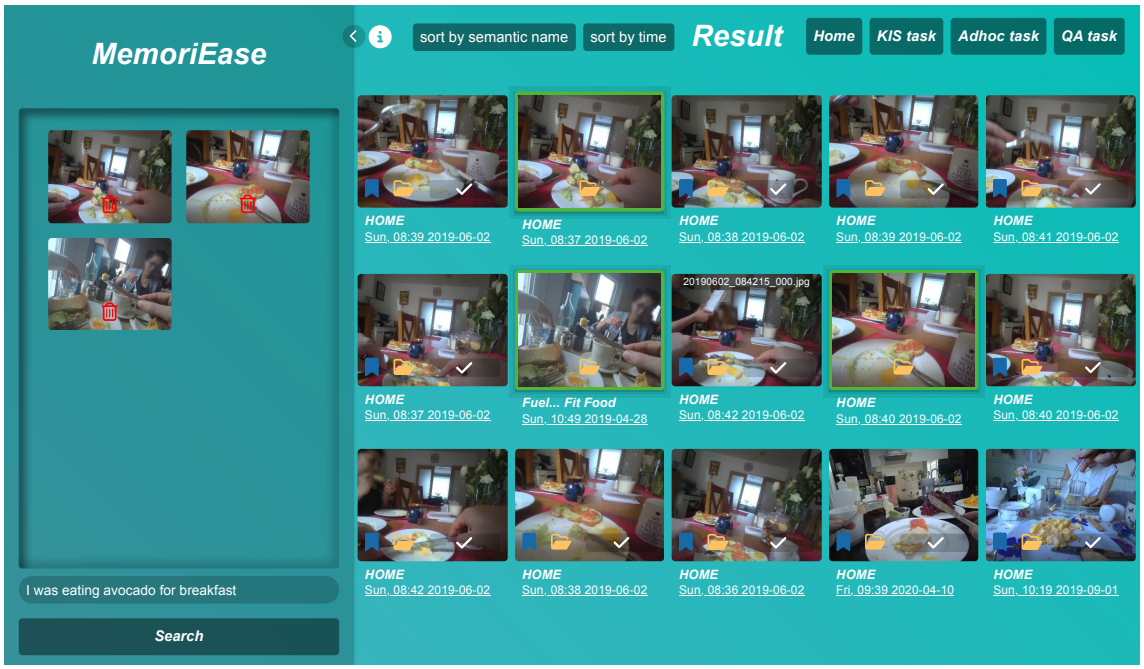


Figure A.3: Image-to-Image search interface.

Table A.1: Model cards for all models used in this thesis.

Type	Model	Params	Provider
Non-reasoning LLM	GPT-4o	undisclosed	OpenAI
	GPT-4o-mini	undisclosed	
	Llama 3.1 8B Instruct	8B	Meta
	Mistral 7B Instruct v0.3	7B	Mistral AI
Reasoning LLM	GPT-o4-mini	undisclosed	OpenAI
	GPT-oss-20B	20B (3.6B active, MoE)	
	Qwen3 14B	14B	Alibaba
	Qwen3 8B	8B	
Vision–language model	LLaVA-NeXT-Interleave 7B	7B	ByteDance
Seq2seq baseline	BART	406M	Meta
	T5	220M	Google
Text embedding	Stella-en-1.5B-v5	1.5B	NovaSearch
Image–text embedding	BLIP2	1.17B	Salesforce
	CLIP	986M	OpenAI
Multimodal cross-encoder	ModernBERT + ViT	235M	Answer.AI, Google
Cross-encoder reranker	MiniLM, Electra-base	14M–22M	Microsoft, Google

the validation split of OpenLifelogQA; **HW**: fixed by the memory of the available GPU; **EMP**: arrived at by inspection of outputs during development, without a systematic search. The rationale behind these choices is given in Section 3.5.

Reproducibility note. All experiments in Chapter 5 and Chapter 6 were conducted on a single NVIDIA A100 GPU with 80 GB of memory; the reranking model in Chapter 4 was trained on a single NVIDIA RTX 4090. Source code for the RAG pipeline and for the reasoning experiments is available in the repositories cited in Sections 5.3 and 6.3. Random seeds were not fixed, and each configuration was run once; the consequences of this are discussed in Section 3.5.

Table A.2: Training hyperparameters, with the provenance of each value.

Component	**Setting**	**Value**	**Source**
Multimodal reranker (Ch. 4)	Optimiser	AdamW	PW
	Learning rate	1×10^{-5}	VAL
	Weight decay	1×10^{-4}	VAL
	Batch size	32	HW
	Max sequence length	128	VAL
	Loss	Binary cross-entropy	PW
	Negative sampling	BM25 top-10, ground truth excluded	PW
	Epochs / early stopping	20 / patience 2	VAL
Supervised fine-tuning: Llama 3.1 8B, Qwen3 8B	Adaptation	Full-weight	VAL
	Optimiser	AdamW	VD
	Learning rate	1×10^{-5}	VAL
	Weight decay	0.01	VD
	Batch size / grad. accum.	4 / 8	HW
Supervised fine-tuning: Qwen3 14B	Adaptation	8-bit, LoRA rank 128, alpha 256	HW
	Learning rate	3×10^{-5}	VAL
	Weight decay	0.05	VAL
	Batch size / grad. accum.	4 / 16	HW
	Max sequence length	8,096	HW
	Epochs / early stopping	5 / 60 eval steps	VAL
GRPO fine-tuning: Llama 3.1 8B, Qwen3 8B	Adaptation	LoRA rank 8, alpha 16	VD
	Optimiser	AdamW	VD
	Learning rate	5×10^{-6}	VD
	Generations per sample	4	VD
	Batch size / grad. accum.	4 / 16	HW
	Epochs	2	VAL

PW = prior work; VD = library/vendor default; VAL = tuned on validation set; HW = hardware constraint.

Bibliography

[1] Mofetoluwa Adeyemi et al. "Zero-Shot Cross-Lingual Reranking with Large Language Models for Low-Resource Languages". In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Ed. by Lun-Wei Ku, Andre Martins, and Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 650–656. DOI: 10.18653/v1/2024.acl-short.59. URL: https://aclanthology.org/2024.acl-short.59/.

[2] Abhimanyu Dubey et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[3] Naushad Alam, Yvette Graham, and Cathal Gurrin. "Memento 4.0: A Prototype Conversational Search System for LSC'24". In: *Proceedings of the 7th Annual ACM Workshop on the Lifelog Search Challenge*. LSC '24. Phuket, Thailand: Association for Computing Machinery, 2024, pp. 82–87. ISBN: 9798400705502. DOI: 10.1145/3643489.3661126. URL: https://doi.org/10.1145/3643489.3661126.

[4] Ahmed Alateeq, Mark Roantree, and Cathal Gurrin. "Voxento-Pro: An Advanced Voice Lifelog Retrieval Interaction for Multimodal Lifelogs". In: *Proceedings of the 7th Annual ACM Workshop on the Lifelog Search Challenge*. LSC '24. Phuket, Thailand: Association for Computing Machinery, 2024, pp. 105–110. ISBN: 9798400705502. DOI: 10.1145/3643489.3661130. URL: https://doi.org/10.1145/3643489.3661130.

[5] Ali Mohamed Nabil Allam and Mohamed Hassan Haggag. "The question answering systems: A survey". In: *International Journal of Research and Reviews in Information Sciences (IJRRIS)* 2.3 (2012).

[6] Adrienne H Andrew, Kevin Eustice, and Andy Hickl. "Using location lifelogs to make meaning of food and physical activity behaviors". In: *2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*. IEEE. 2013, pp. 408–411.

[7] Stanislaw Antol et al. "Vqa: Visual question answering". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2425–2433.

[8] Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. "Transformers for Tabular Data Representation: A Survey of Models and Applications". In: *Transactions of the Association for Computational Linguistics* 11 (2023), pp. 227–249. DOI: 10.1162/tacl_a_00544. URL: https://aclanthology.org/2023.tacl-1.14/.

[9] Payal Bajaj et al. *MS MARCO: A Human Generated MAchine Reading COmprehension Dataset*. 2018. arXiv: 1611.09268 [cs.CL]. URL: https://arxiv.org/abs/1611.09268.

[10] Jack Bandy and Nicholas Vincent. *Addressing "Documentation Debt" in Machine Learning Research: A Retrospective Datasheet for BookCorpus*. 2021. arXiv: 2105.05241 [cs.CL]. URL: https://arxiv.org/abs/2105.05241.

[11] Hangbo Bao et al. *UniLMv2: Pseudo-Masked Language Models for Unified Language Model Pre-Training*. 2020. arXiv: 2002.12804 [cs.CL]. URL: https://arxiv.org/abs/2002.12804.

[12] Michael Blighe and Noel E. O'Connor. "MyPlaces: detecting important settings in a visual diary". In: *Proceedings of the 2008 International Conference on Content-Based Image and Video Retrieval*. CIVR '08. Niagara Falls, Canada: Association for Computing Machinery, 2008, pp. 195–204. ISBN:

9781605580708. DOI: 10.1145/1386352.1386382. URL: https://doi.org/10.1145/1386352.1386382.

[13] Marc Bolaños, Mariella Dimiccoli, and Petia Radeva. "Toward Storytelling From Visual Lifelogging: An Overview". In: *IEEE Transactions on Human-Machine Systems* 47 (2015), pp. 77–90. URL: https://api.semanticscholar.org/CorpusID:499315.

[14] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[15] Andrea LS Bulungu et al. "Validation of a life-logging wearable camera method and the 24-h diet recall method for assessing maternal and child dietary diversity". In: *British Journal of Nutrition* 125.11 (2021), pp. 1299–1309.

[16] Vannevar Bush. "As we may think". In: *SIGPC Note.* 1.4 (Apr. 1979), pp. 36–44. ISSN: 0163-5816. DOI: 10.1145/1113634.1113638. URL: https://doi.org/10.1145/1113634.1113638.

[17] Daragh Byrne et al. "Everyday concept detection in visual lifelogs: validation, relationships and trends". In: *Multimedia Tools Appl.* 49.1 (Aug. 2010), pp. 119–144. ISSN: 1380-7501. DOI: 10.1007/s11042-009-0403-8. URL: https://doi.org/10.1007/s11042-009-0403-8.

[18] Marco Antonio Calijorne Soares and Fernando Silva Parreiras. "A literature review on question answering techniques, paradigms and systems". In: *Journal of King Saud University - Computer and Information Sciences* 32.6 (2020), pp. 635–646. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2018.08.005. URL: https://www.sciencedirect.com/science/article/pii/S131915781830082X.

[19] Danqi Chen et al. *Reading Wikipedia to Answer Open-Domain Questions*. 2017. arXiv: 1704.00051 [cs.CL]. URL: https://arxiv.org/abs/1704.00051.

[20] Jiahan Chen, Da Li, and Keping Bi. "LifeIR at the NTCIR-18 Lifelog-6 Task". In: *ArXiv* abs/2505.20987 (2025). URL: https://api.semanticscholar.org/CorpusID:278912262.

[21] Mark Chen et al. *Evaluating Large Language Models Trained on Code*. 2021. arXiv: 2107.03374 [cs.LG]. URL: https://arxiv.org/abs/2107.03374.

[22] Yi Chen and Gareth J. F. Jones. "Are episodic context features helpful for refinding tasks? lessons learnt from a case study with lifelogs". In: *Proceedings of the 5th Information Interaction in Context Symposium*. IIiX '14. Regensburg, Germany: Association for Computing Machinery, 2014, pp. 76–85. ISBN: 9781450329767. DOI: 10.1145/2637002.2637013. URL: https://doi.org/10.1145/2637002.2637013.

[23] Yi Chen and Gareth JF Jones. "Augmenting human memory using personal lifelogs". In: *Proceedings of the 1st augmented human international conference*. 2010, pp. 1–9.

[24] Zesen Cheng et al. *VideoLLaMA 2: Advancing Spatial-Temporal Modeling and Audio Understanding in Video-LLMs*. 2024. arXiv: 2406.07476 [cs.CV]. URL: https://arxiv.org/abs/2406.07476.

[25] Eunsol Choi et al. "QuAC: Question answering in context". In: *arXiv preprint arXiv:1808.07036* (2018).

[26] Junho Choi et al. "Intelligent healthcare service using health lifelog analysis". In: *Journal of medical systems* 40.8 (2016), p. 188.

[27] Soumyadeb Chowdhury et al. ""My Day in Review": Visually Summarising Noisy Lifelog Data". In: *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. ICMR '15. Shanghai, China: Association for Computing Machinery, 2015, pp. 607–610. ISBN: 9781450332743. DOI: 10.1145/2671188.2749393. URL: https://doi.org/10.1145/2671188.2749393.

[28] Yung-Sung Chuang et al. "Expand, Rerank, and Retrieve: Query Reranking for Open-Domain Question Answering". In: *Findings of the Association for Computational Linguistics: ACL 2023*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 12131–12147. DOI: 10.18653/v1/2023.findings-acl.768. URL: https://aclanthology.org/2023.findings-acl.768/.

[29] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG]. URL: https://arxiv.org/abs/2110.14168.

[30] Duc-Tien Dang-Nguyen et al. "Overview of ImageCLEF lifelog 2017: lifelog retrieval and summarization". In: *CLEF 2017* (2017).

[31] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. arXiv: 2501.12948 [cs.CL]. URL: https://arxiv.org/abs/2501.12948.

[32] Michael Desmond et al. "EvaluLLM: LLM assisted evaluation of generative outputs". In: *Companion Proceedings of the 29th International Conference on Intelligent User Interfaces*. IUI '24 Companion. Greenville, SC, USA: Association for Computing Machinery, 2024, pp. 30–32. ISBN: 9798400705090. DOI: 10.1145/3640544.3645216. URL: https://doi.org/10.1145/3640544.3645216.

[33] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[34] Aiden R Doherty and Alan F Smeaton. "Automatically segmenting lifelog data into events". In: *2008 ninth international workshop on image analysis for multimedia interactive services*. IEEE. 2008, pp. 20–23.

[35] Aiden R. Doherty et al. "Combining image descriptors to effectively retrieve events from visual lifelogs". In: *Proceedings of the 1st ACM International Con-*

ference on Multimedia Information Retrieval. MIR '08. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2008, pp. 10–17. ISBN: 9781605583129. DOI: 10.1145/1460096.1460100. URL: https://doi.org/10.1145/1460096.1460100.

[36] Aiden R. Doherty et al. "Experiences of Aiding Autobiographical Memory Using the SenseCam". In: *Human–Computer Interaction* 27.1-2 (2012), pp. 151–174. DOI: 10.1080/07370024.2012.656050. eprint: https://doi.org/10.1080/07370024.2012.656050. URL: https://doi.org/10.1080/07370024.2012.656050.

[37] Aiden R. Doherty et al. "Investigating keyframe selection methods in the novel domain of passively captured visual lifelogs". In: *Proceedings of the 2008 International Conference on Content-Based Image and Video Retrieval*. CIVR '08. Niagara Falls, Canada: Association for Computing Machinery, 2008, pp. 259–268. ISBN: 9781605580708. DOI: 10.1145/1386352.1386389. URL: https://doi.org/10.1145/1386352.1386389.

[38] Aiden R. Doherty et al. "Passively recognising human activities through lifelogging". In: *Computers in Human Behavior* 27.5 (2011). 2009 Fifth International Conference on Intelligent Computing, pp. 1948–1958. ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2011.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0747563211000963.

[39] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[40] Yiming Du et al. "PerLTQA: A Personal Long-Term Memory Dataset for Memory Classification, Retrieval, and Fusion in Question Answering". In: *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*. Ed. by Kam-Fai Wong et al. Bangkok, Thailand: Associa-

tion for Computational Linguistics, Aug. 2024, pp. 152–164. URL: https://aclanthology.org/2024.sighan-1.18/.

[41] Aaron Duane and Cathal Gurrin. "User Interaction for Visual Lifelog Retrieval in a Virtual Environment". In: *MultiMedia Modeling*. Ed. by Ioannis Kompatsiaris et al. Cham: Springer International Publishing, 2019, pp. 239–250. ISBN: 978-3-030-05710-7.

[42] Chenyou Fan. "EgoVQA - An Egocentric Video Question Answering Benchmark Dataset". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)* (2019), pp. 4359–4366. URL: https://api.semanticscholar.org/CorpusID:203584094.

[43] Min-Huan Fu et al. "Incorporating semantic knowledge for visual lifelog activity recognition". In: *Proceedings of the 2020 international conference on multimedia retrieval*. 2020, pp. 450–456.

[44] Yifan Gao et al. "Generating distractors for reading comprehension questions from real examinations". In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'19/IAAI'19/EAAI'19. Honolulu, Hawaii, USA: AAAI Press, 2019. ISBN: 978-1-57735-809-1. DOI: 10.1609/aaai.v33i01.33016423. URL: https://doi.org/10.1609/aaai.v33i01.33016423.

[45] Yunfan Gao et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: 2312.10997 [cs.CL]. URL: https://arxiv.org/abs/2312.10997.

[46] Jim Gemmell, Gordon Bell, and Roger Lueder. "MyLifeBits: a personal database for everything". In: *Commun. ACM* 49.1 (Jan. 2006), pp. 88–95. ISSN: 0001-0782. DOI: 10.1145/1107458.1107460. URL: https://doi.org/10.1145/1107458.1107460.

[47] Jim Gemmell et al. "MyLifeBits: fulfilling the Memex vision". In: *Proceedings of the Tenth ACM International Conference on Multimedia*. MULTIMEDIA '02. Juan-les-Pins, France: Association for Computing Machinery, 2002, pp. 235–238. ISBN: 158113620X. DOI: 10.1145/641007.641053. URL: https://doi.org/10.1145/641007.641053.

[48] Jiawei Gu et al. *A Survey on LLM-as-a-Judge*. 2025. arXiv: 2411.15594 [cs.CL]. URL: https://arxiv.org/abs/2411.15594.

[49] Rashmi Gupta and Cathal Gurrin. "Approaches for event segmentation of visual lifelog data". In: *International Conference on Multimedia Modeling*. Springer. 2018, pp. 581–593.

[50] Cathal Gurrin, Alan F Smeaton, Aiden R Doherty, et al. "Lifelogging: Personal big data". In: *Foundations and Trends® in information retrieval* 8.1 (2014), pp. 1–125.

[51] Cathal Gurrin et al. "A privacy by design approach to lifelogging". In: 2014. URL: https://api.semanticscholar.org/CorpusID:56287584.

[52] Cathal Gurrin et al. "A Test Collection for Interactive Lifelog Retrieval". In: *MultiMedia Modeling*. Ed. by Ioannis Kompatsiaris et al. Cham: Springer International Publishing, 2019, pp. 312–324.

[53] Cathal Gurrin et al. "An examination of a large visual lifelog". In: *Proceedings of the 4th Asia Information Retrieval Conference on Information Retrieval Technology*. AIRS'08. Harbin, China: Springer-Verlag, 2008, pp. 537–542. ISBN: 3540686339.

[54] Cathal Gurrin et al. "Introduction to the 8th Annual Lifelog Search Challenge, LSC'25". In: *Proceedings of the 2025 International Conference on Multimedia Retrieval*. ICMR '25. Chicago, IL, USA: Association for Computing Machinery, 2025, pp. 2143–2144. ISBN: 9798400718779. DOI: 10.1145/3731715.3734579. URL: https://doi.org/10.1145/3731715.3734579.

[55] Cathal Gurrin et al. “Introduction to the Fifth Annual Lifelog Search Challenge, LSC’22”. In: New York, NY, USA: Association for Computing Machinery, 2022. ISBN: 9781450392389. DOI: 10.1145/3512527.3531439. URL: https://doi.org/10.1145/3512527.3531439.

[56] Cathal Gurrin et al. “Introduction to the seventh annual lifelog search challenge, lsc’24”. In: *Proceedings of the 2024 International Conference on Multimedia Retrieval*. 2024, pp. 1334–1335.

[57] Cathal Gurrin et al. “Introduction to the Sixth Annual Lifelog Search Challenge, LSC’23”. In: *Proceedings of the 2023 ACM International Conference on Multimedia Retrieval*. ICMR ’23. Thessaloniki, Greece: Association for Computing Machinery, 2023, pp. 678–679. ISBN: 9798400701788. DOI: 10.1145/3591106.3592304. URL: https://doi.org/10.1145/3591106.3592304.

[58] Cathal Gurrin et al. “Introduction to the Third Annual Lifelog Search Challenge (LSC’20)”. In: *Proceedings of the 2020 International Conference on Multimedia Retrieval*. ICMR ’20. Dublin, Ireland: Association for Computing Machinery, 2020, pp. 584–585. ISBN: 9781450370875. DOI: 10.1145/3372278.3388043. URL: https://doi.org/10.1145/3372278.3388043.

[59] Cathal Gurrin et al. “NTCIR Lifelog: The First Test Collection for Lifelog Research”. In: SIGIR ’16. Pisa, Italy: Association for Computing Machinery, 2016, pp. 705–708. ISBN: 9781450340694. DOI: 10.1145/2911451.2914680. URL: https://doi.org/10.1145/2911451.2914680.

[60] Cathal Gurrin et al. “Overview of NTCIR-12 Lifelog Task”. In: *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*. 7–10 June 2016. Tokyo, Japan, 2016.

[61] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[62] Dan Hendrycks et al. *Measuring Massive Multitask Language Understanding*. 2021. arXiv: 2009.03300 [cs.CY]. URL: https://arxiv.org/abs/2009.03300.

[63] Duy-Khang Ho et al. "LSC-ADL: An Activity of Daily Living (ADL)-Annotated Lifelog Dataset Generated via Semi-Automatic Clustering". In: *Proceedings of the 33rd ACM International Conference on Multimedia*. MM '25. Dublin, Ireland: Association for Computing Machinery, 2025, pp. 12600–12606. ISBN: 9798400720352. DOI: 10.1145/3746027.3758195. URL: https://doi.org/10.1145/3746027.3758195.

[64] Steve Hodges, Emma Berry, and Ken Woodberry. "SenseCam: A wearable camera that stimulates and rehabilitates autobiographical memory". In: *Memory (Hove, England)* 19 (Oct. 2011), pp. 685–96. DOI: 10.1080/09658211.2011.605591.

[65] Lei Huang et al. "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions". In: *ACM Transactions on Information Systems* 43.2 (2025), pp. 1–55.

[66] Weijing Huang et al. "Generating reasonable legal text through the combination of language modeling and question answering". In: *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 2021, pp. 3687–3693.

[67] Ishwarappa and J. Anuradha. "A Brief Introduction on Big Data 5Vs Characteristics and Hadoop Technology". In: *Procedia Computer Science* 48 (2015). International Conference on Computer, Communication and Convergence (ICCC 2015), pp. 319–324. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.04.188.

[68] Tim Jacquemard. "Ethics of lifelog technology". PhD thesis. Dublin City University, 2014.

[69] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL]. URL: https://arxiv.org/abs/2310.06825.

[70] Lu Jiang et al. *MemexQA: Visual Memex Question Answering*. 2017. arXiv: 1708.01336 [cs.CV]. URL: https://arxiv.org/abs/1708.01336.

[71] Dian Jiao et al. *DuetRAG: Collaborative Retrieval-Augmented Generation*. 2024. arXiv: 2405.13002 [cs.CL]. URL: https://arxiv.org/abs/2405.13002.

[72] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.

[73] Vladimir Karpukhin et al. "Dense Passage Retrieval for Open-Domain Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. DOI: 10.18653/v1/2020.emnlp-main.550. URL: https://aclanthology.org/2020.emnlp-main.550/.

[74] Vladimir Karpukhin et al. *Dense Passage Retrieval for Open-Domain Question Answering*. 2020. arXiv: 2004.04906 [cs.CL]. URL: https://arxiv.org/abs/2004.04906.

[75] Zixuan Ke et al. *A Survey of Frontiers in LLM Reasoning: Inference Scaling, Learning to Reason, and Agentic Systems*. 2025. arXiv: 2504.09037 [cs.AI]. URL: https://arxiv.org/abs/2504.09037.

[76] Omar Khattab and Matei Zaharia. *ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT*. 2020. arXiv: 2004.12832 [cs.IR]. URL: https://arxiv.org/abs/2004.12832.

[77] Basel Kikhia et al. "Building digital life stories for memory support". In: *International journal of Computers in Healthcare* 1.2 (2010), pp. 161–176.

[78] Kiseung Kim and Jay-Yoon Lee. *RE-RAG: Improving Open-Domain QA Performance and Interpretability with Relevance Estimator in Retrieval-Augmented Generation*. 2024. arXiv: 2406.05794 [cs.CL]. URL: https://arxiv.org/abs/2406.05794.

[79] Seongjung Kim et al. "Ubiquitous Healthcare System for Analysis of Chronic Patients' Biological and Lifelog Data". In: *IEEE Access* 6 (2018), pp. 8909–8915. DOI: 10.1109/ACCESS.2018.2805304.

[80] Denis Kocetkov et al. *The Stack: 3 TB of permissively licensed source code*. 2022. arXiv: 2211.15533 [cs.CL]. URL: https://arxiv.org/abs/2211.15533.

[81] Chakravanti Rajagopalachari Kothari. *Research Methodology*. New Age International, 2004.

[82] Gunjan Kumar et al. "Towards activity recommendation from lifelogs". In: *Proceedings of the 16th international conference on information integration and web-based applications & services*. 2014, pp. 87–96.

[83] Hongwook Kwon et al. "Lifelog data-based prediction model of digital health care app customer churn: retrospective observational study". In: *Journal of Medical Internet Research* 23.1 (2021), e22184.

[84] Matthew L Lee and Anind K Dey. "Lifelogging memory appliance for people with episodic memory impairment". In: *Proceedings of the 10th international conference on Ubiquitous computing*. 2008, pp. 44–53.

[85] Jie Lei et al. *TVQA: Localized, Compositional Video Question Answering*. 2019. arXiv: 1809.01696 [cs.CL]. URL: https://arxiv.org/abs/1809.01696.

[86] Mike Lewis et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv: 1910.13461 [cs.CL]. URL: https://arxiv.org/abs/1910.13461.

[87] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in neural information processing systems* 33 (2020), pp. 9459–9474.

[88] Feng Li et al. "Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models". In: *arXiv preprint arXiv:2407.07895* (2024).

[89] Jiahao Nick Li, Zhuohao (Jerry) Zhang, and Jiaju Ma. "OmniQuery: Enabling Question Answering on Personal Memory by Augmenting Multimodal Album Data". In: *Adjunct Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. UIST Adjunct '24. Pittsburgh, PA, USA: Association for Computing Machinery, 2024. ISBN: 9798400707186. DOI: 10.1145/3672539.3686313. URL: https://doi.org/10.1145/3672539.3686313.

[90] Junnan Li et al. *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models*. 2023. arXiv: 2301.12597 [cs.CV]. URL: https://arxiv.org/abs/2301.12597.

[91] Junnan Li et al. *BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation*. 2022. arXiv: 2201.12086 [cs.CV]. URL: https://arxiv.org/abs/2201.12086.

[92] Ga-Young Lim et al. "Lifelog-based daily step counts, walking speed, and metabolically healthy status". In: *Digital Health* 10 (2024), p. 20552076241260921.

[93] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013.

[94] Yankai Lin et al. "Learning entity and relation embeddings for knowledge graph completion". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 2181–2187. ISBN: 0262511290.

[95] Aixin Liu et al. "Deepseek-v3 technical report". In: *arXiv preprint arXiv:2412.19437* (2024).

[96] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL]. URL: https://arxiv.org/abs/1907.11692.

[97] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101.

[98] Antoine Louis, Gijs van Dijck, and Gerasimos Spanakis. "Interpretable long-form legal question answering with retrieval-augmented large language models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 20. 2024, pp. 22266–22275.

[99] Pan Lu et al. *Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering*. 2022. arXiv: 2209.09513 [cs.CL]. URL: https://arxiv.org/abs/2209.09513.

[100] Mary M Lucas et al. "Reasoning with large language models for medical question answering". In: *Journal of the American Medical Informatics Association* 31.9 (2024), pp. 1964–1975.

[101] Tegan Maharaj et al. *A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering*. 2017. arXiv: 1611.07810 [cs.CV]. URL: https://arxiv.org/abs/1611.07810.

[102] Tai Tan Mai et al. "AIQAM'25: The 2nd ACM Workshop on AI-powered Question Answering Systems for Multimedia". In: *Proceedings of the 33rd ACM International Conference on Multimedia*. MM '25. Dublin, Ireland: Association for Computing Machinery, 2025, pp. 14299–14301. ISBN: 9798400720352. DOI: 10.1145/3746027.3762103. URL: https://doi.org/10.1145/3746027.3762103.

[103] Tai Tan Mai et al. "The First ACM Workshop on AI-Powered Question Answering Systems for Multimedia". In: *Proceedings of the 2024 International Conference on Multimedia Retrieval*. ICMR '24. Phuket, Thailand: Association for Computing Machinery, 2024, pp. 1328–1329. ISBN: 9798400706196. DOI: 10.1145/3652583.3658890. URL: https://doi.org/10.1145/3652583.3658890.

[104] Maria Movin and Claudia Hauff. "Zero-Shot Reranking with Large Language Models and Precomputed Ranking Features: Opportunities and Limitations". In: *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '25. Padua, Italy: Association for Computing Machinery, 2025, pp. 2276–2286. ISBN: 9798400715921. DOI: 10.1145/3726302.3730119. URL: https://doi.org/10.1145/3726302.3730119.

[105] Niklas Muennighoff et al. *MTEB: Massive Text Embedding Benchmark*. 2023. arXiv: 2210.07316 [cs.CL]. URL: https://arxiv.org/abs/2210.07316.

[106] Bernd Münzer et al. "lifeXplore at the Lifelog Search Challenge 2018". In: *Proceedings of the 2018 ACM Workshop on The Lifelog Search Challenge*. LSC '18. Yokohama, Japan: Association for Computing Machinery, 2018, pp. 3–8. ISBN: 9781450357968. DOI: 10.1145/3210539.3210541. URL: https://doi.org/10.1145/3210539.3210541.

[107] Jaap MJ Murre and Joeri Dros. "Replication and analysis of Ebbinghaus' forgetting curve". In: *PloS one* 10.7 (2015), e0120644.

[108] Thao-Nhu Nguyen et al. "E-LifeSeeker: An Interactive Lifelog Search Engine for LSC'23". In: *Proceedings of the 6th Annual ACM Lifelog Search Challenge*. LSC '23. Thessaloniki, Greece: Association for Computing Machinery, 2023, pp. 13–17. ISBN: 9798400701887. DOI: 10.1145/3592573.3593098. URL: https://doi.org/10.1145/3592573.3593098.

[109] Thang-Long Nguyen-Ho et al. "LSAT Focus: EAGLE's Embedded Entities Highlighting Technique for NTCIR-18 Lifelog-6". In: (2025).

[110] Thang-Long Nguyen-Ho et al. "T-EAGLE: Capturing Temporal Narratives via Sequence Captioning and Text Matching". In: *Proceedings of the 8th Annual ACM Workshop on the Lifelog Search Challenge*. LSC '25. Association for Computing Machinery, 2025, pp. 23–27. ISBN: 9798400718571. DOI: 10.1145/3729459.3748691. URL: https://doi.org/10.1145/3729459.3748691.

[111] OpenAI. *GPT-4 Technical Report*. Tech. rep. Accessed: 2024-09-11. OpenAI, 2023. URL: https://arxiv.org/pdf/2303.08774v3.pdf.

[112] OpenAI. *GPT-4o System Card*. 2024. arXiv: 2410.21276 [cs.CL]. URL: https://arxiv.org/abs/2410.21276.

[113] OpenAI. *Introducing ChatGPT*. https://openai.com/index/chatgpt/. Accessed: 24-03-2026. 2022.

[114] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL]. URL: https://arxiv.org/abs/2203.02155.

[115] Kishore Papineni et al. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135.

[116] Karmvir Singh Phogat et al. "Zero-Shot Question Answering over Financial Documents using Large Language Models". In: *arXiv preprint arXiv:2311.14722* (2023).

[117] Shraman Pramanick, Rama Chellappa, and Subhashini Venugopalan. *SPIQA: A Dataset for Multimodal Question Answering on Scientific Papers*. 2025. arXiv: 2407.09413 [cs.CL]. URL: https://arxiv.org/abs/2407.09413.

[118] Zhengwei Qiu et al. "Automatically detecting important moments from everyday life using a mobile device". In: *CHI 2011 Workshop on Video Interaction – Making Broadcasting a Successful Social Media*. Workshop paper. Available: https://doras.dcu.ie/16357/. Vancouver, Canada, 2011. URL: https://doras.dcu.ie/16357/.

[119] Zhengwei Qiu et al. "Mining user activity as a context source for search and retrieval". In: *2011 International Conference on Semantic Technology and Information Retrieval*. IEEE. 2011, pp. 162–166.

[120] Martin Rader and Klaus Schoeffmann. "lifeXplore 2025: Is Search in Text Generated by ChatGPT and BLIP2 better than in Embeddings from OpenCLIP?" In: *Proceedings of the 8th Annual ACM Workshop on the Lifelog Search Challenge*. LSC '25. Association for Computing Machinery, 2025, pp. 34–38. ISBN: 9798400718571. DOI: 10.1145/3729459.3748695. URL: https://doi.org/10.1145/3729459.3748695.

[121] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020.

[122] Rafael Rafailov et al. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. 2024. arXiv: 2305.18290 [cs.LG]. URL: https://arxiv.org/abs/2305.18290.

[123] Colin Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *J. Mach. Learn. Res.* 21.1 (Jan. 2020). ISSN: 1532-4435.

[124] D. Rajasekar and Rajesh Verma. *Research Methodology*. Archers & Elevators Publishing House, 2013.

[125] Juan Ramos et al. "Using tf-idf to determine word relevance in document queries". In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. 1. New Jersey, USA. 2003, pp. 29–48.

[126] Ricardo Ribeiro, Alina Trifan, and António J R Neves. "Lifelog Retrieval From Daily Digital Data: Narrative Review". In: *JMIR Mhealth Uhealth* 10.5 (2022), e30517. ISSN: 2291-5222. DOI: 10.2196/30517. URL: http://www.ncbi.nlm.nih.gov/pubmed/35499858.

[127] Stephen Robertson, Hugo Zaragoza, et al. "The probabilistic relevance framework: BM25 and beyond". In: *Foundations and Trends® in Information Retrieval* 3.4 (2009), pp. 333–389.

[128] Luca Rossetto. "OpenLifelogCam - A Low-Cost Open-Source Wearable Camera Platform". In: *Proceedings of the 2024 International Conference on Multimedia Retrieval*. ICMR '24. Phuket, Thailand: Association for Computing Machinery, 2024, pp. 1236–1240. ISBN: 9798400706196. DOI: 10.1145/3652583.3657588. URL: https://doi.org/10.1145/3652583.3657588.

[129] Luca Rossetto. "vitrivr-engine at the NTCIR-18 Lifelog-6 Task". In: *NTCIR Conference on Evaluation of Information Access Technologies*. 2025. URL: https://api.semanticscholar.org/CorpusID:280336990.

[130] Luca Rossetto et al. "A System for Interactive Multimedia Retrieval Evaluations". In: *MultiMedia Modeling*. Ed. by Jakub Lokoč et al. Cham: Springer International Publishing, 2021, pp. 385–390. ISBN: 978-3-030-67835-7.

[131] Florian Ruosch and Luca Rossetto. "A SPARQL in the Dark: Shining a Light on Multimodal Lifelogs with LifeGraph 5". In: *Proceedings of the 8th Annual ACM Workshop on the Lifelog Search Challenge*. LSC '25. Association for Computing Machinery, 2025, pp. 39–43. ISBN: 9798400718571. DOI: 10.1145/3729459.3748694. URL: https://doi.org/10.1145/3729459.3748694.

[132] Khaled Saab et al. *Capabilities of Gemini Models in Medicine*. 2024. arXiv: 2404.18416 [cs.AI]. URL: https://arxiv.org/abs/2404.18416.

[133] Apoorv Saxena, Soumen Chakrabarti, and Partha Talukdar. "Question Answering Over Temporal Knowledge Graphs". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th*

International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 6663–6676. DOI: 10.18653/v1/2021.acl-long.520. URL: https://aclanthology.org/2021.acl-long.520/.

[134] Klaus Schoeffmann. "lifeXplore at the Lifelog Search Challenge 2023". In: *Proceedings of the 6th Annual ACM Lifelog Search Challenge*. LSC '23. Thessaloniki, Greece: Association for Computing Machinery, 2023, pp. 53–58. ISBN: 9798400701887. DOI: 10.1145/3592573.3593105. URL: https://doi.org/10.1145/3592573.3593105.

[135] John Schulman et al. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. 2018. arXiv: 1506.02438 [cs.LG]. URL: https://arxiv.org/abs/1506.02438.

[136] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[137] Salim G Shaikh, B Suresh Kumar, and Geetika Narang. "Recommender system for health care analysis using machine learning technique: A review". In: *Theoretical Issues in Ergonomics Science* 23.5 (2022), pp. 613–642.

[138] Zhihong Shao et al. *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. 2024. arXiv: 2402.03300 [cs.CL]. URL: https://arxiv.org/abs/2402.03300.

[139] Noam Shazeer et al. *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*. 2017. arXiv: 1701.06538 [cs.LG]. URL: https://arxiv.org/abs/1701.06538.

[140] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556.

[141] Karan Singhal et al. "Towards expert-level medical question answering with large language models". In: *arXiv preprint arXiv:2305.09617* (2023).

[142] Shamane Siriwardhana et al. "Improving the Domain Adaptation of Retrieval Augmented Generation (RAG) Models for Open Domain Question Answering". In: *Transactions of the Association for Computational Linguistics* 11 (Jan. 2023), pp. 1–17. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00530. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00530/2067834/tacl_a_00530.pdf. URL: https://doi.org/10.1162/tacl%5C_a%5C_00530.

[143] Charlie Snell et al. *Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters*. 2024. arXiv: 2408.03314 [cs.LG]. URL: https://arxiv.org/abs/2408.03314.

[144] Florian Spiess and Heiko Schuldt. "Multimodal Interactive Lifelog Retrieval with vitrivr-VR". In: *Proceedings of the 5th Annual on Lifelog Search Challenge*. LSC '22. Newark, NJ, USA: Association for Computing Machinery, 2022, pp. 38–42. ISBN: 9781450392396. DOI: 10.1145/3512729.3533008. URL: https://doi.org/10.1145/3512729.3533008.

[145] Guanglu Sun et al. "Video question answering: a survey of models and datasets". In: *Mobile Networks and Applications* (2021), pp. 1–34.

[146] Wang-Chiew Tan et al. "Timelineqa: A benchmark for question answering over timelines". In: *Findings of the Association for Computational Linguistics: ACL 2023*. 2023, pp. 77–91.

[147] Makarand Tapaswi et al. *MovieQA: Understanding Stories in Movies through Question-Answering*. 2016. arXiv: 1512.02902 [cs.CV]. URL: https://arxiv.org/abs/1512.02902.

[148] Ross Taylor et al. *Galactica: A Large Language Model for Science*. 2022. arXiv: 2211.09085 [cs.CL]. URL: https://arxiv.org/abs/2211.09085.

[149] Gemini Team et al. *Gemini: A Family of Highly Capable Multimodal Models*. 2024. arXiv: 2312.11805 [cs.CL]. URL: https://arxiv.org/abs/2312.11805.

[150] Qwen Team. *Qwen3 Technical Report*. 2025. arXiv: 2505.09388 [cs.CL]. URL: https://arxiv.org/abs/2505.09388.

[151] Anirudh Thatipelli, Shao-Yuan Lo, and Amit K. Roy-Chowdhury. "Egocentric and exocentric methods: A short survey". In: *Computer Vision and Image Understanding* 257 (2025), p. 104371. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2025.104371. URL: https://www.sciencedirect.com/science/article/pii/S1077314225000943.

[152] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL]. URL: https://arxiv.org/abs/2307.09288.

[153] Allie Tran and Cathal Gurrin. "MyEachtraX: Semantic Event Retrieval with Time-Weighted Fusion". In: *Proceedings of the 8th Annual ACM Workshop on the Lifelog Search Challenge*. LSC '25. Association for Computing Machinery, 2025, pp. 44–51. ISBN: 9798400718571. DOI: 10.1145/3729459.3748690. URL: https://doi.org/10.1145/3729459.3748690.

[154] Ly-Duyen Tran. "Pushing the Boundaries of Lifelog Retrieval Systems with Question Answering Techniques". PhD thesis. Dublin City University, 2024.

[155] Ly-Duyen Tran et al. "E-Myscéal: Embedding-based Interactive Lifelog Retrieval System for LSC'22". In: *Proceedings of the 5th Annual on Lifelog Search Challenge*. LSC '22. Newark, NJ, USA: Association for Computing Machinery, 2022, pp. 32–37. ISBN: 9781450392396. DOI: 10.1145/3512729.3533012. URL: https://doi.org/10.1145/3512729.3533012.

[156] Ly-Duyen Tran et al. "Interactive Question Answering for Multimodal Lifelog Retrieval". In: *International Conference on Multimedia Modeling*. Springer. 2024, pp. 68–81.

[157] Ly-Duyen Tran et al. "LLQA-lifelog question answering dataset". In: *International Conference on Multimedia Modeling*. Springer. 2022, pp. 217–228.

[158] Ly-Duyen Tran et al. "Myscéal: An Experimental Interactive Lifelog Retrieval System for LSC'20". In: *Proceedings of the Third Annual Workshop on Lifelog Search Challenge*. LSC '20. Dublin, Ireland: Association for Computing Machinery, 2020, pp. 23–28. ISBN: 9781450371360. DOI: 10.1145/3379172.3391719. URL: https://doi.org/10.1145/3379172.3391719.

[159] Khanh-Tung Tran et al. *Multi-Agent Collaboration Mechanisms: A Survey of LLMs*. 2025. arXiv: 2501.06322 [cs.AI]. URL: https://arxiv.org/abs/2501.06322.

[160] Ly Duyen Tran et al. "MyEachtra: Event-Based Interactive Lifelog Retrieval System for LSC'23". In: *Proceedings of the 6th Annual ACM Lifelog Search Challenge*. LSC '23. Thessaloniki, Greece: Association for Computing Machinery, 2023, pp. 24–29. ISBN: 9798400701887. DOI: 10.1145/3592573.3593100. URL: https://doi.org/10.1145/3592573.3593100.

[161] Quang-Linh Tran et al. "MemoriLens: a Low-cost Lifelog Camera Using Raspberry Pi Zero". In: *Proceedings of the 2024 International Conference on Multimedia Retrieval*. ICMR '24. Phuket, Thailand: Association for Computing Machinery, 2024, pp. 1255–1259. ISBN: 9798400706196. DOI: 10.1145/3652583.3657592. URL: https://doi.org/10.1145/3652583.3657592.

[162] Quang-Linh Tran et al. *OpenLifelogQA: An Open-Ended Multi-Modal Lifelog Question-Answering Dataset*. 2025. arXiv: 2508.03583 [cs.MM]. URL: https://arxiv.org/abs/2508.03583.

[163] Tijmen Van Teijlingen, Erik Oudman, and Albert Postma. "Lifelogging as a rehabilitation tool in patients with amnesia: A narrative literature review on the effect of lifelogging on memory loss". In: *Neuropsychological Rehabilitation* 32.10 (2022), pp. 2646–2672.

[164] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[165] Benjamin Warner et al. *Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference*. 2024. arXiv: 2412.13663 [cs.CL]. URL: https://arxiv.org/abs/2412.13663.

[166] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. NIPS '22. New Orleans, LA, USA: Curran Associates Inc., 2022. ISBN: 9781713871088.

[167] Jason Wei et al. *Emergent Abilities of Large Language Models*. 2022. arXiv: 2206.07682 [cs.CL]. URL: https://arxiv.org/abs/2206.07682.

[168] Zhepei Wei, Wei-Lin Chen, and Yu Meng. "InstructRAG: Instructing Retrieval-Augmented Generation with Explicit Denoising". In: *arXiv preprint arXiv:2406.13629* (2024).

[169] Qi Wu et al. "Visual question answering: A survey of methods and datasets". In: *Computer Vision and Image Understanding* 163 (2017), pp. 21–40.

[170] Guangzhi Xiong et al. *Benchmarking Retrieval-Augmented Generation for Medicine*. 2024. arXiv: 2402.13178 [cs.CL]. URL: https://arxiv.org/abs/2402.13178.

[171] Hu Xu et al. "Vlm: Task-agnostic video-language model pre-training for video understanding". In: *arXiv preprint arXiv:2105.09996* (2021).

[172] Li Xu, He Huang, and Jun Liu. *SUTD-TrafficQA: A Question Answering Benchmark and an Efficient Network for Video Reasoning over Traffic Events*. 2021. arXiv: 2103.15538 [cs.CV]. URL: https://arxiv.org/abs/2103.15538.

[173] Qianli Xu et al. "Lifelog image retrieval based on semantic relevance mapping". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 17.3 (2021), pp. 1–18.

[174] Antoine Yang et al. *Zero-Shot Video Question Answering via Frozen Bidirectional Language Models*. 2022. arXiv: 2206.08155 [cs.CV]. URL: https://arxiv.org/abs/2206.08155.

[175] Yang Yang, Hyowon Lee, and Cathal Gurrin. "Visualizing lifelog data for different interaction platforms". In: *CHI '13 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '13. Paris, France: Association for Computing Machinery, 2013, pp. 1785–1790. ISBN: 9781450319522. DOI: 10.1145/2468356.2468676. URL: https://doi.org/10.1145/2468356.2468676.

[176] Zhilin Yang et al. "HotpotQA: A dataset for diverse, explainable multi-hop question answering". In: *arXiv preprint arXiv:1809.09600* (2018).

[177] Shunyu Yao et al. "Tree of thoughts: Deliberate problem solving with large language models". In: *Advances in Neural Information Processing Systems* 36 (2024).

[178] Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. "Pretrained transformers for text ranking: BERT and beyond". In: *Proceedings of the 14th ACM International Conference on web search and data mining*. 2021, pp. 1154–1156.

[179] An-Zi Yen et al. "Visual lifelog retrieval: humans and machines interpretation on first-person images". In: *Multimedia Tools and Applications* 82.24 (2023), pp. 37757–37787.

[180] Antonio Jimeno Yepes et al. "Financial report chunking for effective retrieval augmented generation". In: *arXiv preprint arXiv:2402.05131* (2024).

[181] Xunjian Yin, Baizhou Huang, and Xiaojun Wan. *ALCUNA: Large Language Models Meet New Knowledge*. 2023. arXiv: 2310.14820 [cs.CL]. URL: https://arxiv.org/abs/2310.14820.

[182] Rowan Zellers et al. *Defending Against Neural Fake News*. 2020. arXiv: 1905.12616 [cs.CL]. URL: https://arxiv.org/abs/1905.12616.

[183] Dun Zhang et al. *Jasper and Stella: distillation of SOTA embedding models*. 2025. arXiv: 2412.19048 [cs.IR]. URL: https://arxiv.org/abs/2412.19048.

[184] Mi Zhang and Alexander A Sawchuk. "Human daily activity recognition with sparse representation using wearable sensors". In: *IEEE journal of Biomedical and Health Informatics* 17.3 (2013), pp. 553–560.

[185] Tianyi Zhang et al. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: 1904.09675 [cs.CL]. URL: https://arxiv.org/abs/1904.09675.

[186] Wayne Xin Zhao et al. "A survey of large language models". In: *arXiv preprint arXiv:2303.18223* 1.2 (2023).

[187] Yaoyao Zhong et al. "Video question answering: Datasets, algorithms and challenges". In: *arXiv preprint arXiv:2203.01225* (2022).

[188] Bolei Zhou et al. "Learning Deep Features for Scene Recognition using Places Database". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.

[189] Lijuan Marissa Zhou et al. "ShareDay: A Novel Lifelog Management System for Group Sharing". In: (2013), pp. 490–492.

[190] Liting Zhou et al. "Overview of the NTCIR-17 Lifelog-5 Task". In: *NTCIR Conference on Evaluation of Information Access Technologies*. 2023. URL: https://api.semanticscholar.org/CorpusID:266493788.

[191] Liting Zhou et al. "Overview of the NTCIR-18 Lifelog-6 Task". In: *Proceedings of the 18th NTCIR Conference on Evaluation of Information Access Technologies (NTCIR-18). Tokyo, Japan*. 2025.

[192] Fengbin Zhu et al. "Retrieving and reading: A comprehensive survey on open-domain question answering". In: *arXiv preprint arXiv:2101.00774* (2021).