Neural Entity Linking for Question Answering over Knowledge Graphs

---

Daniel Vollmers

May 20, 2026

Faculty of Computer Science, Electrical Engineering and Mathematics
Heinz Nixdorf Institute, Department of Computer Science
Data Science Group (DICE)

Doctoral Dissertation

Neural Entity Linking for Question Answering over Knowledge Graphs

Daniel Vollmers

1. Reviewer	Prof. Dr. Axel-Cyrille Ngonga Ngomo Department of Computer Science Paderborn University
2. Reviewer	Prof. Dr. Ricardo Usbeck Institute of Information Systems Leuphana University Lüneburg
Supervisors	Prof. Dr. Axel-Cyrille Ngonga Ngomo Dr. Hamada Mohamed Abdelsamee Zahera

May 20, 2026

Daniel Vollmers

Neural Entity Linking for Question Answering over Knowledge Graphs

Doctoral Dissertation, May 20, 2026

Reviewers: Prof. Dr. Axel-Cyrille Ngonga Ngomo and Prof. Dr. Ricardo Usbeck

Supervisors: Prof. Dr. Axel-Cyrille Ngonga Ngomo and Dr. Hamada Mohamed Abdelsamee Zahera

Paderborn University

Data Science Group (DICE)

Heinz Nixdorf Institute, Department of Computer Science

Faculty of Computer Science, Electrical Engineering and Mathematics

Warburger Str. 100

33098 and Paderborn

Abstract

One of the primary uses of the Internet remains the search for accurate information and the exploration of new or unfamiliar topics. Search engines and related applications, which retrieve relevant documents such as web pages in response to user queries, have long been central to information retrieval research. However, the information need of users is often more complex and requires the integration of information from multiple sources, for instance, to search for cost-effective car offers or to answer factual questions such as *"Which museums are located in Paderborn?"*.

This thesis investigates the task of question answering (QA). Recent advances in large language models (LLMs) trained on vast textual corpora have transformed this field. Although LLMs represent the current state of the art, they are prone to hallucinations, generating responses that are not always factually correct or grounded in reliable information. This limitation is especially problematic given the widespread availability of misinformation online.

To address this issue, a promising strategy is to ground LLM predictions in trusted knowledge sources such as text corpora, databases, or knowledge graphs (KGs). In this retrieval-augmented generation (RAG) paradigm, external information is retrieved and provided to the model to improve factual accuracy. When specifically leveraging knowledge graphs, this task is referred to as Knowledge Graph Question Answering (KGQA). This task can be solved by applying a semantic parsing approach to translate natural language questions into executable SPARQL queries. In this context, this thesis explores the application of neural entity linking approaches for question answering over knowledge graphs (KGQA). We develop methods for contextual augmentation, absent keyphrase extraction, and robust retrieval to enhance entity, relation, and type linking (ERL) and analyze how ERL performance affects overall KGQA effectiveness.

Zusammenfassung

Eine der zentralen Anwendungen des Internets besteht weiterhin in der gezielten Suche nach korrekten Informationen und der Erkundung neuer oder bislang unbekannter Themen. In diesem Zusammenhang spielen Suchmaschinen und verwandte Anwendungen, die als Antwort auf Benutzeranfragen relevante Dokumente, wie beispielsweise Webseiten abrufen, seit Langem eine zentrale Rolle in der Forschung. Das Informationsbedürfnis der Nutzenden ist jedoch häufig komplexer und erfordert die Integration von Informationen aus mehreren Quellen, etwa bei der Ermittlung des günstigsten Angebots für ein Auto oder bei der Beantwortung faktischer Fragen wie *"Welche Museen befinden sich in Paderborn?"*

Diese Arbeit beschäftigt sich mit dem Forschungsbereich Question Answering. Jüngste Fortschritte im Zusammenhang der Entwicklung groSSer Sprachmodelle (Large Language Models, LLMs), die auf umfangreichen Textkorpora trainiert werden, haben dieses Forschungsfeld grundlegend verändert. Obwohl LLMs den aktuellen Stand der Technik darstellen, neigen sie zu Halluzinationen, das heiSSt, sie erzeugen Antworten, die nicht immer faktisch korrekt oder auf verlässlichem Wissen beruhen. Diese Einschränkung ist besonders problematisch angesichts der weit verbreiteten Desinformation im Internet.

Zur Bewältigung dieses Problems bietet das anreichern von LLM prompts durch Informationen aus vertrauenswürdigen Wissensquellen, wie Textkorpora, Datenbanken oder Wissensgraphen (Knowledge Graphs, KGs), einen vielversprechenden Ansatz. In diesem Retrieval-Augmented-Generation (RAG)-Paradigma werden zunächst Information aus Wissensquellen abgerufen und dem Modell als zusätzlicher Input zur Verfügung gestellt, um die faktische Genauigkeit zu verbessern. Im Fall von Wissensgraphen wird diese Aufgabe als Question Answering über Wissensgraphen bezeichnet (KGQA). Um Fragen basierend auf den Informationen eines Wissensgraphen zu beantworten, können Semantic Parsing Verfahren angewandt werden, welche eine natürlichsparachliche Frage in eine ausführbare SPARQL Query übersetzen. In diesem Kontext konzentriert sich die Arbeit auf die Erforschung und Anwendung von neuronalen Entity Linking Verfahren, für das Forschungsthema Question Answering über Wissensgraphen (KGQA). Es werden Methoden zur kontextuellen Erweiterung, zur Extraktion fehlender Schlüsselwörter und zum robusten Retrieval entwickelt, um die Identifikation von Entitäten, Relationen und Typen (ERL) zu verbessern. Darüber hinaus wird analysiert, wie ERL die Gesamtleistung von KGQA-Systemen beeinflusst.

Acknowledgement

I would first like to express my sincere gratitude to Prof. Dr. Axel-Cyrille Ngonga Ngomo for giving me the opportunity to pursue my PhD studies within his research group and for his continuous support and patience throughout this journey.

I would also like to thank my second reviewer, Prof. Dr. Ricardo Usbeck, as well as the other members of my thesis committee–Prof. Dr. Roman Dumitrescu, Prof. Dr. Stefan Sauer, and Dr. Michael Röder–for their time and commitment.

Furthermore, I am deeply grateful to my colleagues, especially my supervisor, Dr. Hamada Zahera, and my former supervisor, Dr. Diego Campos Moussallem. I also extend my thanks to all my co-authors who supported my research and contributed to the publications that form the foundation of this thesis.

Finally, I would like to thank my family and friends, especially my parents, for their support throughout my bachelors and masters studies.

Erklärungen

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinnentsprechend übernommen worden sind, sind als solche gekennzeichnet.

Beim Verfassen dieser Arbeit habe ich zudem ChatGPT, Gemini, Grammarly und DeepL verwendet, um die sprachliche Formulierung sowie die Formatierung zu verbessern. Die inhaltliche Verantwortung liegt bei mir als Autor.

Paderborn, May 20, 2026

Daniel Vollmers

Contents

List of Figures

List of Tables

1 Introduction

In recent years, artificial intelligence (AI) has become increasingly popular and is currently a main focus in research and industry. Within the context of AI, this thesis explores the application of neural entity linking approaches for question answering over knowledge graphs (KGQA). KGQA is a vital task for enabling more accurate and efficient access to structured information. Our work is centered around the development of semantic parsing systems, which is one of the leading methods in this research field. Overall, this thesis comprises 11 chapters. Chapter 1 motivates the research conducted in this thesis, introduces research questions, and summarizes the contributions. Chapters 2 and 3 introduce the prerequisites for our research and summarize related works, respectively. Chapters 4-10 present the research contributions in detail. Finally, Chapter 11 summarizes the thesis and discusses future research directions.

1.1 Motivation

A recent study by Kemp (2026) shows that 'finding information' remains the primary purpose for people worldwide to use the internet, and that search engines and web portals are still the third most frequently used type of website or app, behind chat and messaging systems and social networks. Another study shows that manual search engines and social media are still the most commonly used platforms for searching information (Hedges, 2024). In traditional search systems, users submit queries containing only a few keywords to find information for tasks such as initial searches, fact-finding, or navigation (Fishkin, 2017; Heitzman, 2025). However, more complex applications, such as finding and comparing product information or answering complex questions, require combining information from multiple sources. For instance, to decide on a particular product, users may start with an initial search by using a traditional search engine, then collect product information from shopping platforms like Amazon, and read reviews in social networks for further comparison (Heitzman, 2025).

In recent years, AI chatbots trained on data from various online sources, like ChatGPT[1], have become increasingly popular, as they can support users in solving complex tasks that

[1] https://chatgpt.com/

go beyond traditional keyword searches for finding specific information. In this context, a survey by Garrel and Mayer (2023) shows that students at German universities use AI-based tools to clarify questions, explain subject-specific concepts, conduct literature reviews, and translate texts.

An essential task in AI chatbots is question answering (QA). Unlike traditional search engines, which apply information retrieval (IR) to extract relevant documents or passages for a query from an underlying set of documents, question answering aims to generate an answer for a natural language query directly (Zhang et al., 2023). Question answering is a complex task, as it often requires combining information from multiple documents or passages. In recent years, generative AI tools based on decoder-based large language models (LLMs), such as ChatGPT, have enabled significant progress in this research field, as they can learn from diverse data sources during training (Brown et al., 2020). However, since these frameworks tend to hallucination (Huang et al., 2025), it cannot be guaranteed that the predicted answers are accurate. To mitigate this, techniques such as retrieval-augmented generation (RAG) have been proposed, which retrieve data from diverse knowledge sources at inference time and incorporate this knowledge into the generative LLM to predict the final answer (Lewis et al., 2020b). The underlying knowledge sources of RAG systems

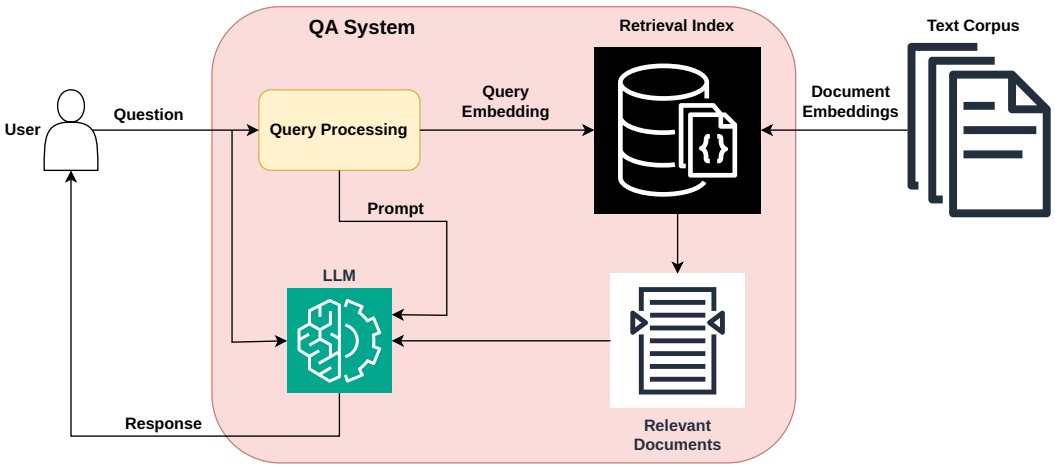


Figure 1.1: RAG-based Question Answering

are often large unstructured text corpora. For the retrieval step, the textual knowledge is usually indexed into modern vector databases such as Croma[2] or Faiss (Johnson et al., 2019), which enables retrieving relevant documents based on text embeddings generated by LLM-based encoder models such as BERT (Devlin et al., 2019). This allows for matching based on semantic similarity, as illustrated in Figure 1.1. Overall, RAG-based QA systems have consistently improved the performance of LLMs on the question answering task over unstructured datasets (Lewis et al., 2020b; Siriwardhana et al., 2023)

[2] https://www.trychroma.com/

Unlike unstructured sources, QA systems can also leverage information from structured knowledge sources, such as relational databases (Shi et al., 2025) or knowledge graphs (KGs) (Lan et al., 2021). Benchmarks in this domain often provide additional challenges, such as complex, multi-hop questions and aggregate queries (Sen et al., 2022; Usbeck et al., 2023), which are still difficult for RAG-based QA systems (Zhuang et al., 2024). In general, the literature distinguishes between two approaches to question answering on structured sources: retrieval-based approaches and semantic parsing (Zhang et al., 2025). Retrieval-based QA utilize retrieval techniques to leverage knowledge from structured sources. Afterward, ranking and generation approaches are applied to generate the final answer (Zhang et al., 2025). The drawback of this approach is that the decision process that leads to an answer being returned is usually not transparent to the user. Furthermore, aggregation functions, such as counting answers or summing solutions, are hard to implement. In contrast, semantic parsing approaches convert questions into logical forms such as database or graph queries that can be executed on databases or triple stores to generate answers (Perez-Beltrachini et al., 2023). The advantage of these approaches is that the query can be displayed to the user, providing an explanation of the underlying logic used to generate the answer. Additionally, common query languages enable the application of aggregation functions.

In this thesis, we focus primarily on question answering over knowledge graphs (KGQA). KGs represent real-world data as graphs, where nodes denote entities and edges describe relations between them (Hogan et al., 2021). Knowledge graphs can be assembled from heterogeneous data sources such as natural language texts or relational databases. A popular example of a knowledge graph is DBpedia (Auer et al., 2007), which is automatically derived from Wikipedia leveraging its link structure and structured information, such as infoboxes. KGs can vary in structure and granularity, making it challenging to search for specific information, since this requires knowledge of the KG's structure and its internal identifiers. To extract knowledge from KGs, knowledge graph question answering (KGQA) systems can provide an easy to use access point (Wu et al., 2019), since they enable users to query KGs without the need to understand the underlying structure. KGQA remains a challenging task, as multiple interconnected steps are required to extract relevant information from the KG, and questions can vary in complexity. The literature distinguishes between simple and complex questions, where simple questions involve only a single fact from the KG. In contrast, complex questions may involve multiple entities in the query, use compound relations, or require traversing multiple hops in the KG (Lan et al., 2021). For example, answering a question such as *"Who are the grandchildren of Helmut Kohl?"* requires first finding the children of the query entity (*"Helmut Kohl"*) and subsequently identifying the children of those individuals. Furthermore, some queries demand filtering, for example:

"Find mountains higher than 3.000 meters in Austria". Other queries may require finding maximum values across a set of entities, e.g., extracting the largest city in a country.

One major challenge for KGQA is identifying entities within the query and mapping them to the KG, which is usually one of the initial stages in a KGQA pipeline. In this context, Entity linking (EL) is the task of finding entity mentions in natural language texts and linking them to a target knowledge source (Ling et al., 2015). EL is generally approached in two steps: named entity recognition (NER), also called mention detection, and entity disambiguation (ED) (Sevgili et al., 2022). The goal of named entity recognition is to identify and classify textual spans referring to entities such as people, locations, and organizations (Yadav and Bethard, 2018). Early systems primary focused on the application of rules to annotate texts (Petasis et al., 2001). In contrast, modern approaches use neural network-based methods, especially various forms of recurrent neural networks (RNNs) (Yadav and Bethard, 2018), and more recently, LLMs (Wang et al., 2025).

Entity disambiguation (ED) is the second step in an EL system. It connects the entity mentions found in the NER step to identifiers in a knowledge graph. The first part of ED is usually candidate generation that extracts a set of candidate entities from the KG (Sevgili et al., 2022). For this purpose, textual representations of entities must be generated, for example, by utilizing entity labels or their corresponding surface forms. These representations can be further extended by adding descriptions, alternative names, synonyms, and abbreviations, which cover more possible entity matches (Sevgili et al., 2022). Candidate generation can be solved using classical or neural IR methods (Moussallem et al., 2018; Zhang et al., 2022c). Finally, the disambiguation step selects a candidate entity for each entity mention by leveraging the local KG context of the candidate entities, which may consist of interlinked entities or connected types. Early approaches applied graph-based disambiguation algorithms (Moussallem et al., 2018), while neural methods employ machine learning-based rerankers leveraging LM encoder models (Wu et al., 2020a) or generative models that align candidates with entity mentions (Zhang et al., 2022c).

For KGQA, it is essential to extract not only entities but also relations from the KG. One major challenge is that relations are often not directly expressed in the input question. For example, a question such as *"Which museums exist in New York"* does not directly mention relations such as *"type"* and *"location"*. Another challenge in KGQA comes from the annotation guidelines commonly used for NER datasets.[3] These guidelines usually define entities as mentions that have a proper name. However, in KGQA, many questions additionally require linking mentions representing KG types (e.g., the term *"museums"* in the previous example), which fall outside the standard definition of named entities.

[3] See, for instance: https://zenodo.org/records/4574199 and https://aclanthology.org/attachments/2024.lrec-main.1262.OptionalSupplementaryMaterial.pdf.

The goal of this thesis is to investigate neural entity linking for question answering. Neural approaches have shown significant improvements over traditional methods in recent years (Sevgili et al., 2022). In particular, LLM-based approaches achieve better performance on the classical entity linking task, as they leverage rich contextual information acquired during pretraining (Wu et al., 2020a). One goal of this thesis is to effectively extract entities and relations from knowledge graphs for given questions. To achieve this, we propose a new concept of LLM-based contextual augmentation, which expands the input context and entity mentions to resolve ambiguity. However, fine-tuning and deploying an LLM for each step in a KGQA pipeline requires substantial computing resources. Especially at inference time, running several LLMs in parallel increases cost and latency. To mitigate this, we show that a single LLM can be jointly fine-tuned for both mention detection and disambiguation. Furthermore, we will demonstrate that an LLM can be jointly trained to generate queries for multiple KGs.

LLM-based retrieval systems are essential components for KGQA systems. However, their performance is usually limited to the domain or scope of the documents seen during training. To make retrieval more robust, we explore noise injection during training as a simple but effective way to improve model generalization. Another goal is to compare traditional question answering systems, which primarily use rules and templates for query generation, with modern LLM-based query generation approaches. To this end, we implement systems for both paradigms and evaluate their performance. Finally, we extend classical entity linking with keyphrase extraction, enabling the model to capture not only entities but also relations and types–elements usually not covered by standard entity linking systems. We also compare the performance of traditional and neural entity and relation linking (ERL) pipelines on the KGQA task.

1.2 Research Questions and Contributions

Knowledge extraction and query generation are two essential components in KGQA frameworks. The output of the knowledge extraction step is commonly used as input for query generation. The primary focus of this thesis is to evaluate how the knowledge extraction performance influences query generation. Consequently, we distribute the research questions into two challenges: Knowledge extraction for KGQA and query generation.

1.2.1 Challenge I: Knowledge Extraction for KGQA

A key difficulty in predicting entities and relations in questions is that these elements are often implicitly mentioned. To solve this problem, we formulate the following research question:

Research Question 1 ($\mathbf{RQ}_1$)

What methods can be used to predict knowledge that is implicit rather than explicitly expressed in questions?

To answer this question, we present a framework that predicts absent keyphrases by linking present keyphrases in the form of n-grams to KGs to extract additional information. Furthermore, we propose the concept of contextual augmentation for entity linking to enrich the context and improve the system's disambiguation performance. For this, we answer the research question:

Research Question 2 ($\mathbf{RQ}_2$)

How can LLM-based augmentation enhance the performance of entity linking systems?

We answer this question by implementing a framework for contextual augmentation and evaluating it against state-of-the-art entity linking methods. Afterward, we adapt this concept to augment the context of questions to predict additional information, such as absent entities, relations, and types. We then evaluate different pipelines, especially dense-retrieval, autoregressive entity linking, and the traditional ED framework MAG (Moussallem et al., 2018), to assess the performance on extracting knowledge for questions, to answer the research question:

Research Question 3 ($\mathbf{RQ}_3$)

How can LLM-based entity linking approaches improve the performance for extracting KG-knowledge for questions?

Furthermore, we implement several LLM-based end-to-end knowledge extraction pipelines, evaluating them against each other as well as traditional approaches for entity and relation linking.

1.2.2 Challenge II: Query Generation

Fine-tuning LLMs is currently one of the leading approaches to translate questions into queries (Banerjee et al., 2022). However, tuning a separate LLM for each dataset and KG is resource-intensive and time-consuming. To improve this, we answer the question:

Research Question 4 ($\mathbf{RQ_4}$)

Can LLMs be jointly fine-tuned for query generation for multiple KGs, and how do these models perform compared to models trained for single KGs?

In contrast to fine-tuning methods, traditional KGQA systems often use templates for query prediction. Therefore, another goal of this thesis is to compare LLM-based query prediction approaches with template-based approaches to answer the research question:

Research Question 5 ($\mathbf{RQ_5}$)

How well can traditional template-based question answering approaches perform on the KGQA task compared to LLM-based approaches?

Query generation approaches depend on the output of the previous knowledge extraction step, therefore their performance is closely tied to the quality of that output. To investigate this, we answer the following research question:

Research Question 6 ($\mathbf{RQ_6}$)

How does the performance of knowledge extraction frameworks influence the performance of query prediction?

We answer this question by evaluating a fine-tuning-based query generation approach that uses the output of the knowledge extraction pipeline from **RQ3** as input. The evaluation focuses on measuring query prediction performance on the end-to-end KGQA task.

1.3 Thesis Outline

- **Chapter 2** presents the relevant background knowledge about the topics of entity linking and knowledge graph question answering. Additionally, all technologies used for the approaches and experiments in the follow-up sections are introduced and described.

- **Chapter 3** describes the relevant state of the art in the research fields of knowledge graph question answering and entity linking. Furthermore, related work to the previously described research questions is summarized.
- **Chapter 4** describes an approach for LLM-based contextual augmentation in entity linking, as well as the training of joint autoregressive models for NER and ED tasks. The results show that contextual augmentation can improve entity linking performance, especially on out-of-domain datasets. We further demonstrate that a jointly trained model can achieve performance comparable to end-to-end models. Finally, the results show that an LLM-based augmentation strategy performs better when NER and ED steps are separated.
- **Chapter 5** presents our results on noisy optimization of encoder-based LMs for the ranking and retrieval tasks, which are helpful for candidate generation of entity linking systems. The results show that noisy optimization can improve the performance of fine-tuned neural ranking approaches.
- **Chapter 6** presents our approach for present and absent keyphrase extraction based on LLMs and knowledge graphs. The results show that LLMs can effectively predict keyphrases and that linking those keywords to KGs to extract additional information can improve performance in absent keyphrase extraction.
- **Chapter 7** presents our approach for joint query prediction on multiple KGs. Our UniQ-Gen approach combines the training data for different KGs and trains a joint model for both graphs. Furthermore, it also introduces the output from entity and relation linking systems into the training data. Our experimental results indicate that presenting this information can improve the robustness of the query generation process. The results also show that a joint fine-tuned model performs as well as specialized models for single KGs.
- **Chapter 8** presents our baseline end-to-end KGQA system based on isomorphic graph patterns named TeBaQA. It applies a template-based approach for query generation, using supervised template classification, a rule-based method for filling templates with KG resources, and an algorithm to rank candidate queries. TeBaQA can partly outperform traditional rule-based approaches. In the context of this thesis, it serves as a baseline model for identifying weaknesses in traditional systems.
- **Chapter 9** investigates the influence of different entity and relation linking systems on the performance of KGQA systems. The results show that neural LLM-based approaches achieve better performance on entity and relation linking for questions and, ultimately, for KGQA systems.

- **Chapter 10** evaluates how queries can enhance the performance of answer verbalization during fine-tuning of LLMs, which is an application of semantic parsing-based KGQA systems. The results show that introducing entities and relations into the LLM input can improve the performance of answer verbalization approaches.
- **Chapter 11** summarizes the main results of the thesis and discusses future research directions.

1.4 Publications

The chapters 4 to 10 are mainly based on the following research papers published at international peer-reviewed conferences:

1. Daniel Vollmers, Rricha Jalota, Diego Moussallem, Hardik Topiwala, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck. 2021. Knowledge graph question answering using graph-pattern isomorphism, pages 103–117. IOS Press
2. Hamada M. Zahera, Daniel Vollmers, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. 2022. MultPAX: Keyphrase extraction using language models and knowledge graphs. In The Semantic Web – ISWC 2022, pages 303–318, Cham. Springer International Publishing
3. Daniel Vollmers, Parth Sharma, Hamada M. Zahera, and Axel-Cyrille Ngonga Ngomo. 2024. Enhancing answers verbalization using large language models. In Proceedings of the 20th International Conference on Semantic Systems, 17–19 September 2024, Amsterdam, The Netherlands. SEMANTICS 2024
4. Daniel Vollmers, Nikit Srivastava, Hamada M. Zahera, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2025b. UniQ-Gen: Unified query generation across multiple knowledge graphs. In Knowledge Engineering and Knowledge Management, pages 174–189, Cham. Springer Nature Switzerland
5. Daniel Vollmers, Hamada Zahera, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2025c. Contextual augmentation for entity linking using large language models. In Proceedings of the 31st International Conference on Computational Linguistics, pages 8535–8545, Abu Dhabi, UAE. Association for Computational Linguistics

6. Daniel Vollmers, René Speck, Hamada M. Zahera, and Axel-Cyrille Ngonga Ngomo. 2025a. Evaluation of entity and relation linking for question answering over knowledge graphs. In Proceedings of the 13th Knowledge Capture Conference 2025, K-CAP '25, pages 211–214, New York, NY, USA. Association for Computing Machinery

7. Daniel Vollmers, Arnab Sharma, and Axel-Cyrille Ngonga Ngomo. 2026. Evaluating noisy optimization in finetuning lms for neural ranking. In NLDB 2026. NLDB 2026

During his PhD studies, the author contributed to the following publications not included in this thesis:

1. Svetlana Pestryakova, Daniel Vollmers, Mohamed Ahmed Sherif, Stefan Heindorf, Muhammad Saleem, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2022. CovidPubGraph: A fair knowledge graph of covid-19 publications. Scientific Data

2. Rricha Jalota, Daniel Vollmers, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2021. LAUREN - knowledge graph summarization for question answering. In 2021 IEEE 15th International Conference on Semantic Computing (ICSC), pages 221–226

3. Ria Hari Gusmita, Rricha Jalota, Daniel Vollmers, Jan Reineke, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck. 2019. QUANT - question answering benchmark curator. In Semantic Systems. The Power of AI and Knowledge Graphs, pages 343–358, Cham. Springer International Publishing

4. Umair Qudus, Michael Röder, Daniel Vollmers, and Axel-Cyrille Ngonga Ngomo. 2024. ExPrompt: Augmenting prompts using examples as modern baseline for stance classification. In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM '24, pages 3994–3999, New York, NY, USA. Association for Computing Machinery

5. Arnab Sharma, Daniel Vollmers, and Axel-Cyrille Ngonga Ngomo. 2025. Calibrating language models for neural ranking under noisy supervision with relaxed labels. In Proceedings of the 2nd Workshop on Uncertainty-Aware NLP (UncertaiNLP 2025), pages 259–272, Suzhou, China. Association for Computational Linguistics

2 Background

This chapter discusses the most important prerequisites relevant for this thesis from the research fields of NLP and knowledge graphs. In the first step, basic NLP tasks are introduced as a foundation for more complex tasks, such as entity linking. Afterward, we provide an introduction to neural networks and LLMs, which build the backbone of most current state-of-the-art NLP approaches. Next, we introduce knowledge graphs and according KG-querying approaches, since KGs serve as the data model for all the methods described in this thesis. This includes the definition of KGs and the SPARQL query language. Additionally, we describe core tasks for knowledge graph question answering, including information retrieval, entity linking, keyphrase extraction, and semantic parsing. Finally, relevant evaluation measures are summarized.

2.1 Basic Natural Language Processing Tasks

Natural language processing is the task of learning, understanding, and producing human language content by employing computational technologies (Hirschberg and Manning, 2015). Initially, researchers employed symbolic approaches by developing vocabularies and rules that can be leveraged by computers. However, this rule-based paradigm proved to be a challenging task due to the ambiguity and context-dependent interpretation of human language. Additionally, increases in computing power and the availability of large linguistic corpora shifted NLP toward statistical modeling (Hirschberg and Manning, 2015). The NLP research field covers a wide range of tasks that involve different units of human language. For instance, text-to-speech aims to synthesize natural speech from text (Tan et al., 2021) and is an essential component in modern speech assistants. Other tasks are purely text-based, such as text summarization, automatic machine translation, and question answering. Many NLP tasks serve as subtasks for larger systems (Khurana et al., 2023). For instance, a speech assistant can use a question answering system to answer user questions, leveraging a text-to-speech system to process the QA system's answers and generate audio output. However, the QA system itself will use different subtasks, such as information retrieval or entity linking. We will describe the question answering task in Section 2.4.3 and the subtask entity linking and retrieval in the Sections 2.4.1 and 2.4.2, which are the primary focus of

this thesis. In the following, we focus on basic NLP tasks and techniques that form the foundation for the higher-level NLP tasks addressed in this thesis.

Tokenization To process natural language texts, they are commonly split into smaller units called tokens. A widely used technique segments text based on whitespace and punctuation, yielding an ordered sequence of tokens referred to as *unigrams*. This process, known as whitespace tokenization, retains the original textual order. In some NLP applications, these unigrams can also be combined into larger units consisting of n unigrams, denoted as *n-grams*. In particular, text similarity measures like BLEU (Section 2.5) use n-grams to better capture the context of word usages (Papineni et al., 2002).

Machine learning approaches typically require a finite input vocabulary to segment the input into processable units (Sennrich et al., 2016). This vocabulary can be obtained from a training corpus by applying a whitespace tokenizer to each text in the corpus and generating the input vocabulary V_{in} from all generated tokens. Afterward, a unique ID is assigned to each token of V_{in}. This approach has two significant drawbacks: First, at inference time, sequences may contain tokens that are not part of the vocabulary. This can be handled by mapping these words to special *unknown* tokens. Second, for a large corpus, this can result in a vast vocabulary, especially when a large training corpus is used. To mitigate these problems, tokenization approaches were developed that apply a subword segmentation algorithm. This method, called byte-pair encoding (BPE) (Sennrich et al., 2016), breaks words into smaller subword units. The vocabulary is constructed by iteratively merging frequent character pairs from the corpus until the desired vocabulary size is reached (Zouhar et al., 2023). This method alleviates the out-of-vocabulary issue, as rare words can be segmented into individual characters. A related method is the WordPiece algorithm used in BERT (Devlin et al., 2019). The approach is similar to BPE, but it applies a different strategy to select a pair for merging. Instead of merging the most frequent pair, it prefers pairs where the individual parts are less frequent in the vocabulary. For this, a score is computed by dividing the frequency of a pair by the product of the frequencies of its components.

Part-of-Speech Tagging Part-of-speech tagging (PoS) is the task of assigning each word in a sequence to one unique grammatical category (Machado and Ruiz, 2024). The most commonly set of PoS tags for English are defined by the Penn Treebank (Taylor et al., 2003). PoS tagging lays the foundation for many feature engineering approaches, as it is enables to derive statistical features from an input text, such as the number of nouns or verbs, or to identify numerical information. PoS tagging can be solved by sequence-to-sequence machine learning approaches on a tokenized sentence as described in Section 2.2 (Kanakaraddi

and Nandyal, 2018). In this thesis, PoS tagging is used to extract features from questions for query-template predictions in Chapter 8.

Lemmatizing Lemmatization is the process of determining a base or dictionary form (lemma) for a given surface form (Kanerva et al., 2019). For example, the verbs *am*, *are*, and *is* would be mapped to the word *be*. Lemmatization normalizes morphological variants by mapping different grammatical forms of a word to a single dictionary headword (Kanis and Skorkovská, 2010). This technique can help obtain more precise word usage statistics, for example, in a traditional information retrieval system. A related task is stemming, which reduces words to their base form (Samir and Lahbib, 2018).

Due to the ambiguity of languages, the lemmas or stems of words depend on their context, so that, in addition to rule-based approaches that rely on linguistic resources such as WordNet (Miller, 1994), sequence-to-sequence approaches can also be used for lemmatization (Kanerva et al., 2019). In the context of this thesis, lemmatization is used alongside PoS tagging to extract features for predicting query templates.

Text Classification Tasks Text classification assigns labels to various textual units, such as sentences or documents, and powers diverse applications like spam detection and text analysis (Minaee et al., 2021). More specifically, for an input text S, the goal is to predict a single class $c \in C$, where C is the set of classes. To address this task, deep-learning-based neural network approaches, as described in Section 2.2.1, can be used to predict a probability for each class $c \in C$ for an input sequence S. In our work, text classification is used to select query templates for input questions within Chapter 8.

Sequence-to-Sequence Tasks NLP tasks that require predicting an output sequence, like the PoS-tagging task mentioned before, are usually tackled by sequence-to-sequence machine learning models (Sutskever et al., 2014). These models transform an input sequence, such as the words within a natural language text, into another output sequence. More specifically, a function F is learned for an input sequence S consisting of n tokens $(t_1, ..., t_n) \in V_{in}$ that generates a new sequence S' of m tokens $(t'_0, ..., t'_m) \in V_{out}$, where V_{in} and V_{out} are the input and output vocabularies, respectively. For instance, in the PoS task, V_{out} is the set of PoS tags, and the input and target sequences have the same length. For other tasks, such as text summarization, V_{out} can be the same as V_{in}, but the target sequences likely have a different length than the input.

Text and Word Representation Most machine learning approaches require text to be mapped into a continuous vector representation. An early approach for this is one hot encoding, which generates an encoding vector for each token in the sequence of the form $X_i \in \mathbb{R}^{|V_{in}|}$ and each word in V_{in} is represented by one unique vector, such that the according entry of a word i is set to 1, all others are set to 0 (Naseem et al., 2020). As an alternative a bag-of-words model (BoW) generates a single encoding vector of size $|V_{in}|$, where the entries are the number of occurrences of the word in the input sequence (Verdonck et al., 2021). This encoding represents an unordered collection of words. It does not capture the word order of sequences, so it is mainly used for text classification. Another approach in this context is term frequency-inverted document frequency (TF-IDF), which is often used in information retrieval systems. We will further discuss information retrieval in Section 2.4.1.

Both models have two major drawbacks: their outputs are large, sparse vectors with many zeros, and the vector representations do not capture the semantic and syntactic meaning of words (Naseem et al., 2020). To address these issues, Mikolov et al. (2013) introduced learned word representations, known as Word-2-Vec, to capture the syntactic and semantic meaning of words based on their context of use.

There are two variants of this model: continuous bag of words (CBOW) and skip-gram. Continuous bag of words tries to predict a main word based on its context, while skip-gram tries to predict the context words based on the main word. Both models apply shallow neural networks to learn vector representations, such that words with similar contexts appear close to each other in the vector space (Verdonck et al., 2021). In the literature, there also exist extensions of these models, such as FastText (Bojanowski et al., 2017), which uses subwords instead of embedding single words. Relying on subwords reduces training time because the vocabulary is smaller. Another extension is GloVe, which leverages co-occurrence statistics from the words in the corpus to compute embedding vectors (Pennington et al., 2014).

2.2 Language Models and Neural Networks

Machine learning approaches build the backbone of most current NLP systems. Early approaches used sequence-to-sequence machine learning models such as hidden Markov models (HMMs) (Rabiner and Juang, 1986) or conditional random fields (CRFs) (Lafferty et al., 2001) to process sequences; furthermore, classification approaches such as random forests or decision trees were commonly used for text classification tasks (Islam et al., 2019). Today, neural networks are used for the majority of current NLP systems. In the following, we introduce basic neural network architectures: feed-forward networks (FFNs)

and recurrent neural networks (RNNs). FFNs are also part of Transformer architecture as presented in Section 2.2.4. RNNs, especially encoder-decoder networks, lay the foundation and central intuition for the development of Transformer-based models; thus, we provide a brief description of RNNs and the encoder-decoder architecture in this section.

2.2.1 Neural Networks

The neural networks considered in this thesis consist of a set of layers of neurons. The first layer is usually called the input layer, and the last layer is called the output layer. The layers between these two are hidden layers (Schmidhuber, 2014). A simple yet important type of neural network is the feed-forward neural network, where the information flows in only one direction (Schmidhuber, 2014). An input layer X_i with n neurons can be represented as a

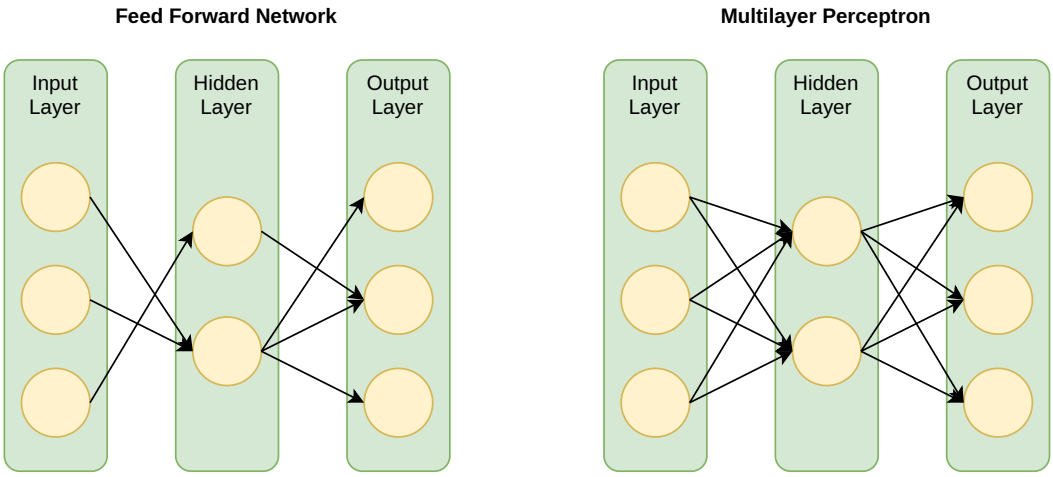


Figure 2.1: Basic Neural Network Architectures

vector such as $X_i \in \mathbb{R}^n$. In a hidden layer, the following computation is applied to compute the output state h_{hidden}:

$$h_{hidden} = \phi(X_i W_{i,d} + b_d)\,. \tag{2.1}$$

Where $W_{i,d} \in \mathbb{R}^{i \times d}$ is a weight matrix with i inputs from the previous layer and d own neurons, $b_d \in \mathbb{R}^d$ is a bias vector, and an activation ϕ is a function as presented below. This computation generates the output vector h_{hidden}, which is applied in follow-up layers. Note that this computation assumes that layers are fully connected; in that case, the feed-forward network is often called a multilayer perceptron. For sparse connectivity, unused connections can be disabled by setting corresponding weights in $W_{i,d}$ to zero. The last layer is usually denoted as the output layer h_{out}. These basic form of neural networks is described in Figure 2.1.

The activation function ϕ determines the output of a neuron by applying a non-linear transformation to its weighted input. When no activation is applied, this is usually called a linear projection, which is often used to change the dimensionality of vectors.

An activation function should fulfill multiple properties to enable learning in neural networks, such as differentiability, low computational complexity, and non-linearity (Dubey et al., 2021). The following functions are common choices for activation:

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = 1 - \frac{2}{e^{2x}+1}, \tag{2.2}$$

$$sigmoid(x) = \frac{1}{1+e^{x}}, \tag{2.3}$$

$$ReLu(x) = max(0,x), \tag{2.4}$$

$$GELU(x) = x \cdot \frac{1}{2} + erf\left[x/\sqrt{2}\right] \approx 0.5x(1+Tanh(\sqrt{2/\pi}\cdot(x+0.044715x^3))), \tag{2.5}$$

$$GeGLU(x) = x \cdot sigmoid(x) + GELU(x), \tag{2.6}$$

$$GLU(x) = sigmoid(xW+b) \otimes (xV+c), \tag{2.7}$$

$$Swish_\beta(x) = x \cdot sigmoid(\beta x) = \frac{x}{1+e^{-\beta x}}, \tag{2.8}$$

$$SwiGlu(x) = Swish(xW+b) \otimes (xV+c). \tag{2.9}$$

Tanh, sigmoid, and ReLu (Nair and Hinton, 2010) are relatively simple functions that are built upon by more complex functions such as GLU or GeGLU (Dauphin et al., 2017; Shazeer, 2020). Notably, ReLU is not differentiable at 0. In practice, the derivative at 0 is conventionally set to either 0 or 1, which does not significantly impair training. GeLu (Hendrycks and Gimpel, 2023) uses the Gaussian error function (erf), which is computed with the approximation presented above. More complex functions, such as GeGlu, are combinations of usually two error functions. The functions GLU and SwiGlu also contain learnable parameters (W, V, b, and c). Finally, SwiGlu (Shazeer, 2020) is a variation of GLU that applies the Swish function (Ramachandran et al., 2017) instead of sigmoid. Swish contains the parameter β, which can be fixed or trainable. For further details on activation functions, we refer to the literature (Dubey et al., 2021).

During the training phase, the neural network's parameters must be updated. For this, the difference between the predicted and expected outputs of the model has to be estimated. These functions are commonly denoted as loss functions. The choice of the loss function is crucial for model performance and highly depends on the task the model is trained on. A loss function must fulfill different properties such as convexity, differentiability, and

robustness. The standard loss function for the majority of tasks solved within this thesis is the categorical cross-entropy loss $\mathscr{L}_{CE}$ (Mao et al., 2023), which is computed as:

$$\mathscr{L}_{CE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C} y_{i,j} log(\hat{p})\,. \tag{2.10}$$

Here n is the number of samples within a batch, C the number of classes, $y_{i,j} \in 0,1^C$. $\sum_j y_{i,j}$ are the one hot representations of the correct classes, and $p_{i,j}$ is the predicted probability distribution for the sample i over the classes C. The objective of this loss function is to heavily penalize low-confidence predictions for the correct class. The parameter optimization is executed by gradient-based algorithms, which usually apply variations of the stochastic gradient descent (SGD) algorithm (Robbins, 1951) that updates a parameter θ based on a pre-defined learning rate α and the gradient g_t, which is computed by back-propagation (Rumelhart et al., 1986), at time step t, such as:

$$g_t = \nabla_\theta Loss(\theta_{t-1}), \text{ with} \tag{2.11}$$

$$\theta_t = \theta_{t-1} - \alpha g_t\,. \tag{2.12}$$

Modern optimization algorithms still set up on SGD. The most commonly used variant is the Adam optimizer (Kingma and Ba, 2015), which enables the computation of individual learning rates for different parameters.

2.2.2 Recurrent Neural Networks

As described before, a sequence-to-sequence model predicts an output for each single element of the input sequence based on its vector representations X_i. To implement a neural network for sequences, the prediction of a single token usually depends on the predictions of surrounding tokens. Simple FFN model cannot capture these dependencies (Morteza-pour Shiri et al., 2024). To solve this, recurrent neural networks RNNs played a crucial role for early neural sequence-to-sequence models. These models represent the context C_i of the word s_i as a hidden state H_i. For computing the hidden state H_i, it leverages the hidden state H_{i-1} calculated from the previous word s_{i-1} and the word s_i itself as illustrated in Figure 2.2. Formally, given a weight matrix $W_{d,h} \in \mathbb{R}^{d\times h}$ and a hidden-state-to-hidden-state matrix $W_{h,h} \in \mathbb{R}^{h\times h}$, where h denotes the number of hidden units and d the number of inputs. For an input vector $X_i \in \mathbb{R}^d$, a hidden state H_i is computed as:

$$H_i = \phi(X_t W_{x,h} + H_{t-1} W_{h,h} + b_h)\,, \tag{2.13}$$

where ϕ is an activation function and $b_h \in \mathbb{R}^{1\times h}$ a bias parameter (Schmidt, 2019). The output $O_t \in \mathbb{R}^{|V_{out}|}$ for a word is computed as:

$$O_t = \phi(H_t W_{h,o} + b_o), \tag{2.14}$$

where $W_{h,o} \in \mathbb{R}^{d\times|V_{out}|}$ (Schmidt, 2019). Using this form of neural networks, the hidden

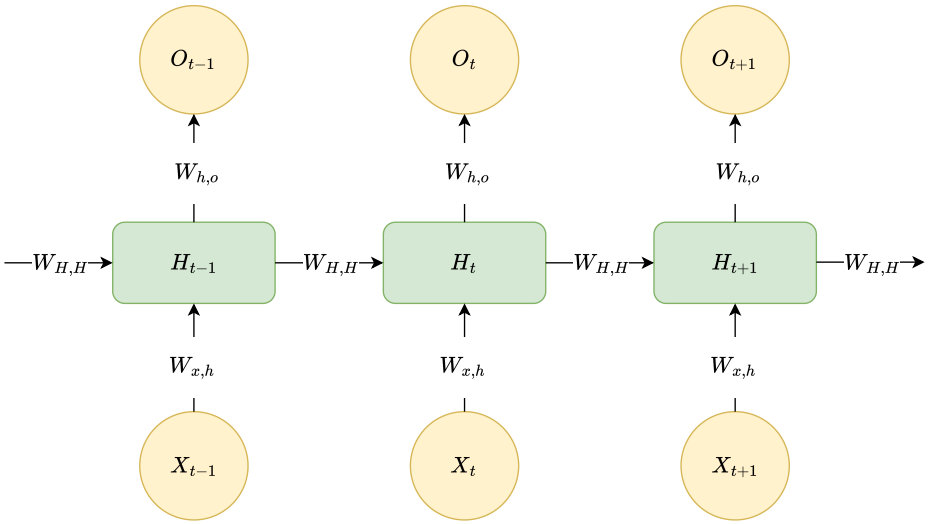


Figure 2.2: Recurrent Neural Network

state H_t is recursively updated by the previous words. This vanilla implementation of RNNs has two main drawbacks: First, the hidden states are computed only based on the earlier words, while later words in the sequence are not considered. This can be easily solved by computing the hidden states bi-directionally from both sides of the sequence. Second, training RNNs is difficult in practice, since at training time for very long sequences and a vast set of layers, gradients of parameters can get extremely small or large, which leads to the problem that layers in the beginning of the networks only get minor updates (in case of small gradients) or the optimization is unstable (in case of large gradients). This problem is commonly known as the vanished ore exploding gradient problem (Bengio et al., 1994; Pascanu et al., 2012). At inference time, this leads to the problem that the recursively computed hidden states lose the information from earlier words within the sequence. To partly solve this problem, LSTMs became popular, which use gated cells to control the flow of information within the model (Hochreiter and Schmidhuber, 1997; Schmidt, 2019).

2.2.3 Encoder-Decoder Architecture

The above-described implementation of RNNs computes an output for each single word; thus, only output sequences with the same size as the input sequence can be predicted. This

makes them impractical for tasks like machine translation or text summarization, where the length of the output sequence can be different from the length of the input sequence. To solve this, the encoder-decoder architecture became important (Cho et al., 2014). This architecture consists of two components: an encoder and a decoder, where the encoder compresses the input into a latent representation, and the decoder reconstructs or generates the target sequence. In the classical architecture, an RNN is used, but unlike the classical architecture, it only computes the hidden states $H_0, ..., H_n$ without predicting outputs. In the following, we denote the encoder's last hidden state as HE_n. The decoder applies HE_n as input for the first decoder state HD_0, and computes the output token by token. For this, two special tokens for the start and end of the output sequence have to be introduced. We describe these tokens as $t-start$ and $t-end$, where $t-start$ is used as input for HD_0, such that HD_0 can be computed as:

$$HD_0 = \phi(X_{t-start}W_{x,h} + HE_nW_{h,h} + b_h), \tag{2.15}$$

where $X_{t-start}$ is the embedding of $t-start$. Afterward, $O_{t-start}$ can be computed. The follow-up decoding steps use the outputs of the previous decoding steps, HD_{t-1} and O_{t-1}, as inputs. This process is repeated until the model predicts the token $t-end$. Obviously, this architecture is heavily affected by the vanished and exploding gradient problem for long sequences, since, in addition to the encoder component, which uses n recurrences like the classical sequence-to-sequence model, an additional decoder is used, which even introduces more layers and recurrences (Bahdanau et al., 2014). Furthermore, compressing the entire input sequence into a single vector HE_n creates an information bottleneck, particularly problematic for long sequences. To address this, the attention mechanism was developed, enabling the decoder to attend to all hidden states computed during the encoding process (Bahdanau et al., 2014). Attention also underpins state-of-the-art Transformer models, which form the foundation for most of the approaches used in this thesis. We describe the Transformer architecture, including attention and common adaptations, in the follow-up section.

2.2.4 Large Language Models

The Transformer architecture is the foundation of all current state-of-the-art language models. The architecture from Vaswani et al. (2017) is an encoder-decoder model. Unlike previous approaches, it does not use a recurrent neural network but relies entirely on attention. The architecture is presented in Figure 2.3.

The encoder is a stack of layers, where each layer consists of two sub-layers. The first sub-layer contains the attention mechanism, commonly called self-attention, since the

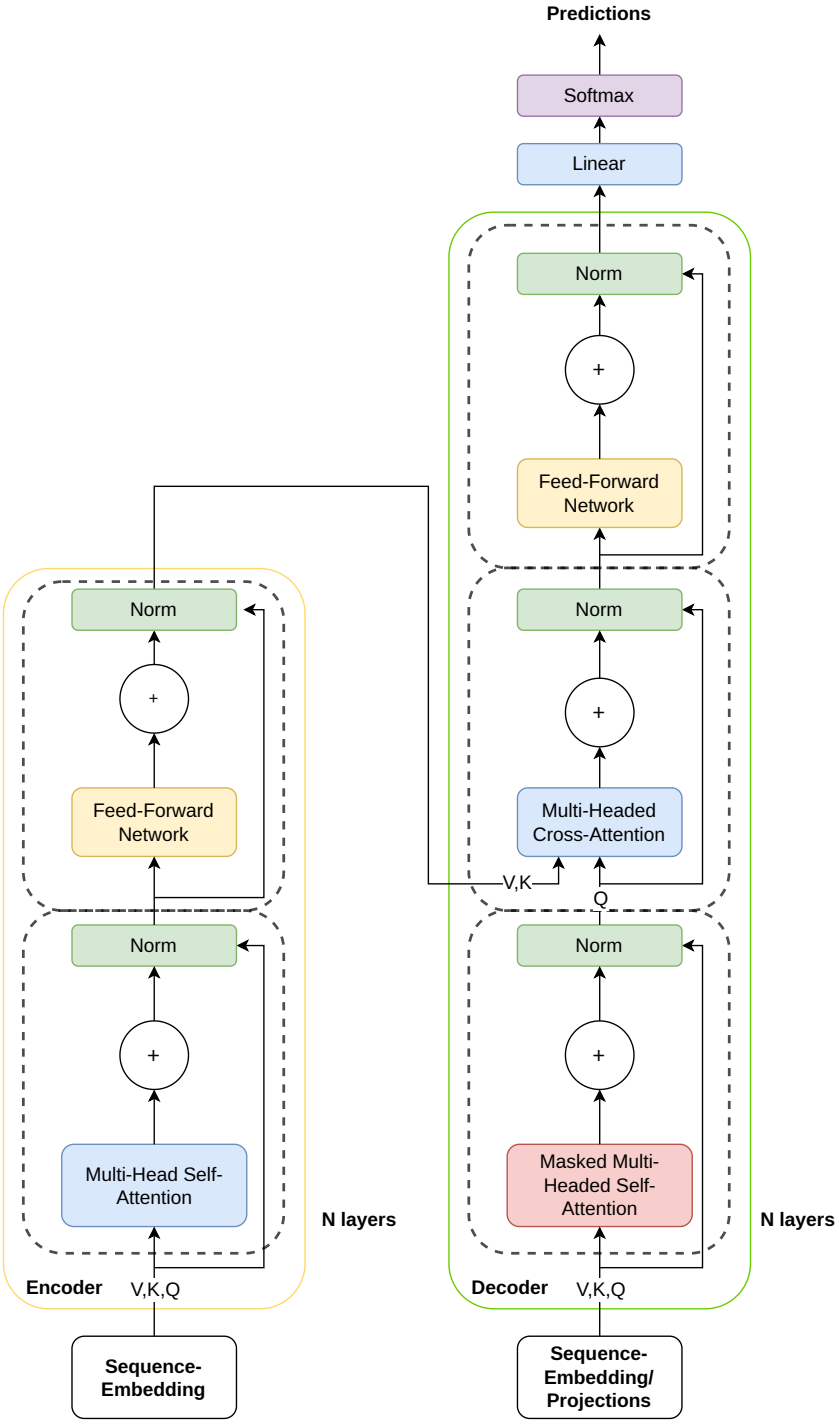


Figure 2.3: The Transformer architecture based on Vaswani et al. (2017).

attention scores are computed over a single sequence. The second layer is a position-wise fully connected feed-forward layer, which applies two linear transformations and uses ReLu

activation. Additionally, a residual connection is used to bypass the inputs around both sub-layers.

The decoder layer additionally contains a third sub-layer that computes attention over the encoder's output, usually denoted as cross-attention. Additionally, masking is applied in the self-attention step and the outputs are offset by one position. The offset and the masking ensure that predictions for a position i only depend on the known outputs at positions less than i (Vaswani et al., 2017). The FFN network is already introduced in Section 2.2.1. Finally, to predict outputs, a linear projection is applied together with the softmax function to predict the output vector $o \in \mathbb{R}^{|V_{out}|}$. For a vector $v \in \mathbb{R}^K$ softmax is defined as:

$$softmax(v)_j = \frac{e^{v_j}}{\sum_{k=1}^{K} e^{v_k}}. \tag{2.16}$$

The two major innovations of Transformer are the multi-head attention mechanism and the introduction of positional encodings instead of recurrences. These components are presented in the follow-up paragraphs. Furthermore, we will describe commonly used variations of these components.

Attention The core part of Transformer is the attention mechanism presented in Figure 2.4 The inputs are three matrices that are usually called **Q**ery, **K**ey, and **V**alue. It applies dot-product attention with the scaling factor $\sqrt{d_k}$, where d_k denotes the dimensionality of queries and values. The masking is optional and only applied in the decoder as described before. The overall attention can be described by the following equation (Vaswani et al., 2017):

$$\text{Attention}(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{2.17}$$

Instead of applying a single attention function, Transformer uses multiple heads, which allow the model to attend to information from different subspaces at different positions (Figure 2.4). To achieve this, the queries, keys, and values are linearly projected h times by parameter matrices $W \in \mathbb{R}^{d_{model} \times d_k}$. Where h describes the number of heads. Afterward, the scaled dot-product attention is applied to all these projections. The outputs from all heads are concatenated and finally projected again with $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ (Vaswani et al., 2017), which can be formalized as follows:

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O, \tag{2.18}$$

$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V). \tag{2.19}$$

For a memory-efficient implementation, Dao et al. (2022) describe Flash Attention, which distributes queries, keys, and values into blocks such that the attention output can be

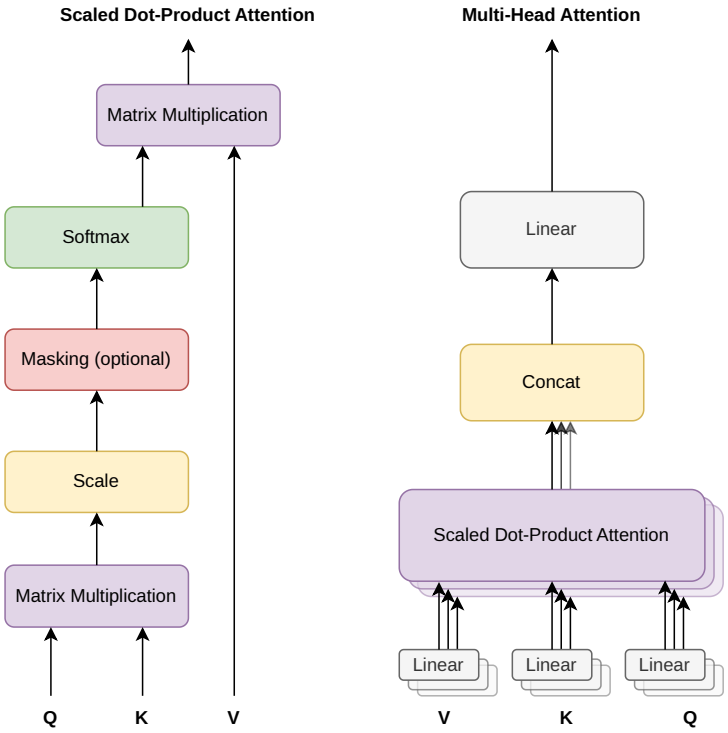


Figure 2.4: Transformer Attention based on Vaswani et al. (2017).

computed over each of these blocks instead of the whole attention matrix. This computation prevents the need to write and read large matrices between the high-bandwidth RAM and the static volatile SRAM of the GPU. The intuition behind this method requires a deeper understanding of computer hardware; thus, we refer the reader to the literature (Dao et al., 2022).

In recent years, many variations of the attention mechanism have been implemented. In the following, we will summarize a few of them.

- **Approximate Attention based on Locality Sensitive Hashing (LSH-Attention):**

  Since Softmax is dominated by the largest elements, Kitaev et al. (2020) suggest that it is sufficient to compute the attention for each query $q_i \in Q$ only with the closest keys in K. For this they propose to apply locality sensitive hashing, so that queries and keys can be hashed into hash-buckets, such that close vectors share the same bucket with a high probability and that the buckets have a similar size. The attention for a single query q_i can be computed as:

$$o_i = \sum_{j \in P_i} exp(q_i k_j - z(i, P_i)) v_j \text{ with } P_i = \{j : i > j\}, \tag{2.20}$$

where P_i is the set that the query q at position i attends and z denotes the partition function. Since the buckets should have similar sizes, it is ensured that the hash-function $h(q_j) = h(k_j)$ holds. Additionally, queries are sorted by bucket number and within each bucket by position. After sorting, the queries are batched into chunks, with m consecutive queries, and the attention for a query is computed against each of the other queries in the chunk and one chunk before (Kitaev et al., 2020).

- **Sliding Window Attention:** Sliding window attention employs a fixed-sized window of attention of size w around each token. Attention is computed to $0.5w$ tokens on both sides (Beltagy et al., 2020). This attention mechanism exploits the stacked-layer architecture of transformer models, such that in each layer k, tokens from the input layer can be accessed with a maximum distance of $w \cdot n$ (Jiang et al., 2023). To give an example: with window size $w = 256$ and 8 layers, each token can theoretically access information from tokens with a maximum distance of $256 \cdot 8 = 2048$ positions. This architecture reduces the quadratic complexity of the attention function to $O(n \times w)$, where n is the length of the input sequence. To balance between efficiency and representation performance, different values for w can be used for each layer.

  It is further possible to delate the sliding window, such that w contains gaps of size d. Additionally, some parts of the model, such as the first token of the sequence, which is often used for sequence classification (Devlin et al., 2019), can be set to full attention, such that it attends to all tokens in the sequence and vice versa (Beltagy et al., 2020).

- **Multi-Query Attention and Grouped-Query Attention:** Each query head uses linear projections for queries, keys, and values. Unlike in multi-query attention, only one projection is used for queries and values (Shazeer, 2019). In a more general form, the queries are distributed equally into g groups, which share a single key and value head. Overall, this technique can reduce the memory bandwidth of the attention function, while the overall performance and speed can be balanced over the number of groups (Ainslie et al., 2023). Furthermore, Ainslie et al. (2023) have shown that it is possible to uptrain a multi-head attention model to a multi-query attention model by applying mean pooling on the key and value heads.

Positional Encodings Since Transformer does not use recurrence, the order of the positions of a token has to be embedded to represent the order of the tokens in the input sequence (Vaswani et al., 2017). The original transformer model applies the sine and cosine

functions with different frequencies for fixed positional encodings (Vaswani et al., 2017) as:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \text{ and} \tag{2.21}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}), \tag{2.22}$$

where pos denotes the position and i the dimension in the encoding vector. Sinusoids with different wavelengths represent the dimensions in the position encoding vector. Sine and Cosine functions are used alternately for even and odd dimensions. The wavelengths form a geometric progression, such that for a fixed offset k the encoding of the position PE_{pos+k} can be represented as a linear function of PE_{pos}. To generate the sequential input, the token embedding is concatenated with the corresponding positional embedding for the first transformer layer (Vaswani et al., 2017).

With the evolution of LLMs, relative positional embeddings also became popular (Raffel et al., 2020), which consider the pairwise relationships between input tokens (Shaw et al., 2018). In the following, we adapt the notations of Su et al. (2024b). The computation of query vector of the m^{th} position q_m and the key and value vectors of the n^{th} position k_n and v_n in the original transformer can be described as:

$$q_m = f_q(x_m, m), \tag{2.23}$$

$$k_n = f_k(x_n, n), \tag{2.24}$$

$$\text{and } v_n = f_v(x_n, n), \tag{2.25}$$

where x_i is the i^{th} token embedding, and m and n the positional embedding for the m^{th} and n^{th} position. f_q, f_k, and f_v are the corresponding linear layers of the attention head implemented by weight matrices W_q, W_k, and W_v. Given that, the attention score $a_{m,n}$ between the m^{th} and n^{th} tokens (including the softmax function) and the attention output o_m for the m^{th} token are computed as:

$$a_{m,n} = \frac{exp(\frac{q_m^T k_n}{\sqrt{d}})}{\sum_{j=1}^{N} exp(\frac{q_m^T k_j}{\sqrt{d}})} \text{ and} \tag{2.26}$$

$$o_m = \sum_{n=1}^{N} a_{m,n} v_n . \tag{2.27}$$

The term $q_m^T k_n$ can be decomposed into:

$$q_m^T k_n = x_m^T W_q^T W_k x_n + x_m^T W_q^T W_k p_n + p_m^T W_q^T W_k x_n + p_m^T W_q^T W_k p_n , \tag{2.28}$$

with absolute positional encodings p_m and p_n. In the literature, there exist different options to replace the absolute positional encoding by modifying the last three terms of the previous equation, e.g, by replacing p_n by a relative encoding p'_{m-n} and introducing learnable vectors (Su et al., 2024b). In this thesis, we apply the T5 model (Raffel et al., 2020) for fine-tuning, which omits the second and third term, and introduce different projection matrices U_q and U_k and a trainable bias $b_{i,j}$ (Su et al., 2024b):

$$q_m^T k_n = x_m^T W_q^T W_k x_n + p_m^T U_q^T U_k p_n + b_{i,j}\,. \tag{2.29}$$

However, with the development of larger LLMs and the introduction of chatbots, efficient memory management within the inference phase became an important topic. To address this, Kwon et al. (2023) developed PageAttention, which partitions the keys and values into separate KV blocks, requiring that the encoding of a token does not change while generating new tokens, which does not hold for relative positional encodings. For this, Su et al. (2024b) introduced rotary positional encodings, which combine absolute and relative positional encoding by a rotation matrix and incorporate the explicit relative position dependency in the self-attention formulation. Su et al. (2024b) define the rotary matrix for d dimensions and a position m as:

$$R^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}. \tag{2.30}$$

The parameters Θ are set to $\{\theta_i = 10000^{-2(i-d)/d_{model}}\}$ with $i \in [1, ..., d_{model}/2]$, which is similar to the absolute positional encodings. The term $q_m^T k_n$ can be calculated as:

$$q_m^T k_n = (R^d_{\Theta,m} W_q x_m)^T (R^d_{\Theta,n} W_k x_n) = x^T W_q R^d_{\Theta,n-m} W_k x_n\,, \tag{2.31}$$

with $R^d_{\Theta,n-m} = (R^d_{\Theta,m})^T R^d_{\Theta,n}$ (Su et al., 2024b).

2.2.5 LLM Implementations

In recent years, a huge set of (large) LMs has been developed, which all apply the transformer architecture. Unlike the original paper, many state-of-the-art LLMs use only the encoder or the decoder of the original transformer model. A transformer-based encoder is usually used to generate text-based embeddings, which can, for example, be used to calculate text similarity (Reimers and Gurevych, 2019). In contrast, decoder models like GPT (Radford and Narasimhan, 2018) have become popular for generative AI applications such as chatbots and question answering. In the following, we present the LLMs used throughout this thesis.

BERT (Devlin et al., 2019) BERT is an encoder-only language model pretrained on two tasks. The first task is masked language modeling (MLM), where the model learns to predict masked tokens from a bidirectional context. During training, 15% of the tokens are replaced by a special *[Mask]* token, and the goal is to predict the correct tokens for these masked tokens. The other task is next-sentence prediction, in which the model predicts the next sentence from a set of candidate sentences for an input sentence. For the input representation, an additional segment embedding is introduced, enabling splitting input texts into multiple segments, such as sentences. The segments are separated by a special *[SEP]* token. For the input representation, the token, segment, and position embeddings are concatenated. BERT also prepends a *[CLS]* token at the beginning of each sequence, which is used to aggregate sequential representations for the next sentence prediction task. The pretraining on the next sentence prediction task makes it useful for retrieval and ranking applications, as described in Section 2.4.1.

Text-to-Text Transfer Transformer (T5) (Raffel et al., 2020) T5 is an encoder-decoder language model, commonly used as a fine-tuned model for multiple downstream sequence-to-sequence NLP tasks, such as question answering, sentence completion, and sentiment analysis. The pretraining dataset is generated from Common Crawl data[1]. T5 applies some modifications compared to the original transformer model. Instead of absolute positional encoding, a simplified version of relative positional encodings is used, as described in the previous section. Additionally, in layer normalization, activations are only rescaled, and an additive bias is added, and the layer normalization is placed outside the residual path.

BART (Lewis et al., 2020a) BART is an encoder-decoder LM which can be fine-tuned for various downstream tasks such as sequence classification, token classification, sequence

[1] https://commoncrawl.org/

generation, and machine translation. Similar to BERT, it is pretrained on token masking, but also applies other pretraining tasks, in particular, token deletion, text infilling, sentence permutation, and document rotation. BART mainly follows the original Transformer architecture, but uses GeLU instead of ReLU activation. Two different versions of the model were trained. The smaller model uses six layers in the encoder and decoder, while the larger model uses 12 layers in both.

Generative Pretrained Transformer (GPT) GPT is a family of LLMs developed by OpenAI[2] since 2018. All GPT models are decoder-only models. Released between 2018 and 2024, the GPT family has grown from 117M parameters (GPT-1) to over 1 trillion parameters (GPT-4). The vocabulary is generated using byte-pair encoding. Compared to the original transformer architecture, the GELU activation function is applied. From GPT-2 onward, trained positional encodings were introduced. The size of the different models and the maximum context windows were increased with each new generation of GPT. GPT-1 (Radford and Narasimhan, 2018) and GPT-2 (Radford et al., 2019) were pretrained on the language modeling task, such that the model can be fine-tuned to specific tasks afterward. With GPT-3 (Brown et al., 2020), in-context learning was introduced where the models learns to solve a task by learning from a small set of example prompts. GPT-3.5 is an improved version of GPT, which applies reinforcement learning from human preferences, so that the model can be refined based on human feedback. GPT-4 (OpenAI et al., 2024) enabled the processing of images, and the context size was increased. In GPT-5[3], the context window was extended, and the reasoning and code-writing capabilities were improved.

LlaMA (Touvron et al., 2023) Similar to GPT, LlaMA is a decoder-based generative LLM developed by Meta. Unlike GPT, it is trained only on publicly available data. In its architecture, normalization is computed only on the input of each transformer sub-layer, not on its outputs. It applies the SwigGLU activation function instead of ReLU and uses rotary positional embeddings instead of absolute positional embeddings. For tokenization, byte-pair encoding is applied. Meanwhile, three versions of LlaMA were developed, with model sizes ranging from 7 B to 64 B parameters.

Gemma (Team et al., 2024) Gemma is a family of lightweight models developed by Google, which are also used to create the Gemini models. Gemma offers models ranging from 2B to 27B parameters, designed for efficiency on consumer hardware. For small-scale models, multi-query attention is applied. Similar to LlaMA, rotary positional encodings

[2] https://openai.com/de-DE/
[3] https://openai.com/de-DE/gpt-5/

are used. The activation function is the GeGLU activation function. The newest version of Gemma[4]can handle up to 128k context tokens.

2.3 Knowledge Graphs and Querying

A Knowledge Graph (KG) can be defined as a graph of data intended to accumulate and convey knowledge of the real world (Hogan et al., 2021). Its nodes represent entities of interest and edges represent relations between these entities (Hogan et al., 2021). In this context, a data graph conforms to a graph-based data model. In practice, various graph data models are used, including directed edge-label graphs, where edges are directed and labeled; heterogeneous graphs, where the source and target nodes of an edge can have different types; and property graphs, which allow property-value pairs, where target nodes can be labels. Furthermore, multiple graphs can be combined into one graph dataset. The standard data model for KGs is the Resource Description Framework (RDF)[5]. RDF provides a standardized framework for knowledge representation, enabling interoperability across different KG systems. It defines three different types of nodes: IRIs, literals, and blank nodes (Hogan et al., 2021). IRIs must follow the definition of Internationalized Resource Identifiers.[6] Literals are values such as strings and numbers that are not represented by IRIs. Literals consist of a lexical form in Unicode and a datatype represented as an IRI. A String can have an additional language tag. For instance, the string *"Albert Einstein"@en* has lexical form *"Albert Einstein"*, datatype *"xsd:string"*, and language tag *"en"*. Blank nodes are resources that are not further defined by an IRI. They have only a local scope within a file or an RDF graph. IRIs, blank nodes, and literals are pairwise disjoint sets.[7]

In an RDF Graph, statements about resources are described in the form of RDF triples in the form <*subject*> <*predicate*> <*object*>. For example, in Figure 2.5, the statement <*Leonardo da Vinci*> <*notable work*> <*Mona Lisa*> describes the relationship between the two nodes *"Leonardo da Vinci"* and *"Mona Lisa"*. More formally, let R denote the set of IRIs, B the set of blank nodes, and L the set of literals. A knowledge graph consists of a set of triples $G = \{(s,p,o) \subseteq (R \cup B) \times R \times (R \cup B \cup L)\}$.

[4] https://deepmind.google/models/gemma/gemma-3/
[5] https://www.w3.org/TR/rdf11-concepts/
[6] https://www.ietf.org/rfc/rfc3987.txt
[7] https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

The resources R_G, properties P_G literals L_G and blank nodes B_G within a KG G can be described by the following sets:

$$R_G = \{s|(s,p,o) \in G \text{ and } s \in R\} \cup \{o|(s,p,o) \in G \text{ and } o \in R\}, \tag{2.32}$$

$$P_G = \{p|(s,p,o) \in G \text{ and } p \in R\}, \tag{2.33}$$

$$B_G = \{s|(s,p,o) \in B \text{ and } s \in B\} \cup \{o|(s,p,o) \in G \text{ and } o \in B\}, \tag{2.34}$$

$$L_G = \{o|(s,p,o) \in G \text{ and } o \in L\}. \tag{2.35}$$

The nodes in a graph G are usually organized into groups called classes, and nodes within the groups are called instances of this class. The set of classes C_G in G is a subset of the resources R_G within a graph: $C_G \subseteq R_G$. C_G and P_G are usually described by a vocabulary. For this, the RDF Schema provides a basic vocabulary for modeling RDF data.

Table 2.1: RDF Schema

Class/Property	Identifier	Description
Class	rdfs:Resource	Superclass of all resources R_G
Class	rdfs:Class	Class of all Classes in C_G
Class	rdfs:Literal	Class of all Literals L_G
Class	rdfs:Datatype	Class of all datatypes for the Literals in L_G
Class	rdf:Property	Class of all properties P_G
Property	rdfs:range	Objects targeted with a property are instance of one or more classes
Property	rdfs:domain	Subjects using this property are instance of one or more classes
Property	rdfs:label	Human-readable version of a resource name
Property	rdf:type	A resource is an instance of a class

We summarize some essential classes and relations in table 2.1. An ontology often defines the vocabulary used within a graph. To this end, the Web Ontology Language provides the vocabulary for building these ontologies. RDF graphs can be encoded with the N-triples[8] or the Turtle[9] text format or as XML documents. The goal of KGQA is to answer questions on RDF Graphs. For example, on the graph in Figure 2.5, the question *"Where was Leonardo da Vinci educated?"* can be answered by traversing the directed edge *"educated at"*. In a semantic parsing KGQA system, this can be achieved by translating a given text question into a formal query. To this end, a query language is needed that can be executed on a graph database, such as a triple store. For KGs, SPARQL is the most widely used query language. In the following, we provide a brief description of the most relevant features used in KGQA benchmarks for semantic parsing.

[8] https://www.w3.org/TR/n-triples/
[9] https://www.w3.org/TR/turtle/

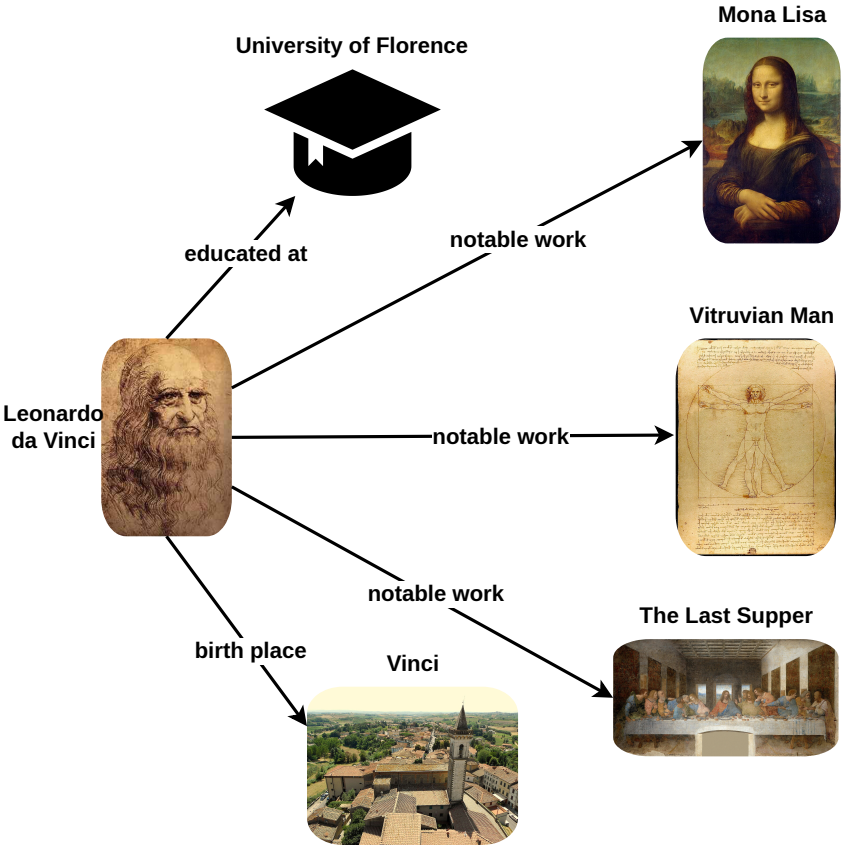


Figure 2.5: Knowledge Graph

2.3.1 SPARQL

The SPARQL query language is designed to query data, stored in RDF graphs.[10] Thus, it supports querying over conjunctions and disjunctions of graph patterns, including aggregations and subqueries. The result set is a table, where each column corresponds to a variable defined in the *SELECT* clause of the query. Alternatively, an *ASK* query produces a boolean result. In the following, we describe essential characteristics of the SPARQL query language.

A basic graph pattern contains triple patterns, which are the same as RDF triples, except that they can contain variables for the subject, predicate, and object. A query with a single triple pattern can be formulated as follows:

```
1  SELECT ?result WHERE {
2    <https://dbpedia.org/resource/Germany>
3    <http://dbpedia.org/ontology/currency> ?result.
4  }
```

[10]The descriptions in this section are based on the following web document: https://www.w3.org/TR/sparql11-query/

In this query, the *SELECT* clause describes the variables to be displayed in the results, and the *WHERE* clause contains the graph pattern to match against the graph data. The term *?result* defines a variable. IRIs start with the character *"<"* and end with *">"*. The object of a triple in RDF can be a literal, a URI, or a blank node. Consequently, a SPARQL query can also contain literals (we omit the description of blank node queries here, as these are not used in the KGQA benchmarks selected for our experiments). To shorten URIs in the graph pattern, prefixes can be defined. The usage of literals and prefixes is presented in the following example:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?result WHERE {
  ?result  rdfs:label "John Doe"@en.
}
```

Instead of a single basic graph pattern, a graph pattern can also contain multiple groups of graph patterns delimited by braces (*{group_1},{group_2}*), enabling the introduction of optional and alternative patterns. An optional pattern is left-associative and allows adding information to the result set if it is available in the graph.

```
SELECT ?person ?date WHERE {
  ?person rdf:type dbo:Person:
  OPTIONAL{
    ?person dbo:deathDate ?date
  }
}
```

Alternatives can be used, to combine multiple graph patterns, E.g in the query below all persons, who are born in Austria or Germany would match the triple pattern:

```
SELECT ?person WHERE {
  {?person dbo:birthPlace dbo:Germany}
  UNION
  {?person dbo:birthPlace dbo:Austria}
}
```

Filters can be used to restrict the results of graph patterns. Filters always apply for the whole group, where they are defined. Filters can also be defined as graph patterns, using the terms *EXISTS* and *NOT EXISTS*. For instance, the following query contains two filters to query people who were born after 1989 and are still alive:

```
SELECT ?person WHERE {
  ?person dbo:birthDate ?date
  FILTER (?date = "1989-01-01"^^xsd:date)
  FILTER NOT EXISTS {?person dbo:deathDate ?death}
}
```

SPARQL also allows the usage of property paths, which describe the path between two nodes in a graph. For example, the query:

```
?SELECT ?person, ?country WHERE{
  ?person dbo:birthPlace/dbo:country ?country
}
```

applies a path sequence with two hops to extract a person's country of birth. Other path expressions allow describing alternative paths "|" or matching paths multiple times, like "$*$" for zero or more and "$+$" for one or more matches of a path. Paths are not allowed to contain variables. Another useful feature is aggregation, for example *COUNT*, *MIN*, *MAX*, and *AVG*. Aggregates can be calculated for the entire result set, or results can be grouped into one or more groups so that the aggregation function is computed for each group separately. Aggregations can also be used in the *SELECT* part of the query; for example, to count results. Furthermore, the *HAVING* keyword can be used to apply filters on aggregations. As an example, the following query finds persons with more than two grandchildren:

```
SELECT DISTINCT ?parent WHERE {
  ?parent dbo:child ?child1.
  ?child1 dbo:child ?child2.
}
GROUP BY ?parent
HAVING ( COUNT(?child2) > 2 )
```

The SPARQL Algebra defines the semantics of the SPARQL query execution. An algebraic expression is formed by parsing the SPARQL string into a syntax tree. The following two listings show a SPARQL query with the corresponding algebraic expression in the ARQ algebra, which is used by the Apache Jena RDF framework.[11]

```
SPARQL-Query
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>

SELECT DISTINCT  ?result
WHERE
{
  wd:Q696071  wdt:P50  ?author .
  ?author   wdt:P800  ?result .
  ?result   wdt:P921  wd:Q16003532
  FILTER ( ?result != wd:Q69071 )
}
```

```
ARQ-Algebra
(base <http://example/base/>
(prefix ((wd: <http://www.wikidata.org/entity/>)
```

[11] https://jena.apache.org/documentation/query/algebra.html

```
    (wdt: <http://www.wikidata.org/prop/direct/>))
    (distinct
    (project (?result)
    (filter (!= ?result wd:Q69071)
    (bgp
    (triple wd:Q696071 wdt:P50 ?author)
    (triple ?author wdt:P800 ?result)
    (triple ?result wdt:P921 wd:Q16003532)
    ))))))
```

Execution of SPARQL Queries Query engines that execute SPARQL queries are usually called triple stores. Popular implementations are, for example, Virtuoso[12], Fuseki[13] and the tensor-based triple store Tentris[14] (Bigerl et al., 2020). In addition, also popular RDF frameworks such as Apache Jena[15] and RDFLib[16] are capable of indexing RDF graphs and executing SPARQL queries. These SPARQL engines first translate a query into an algebraic expression by mapping it to algebraic symbols. The translation process is carried out iteratively by replacing graph patterns with operators in the appropriate algebra. Finally, the algebraic expression is evaluated on a dataset, using the semantics of the introduced operators, to generate a result set. We omit these details here, since the goal of this thesis is only the generation of SPARQL query strings and refer to the according literature (Cyganiak, 2005; Pérez et al., 2009).[17]

2.3.2 Knowledge Graphs for KGQA

In this thesis, we apply the knowledge graphs Wikidata, DBpedia, and Freebase for our experiments on KGQA, which are presented in the upcoming paragraphs.

DBpedia DBpedia (Auer et al., 2007) is a knowledge graph that is automatically generated from Wikipedia[18], by leveraging extraction frameworks. In the early stages of the project, RDF data was extracted mainly from the structured elements of Wikipedia, such as infoboxes, and from links between DBpedia pages or to external web pages. Later, the DBpedia ontology was introduced, enabling the mapping of infobox terms to improve data quality. Furthermore, additional extractors were introduced, for example, to extract person data or

[12] https://virtuoso.openlinksw.com/
[13] https://jena.apache.org/documentation/fuseki2/
[14] https://tentris.io/
[15] https://jena.apache.org/
[16] https://github.com/RDFLib/rdflib
[17] https://www.w3.org/2001/sw/DataAccess/rq23/rq24-algebra.html
[18] https://en.wikipedia.org/wiki/Main_Page

to process disambiguation pages (Lehmann et al., 2015b). DBpedia is available in many languages. The data is provided as RDF dumps[19] and regular releases are published over the DBpedia Databus platform (Hofer et al., 2020). Furthermore, a public triple store is available, which can be accessed via SPARQL query language.[20]

Freebase Freebase (Bollacker et al., 2008) was a publicly available knowledge graph for storing human knowledge, launched by Metaweb in 2007 (later Metaweb was acquired by Google). Similar to Wikidata, it was collaboratively maintained. The Freebase project was stopped in 2016. However, the latest data is still available.[21] Despite discontinuation, Freebase remains important for benchmarking. Currently, the data is still used as a KG for benchmarking datasets like WebQuestions (Berant et al., 2013) and GrailQA (Gu et al., 2021) in the domain of KGQA.

Wikidata Wikidata (Vrandečić and Krötzsch, 2014) is an open KG which is operated by the Wikimedia Foundation.[22] Different from DBpedia, it is not automatically generated. Instead, the data and the schema are developed and controlled by the community, similar to Wikipedia. As of 2026, Wikidata contains around 120 million items and is actively maintained by thousands of contributors worldwide.[23] In the beginning, the primary purpose of Wikidata was to support Wikipedia's development, especially to enrich articles with data (Vrandečić and Krötzsch, 2014). In later years, Wikidata was also used to support other Wikimedia projects. Furthermore, after the Freebase project ended, the data was migrated into Wikidata (Vrandečić et al., 2023). Meanwhile, Wikidata is used as a knowledge source in a wide range of research domains, such as NLP, the Semantic Web, and machine learning.

2.4 Core Tasks in Knowledge Graph Question Answering

In this section, we describe core tasks for the KGQA problem, including ranking and retrieval, which are core techniques in many entity linking systems (Wu et al., 2020a; Zhang et al., 2022c). Afterward, we describe the entity linking and key phrase extraction tasks, the

[19] http://dev.dbpedia.org/Download_DBpedia
[20] https://dbpedia.org/sparql
[21] https://developers.google.com/freebase
[22] https://wikimediafoundation.org/
[23] https://www.wikidata.org/wiki/Wikidata:Statistics

main components used in this thesis to extract relevant knowledge from the KG. Finally, we describe semantic parsing as the main KGQA approach studied in this thesis.

2.4.1 Information Retrieval

Manning et al. (2008) defines information retrieval (IR) as the task of finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers). The input of a text-based IR system is a query consisting of a sequence of words $w_1, ...w_n$ and the goal is to extract a set of documents D' from a text corpus D, such that for each $d \in D'$ the probability $P(d \mid q, D)$ is maximized (Ramos, 2003).

Traditional Approaches In the literature, various approaches exist for solving this problem. An early but still popular approach is TF-IDF (term frequency-inverse document frequency) (Baeza-Yates and Ribeiro-Neto, 1999), which calculates the TF-IDF score for a word w and a document d as follows:

$$w_d = TF_{w,d} \cdot IDF_{w,d} = f_{w,d} \cdot log\left(\frac{|D|}{f_{w,D}}\right). \tag{2.36}$$

Here, $f_{w,d}$ is defined as the number of times a word w appears in the document d. The term $f_{w,D}$ describes the number how many documents of the corpus D contain the word w and $|D|$ describes the size of the corpus. This weighting scheme gives higher scores to terms that are frequent in a document but rare across the corpus. Finally, for a given query q consisting of words w_i, a document d can be scored by calculating the sum over all $w_{i,d}$ (Ramos, 2003):

$$score = \sum_i w_{i,d}\,. \tag{2.37}$$

In practice, there exist many variations of TF-IDF. A popular extension is the BM25 (Robertson et al., 1994) algorithm. BM25 improves upon TF-IDF by addressing two issues: (1) diminishing returns for term frequency (saturation), and (2) normalization over the document length. BM25 applies the following scoring function:

$$BM25 = \sum_w IDF_w \cdot \frac{f_{w,d} \cdot k_1 + 1}{f_{w,d} + k_1 \cdot (1 - b + b \cdot \frac{fieldLen}{avgFieldLen})}\,. \tag{2.38}$$

The expression $1 - b + b \cdot \frac{fieldLen}{avgFieldLen}$ calculates the relative length of a document compared to the average document length and applies the scaling parameter b. If the document length is larger than the average score, the final score for the document is decreased, if

it is smaller than the average length, the score is increased.[24] Additionally frequency saturation is applied, so that the an increase of $f_{w,d}$ only effects the score until a saturation is reached. This is controlled by the parameter k_1. Lower values for k_1 lead to a quicker saturation, higher values to a slower saturation.[25] In all these approaches, the input text has to be segmented into words. To this end, a whitespace tokenizer can be applied as already described in Section 2.1. A drawback of these approaches is that queries and documents are only compared at the word level, which is not always sufficient, as queries and documents can contain synonyms such as *skill* and *competence* that cannot be mapped directly. Additionally, it is possible that words in a query do not exactly match the terms in a document, even when there is a connection; for example, the terms *index* and *search-index* refer to the same concept. To solve this, often synonym dictionaries and weak matching approaches are applied.

Neural Approaches In recent years, neural approaches for information retrieval have become increasingly popular. They apply neural network architectures to represent queries and documents. Especially the introduction of encoder-based language models such as BERT (Devlin et al., 2019) led to the development of neural IR approaches that outperform traditional systems (Nogueira and Cho, 2019), as these approaches are capable of capturing the contexts of terms in queries and documents, enabling the handling of synonyms and ambiguous terms. LLM-based information retrieval is usually implemented as a two-stage

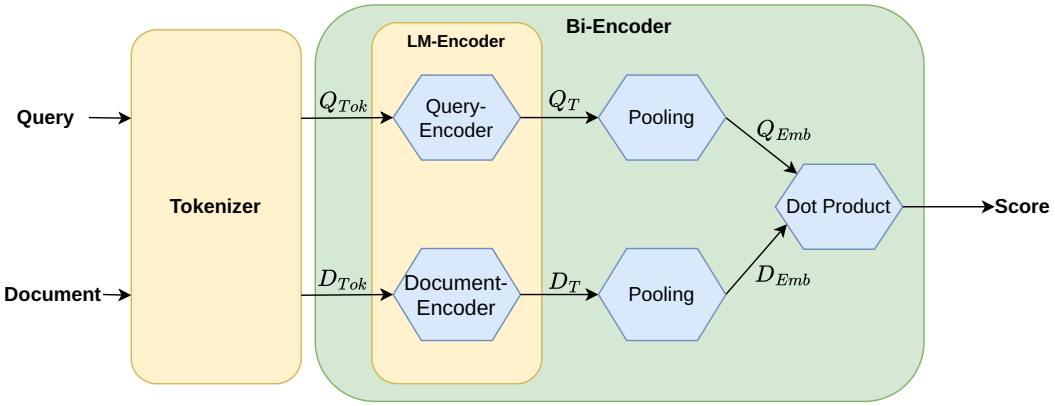


Figure 2.6: LM-based Bi-Encoder

approach: a retrieval step (also referred to as first-stage retrieval), followed by a re-ranking step (Guo et al., 2022). In the retrieval step, the task is to extract the top n documents from the entire document corpus D. In this context, n is usually set to 30, 100, or even 1.000

[24] https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables

[25] https://www.elastic.co/guide/en/elasticsearch/guide/current/pluggable-similarites.html

documents. This step is usually solved with a bi-encoder module (Peng et al., 2022). This bi-encoder generates an embedding for the query and document independently by applying an encoder-based LM. The workflow is described in Figure 2.6. For a given document d and a query q, q and d are tokenized. Afterward, the tokenized sequences Q_{tok} and D_{tok} are independently encoded by the LM using a *Query-Encoder* and a *Document-Encoder*. Both functions apply an LLM-based encoder model to generate sequence embeddings Q_T and D_T for all input tokens in Q_{Tok} and D_{tok}. Both functions can be the same, but can also append additional tokens, or perform a transformation on the input tensors before generating Q_T and D_T. More formal, for a an encoder model F and the tokenized query sequence Q_{Tok} of length l_q and D_{Tok} of length l_d, the sequence embeddings Q_T and D_T are computed as

$$Q_T = F(Q_{Tok}) \text{ and } D_T = F(D_{Tok})\,. \tag{2.39}$$

The results are sequence embeddings of size $l_q \times d_{model}$ for Q_T and $l_d \times d_{model}$ for D_T. For computing scores, a single-vector representation is required. For this, a pooling function is applied to generate the embeddings of Q_{Emb} and D_{Emb}. To represent the whole sequence, BERT's pooling mechanism is applied, which performs a linear projection on the `[CLS]` token embedding. (see Section 2.2.5). The query embedding Q_{Emb} and the document embedding is computed as:

$$Q_{Emb} = Q_{T_0} W \text{ and } D_{Emb} = D_{T_0} W\,, \tag{2.40}$$

where $W \in \mathbb{R}^{d_{model} \times d_{model}}$ of Q_T and D_T. Furthermore, additional dense layers can be included in the pooling step to compute additional features. The final score is obtained by computing the *dot Product* between Q_T and D_T

$$Score = Q_{Emb} \cdot D_{Emb}\,. \tag{2.41}$$

This architecture enables generating document embeddings offline. Thus, at inference time, only the query embeddings need to be computed, and documents can be retrieved using k-nearest neighbors (KNN) on the document embeddings computed offline (Peng et al., 2022).

In the second step, usually called re-ranking, the goal is to order the documents extracted by the retrieval step by assigning final scores. To this end, a cross-encoder architecture is applied.

Unlike the bi-encoder , the cross-encoder concatenates the input sequences from queries and documents. After tokenization, the joint tokens J_{Tok} are encoded by the *LM-Encoder*. Similar to the bi-encoder, a pooling step is applied to generate the embedding of the *[CLS]* token. The formal computation of J_{Emb} is the same as that of D_{Emb} and Q_{Emb} in the bi-

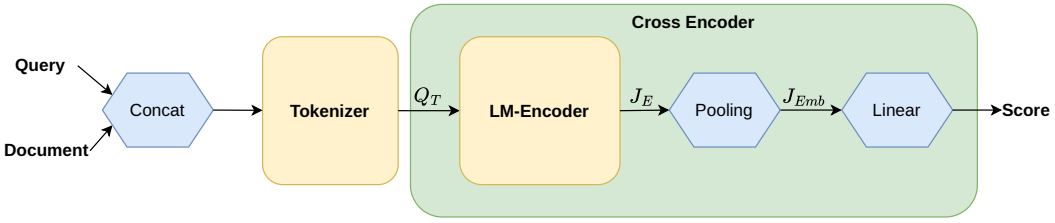


Figure 2.7: LM-based Cross-Encoder

encoder . Since only a single vector is computed, a different scoring mechanism is required. For this, a linear projection is used to obtain a scalar value as the final cross-encoder score:

$$Score = J_{Emb} W_s \,, \tag{2.42}$$

where $W_s \in \mathbb{R}^{d_{model} \times 1}$ is the linear layer as described in Figure 2.7.

In general, cross-encoders achieve a higher accuracy compared to bi-encoders at the cost of a higher complexity, as they have to execute forward passes for all query document pairs at inference time, while bi-encoders only have to compute embeddings for input queries, since document embeddings can be precomputed, reducing retrieval to a nearest-neighbor search (Chiu and Shinzato, 2022; Peng et al., 2022). Further details about the training process of the presented neural ranking models are presented in Chapter 5.

2.4.2 Entity Linking and Keyphrase Extraction

The entity linking task traditionally involves two main steps: (I) named entity recognition and (II) entity disambiguation, as described in the following sections. The goal is first to annotate entities such as persons, locations, and organizations in an input document and, in a second step, to link those entities to identifiers in a KG. The set of target entities E_G of classical entity linking systems is usually the set of resources R_G of a knowledge graph G without the set of classes C_G:

$$E_G = R_G \setminus C_G \,. \tag{2.43}$$

Named Entity Recognition (NER) Named entity recognition identifies and extracts entity mentions (e.g., people, organizations, locations) from text. Given a document $D = (t_1, t_2, \cdots, t_k)$, consisting of k tokens from the input vocabulary V_{in}, the goal is to identify a subset of tokens $M = m_1, m_2, \cdots, m_i$ representing entity mentions. The NER function maps D to a list of entity mention spans M (Sevgili et al., 2022):

$$\text{NER} : C \rightarrow M^n \,. \tag{2.44}$$

In practice, this problem is modeled as a sequence-to-sequence task as described in Section 2.1, meaning that for an input document D, a new sequence with the same length is predicted. The output vocabulary $V_{out} = \{B, I, O\}$ contains the tags from the BIO-tagging scheme, where B denotes that a token is the beginning of a named entity, I denotes that a token is inside an entity, and O that a token is outside of an entity (Keraghel et al., 2024). For example, for the token sequence *"North,Rine,-,Westphalia,is,a,state,in,Germany"* the sequence *"B,I,I,I,O,O,O,O,B"* is the target output sequence.

In the next step, the entity mentions can be extracted from the original sequence based on the predicted output sequence, such as [North, Rine, -, Westphalia] and [Germany]. Finally, for each mention, an entity type is predicted. Among others, common named entity types are `Location (LOC)`, `Person(PER)`, and `Organization(ORG)`. In our example, the type `Location (LOC)` would be annotated for both entity mentions. Popular supervised approaches for the NER task include traditional sequence-to-sequence machine learning models such as hidden Markov models (HMMs), conditional random fields (CRFs), and LSTMs. However, with the rise of Transformer-based LLMs, LLM-based approaches became the leading approach for NER (Keraghel et al., 2024).

Entity Disambiguation (ED) Entity disambiguation is the task of resolving any potential ambiguities of entity mentions extracted by a NER system, by linking each mention m_i to the correct entity e_i in a knowledge graph, using similarity measures and contextual information (Sevgili et al., 2022):

$$\text{ED} : [(m_1, \cdots, m_n)] \rightarrow (e_1, \cdots, e_n) \text{ with } (e_1, \cdots, e_n) \in E_G \,. \tag{2.45}$$

For example, the entity mention *"North Rhine-Westphalia"* can be mapped to the Wikidata entity `wd:Q1198` and the mention *Germany* to `wd:Q183`. This process is usually solved as a two-step approach, as described in Figure 2.8. In a first step, a set of candidate entity links is extracted from a retrieval index by applying IR. The retrieval index contains all entities represented in verbalized form, for instance, by composing documents from entity surface forms and textual descriptions. In a second step, a final disambiguation maps each candidate mention to one of the candidate entities. In practice, ED is a challenging task because each step depends on the output of previous ones; the process fails if the correct entity is missing from the candidate set or if the provided context is insufficient for disambiguation.

End-to-End Entity Linking Alternative approaches describe EL as an end-to-end task by omitting the NER step. This joint modeling avoids error propagation from the NER component to disambiguation. These frameworks are usually separated into a retrieval and a

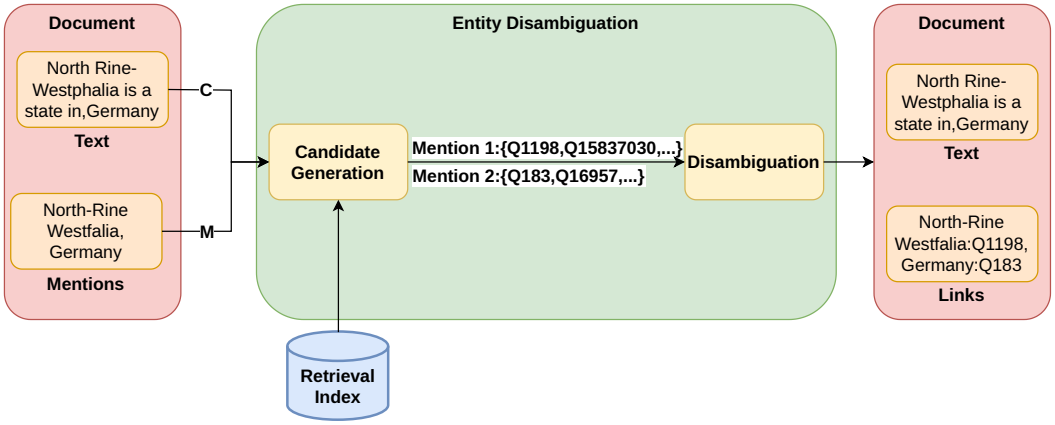


Figure 2.8: Entity Disambiguation

reading phase, such that the retrieval step extracts candidate entities directly from the input sequence. Afterward, the reader identifies mentions in the original input sequence for each candidate and applies re-ranking to obtain the final mention-entity mapping (Zhang et al., 2022c).

As an alternative, the reader can also apply a generative LLM, which takes the candidates and the original sequence as input and predicts a new sequence with annotated entities, including links (De Cao et al., 2021).

Keyphrase Extraction A related task to NER is keyphrase extraction, where the goal is to generate a set of keyphrases $K = \{k_1, k_2, ..., k_t\}$ that describe an input document D (Hasan and Ng, 2014). Unlike NER, keyphrase prediction is not limited to extracting phrases from the text; it also includes the generation of phrases that do not necessarily appear verbatim in the source text, which is often denoted as keyphrase generation. For instance, the abbreviation *"NRW"* is a valid keyword for the input text *"North Rhine-Westphalia is a state in Germany"*. Keyphrase extraction also plays a vital role in information retrieval, as keywords can serve as a summary of the text, helping retrieve documents by removing many filler words and unimportant information from the query. Keyphrase extraction is usually solved in two steps: in the first, a list of candidate keyphrases is generated, and in the second step, these candidate keyphrases are ranked using supervised or unsupervised ranking approaches (Hasan and Ng, 2014). In this thesis, we show that the disambiguation process of an EL system can also be applied to keyphrases rather than mentions to extract additional information. We will further discuss this in Chapter 6.

2.4.3 Semantic Parsing

A semantic parsing KGQA system answers natural language user questions on a KG by translating the input question into an executable query (Lan et al., 2021). Semantic parsing requires translating the input query into the target query language, e.g, SPARQL (Section 2.3.1). This translation is challenging because: (1) questions are often ambiguous, (2) the formal query language is complex, and (3) the mapping requires understanding both natural language and graph structure. The expressiveness of SPARQL enables answering various types of questions, such as factoid questions with a single answer, and those that require one or multiple hops through the KG. Other queries can extract one or more entities or literals from the KG that satisfy specific restrictions to answer questions such as *"Which Bundesliga teams are located in Bavaria?"*. Furthermore, certain queries are boolean questions that require a binary response, such as true or false. This can comprise one or multiple facts in the KG. Finally, aggregate queries enable answering questions that require counting, such as *"How many Oscars has Leonardo DiCaprio received?"*, as well as superlative queries, such as *"What is the highest building in Europe?"*.

In general, semantic parsing involves the four steps presented in Figure 2.9: Question processing, knowledge extraction, query generation, and answer generation, potentially followed by an optional verbalization step. The question processing step employs NLP techniques such as keyphrase extraction, named entity recognition, and answer type detection to the input question (Lan et al., 2021). The knowledge extraction step retrieves the relevant information from the KG based on the output from the query processing step. It includes entities, relations, and types as described in Section 2.4.4. Other options include extracting complete triples from the KG to generate a subgraph or retrieving triple patterns with the support of a triple store.

In the query generation step, one or multiple candidate queries are generated. These queries are executed on a triple store to retrieve answers. In cases of multiple queries, this also includes an answer-ranking step. For the query generation step, we will utilize two different approaches in this thesis. The first approach applies text classification to map queries to a set of predefined query templates and is presented in Chapter 8. The other approach is a sequence-to-sequence approach based on fine-tuned LLMs and is described in Chapter 7.

Within a QA system, an additional verbalization step can be applied to enhance a natural interaction with the user. We discuss verbalization approaches in detail in the Chapters 3 and 10.

As described in Figure 2.9, output from one of the steps is not only used in the follow-up processing step, but can also be forwarded to steps later in the pipeline. This forwarding is especially useful to provide further context for different processing steps.

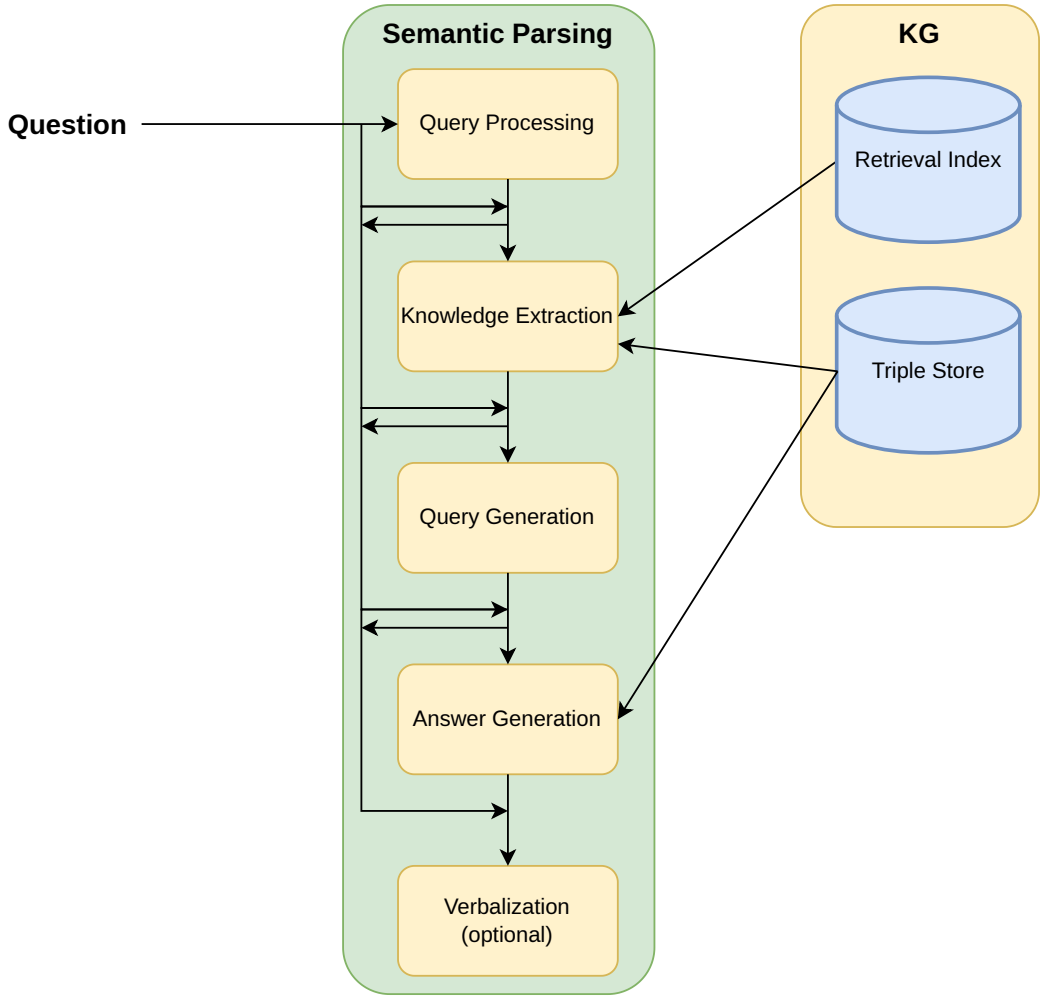


Figure 2.9: Knowledge Graph Question Answering Process

KG information is usually provided in two forms: In a triple store and as a retrieval index. The retrieval index is used to store literal information from the set L_G of a KG G, as introduced in Section 2.3. This enables the application of entity linking and retrieval techniques to map textual information from questions to G. The triple store is a graph database, as described in Section 2.3.1, used to execute the final queries, but can also be used to extract triples or query patterns during the knowledge extraction step.

2.4.4 Knowledge Extraction for Semantic Parsing

The primary focus of this thesis is the implementation of the query processing and the knowledge extraction step, and evaluating the influence of these two steps on the rest of the KGQA pipeline. Figure 2.10 shows a possible connection between the input question, the KG, and the expected output query for the example question *"Which other paintings*

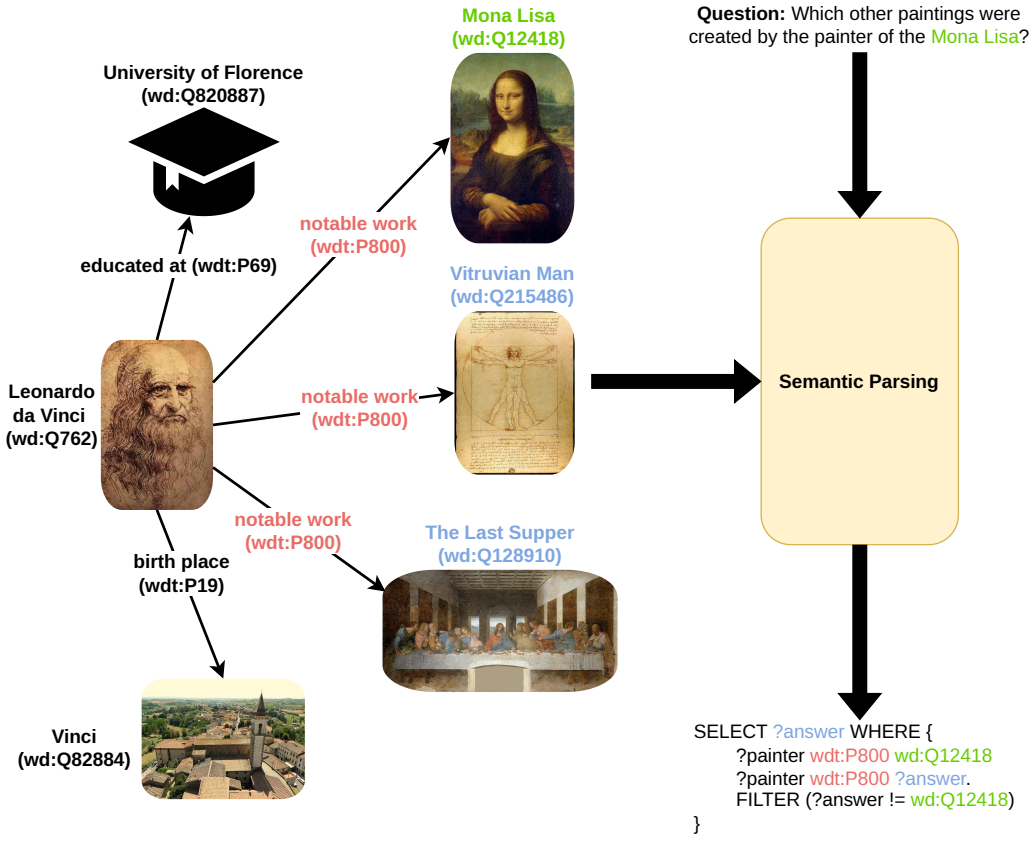


Figure 2.10: KGQA Challenge

were created by the painter of the Mona Lisa?". To answer this question, two hops have to be executed. Here, the entity *"Mona Lisa"* describes the query entity. The query entity is directly mentioned in the input question and can be extracted by an entity linking system. The edge *"notable work"* between the entities *"Mona Lisa"* and *"Leonardo Da Vinci"* describes the edge for the first hop represented by the first triple pattern in the query. The second hop applies the same edge to find the other *"notable works"* for *"Leonardo da Vinci"* as described by the second triple pattern. Finally, a filter is applied to ensure that the query entity *"Mona Lisa"* is excluded from the result set.

The small example already shows that the resources that have to be extracted from the KG are not always explicitly mentioned in the input question. This, and the fact that questions are usually relatively short texts with only a small amount of context, make knowledge extraction for semantic parsing a challenging task. Importantly, classical entity linking focuses only on entities mentioned explicitly in a text. However, KGQA requires linking to properties and types that are often implicit, posing additional challenges. The classical definition of entity linking is limited to entities mentioned explicitly in an input sequence, and the target entities of the disambiguation step are limited to the set E_g, which does not

include the set of classes C_G and properties P_G of a knowledge graph G as described in Section 2.3. Consequently, traditional EL-disambiguation approaches have to be updated to extract KG resources from the sets R_G for additional type extraction and from P_G for property extraction, which are usually denoted as relations in this context. This combined task is referred to as joint entity and relation linking (ERL) (Dubey et al., 2018). Additionally, we will adapt LLM-based keyphrase extraction methods to address the challenge of extracting implicitly mentioned KG resources and properties within the ERL task.

2.5 Evaluation Metrics

In this section, we present performance metrics used for evaluating entity linking, ranking, and question answering approaches. Furthermore, we present measurements for evaluating text similarity, which are important for the evaluation of answer verbalization approaches.

Precision, recall, and F1 (Powers, 2011) measures are commonly used for evaluating entity linking and question answering approaches. These measures are computed based on a two-class confusion matrix:

		True Values	
		Correct	False
Prediction	Correct	True Positive (TP)	False Positive (FP)
	False	False Negative(FN)	True Negative(TN)

The precision is defined as:

$$Precision = \frac{TP}{TP+FP} \tag{2.46}$$

and measures the fraction of all correctly predicted instances among all predicted instances. The recall is defined as:

$$Recall = \frac{TP}{TP+FN}$$

and measures the fraction of all correctly predicted instances among all correct instances. Finally, F1 measure is defined as the harmonic mean of the precision and recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{2.47}$$

F1 balances precision and recall, giving equal weight to both. It is particularly useful when classes are imbalanced.

In the context of entity linking, there exist two different ways to compute these measures. Firstly, in the micro setup, these measures are calculated across all entities in the dataset in a single step. In the macro setup, scores are calculated per document, and the final score is computed as the average of all individual document scores. Micro-averaging is preferred for imbalanced datasets where the overall performance is important. Macro-averaging is preferred when all instances should be treated equally, regardless of frequency. Furthermore, many papers report scores only for entities in the target KG (*in-wiki*). In this case, the scores are denoted as *In-KG* scores (Röder et al., 2018).

The definitions of macro- and micro-scores are similar across the KGQA tasks. Here, the instances used for scoring are the gold answers from the KGQA datasets and the predicted answers from the KGQA systems. For some datasets from the QALD Challenges[26], it is common to also report the F1 score specifically for those questions where the KGQA system generated an answer. This metric is referred to as the QALD F1 score (Usbeck et al., 2019).

For ranking algorithms, the task is to extract the top n documents from a text corpus. To this end, in addition to the previously described scores, the MRR score can be computed, which is defined as:

$$MRR = \frac{1}{d}\sum_{i=1}^{d}\frac{1}{r_i}, \tag{2.48}$$

where d is the set of queries and r_i is the rank of the first relevant document of the $i-th$ query (Schwartz, 2021). This measure focuses on the rank rather than the number of correctly or incorrectly extracted documents, as the rank plays a crucial role in information retrieval. MRR heavily weights top-ranked documents in the prediction. A top-ranked document contributes with a value of 1.0, while a document at position 10 only contributes with a value of 0.1.

In the context of automatic evaluation for text generation, generated outputs are compared against ground-truth references from a benchmark. To this end, we employ the BLEU and METEOR scores, which are described below.

Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002) : This metric measures the n-gram overlap between generated sentences and ground-truth references. The BLEU metric also includes a brevity penalty based on the lengths of the reference and generated sentences, penalizing when the generated text is shorter than the reference text. The brevity penalty is calculated over the whole corpus to avoid penalizing shorter sentences. The BLEU score ranges from 0 to 1, where 1 is the best score, indicating that the reference and generated text are identical.

[26] https://www.nliwod.org/challenge

Formally, for a sentence s and a set of reference sentences C, BLEU first calculates the modified n-gram precision, based on the set of matched n-grams R between s and all sentences in C:

$$p_n = \frac{\sum_{R \in C} \sum_{n \in R} Count_{clip}(n)}{\sum_{R \in C} \sum_{n \in R} Count(n)} . \tag{2.49}$$

Here, the $Count_{clip}()$ function counts an n-gram in the candidate sequence only as many times as it appears in the reference sequence. The brevity penalty is calculated based on the length of a candidate sentence l_s and a reference sentence l_r as:

$$BP = \begin{cases} 1, \text{ if } l_s > l_r \\ e^{1-l_r/l_s}, \ l_s \leq L_r \ . \end{cases} \tag{2.50}$$

Finally, BLEU is calculated as:

$$BLEU = BP \cdot exp\Big(\sum_{n=1}^{N} w_n \cdot log(p_n) \Big) , \tag{2.51}$$

with N denoting the maximum n-gram length of N and w_n as positive weights summing to 1 (Papineni et al., 2002).

Metric for Evaluation of Translation with Explicit Ordering (METEOR) The METEOR score was introduced by Banerjee and Lavie (2005) and evaluates the similarity between the hypothesis (i.e., generated text) and the reference text by considering chunks of text. Unlike other metrics, METEOR can incorporate semantic similarity, accounting for similar words rather than just exact matches. Different from BLEU, METEOR emphasizes recall over precision (9:1 ratio), making it more lenient toward verbosity but stricter on omissions. METEOR calculates the harmonic mean of precision and recall and counts exact word matches between the hypothesis and the reference. Additionally, it penalizes incorrect word order, as the order of words in a sentence is essential for grammatical correctness and meaning. In the first step, METEOR calculates an alignment between the unigrams in the candidate and the reference texts. This is done in multiple stages that apply different processing pipelines, starting with exact matches and then applying stemming and synonym identification. After alignment is complete, the unigram precision P and recall R are calculated. The harmonic mean between P and R is calculated as:

$$F_{mean} = \frac{10PR}{R + 9P} . \tag{2.52}$$

To take into account longer matches, the fewest possible number of chunks in the candidate sequence is calculated, and a penalty is computed. The penalty and final METEOR scores are computed as:

$$Penalty = 0.5 \cdot \left(\frac{\#chunks}{\#unigrams\_matched} \right)^3 \text{ and} \tag{2.53}$$

$$METEOR = F_{mean} \cdot (1 - Penalty) . \tag{2.54}$$

3 State of the Art

This chapter summarizes the related work in all research areas with respect to our research questions. The first part outlines current developments in knowledge extraction, providing the methodological basis for retrieving relevant information from a KG for question answering. This includes information retrieval, entity linking, and keyphrase extraction. In the second part, we describe the state of the art in question answering, primarily semantic parsing approaches. This includes traditional approaches and neural LLM-based approaches. Furthermore, we describe multi-KG query generation and verbalization approaches.

3.1 Knowledge Extraction

In this section, we describe the state of the art of the core technologies used in this thesis for extracting KG knowledge for question answering. These concepts were already introduced in section 2.4 and comprise the topics information retrieval, entity linking, and keyphrase extraction.

3.1.1 Information Retrieval and Noise-driven Optimization

The goal of information retrieval is to retrieve a set of relevant documents from a large corpus given an input query, as introduced in Section 2.4.1. Traditional retrieval systems rely on term-based retrieval approaches, such as TF-IDF and BM25. These techniques are also implemented in many popular search frameworks such as Apache Lucene (Bialecki et al., 2012) and Elasticsearch[1]. Traditional term-based retrieval techniques have drawbacks, such as polysemy, synonymy, and lexical gaps (Hambarde and Proença, 2023), limiting their performance. These limitations encouraged the development of neural approaches, which are the primary focus of this thesis. Within this section, we describe the state of the art in the area of neural ranking and retrieval. Furthermore, we summarize noise-driven optimization methods, since one goal of this thesis is to investigate, if these methods can improve the robustness of neural retrieval and ranking approaches as described in chapter 5.

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

Neural Ranking and Retrieval Recent advancements in the development of LLMs have significantly contributed to the area of information retrieval, which is commonly distributed into a retrieval and re-ranking stage (Hambarde and Proença, 2023) as already discussed within section 2.4.1. For instance, Reimers and Gurevych (2019) introduced Sentence-BERT, a model that generates semantically meaningful sentence embeddings using a Siamese network structure, enabling efficient semantic similarity comparisons and clustering. The effectiveness of fine-tuning BERT for passage re-ranking, achieving state-of-the-art results on the MS MARCO passage retrieval task, has been demonstrated by Nogueira and Cho (2019). Wang et al. (2022) proposed E5, a family of text embeddings trained through contrastive learning on a large-scale text pair dataset, achieving strong performance in tasks requiring single-vector text representations. More recently, Déjean et al. (2024) explored the comparative strengths of cross-encoder rerankers and large language models (LLMs), shedding light on trade-offs between efficiency and ranking effectiveness in different retrieval settings. Since the development of Large Language Models (LLMs) is currently dominated by decoder-based architectures like GPT, Ma et al. (2024) proposed a two-stage IR framework, fine-tuning bi-encoder and cross-encoder variants of LLaMA. Instead of using a prepended *[CLS]* token for the sequence representation, they use the embedding of the last token because LLaMA uses an unidirectional attention mechanism. Unlike classical indexing-based approaches, a new branch of approaches aims to train LLMs directly to predict identifiers (Chen et al., 2022; Tay et al., 2022), which requires representing document identifiers so they can be memorized by LLMs. For re-ranking, also unsupervised prompt-based approaches were developed (Zhu et al., 2025). For instance, Sun et al. (2023) prompts an LLM to rank documents given an input query.

These information retrieval methods also play a crucial role in entity linking. For instance, Zhang et al. (2022c) introduced EntQA, which reformulates entity linking as a question answering problem, combining dense retrieval and reading comprehension to enhance mention disambiguation. Similarly, Wu et al. (2020b) proposed a dense retrieval-based entity linking approach that improves ranking precision by leveraging powerful transformer-based encoders. While most of these works focus on incorporating sophisticated architectures to train more effective ranking models, considering the loss landscape to achieve better generalization is relatively underexplored.

Noise-driven Optimization. Adding noise to the gradient descent approach has been widely explored in the literature for its ability to enhance optimization in non-convex settings by introducing noise into the gradient descent process (Jin et al., 2017, 2021; Polyak and Tsybakov, 1990; Smith et al., 2020; Zhou et al., 2019). For instance, Smith et al. (2020) demonstrated that the noise inherent in SGD helps converge to minima with lower curvature, leading to better generalization. Similar findings were reported by Zhang et al. (2019a), who empirically demonstrated that the tendency of SGD to find flatter minima is inherently

linked to its stochastic nature. Moreover, the work of Bradley and Gomez-Uribe (2021) has established a connection between the level of noise in SGD (influenced by factors such as batch size and learning rate) and its ability to find flat minima, thereby improving the generalization of models. Jin et al. (2017) demonstrated that escaping saddle points can be achieved by discretizing Langevin dynamics (Fogedby, 1994), where noise contributes to diffusion. Zhou et al. (2019) further showed that parameter perturbations during optimization help escape spurious local minima in non-convex settings. Adding noise before gradient computation, initially used for smoothing non-smooth convex objectives (Nesterov and Spokoiny, 2017; Polyak and Tsybakov, 1990), has also shown promise in non-convex scenarios by biasing optimization toward flatter minima, which enhances generalization (Liu et al., 2021a; Orvieto et al., 2022). The link between flat minima and better generalization has been extensively highlighted by Keskar et al. (2017) and Chaudhari et al. (2019). However, traditional methods rely on uncorrelated perturbations, which may hinder convergence or result in suboptimal optimization paths. Anticorrelated noise has emerged as a promising alternative, further biasing optimization towards flatter minima (Orvieto et al., 2022).

Although a vast amount of work has been done exploring the usage of introducing noise to the gradient descent approach, either in parameters or in labels, most of the work has focused on vision-related tasks. This approach has been relatively less explored in the NLP domain.

3.1.2 Entity Linking

Entity linking typically involves two phases: named entity recognition and entity disambiguation, as described in Section 2.4.2. During the NER phase, many existing approaches use standard named entity recognition to identify relevant spans in text. In the disambiguation phase, candidate entities are generated from a knowledge graph and linked to their most appropriate matches, often using information retrieval approaches as described earlier. In the literature, many entity linking systems focus only on the disambiguation phase by relying on existing named entity spans, while others describe end-to-end pipelines. In the following, we describe state-of-the-art approaches from both of these research directions.

Disambiguation Approaches Traditional non-neural approaches, such as MAG (Moussallem et al., 2018), rely on pre-built indices for effective entity disambiguation. For example, MAG employs five specialized indices (*surface forms, personal names, rare references, acronyms*, and *contextual information*) to retrieve candidate entities. To refine the candidate set, MAG applies an additional rule-based filtering step. Finally, MAG generates a subgraph based on the retrieved candidates and applies the HITS (Kleinberg, 1999)

algorithm to compute the final entity annotations. In contrast, DoSeR (Zwicklbauer et al., 2016) incorporates text-based retrieval with surface forms, a Word2Vec embedding model, and a priori probabilities derived from occurrence frequencies. For candidate generation, it employs a similar candidate-expansion strategy to MAG. Other approaches, such as Mulang et al. (2020), introduce context information to the input sequence in the form of verbalized triples extracted from the knowledge graph. Meanwhile, Raiman and Raiman (2018) incorporates type information into the disambiguation process to improve accuracy. Moreover, BLINK (Wu et al., 2020a) adopts an embedding-centric approach to candidate generation, employing a bi-encoder to generate representations for both candidates and mentions. Entity search within the index is executed through KNN-Search based on context embedding vectors. Other methods, similar to BLINK, such as Lai et al. (2022), combine text-based retrieval with deep neural embedding models for entity disambiguation.

Alternatively, Parravicini et al. (2019) propose an embedding-based approach, where node embeddings, derived using the *word2vec* algorithm, assess vertex similarity. This method involves evaluating candidates using tuples, where each tuple associates a candidate entity with its corresponding mentions in the document. A global similarity score, calculated from these node embeddings, determines the score for each tuple. In contrast, some approaches use re-ranker models to compute embeddings, taking into account both the mention's context and candidate entities. These models employ a feedforward layer to re-rank candidates (Lai et al., 2022; Wu et al., 2020a). Lastly, Xin et al. (2024) introduced the first framework relying on LLM-based context augmentation for entity disambiguation, focusing solely on enhancing the context of entity mentions without addressing NER. This method, however, is computationally intensive because it requires augmenting each mention individually.

End-to-End Entity Linking Unlike the traditional two-step entity linking pipeline, recent approaches omit the NER step and directly extract or annotate entity candidates from the target KG. These approaches often employ fine-tuned autoregressive models (Zhang et al., 2022c). An early work from Kolitsas et al. (2018) integrates word and entity embeddings based on Word2Vec and and an LSTM (Hochreiter and Schmidhuber, 1997) to incorporate contextual information to predict mention-entity pairs. Furthermore, van Hulst et al. (2020) introduced an approach that predicts coherence scores to align entity annotations.

Different from these early methods, modern entity linking methods mainly apply fine-tuned LLMs for this task. For instance, the GENRE framework (De Cao et al., 2021) leverages a fine-tuned BART model for autoregressive annotation of input sequences, using an offline prefix trie derived from Wikipedia titles and applies constrained decoding to reduce the search space. In contrast, EntQA (Zhang et al., 2022c) uses a vector-based search index similar to BLINK to identify entities within the input sequence. During the disambiguation

phase, it computes candidate spans for each entity and selects the highest-scored candidate for linking. Our work presented in chapter 4 differs from these methods by addressing the entire entity linking pipeline, including NER. It introduces an LLM-based augmentation approach to improve entity disambiguation, particularly on out-of-domain datasets.

3.1.3 Keyphrase Extraction

In this thesis, we use unsupervised approaches to extract keyphrases and implicitly mentioned KG resources from questions, which can also be denoted as absent-keyphrase extraction. Consequently, we summarize the state of the art in these two research directions.

Unsupervised Keyphrase Extraction Several approaches have recently been developed for extracting keyphrases in an unsupervised setting without the need for annotated data. For example, statistical approaches such as TF-IDF and YAKE (Campos et al., 2020) compute statistical features (e.g., word frequencies and co-occurrences) to find meaningful words as candidates for present keyphrases. Moreover, graph-based approaches like TextRank (Mihalcea and Tarau, 2004) construct a graph representation of text, where words are represented as nodes and their co-occurrences as edges. Thereafter, a node ranking algorithm (e.g., PageRank) is used to sort words and return top-k words as candidate keyphrases. Bougouin et al. (2013) proposed TopicRank, a graph-based approach similar to TextRank. In the first step, candidate phrases are clustered into topics and then ranked based on their importance in the document.

Recent studies have demonstrated that embedding-based models can achieve significant results in extracting keyphrases. For example, the EmbedRank (Bennani-Smires et al., 2018) approach uses part-of-speech tags to extract potential keyphrases from an input document. Furthermore, EmbedRank uses a pretrained embedding model to represent both phrases and an input document as low-dimensional vectors. Candidate keyphrases are ranked based on their cosine similarity scores with the document's embedding vector. Although pretrained language models have shown promising performances for extracting present keyphrases, they have failed to generate absent keyphrases from their lexical corpus. Furthermore, Liang et al. (2021) noted that embedding-based models ignore local information within a document. Accordingly, they developed a jointly trained model to incorporate both global and local document context. In the global view, their approach represents candidate keyphrases and the input document as low-dimensional vectors in a single semantic space. After that, the similarity between each candidate keyphrase and the document is computed. In the local context, the authors built a graph structure based on the document context, where nodes

represent phrases and edges represent their similarities. Finally, the output keyphrases are ranked based on this global and local information.

Absent Keyphrase Extraction Many previous approaches rely on sequence-to-sequence models with an encoder-decoder architecture to generate absent keyphrases (Chen et al., 2019). By doing so, sequence-to-sequence models can decode not only keyphrases that appear in source text, but also those that may be absent, i.e. keyphrases that are not explicitly mentioned in the text. However, additional mechanisms need to be integrated to improve the generation of absent keyphrases. For example, Ye et al. (2021) applied a graph neural network (GNN) to capture knowledge from related references in scholarly publications. A neural topic model is employed by Wang et al. (2019) to expand the context of the decoding component to generate more absent keyphrases.

It is noteworthy that Zhao et al. (2021) achieved significant results in extracting keyphrases by dividing this task into two sub-tasks: Present keyphrase extraction and absent keyphrase generation. Furthermore, the authors propose a multitask approach to select, guide, and generate keyphrases. In the select module, the authors use a BiLSTM to predict whether a sentence has a keyphrase or not. Then, a guider network is employed to leverage attention information and memorize the selector's predictions. Finally, this information is fed to a generator network to generate absent keyphrases by selecting words from both the source text and a predefined vocabulary. In addition to these fully-supervised approaches, some unsupervised methods have achieved promising results in generating keyphrases without the need for labeled data. Shen et al. (2022) observed that many keyphrases absent from an input document appeared in other related documents. Therefore, they constructed a phrase bank of all keyphrases in a corpus. Then, they identified present keyphrases in relevant documents as candidates for absent keyphrases for the input document. In addition, they employed present keyphrases as silver labels to train a sequence-to-sequence model. Finally, all keyphrases (both present and absent) were ranked based on their lexical and semantic similarity to an input document.

3.2 Knowledge Graph Question Answering

The second part of our research focuses on the KGQA task. For KGQA there exist two different kind of approaches namely retrieval-based approaches and semantic parsing (Wu et al., 2019). Retrieval-based approaches first retrieve the relevant context in form of a subgraph (Ding et al., 2024) and try to answer input questions based on the extracted context. In contrast, semantic parsing approaches convert an input question into an executable query, which is then run against a triple store to retrieve the final answer. This thesis mainly

targets semantic parsing, thus we focus on query generation approaches within this section. Additionally, this thesis aims to compare classical and LLM-based KGQA approaches and to develop a method for generating queries across multiple knowledge graphs. As an application of semantic parsing KGQA, we also target the use case of answer verbalization, which can enable a more human-like interaction with users. Consequently, we summarize related works for all these research directions within this section.

3.2.1 Traditional Approaches

Traditional approaches often rely on rule-based processing pipelines to generate SPARQL queries. An early rule-based approach is gAnswer2 (Zou et al., 2014), which treats semantic parsing as a subgraph-matching problem. Also following the rule-based paradigm, QAnswer (Diefenbach et al., 2020) utilizes the semantics embedded in the underlying KG (DBpedia and Wikidata) and employs a combinatorial approach to create SPARQL queries from natural language questions. QAnswer produces an extensive set of candidate SPARQL queries, which necessitates learning an effective ranking model from large-scale training datasets. Commonly both approaches try to utilize the structure of the KG for query generation. In contrast to this approach, other popular approaches rely on SPARQL templates for query generation. One example is the approach of Hao et al. (2018), which introduced a pattern-revising QA system for answering simple questions. It relies on pattern extraction and joint fact selection, enhanced by relation detection, to rank candidate subject-relation pairs. NEQA (Abujabal et al., 2018) is a template-based KGQA system, which uses a continuous learning paradigm to answer questions from unseen domains. Apart from using a similarity-based approach for template matching, it also relies on user feedback to improve over time. Since it relies on active learning of sub-parts, it may miss the semantic connections between question parts. Also, since many templates are learned, it can fail to generalize well to other domains or KB structures. Another approach from Zheng and Zhang (2019) uses structural query patterns, a coarse-granular version of SPARQL basic graph patterns, which are later augmented with different query forms and thus can also generate SPARQL query modifiers. The closest work to our approach, as presented in Chapter 8, is the one by Athreya et al. (2020). By using a tree-based RNN to learn templates directly from LC-QuAD v1, the system becomes closely coupled to its specific SPARQL structures, thereby limiting its generalizability to different KGs or datasets. Next to supervised learning approaches, unsupervised and traditional approaches have also been developed. An example for these branch of approaches is QAMP (Vakulenko et al., 2019), which implements an unsupervised message-passing system that uses a simple approach to information extraction. QAMP outperforms QAnswer on LC-QuAD v1.

Overall, traditional approaches lack performance compare to modern LLM-based approaches. One of the main shortcomings is the adaption to new datasets and KGs, since this often requires to update rule-based pipelines. Furthermore, questions often include a lot of implicit information, which makes the extraction of KG resources challenging for traditional term-based retrieval pipelines.

3.2.2 LLM-based approaches

In recent years, LLM-based approaches have become the leading technology for question answering over knowledge graphs. In general, there are two different kinds of approaches, namely fine-tuning and prompting with generative LLMs, which are described below. Works based on fine-tuning, such as the approaches of Borroto et al. (2021) and Banerjee et al. (2022) treat the query generation problem as a translation problem. The task is to translate a natural language question into a SPARQL query. As inputs, these models use the question itself, along with linked knowledge such as entities and relations from the knowledge graph (Banerjee et al., 2022). The target of the query prediction is usually a SPARQL query. Flipped relations are a common problem in query generation, since entities can be placed either at the head or the tail of a triple pattern. Su et al. (2024a) solves this problem by applying triplet-order-sensitive pretraining. Another limitation is the reliance of benchmarks on predefined templates for query construction, which hinders the ability of fine-tuned models to generalize to novel query structures. To mitigate this, Diallo and Zouaq (2025) augment natural language questions with structured semantic information, by applying a frame detection approach, to improve generalization.

Other approaches iteratively predict and rank queries as S-expressions (Shu et al., 2022; Ye et al., 2022). For example, the RnG-KBQA (Ye et al., 2022) framework combines ranking and query generation to predict queries. Unlike translation-based approaches for Wikidata question answering, candidate queries are generated and introduced into the generation model. Other approaches, such as Pangu (Gu et al., 2023), construct queries by iteratively extending and ranking a set of query sub-plans. These models share the characteristic of computing a large number of sub-plans, which demand significant resources in terms of GPU memory and time, making an end-to-end implementation usually unavailable.

As an alternative to fine-tuning, large generative LLMs enable query generation via prompting. In this context, techniques such as retrieval-augmented generation are used to extract knowledge from the KG. A popular approach in this context is to retrieve similar queries from the benchmark's training split and incorporate them as examples into the prompt. This enables the LLM to leverage these patterns to generate a query with a similar structure. An early approach in this direction from Kovriguina et al. (2023) uses a set of guiding

examples and augment these examples with subgraph information. However, their results show that this approach does not outperform existing fine-tuning approaches. One of the state-of-the-art approaches in the area of prompt-based query generation from Zahera et al. (2024) applies the chain-of-thought technique. It generates prompts based on the KG context, which includes entities, relations, and an example query retrieved from the training data. To find similar question-query pairs k-means clustering is used to group similar questions into clusters, based on LLM-based text embeddings. At inference time, each cluster is represented by a single question-query pair and the most similar example is chosen as input for the prompt. More recently Walter and Bast (2025) developed a zero-shot SPARQL generation framework for question answering. It provides multiple functions for the interaction between the LLM and the KG, which includes searching for entities, relations, objects, or triples, but also for finding similar queries as discussed earlier. Furthermore, also the introduction of feedback is investigated. Their results show impressive improvements for the SPARQL generation task. However, these kind of technique also requires many LLM calls, which makes it slower compared to most fine-tuning-based approaches.

3.2.3 Query Generation on Multiple Knowledge Graphs

Since the inception of the semantic web, various methods have been developed for semantic parsing across multiple or interlinked knowledge graphs. One early method, PowerAqua by Lopez et al. (2006), uses semantic similarity between ontology terms and user queries to generate query triples, which are used by an inference engine to retrieve answers. The updated version of Lopez et al. (2012) works on large KGs like DBpedia[2] but struggles with scalability, performance, and effectiveness, especially with large-scale data, complex queries, and the integration of data from different ontologies, requiring significant effort for updates and maintenance. SINA, introduced by Shekarpour and Auer (2014), is a data-semantics-aware keyword search approach that converts natural language and keyword queries into SPARQL queries to access interlinked KGs in the Linked Open Data Cloud[3]. It uses a hidden Markov model for query disambiguation and resource identification, leveraging the topology of linked data to construct federated SPARQL queries for information retrieval from multiple KGs. However, this process is computationally complex, and the keyword-based approach can overlook the question's syntax. Another approach by Zhang et al. (2016) employs a rule-based method to handle queries across multiple KGs by identifying resources, forming triple patterns, aligning variables, and performing joint inference to create accurate SPARQL queries. However, string matching for entity linking can lead to

[2] https://www.dbpedia.org/
[3] https://lod-cloud.net/

mismatches and missed entities, since it cannot disambiguate similar names or accurately handle naming variations.

Similar to single KG sematic parsing, more recent approaches adapted neural networks for multi-KG query generation. For instance MULTIQUE (Bhutani et al., 2020) uses neural networks for semantic parsing to handle complex queries by combining curated and extracted KGs. These approaches rely heavily on textual data, making it challenging to manage and computationally expensive to extract relevant information on demand. Neelam et al. (2022) introduced SYGMA, which streamlines query generation through KG-agnostic *Question Understanding* and KG-specific *Question Mapping & Reasoning*. It uses abstract meaning representation to generate a KG-agnostic lambda expression based on the transformer model. This expression is refined with specific KB details before being converted into a SPARQL query using a rule-based system. SYGMA's modular design aids generalization but is sensitive to individual module performance, particularly relation linking. Another popular approach to deal with multiple isolated datasets is to apply federated learning (Zhang et al., 2022b; Zhu et al., 2023). Following this idea, Chen et al. (2021) present an approach for federate learning over heterogeneous question answering datasets. The core idea is to first learn local models for each individual dataset and then learn a global model by aggregating the local model updates, such that training data does not need to be uploaded to a global server, which preserves the privacy of training data. With the development of larger LLMs also RAG-based approaches became popular for question answering on multiple KGs. Specifically, Emonet et al. (2025) developed an RAG-based approach for federated query generation across multiple SPARQL endpoints, leveraging indexed metadata and exemplary queries retrieved from the respective sources.

3.2.4 Answer Verbalization

To verbalize answers from KGQA systems, common approaches consider an encoder-decoder architecture. For example, the VOGUE framework (Kacupaj et al., 2021b) takes inputs from both the question and the logical form via a dual-encoder model. These inputs are then combined via cross-attention, and a hybrid decoder generates the final natural-language sequence. Recent text generation methods use pretrained language models like T5 (Raffel et al., 2020) and BART (Lewis et al., 2020a) for various NLP tasks. For instance, Montella et al. (2022) applied transfer learning with pretrained models such as T5 and BART to verbalize answers from KGQA systems, using questions and answers as inputs. They also applied a masking technique to improve the generalization of test datasets. Their results indicate that transfer learning improves answer verbalization performance compared to the VOGUE model.

4 Contextual Augmentation for Entity Linking using Large Language Models

This chapter contributes to the following research questions:

- **RQ**$_1$: *What methods can be used to predict knowledge that is implicit rather than explicitly expressed in questions?*
- **RQ**$_2$: *How can LLM-based augmentation enhance the performance of entity linking systems?*

In this context, an LLM-based contextual augmentation approach for expanding entity mentions is implemented and evaluated. Additionally, a joint model for NER and ED is developed and evaluated against an alternative end-to-end model. The content is based on our work Vollmers et al. (2025c), where the author designed, implemented, and evaluated the approach and co-wrote the corresponding paper.

4.1 Overview

Entity linking (EL), as introduced in Section 2.4.2, is an essential step in the pipeline of most KGQA systems, enabling linking entities mentioned in the question to KGs. An important challenge in linking entities for KGQA is that questions are usually short texts containing only one or two entities. At the same time, state-of-the-art EL systems such as De Cao et al. (2021) and Zhang et al. (2022c) rely on the context of an entity mention to generate a vector representation of the entity mention or the input sequence. The problem of a short context window is that traditional methods often struggle to handle long-tail entities, such as rare or polysemous entities, making it difficult to accurately disambiguate them when there is only minimal contextual information in the input sequence (Jiang et al., 2021).

For example, consider the entity *"Jaguar"*. In a general context, *"Jaguar"* could refer to the animal, the car brand, or even a sports team. However, in a domain-specific context, such as a biology research paper, *"Jaguar"* would most likely refer to the animal. Traditional methods often struggle with such nuances, leading to frequent disambiguation errors. For

instance, in the sentence *"Angelina met her partner Brad and her father Jon in AK"*, the use of first names alone introduces significant ambiguity, complicating the entity linking process.

In this chapter, we propose an LLM-based augmentation strategy to enrich the context of entity mentions to improve the disambiguation performance. Our approach expands entity mentions by prompting the LLM to extend them to their canonical Wikipedia titles, thereby replacing ambiguous entity spans with more explicit, linkable ones. For example, the spans *"Angelina"*, *"Brad"* and *"Jon"* should be expanded to *"Angelina Jolie"*, *"Brad Pitt"* and *"Jon Voight"*. Additionally, our strategy replaces abbreviations such as *"AK"* for the state of Alaska with the full name *"Alaska"*. To link entities in out-of-domain datasets, we use an autoregressive model (De Cao et al., 2021) that generates entity names token by token based on context. This approach allows the model to adapt to new or unseen entities by leveraging surrounding text, thereby improving the accuracy of identifying and linking entities absent from the training data. We conducted several experiments on different benchmarks to evaluate the performance of our approach against various baselines. Specifically, we experimented with two models: an end-to-end model that directly links entities and a traditional two-step approach that first identifies entity spans, before applying the disambiguation step.

At this stage, we evaluate the augmentation approach on the classical entity linking setup described in Section 2.4.2 and on traditional entity linking datasets. In a later step, we will combine the augmentation with keyphrase generation and neural ranking to implement an effective linking approach for questions.

The evaluation results for the classical EL setup demonstrate that our approach significantly outperforms baselines by a large margin across the benchmarking datasets. We summarize the main contributions in this chapter as follows:

- We propose an LLM-based entity linking approach that leverages zero-shot prompting, achieving state-of-the-art results across most out-of-domain evaluation datasets.
- We evaluate different LLM-based augmentation strategies for entity linking and compare their effectiveness across the two-step and end-to-end approaches.
- We evaluate the performance of a joint entity recognition and disambiguation model and compare it with end-to-end EL models.
- We make our source code and fine-tuned models publicly available on GitHub.[1]

[1] https://github.com/dice-group/AugmentedEL

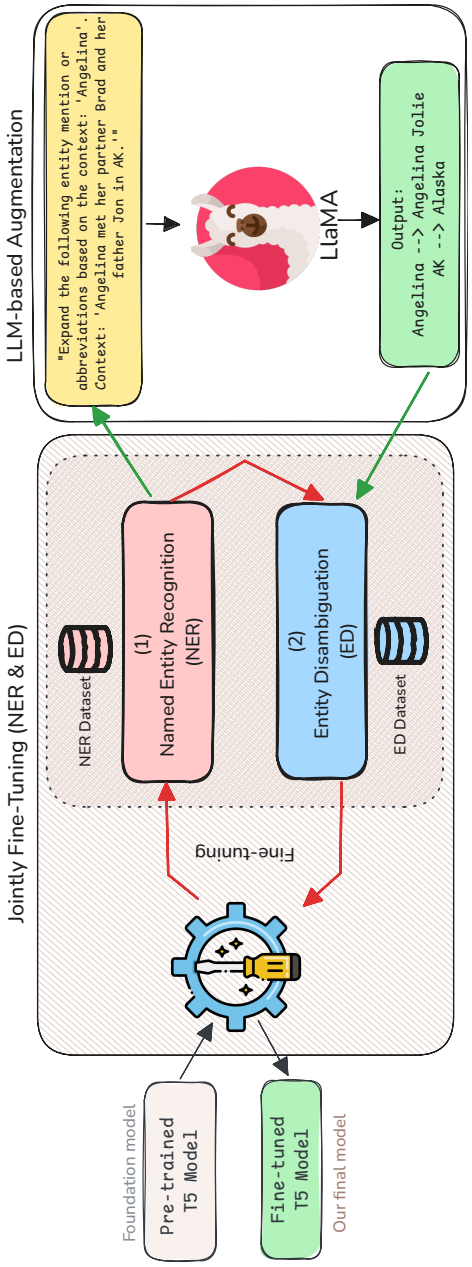


Figure 4.1: The architecture of our approach, including joint fine-tuning and LLM-based augmentation.

4.2 Approach

This section outlines our approach for entity linking using a fine-tuned T5 model and contextual augmentation using LLMs. In this chapter we use the classical task definition of entity linking as described in Section 2.4.2. First, we provide a description of our approach with the main components, including: `Named Entity Recognition (NER)`,

`LLM-based Augmentation`, `Entity Disambiguation (ED)`, and the `Joint Fine-tuning of (NER&ED)`. Afterward, we describe our strategy to mitigate LLM hallucination and the ablations of our approach to assess the performance of LLM-based augmentation. The architecture of our approach can be derived from Figure 4.1.

4.2.1 Architecture

We employ the T5 model as the *foundation architecture* in our approach and fine-tune it on NER and ED tasks. This joint fine-tuning enables our model to leverage shared knowledge across both tasks, thereby improving overall performance. The following sections summarize the main components of our approach.

Named Entity Recognition We use the T5 model with a transformer architecture to capture contextual information from both preceding and succeeding text. Furthermore, the T5 model treats the NER task as a text-to-text problem, where both the input and the output are sequences of text. Initially, the input text is tokenized and processed by the model's encoder, which generates contextual embeddings based on surrounding words. The decoder then uses these embeddings to produce an output sequence with entity tags.

In our approach, we fine-tune the T5 model on annotated datasets with entity annotations, enabling it to learn accurate tagging based on context. For instance, the sentence *"Angelina met her partner Brad and her father Jon in AK"* is transformed into *"[BEGIN_ENT] Angelina [END_ENT] met her partner [BEGIN_ENT] Brad [END_ENT] and her father [BEGIN_ENT] Jon [END_ENT] in [BEGIN_ENT] AK [END_ENT]"*.
In this output, *"Angelina"*, *"Brad"*, *"Jon"*, and *"AK"* are recognized as entities and marked with an annotation tag.

LLM-based Augmentation To further augment the NER process, we employ the LlaMA-3 model to perform contextual augmentation on mention spans and expand on the entities detected by the NER step. The core idea is to replace ambiguous or incomplete entity mentions with more precise and recognizable forms, such as full names or specific titles. This is achieved by prompting the LlaMA-3 model to generate these extended forms based on the given context. Our prompt includes a structured input with the entity mention and its surrounding context. For example, consider the entity mention *"Angelina"* in the sentence: *"Angelina met her partner Brad and her father Jon in AK."* . We prompt the LlaMA-3 model to expand entity mentions as follows:

LLM Prompt

```
Expand the following entity mention 'Angelina' and
abbreviations 'AK' based on the context:  .
context:  "Angelina met her partner Brad and her father
Jon in AK."
```

In response, the LlaMA-3 model expands the entities (e.g., *"Angelina"*$\longrightarrow$*"Angelina Jolie"* and for the abbreviation *"AK"*$\longrightarrow$*"Alaska"*. This approach allows us to replace ambiguous mentions with their more specific counterparts, thereby improving the ability of the model to link entities in the follow-up entity disambiguation step.

Entity Disambiguation Entity disambiguation is crucial for resolving ambiguities when multiple entities share the same name. In our approach, the T5 model addresses this by encoding the context around an entity mention and producing a representation that captures both semantic (e.g., word meaning) and syntactic (e.g., sentence structure) information to determine the correct entity based on the surrounding text. Then, it decodes this representation to predict the correct entity from a set of candidates, using attention mechanisms to focus on relevant parts of the input text. Formally, given a context C, the model generates an output sequence T with entity mentions $m \in M^n$ and their corresponding URIs $e \in E^n$, represented by their titles in a target knowledge graph. The output structure is defined as:

$$[\text{BEGIN\_ENT}]\ m\ [\text{END\_ENT}][\text{title}(e)], \tag{4.1}$$

where $m \in M^n$ denotes an entity mention and $e \in E^n$ represents the associate URI.

Unlike traditional methods, our approach is unique in that it leverages a single T5 model to perform NER and ED in sequence. First, the T5 model generates an intermediate sequence I for NER:

$$\mathrm{I} = [\text{BEGIN\_ENT}]\ m\ [\text{END\_ENT}] | m \in \mathrm{M}^n. \tag{4.2}$$

Afterward, it predicts the target output based on the NER step's output. At inference time, the target output is expanded by our augmentation strategy as presented in section 4.2.1.

Jointly Fine-tuning (NER & ED) To fine-tune the T5 model for both NER and entity disambiguation, we create two distinct training sets: one for NER and another for disambiguation. For the NER task, the input is a text sequence without annotations, appended with the suffix *"target_ner"*. The target sequence is the same text with annotated entity mentions, as detailed in Section 4.2.1. For the disambiguation task, the input is a text sequence with

annotated entities, appended with the suffix *"target_el"*, and the target sequence includes the corresponding Wikipedia entity labels, as described in section 4.2.1.

We combine the NER and disambiguation samples into a single dataset and jointly fine-tune the model for both tasks. Additionally, we integrate the NER and entity disambiguation tasks within a unified framework (see Figure 4.1). This integration enhances the robustness and accuracy of our approach for identifying and disambiguating entities, thereby improving entity linking performance.

4.2.2 Mitigating LLMs Hallucination

During our implementation, we found that LLM hallucination is a critical problem, especially in the augmentation and disambiguation phase. In the disambiguation step, the LLM model occasionally predicts labels that do not exist in Wikipedia. To avoid this, we generated a dictionary that maps all Wikipedia titles to their URIs. By applying this dictionary, we can omit all annotations that do not have an exact match in the dictionary. In the augmentation step, hallucination occurs when the LLM model provides augmentations for spans that do not exist in the sequence or are not annotated by the NER step. To avoid misleading expansions, we only consider those spans for expansion that precisely match one of the annotated spans from the NER step. Unlike the augmentation and disambiguation steps, we did not encounter problems with hallucination in the NER step.

4.3 Ablations

We conducted various ablations of our model to evaluate the impact of augmentation in different setups of EL models, as follows:

End-to-End Foundational Model Recent studies focus on developing entity linking models that omit the mention detection step and directly predict the entity set (De Cao et al., 2021; Zhang et al., 2022c). In the end-to-end (E2E) approach, an expanded set of entities is used to compute E^n for the context C directly.

To set up the end-to-end approach, we trained our T5 model to directly predict the target sequence T from the input context C. This approach aligns with the experimental setup of De Cao et al. (2021) for end-to-end entity linking. However, our implementation uses the T5-base model rather than the BART model. We selected the T5-base model due to its superior performance without requiring a prefix trie. Additionally, we use an augmentation

strategy with two inference steps for the model: first, the model identifies entity mentions to be expanded, then performs final entity disambiguation. After the first step, we extract entity mentions from the output sequence, ignoring the predicted titles. Subsequently, we use the same LLM prompt, as presented in Section 4.2.1, to find possible expansions. To integrate these expansions into the sequence, we replace mentions with their expanded forms, omitting the tags used to annotate entity mentions, similar to the foundational model.

NER Augmentation To improve the performance of our Entity linking system, we introduce an LLM prompt to find additional entity spans in the input texts. We use the following instructions to guide the LLM in finding accurate entity spans from the input text:

LLM Prompt

```
Please generate one list with all entities from the
following text in JSON format, excluding numbers.  Do
not format the json output:
context:  "Angelina met her partner Brad and her father
Jon in AK."
```

We then apply a regular expression, similar to the first expansion strategy, to extract the new entity spans. Since we cannot rely on specified indices in the LLM output, we only consider spans that exactly match the original input sequence. To avoid overlapping annotations, we order the newly extracted spans by length in descending order and add expansions only where no surrounding annotation is present.

Alternative NER Approaches We conducted experiments using the state-of-the-art NER framework Flair (Akbik et al., 2019) for the mention detection step. Additionally, we experimented with a hybrid approach that employs an end-to-end foundational model for mention detection and then introduces those entities into the disambiguation step.

4.4 Experiments

We conducted our experiments to answer $\mathbf{RQ}_2$: *How can LLM-based augmentation enhance the performance of entity linking systems?*. The experiments also contribute to $\mathbf{RQ}_1$: *What methods can be used to predict knowledge that is implicit rather than explicitly expressed in questions?*. This is because contextual augmentation is used to predict additional implicit knowledge about entities. For a detailed analysis, we divide this question into the following subquestions:

$\mathbf{S}_1$ How well does the presented approach perform compared to state-of-the-art baselines?

$\mathbf{S}_2$ How does the LLM-based augmentation impact foundational models?

$\mathbf{S}_3$ Which augmentation strategy works best?

4.4.1 Experimental Setup

We fine-tuned the T5 model as a foundational model on the KILT dataset. The fine-tuning was conducted on four NVIDIA A100 GPUs, each with 40GB of memory, for one week. Following this, we further trained the model on the AIDA-train dataset for up to 125 epochs with an early stopping strategy. We used the AIDA-test-A split as the development set. We chose the base version of the T5 model since its size is comparable to the models used by baseline approaches. The T5 implementation was obtained from Hugging Face[2]. Our training setup mirrors that of the baseline approaches; no additional datasets were used for further training. The end-to-end foundational model, as detailed in Section 4.3, was trained under the same conditions. For inference, we deployed the foundational model and the LLM for the augmentation strategy on a separate machine equipped with two NVIDIA H100 GPUs. This setup enabled the use of large models such as LlaMA-3 with 70 billion parameters.

4.4.2 Evaluation

We conducted our experiments using the GERBIL framework (Röder et al., 2018), which benchmarks entity linking on various datasets. We configured an A2KB experiment and integrated our approach as a web service. We report the InKG micro-F1 scores, a standard metric in the literature. This metric considers only entities with a corresponding link in the target knowledge graph, thereby excluding *out-of-wiki* entities from the evaluation (Röder et al., 2018). For our baseline experiments, we focused only on approaches evaluated in an end-to-end setup, including NER. This inclusion is critical, as NER output significantly affects entity disambiguation performance.

4.4.3 Datasets

To measure the performance on the end-to-end entity linking task, we rely on commonly used datasets from the literature (De Cao et al., 2021; Zhang et al., 2022c). These datasets

[2] https://huggingface.co/docs/transformers/model_doc/t5

Table 4.1: Dataset Statistics

Dataset	#InKG entities	#Docs
AIDA-test-B	4,485	230
Der	201	183
KORE 50	139	48
MSNBC	737	20
N3-Reuters-128	626	115
N3-RSS-500	515	425
OKE-2015	481	100
OKE-2016	221	55

are: AIDA-test-B (Hoffart et al., 2011) (AIDA), Derczynski (Derczynski et al., 2015) (DER), KORE 50 (Hoffart et al., 2012) (K50), MSNBC, NS3-Reuters-128, NS3-Reuters-500 (Röder et al., 2014) (R-128,R-500), and the OKE challenge datasets OKE-2015 and OKE-2016 (Nuzzolese et al., 2015). Table 4.1 provides detailed statistics of these datasets, including the number of entities that have a corresponding entry in the knowledge graph (#InKG entities) and the number of documents (#Docs). The AIDA-test-B dataset contains the largest number of entities with corresponding entries in the knowledge graph. Given that our models are trained on the AIDA-train dataset, AIDA-test-B serves as an in-domain dataset. All other datasets are considered out-of-domain. As knowledge source, we used the 2019-Wikidata-dump from the KILT dataset[3], which is commonly used by state-of-the-art entity linking systems.

4.4.4 Comparison to Baseline Approaches ($\mathbf{S}_1$)

In this section, we compare the performance of our model against state-of-the-art baselines across various datasets. We obtained the baseline scores from Zhang et al. (2022c). Our evaluation focuses on three setups that demonstrate the most stable results: the foundational T5 model with entity span expansion ($\text{T5}_{(NER+ED)}$), a combination of the T5 model with the Flair framework for NER (Flair & $\text{T5}_{(ED)}$), and an end-to-end T5 ablation model (Section 4.3) with span expansion (E2E T5).

Table 4.2 reports the evaluation results across all datasets. Unlike traditional entity linking systems such as Hoffart et al. (2011) and van Hulst et al. (2020), which use separate components for span detection and entity disambiguation, our models do not generate candidate entity sets. Instead, we use the fine-tuned T5 models with LLM-augmentation for entity expansions for contextual information. Our model achieves the best performance

[3] https://github.com/facebookresearch/KILT

Table 4.2: Performance comparison against baseline approaches using InKG micro F1 ($\mathbf{S}_1$). $T5_{(ED)}$ is fine-tuned for Entity disambiguation, while $T5_{(NER+ED)}$ is jointly fine-tuned for both NER and disambiguation. The best result is highlighted in bold.

(a) Results on AIDA, MSNBC, Der, and K50

Approach	AIDA	MSNBC	Der	K50
Hoffart et al. (2011)	72.8	65.1	32.6	55.4
Steinmetz and Sack (2013)	42.3	30.9	26.5	46.8
Moro et al. (2014)	48.5	39.7	29.8	55.9
Kolitsas et al. (2018)	82.4	72.4	34.1	35.2
van Hulst et al. (2020)	80.5	72.4	41.1	50.7
De Cao et al. (2021)	83.7	**73.7**	54.1	60.7
Zhang et al. (2022c)	**85.8**	72.1	52.9	64.5
Flair & $T5_{(ED)}$	71.4	61.3	51.6	**72.7**
$T5_{(NER+ED)}$	71.6	69.3	**55.7**	70.6
E2E T5	69.0	64.2	53.7	64.3

(b) Results on R-128, R-500, OKE 2015, OKE 2016, and Average

Approach	R-128	R-500	OKE 2015	OKE 2016	Avg.
Hoffart et al. (2011)	46.4	42.4	63.1	0.00	47.2
Steinmetz and Sack (2013)	18.1	20.5	46.2	46.4	34.7
Moro et al. (2014)	23.0	29.1	41.9	37.7	38.2
Kolitsas et al. (2018)	50.3	38.2	61.9	52.7	53.4
van Hulst et al. (2020)	49.9	35.0	63.1	58.3	56.4
De Cao et al. (2021)	46.7	40.3	56.1	50.0	58.2
Zhang et al. (2022c)	54.1	41.9	61.1	51.3	60.5
Flair & $T5_{(ED)}$	**54.5**	56.3	**66.6**	**61.5**	**62.0**
$T5_{(NER+ED)}$	51.7	**56.6**	59.4	58.5	61.7
E2E T5	51.9	57.3	61.6	58.4	60.1

on all datasets except from AIDA and MSNBC. On the MSNBC dataset, our model's performance is comparable to the state of the art. However, we observed a performance drop on the in-domain AIDA-test-B dataset compared to the baselines. Interestingly, the expansion strategy is less effective on the AIDA and MSNBC datasets. This might be because our foundational models were trained on the AIDA data, so the expansion did not provide substantial new contextual information for the input sequences. The MSNBC dataset, which includes 20 news articles, is similar to the in-domain AIDA-test-B dataset. The other datasets often contain much shorter texts with fewer entities and are not exclusively based on news articles.

Our findings indicate that the greater the difference between a dataset and our training data, the better our model performs relative to the baselines. Notably, on the highly ambiguous KORE 50 dataset, our joint mention detection and disambiguation strategy improves the F1 score by over 6 percentage points, increasing to an 8-point gain when integrated with the third-party Flair NER framework.

4.4.5 Evaluation of Foundational Models ($\mathbf{S}_2$)

To address this question, we evaluated the performance of various foundational models, especially T5, and investigate how our LLM-based augmentation strategy affects their performance. We employ the LlaMA-3 model for entity expansion (e.g., *"Angelina" to "Angelina Jolie"*) to facilitate entity disambiguation, as described in Section 4.4.6.

Table 4.3 reports the evaluation results for this experiment. We employed the same models as in the previous section, along with the mixed foundational model (Mixed Model) presented in 4.3. By analyzing the results of both the foundational models and those enhanced with our expansion strategy, we observed significant improvements on most out-of-domain datasets. The augmented version of the traditional setup, which separates mention detection and evaluation, outperforms the augmented end-to-end model on five out of eight datasets. For the other three datasets, the performance difference between the two setups was negligible. However, without augmentation, the end-to-end model generally performs better than the traditional setup. This finding suggests that our LLM-based entity expansion is particularly effective with a traditional setup for disambiguating and enables linking entities to knowledge graphs more accurately.

4.4.6 Evaluation of Augmentation Strategies ($\mathbf{S}_3$)

Table 4.4 shows the results for the two augmentation strategies –mention expansion and NER expansion– for both the foundational model ($T5_{(NER+ED)}$) and the combination of Flair and the foundational model (Flair & $T5_{(ED)}$). As previously described, the mention expansion consistently improves the performance of the entity linking system. Conversely, the NER expansion strategy shows variable efficacy across some datasets. While it enhances performance on some datasets, it underperforms relative to standalone mention expansion on others. This variation in performance may originate from differences in annotation strategies across datasets. On the Der dataset, results are notably inconsistent. The NER expansion strategy identifies more entities than are accounted for in the benchmark, which degrades performance relative to the foundational model and other strategies.

Table 4.3: Comparison of different foundational models ($\mathbf{S}_2$). The first four rows present the results when no augmentation strategy is applied. The other rows present results when the entity expansion strategy (Section 4.2.1) is applied. The best result is highlighted in bold.

(a) Results on AIDA, MSNBC, Der, and K50

Model	AIDA	MSNBC	Der	K50
Flair & $T5_{(ED)}$	**73.5**	67.0	48.8	50.0
$T5_{(NER+ED)}$	67.4	61.5	53.9	50.0
E2E T5	68.0	63.0	50.4	49.6
Mixed Model	66.8	61.9	53.5	48.5
Flair & $T5_{(ED)}$ aug.	71.4	61.3	51.6	**72.7**
$T5_{(NER+ED)}$ aug.	71.6	**69.3**	**55.7**	70.6
E2E T5 aug.	69.0	64.2	53.7	64.3
Mixed Model aug.	69.0	63.1	55.0	69.6

(b) Results on R-128, R-500, OKE 2015, OKE 2016, and Average

Model	R-128	R-500	OKE 2015	OKE 2016	Avg.
Flair & $T5_{(ED)}$	41.0	55.1	58.9	55.0	56.2
$T5_{(NER+ED)}$	47.6	55.2	55.9	56.0	56.0
E2E T5	49.1	56.8	56.1	56.5	56.1
Mixed Model	49.2	55.3	56.8	50.0	55.2
Flair & $T5_{(ED)}$ aug.	**54.5**	56.3	**66.6**	**61.5**	**62.0**
$T5_{(NER+ED)}$ aug.	51.7	56.6	59.4	58.5	61.7
E2E T5 aug.	51.9	57.3	61.6	58.4	60.1
Mixed Model aug.	52.3	**58.1**	60.6	53.5	60.2

To further explore this issue, we conducted an additional experiment using only the LLM expansion strategy for entity extraction, labeled as "LLM-only" in Table 4.4. The results indicate that the LLM-only strategy performs well on short texts in the KORE 50 dataset but underperforms on other datasets when not combined with the T5 model. We retain the T5 model for predicting final identifiers, since using mention expansion alone led to hallucinations, making many predicted identifiers untraceable in the title dictionary.

4.4.7 Comparison of Different LLM Models

As an additional experiment, we evaluated which LLM models perform best for our expansion strategy on entity mentions. We evaluated models of various sizes to assess the impact of parameter scale on performance. The LlaMA-3 model with 70 billion parameters performed best among all other models. Our results demonstrate that the effectiveness of

Table 4.4: Evaluation of different augmentation strategies ($\mathbf{S}_3$). Models with NER-exp use both NER and mention expansion, whereas other models use only mention expansion. The best result is highlighted in bold.

(a) Results on AIDA, MSNBC, Der, and K50

Model	AIDA	MSNBC	Der	K50
$T5_{(NER+ED)}$	**71.6**	69.3	**55.7**	70.6
$T5_{(NER+ED)}$, NER-exp	68.0	**70.3**	37.4	69.3
Flair & $T5_{(ED)}$	71.4	61.3	51.6	72.7
Flair & $T5_{(ED)}$, NER-exp.	67.4	57.0	37.1	**76.1**
LLM-only	63.7	62.6	36.4	75.1

(b) Results on R-128, R-500, OKE 2015, OKE 2016, and Average

Model	R-128	R-500	OKE 2015	OKE 2016	Avg.
$T5_{(NER+ED)}$	51.7	**56.6**	59.4	58.5	61.7
$T5_{(NER+ED)}$, NER-exp	53.0	45.4	44.6	57.4	55.7
Flair & $T5_{(ED)}$	**54.5**	56.3	**66.6**	**61.5**	**62.0**
Flair & $T5_{(ED)}$, NER-exp.	51.7	46.5	62.8	61.2	57.5
LLM-only	49.3	47.3	61.7	59.6	57.0

Table 4.5: Comparison of different models for the expansion. The best result is highlighted in bold.

(a) Results on AIDA, MSNBC, Der, and K50

Model	AIDA	MSNBC	Der	K50
LlaMA-3 70B	**71.6**	**69.3**	**55.7**	**70.6**
LlaMA-3 8B	69.6	67.7	53.2	53.3
LlaMA-2 70B	70.2	68.1	51.5	57.9
LlaMA-2 7B	70.3	70.0	53.0	51.5
Mistral	70.7	67.5	54.2	48.3

(b) Results on R-128, R-500, OKE 2015, OKE 2016, and Average

Model	R-128	R-500	OKE 2015	OKE 2016	Avg.
LlaMA-3 70B	**51.7**	**56.6**	**59.4**	**58.5**	**61.7**
LlaMA-3 8B	48.1	54.7	53.5	49.1	56.2
LlaMA-2 70B	48.2	57.4	57.2	50.5	57.6
LlaMA-2 7B	47.7	54.7	55.6	54.3	57.1
Mistral	47.6	56.1	56.1	56.1	57.1

the expansion strategy scales with the number of model parameters. The Mistral and the LlaMA-3 model with 8 billion parameters achieve similar performance. The main reason for the difference in performance is that the larger models produce more stable outputs than

the smaller ones. In the larger models, the JSON output usually has quite similar formatting across all documents. In comparison, in the smaller models, the JSON output was slightly different from document to document, making it hard to extract the expansions from the output. Furthermore, the output was not always complete, as some entities were missing, especially in larger sequences.

4.5 Limitations

Similar to the approach of De Cao et al. (2021), we use Wikipedia as the main knowledge source, where unique titles facilitate entity identification. However, many knowledge graphs, such as Wikidata (Vrandečić and Krötzsch, 2014) do not provide unique entity labels. Existing benchmarks and training datasets are also based on Wikipedia, making text-based features sufficient for efficient entity linking. However, within knowledge graphs, the graph-based structure is crucial for disambiguation. For instance, the German state of Berlin and the city of Berlin share the same label but are distinct entities, making interconnected entities crucial for disambiguation. The current benchmarking datasets used for evaluation are outdated and do not address newer challenges, such as predicting rare entities. Additionally, it is unclear to what extent these datasets have been included in the training data of current LLMs. Therefore, future research should focus on developing new datasets that also address the limitations of modern LLMs.

4.6 Conclusion

In this chapter, we presented our approach for entity linking using a jointly fine-tuned model and LLM-based contextual augmentation. In particular, our approach employs a fine-tuned T5 model that integrates NER and disambiguation tasks into a unified framework, reducing resource demands compared to separate models for each task. Although this setup may slightly degrade performance, our experiments showed that the resulting performance loss is marginal, mainly due to unseen entities. Furthermore, our approach leverages the LlaMA-3-70B model to expand entity mentions with contextual augmentation. The evaluation results demonstrate that LLM-based augmentation significantly improves the performance on out-of-domain datasets, achieving state-of-the-art results compared to traditional two-step methods (i.e., entity recognition and disambiguation). Our experiment with different-sized LLMs has shown that larger LLMs perform better than smaller ones for context augmentation, producing more consistent output with fewer variations. Evaluating LLM-based disambiguation strategies for rare entities remains a key area for future research.

This task necessitates the development of new benchmark datasets, as current resources primarily target well-known entities.

In the context of this thesis, we will apply the introduced techniques for contextual augmentation to expand the context of questions. Since semantic parsing for KGs also requires predicting knowledge that is not explicitly mentioned in a question (Section 2.4.3), we will also introduce techniques to improve the robustness of retrieving information from the KG and apply LLMs to predict terms from the KG in the form of keyphrases, to extract additional knowledge, which can be used as additional context for a question.

5 Evaluating Noisy Optimization in Fine-tuning LMs for Neural Ranking

This chapter describes an approach for improving the robustness of neural retrieval and ranking methods introduced in Section 2.4.1. In the context of this thesis, this approach is employed to implement knowledge extraction pipelines, which are used to answer the following research questions:

- **RQ**$_1$: *What methods can be used to predict knowledge that is implicit rather than explicitly expressed in questions?*
- **RQ**$_3$: *How can LLM-based entity linking approaches improve the performance for extracting KG-knowledge for questions?*

To improve the robustness of LM-based encoder models, noise is introduced into the model parameters during the fine-tuning process. Different noise injection strategies are applied to the parameters of the model to evaluate whether they can improve its robustness. The content is based on Vollmers et al. (2026), where the author co-designed, implemented, and co-evaluated the approach and co-wrote the corresponding research paper.

5.1 Overview

As presented in Chapter 2, ranking models are an essential component within KGQA and entity linking systems, especially for retrieving candidate entities or relations from a KG. In Section 2.4.1, we have introduced two popular neural ranking architectures, namely bi-encoder and cross-encoder. Existing research has primarily focused on improving these neural architectures (Yamada et al., 2020), leveraging zero-shot learning (Sil et al., 2018), and incorporating external features like entity types and priors (Fang et al., 2019; Ganea and Hofmann, 2017; Zhang et al., 2020). However, less attention is paid to the optimization techniques used during training to fine-tune the models. More specifically, adaptive optimizers like Adam often face challenges such as local minima and saddle points, particularly in

high-dimensional, sparse, and imbalanced datasets, leading to suboptimal rankings (Kunstner et al., 2024). This can essentially reduce the model's overall effectiveness. In this chapter, we explore noise injection into model parameters, building upon existing work that utilizes this technique to improve generalization (Jin et al., 2017, 2021; Liu et al., 2021a; Orvieto et al., 2022, 2023; Polyak and Tsybakov, 1990; Zhou et al., 2019). Essentially, our goal is to address saddle point issues and achieve faster convergence towards better-generalized models by applying noise injection to the parameters during fine-tuning of the ranking models. The role of noise in the optimization step, particularly in stochastic gradient descent (SGD) (Robbins, 1951), has been a subject of extensive study, particularly in its impact on generalization (Jin et al., 2017, 2021; Polyak and Tsybakov, 1990; Zhou et al., 2019). It is widely recognized that the intrinsic stochastic noise in SGD, arising from minibatch sampling, tends to guide the optimization process towards flatter minima, which are generally associated with better generalization performance. A recent work by Orvieto et al. (2023) proved that noise injection during optimization can lead to better performance in vision models. Although such techniques have been extensively explored in the ML domain (Liu et al., 2021a; Orvieto et al., 2022), to the best of our knowledge, they have not yet been applied to information retrieval, particularly for ranking models.

In this chapter, we study the effect of injecting noise into the parameters of bi-encoder and cross-encoder models during fine-tuning to improve ranking task performance. More specifically, we inject noise into the model parameters after the backpropagation step but prior to the weight update. To this end, we first introduce the formal notion of parameter-level noise injection by considering three distinct approaches: (i) uncorrelated noise, (ii) anticorrelated noise based on previously added noise, and (iii) anticorrelated noise leveraging the loss gradient. We introduce an algorithm that incorporates these noise injection strategies directly into the parameter optimization process. Additionally, for ranking tasks, we study the effect of noise injection on the pairwise loss, in addition to applying noise injection and using cross-entropy loss. To this end, we also formally show that adding Gaussian noise to the parameters of the model in the optimization step, while considering pairwise loss, can work as an explicit regularization.

Apart from theoretical studies and formalizations, we perform extensive evaluation by applying our noise injection on the model parameters while optimizing them with the Adam optimizer (Kingma and Ba, 2015). To this end, we conduct an extensive evaluation to demonstrate the effectiveness of noise injection for ranking models, considering both the cross-entropy and pairwise losses. Our evaluations suggest that introducing noise into the optimization process can accelerate convergence and help avoid overfitting. Most importantly, we find that noise injection leads to substantial improvement in terms of recall. Our contribution in this work can be summarized as follows.

- We formalize three different noise injection approaches for the Adam optimizer used in ranking models.
- We introduce noise injection approaches that leverage pairwise ranking loss, specifically adapted to ranking tasks.
- We empirically evaluate several noise injection approaches for the Adam optimizer across bi-encoder and cross-encoder models trained on four ranking datasets.
- We give a comparative analysis showing the effectiveness of different noise injection approaches.
- Our code is publicly available.[1]

5.2 Methodology

In our work, we insert noise into the model parameters immediately after backpropagation, but prior to the weight update during training. Specifically, given a training sample $x^{(i)}$, the model first predicts an output $\hat{y}^{(i)}$, and let us assume the original output is $y^{(i)}$. After computing the loss $L(\hat{y}^{(i)}, y^{(i)}) = \hat{y}^{(i)} - y^{(i)}$, the gradient of the loss is computed, i.e., the backpropagation step is applied. Thereafter, we insert the noise into the parameters. This noise herein is added before updating the parameters using the moment estimates, allowing us to perturb the parameter space. After inserting the noise, the Adam optimizer is applied to update the model parameters. However, this update is now performed considering the noise-injected parameters, thereby allowing the optimizer to explore parameter regions it might not have otherwise encountered. As mentioned in previous work (Orvieto et al., 2023), this approach can escape from sharp minima and guide the training trajectory toward flatter, more generalizable solutions. Next, we formalize different noise injection strategies.

Noise Injection In our work, we inject noise by perturbing the parameters of the encoder components, namely, the query and candidate encoders in a dual bi-encoder as introduced by Wu et al. (2020a), and the shared language model in E5 (Wang et al., 2022). For the cross-encoder model, we consider the parameters of both the language model and the scoring function. To this end, the parameters of a ranking model, including the encoder and the scoring function, are denoted as θ_l.[2] Herein, we consider three different types of noise injection approaches: (i) uncorrelated noise, (ii) anticorrelated noise using the previous

[1] https://github.com/dice-group/RobustRanking

[2] Note that, for simplicity, we write language model while describing noise injection mechanism for the rest of the paper.

term, and (iii) anticorrelated noise using the gradient. Each of these is discussed in detail below.

Uncorrelated Noise Injection In this case, a standard Gaussian distribution is used to generate noise with mean zero and standard deviation 1 (which in this case is an identity vector $\mathbf{I}$), therefore, the distribution is defined as $\mathscr{G}(0,\mathbf{I})$. After generating the noise, it is added to the parameters during the optimization step. Consider the noise generated through a Gaussian distribution as ε_l for the parameters of the language model and the feed-forward network, respectively. Formally,

$$\theta_l' = \theta_l + \xi_l \cdot \varepsilon_l, \tag{5.1}$$

where ξ_l is the learning rate. Herein, the learning rate scales the noise in line with the step sizes in the optimization.

Anticorrelated to Previous Noise Unlike standard Gaussian noise, which is independent across iterations, anticorrelated noise integrates historical noise information, allowing for more advanced perturbations during training. More formally,

$$\theta_l' = \theta_l + \xi_l \cdot (-\alpha_l \cdot \varepsilon_l^{t-1} + \varepsilon_l^t), \tag{5.2}$$

where $\alpha_l = \frac{\alpha_0}{1+\|\nabla_{\theta_l}(L)\|}$ dynamically adjusts based on the gradient norm, $\varepsilon_l^t \sim \mathscr{G}(0,\mathbf{I})$ is the Gaussian noise term at iteration t, and ε_l^{t-1}, is noise from the previous iteration. The hyperparameter α controls the influence of past noise. By modulating the impact of noise based on previous iterations, this method can help the optimizer avoid sharp minima and explore flatter regions of the loss landscape, promoting better generalization. In practice, the choice of α_0 is crucial: too high a value may overly dampen the noise, limiting exploration, while too low a value might not sufficiently leverage the anticorrelated structure, resembling more random perturbations akin to Gaussian noise.

Anticorrelated to Gradient In this method, the noise is directly anticorrelated to the gradient of the loss function, acting as an adaptive regularizer. By aligning the noise structure with the gradient information, the model can balance stability during steep descent with exploration as it approaches flatter regions of the loss landscape. Formally,

$$\theta_l' = \theta_l + \xi_l \cdot (-\beta_l \cdot \nabla_{\theta_l}(L) + \varepsilon_l), \tag{5.3}$$

where $\beta_l = \frac{\beta_0}{1+\|\nabla_{\theta_l}(L)\|}$ dynamically adjusts the anticorrelation strength based on the gradient norm, and β_0 is a hyperparameter controlling the overall scale of anticorrelated noise. The dynamic nature of β_l introduces an adaptive mechanism that reduces noise when the gradient

is large and increases noise when the gradient is small. Specifically, when $\|\nabla_{\theta_l}(L)\|$ is large (i.e., during steep descent), β_l becomes small, minimizing the impact of anticorrelated noise. When $\|\nabla_{\theta_l}(L)\|$ is small (i.e., near convergence or in flat regions), β_l increases, emphasizes anticorrelated noise, promoting exploration and helping the model escape shallow minima or flat regions that may hinder optimization. The primary objective of introducing β is to scale the gradient's influence, not to adaptively scale the noise. In steep regions (large gradients), it prevents huge steps, while in flatter regions (small gradients), it enhances noise for exploration, ensuring stable optimization. Note that the two anticorrelated noise approaches discussed above are shown to be escaping the local minima of the loss landscape by Orvieto et al. (2022). However, their study focused on deep learning models using SGD as the optimization method, whereas we consider encoder models with the Adam optimizer.

Noise Injection in Training Loop Algorithm 1 outlines our overall noise injection approach to the Adam optimizer. In each epoch, for a given training example $(x^{(\mathbf{i})}, y^{(\mathbf{i})})$, the loss is computed from the prediction $\hat{y}^{(i)}$. After that, the backpropagation is applied by computing the gradients of the loss. After sampling Gaussian noise ε_l either, uncorrelated or anticorrelated noise δ_l is added to the gradient of the parameters. Thereafter, parameters θ_l are updated using the Adam optimizer, incorporating gradients and noise scaled by the adaptively adjusted learning rate ξ. Herein δ depends on the approach, for instance, for uncorrelated noise, $\delta_l = \varepsilon_l$, while for anticorrelated gradient noise, $\delta_l = -\beta_t \cdot \nabla_{\theta_l}(L) + \varepsilon_l$. After the noise addition is complete, the Adam optimizer is applied using the first and second moment estimates.

Algorithm 1: Training loop with noise injection for Document Ranking

Input: Initial parameters θ_l; Training data X, Y; Learning rate ξ_l; Number of epochs `num_epochs`

Output: Updated parameters θ_l after training

Initialize : Parameters θ_l

for `each epoch` **do**

  for `each training sample` $(x^{(i)}, y^{(i)}) \in (X, Y)$ **do**

    Get Prediction for $x^{(i)}$: $\hat{y}^{(i)} = \Psi(x^{(i)}; \theta_l)$;

    Compute loss: $L(\hat{y}^{(i)}, y^{(i)}) = \hat{y}^{(i)} - y^{(i)}$;

    Calculate the gradient of the loss (Backpropagation):;

    $\nabla\theta_l(L) = \frac{\partial L(\hat{y}^{(i)}, y^{(i)})}{\partial \theta_l}$;

    Sample Gaussian noise: $\varepsilon_l \sim \mathscr{G}(0, \mathbf{I})$;

    Add noise to parameters:;

    $\theta_l \leftarrow \theta_l - \xi_l \cdot (\nabla\theta_l(L) + \delta_l)$;

    Update parameters using Adam

return θ_l (final learned parameters)

5.3 Experiments

In this chapter, we aim to investigate whether the noise injection to the Adam optimizer can help to improve the performance of different types of ranking and retrieval models on diverse datasets and tasks, especially entity ranking for KGQA. The experiments in this chapter contribute to $\mathbf{RQ}_1$: *What methods can be used to predict knowledge that is implicit rather than explicitly expressed in questions?*, since neural retrieval can help to find additional knowledge in KGs. Additionally, they contribute to $\mathbf{RQ}_3$: *How can LLM-based entity linking approaches improve the performance for extracting KG-knowledge for questions?*, because neural retrieval is a submodule of neural entity linking frameworks. Specifically, we answer the following subquestions:

- $\mathbf{S}_1$ Does noise injection help in improving the performance of neural ranking models?
- $\mathbf{S}_2$ Which noise injection strategy among the three presented in Section 5.2 performs the best?
- $\mathbf{S}_3$ Can noise injection lead to faster convergence and reduce overfitting?

We first describe benchmarking datasets and our evaluation setup, and afterward answer these subquestions and provide a detailed discussion.

5.3.1 Datasets

For our evaluation, we apply datasets that provide a sufficient amount of training data to investigate the influence of introducing noise in the optimization. Overall, we used four datasets for the evaluation (three for the entity ranking task and one for document ranking), which are described below. The AIDA (Hoffart et al., 2011) dataset is widely used in the NLP community for evaluating entity linking frameworks. The documents in the dataset are news articles, and entities, linked to Wikipedia. To keep the same data for all experiments, we linked all entities to Wikidata, by applying the according links provided by Wikidata. Furthermore, we investigate the performance on shorter sequences, especially questions. For this reason, we used the knowledge graph question answering datasets LC-QuAD 2.0 (Dubey et al., 2019) and Mintaka (Sen et al., 2022). LC-QuAD 2.0 is a semi-automatically generated dataset. For generating question-query pairs, the authors rely on pairs of question-query templates, generated by crowd workers and automatically filled with entities and relations from Wikidata (Vrandečić and Krötzsch, 2014). Mintaka is fully generated by crowd workers and already provides question entities, which are used as target entities in our experiments. The entities from the LC-QuAD datasets are extracted from the provided SPARQL queries. All questions are linked to the Wikidata. For the

document ranking task, we apply the popular MS MARCO (Craswell et al., 2021) dataset, which is widely used to evaluate NLP tasks such as question answering, passage ranking, and document ranking.[3] Table 5.1 presents some statistics about these datasets.

Table 5.1: Dataset Statistics

Dataset	#Documents Entities	#Train Queries	#Test Queries
LC-QuAD 2.0	26,164	23,601	5,976
Mintaka	8,381	13,937	3,991
AIDA	3,072	964	216
MS MARCO	3,213,835	367,013	5,193

5.3.2 Training and Evaluation Setup

For our experiment, we evaluated three models: two apply the bi-encoder architecture and one applies the cross-encoder architecture. Both bi-encoder models, i.e., E5 (Wang et al., 2022) and dual bi-encoder (Wu et al., 2020a), rely on BERT (Devlin et al., 2019). E5 uses one encoder model for the query and the documents, and is already pretrained on the ranking task. For the second model, which we term the dual bi-encoder, we use an architecture that applies two encoder models, where one is used to encode queries and one is used to encode documents. This model uses BERT as a foundational model for both the query and document encoders. For the cross-encoder, we use BERT as well. All foundational BERT models share the same architecture, i.e., the number of parameters remains the same. For this, we use the version from Hugging Face[4] as BERT implementation and apply in-batch negative sampling (Wu et al., 2020a) during training. We trained these models on a server with 128 GB of RAM and an NVIDIA RTX H100 GPU with 80 GB of RAM. Instead of switching to hard negatives, we indexed all entities into a Faiss index (Douze et al., 2025) at each half-epoch to extract hard negatives. Furthermore, during training, considering the AIDA, Mintaka, and LC-QuAD datasets, we generated each batch including the documents with high-scoring documents for each query. Herein, we maintained the in-batch strategy throughout training, with random negatives used in the first half-epoch. We trained the model for ten epochs using the default learning rate and parameters for the dual bi-encoder and E5 models. For the evaluation of the biencoder models, the document embeddings are generated by the model and indexed by applying the Faiss framework.

[3] https://microsoft.github.io/msmarco/
[4] https://huggingface.co/docs/transformers/model_doc/bert, https://huggingface.co/intfloat/e5-base-v2

For the MS MARCO dataset, we apply a similar strategy. However, instead of applying all documents to compute the hard negatives, we randomly selected 10,000 negative documents from the entire corpus each time we re-indexed the Faiss index. For each query in the training batch, we selected up to four negatives from the index and one positive document. We used two queries per batch and kept the in-batch training strategy for the other datasets as well. Similar to bi-encoder models, we trained the cross-encoder model for 10 epochs with 1000 optimization steps per epoch. Since the main objective of this work is to evaluate the influence of noise on optimization, we had to fine-tune and evaluate a large set of models. For this reason, we decided to evaluate the model on a split of 125,000 randomly selected documents from the whole MS MARCO document corpus.

For evaluating and training cross-encoder models, we used a fine-tuned model of E5 without noisy optimization to compute negatives. For all queries, we generated 100 negative samples for each in both the training and test sets. At training time, we selected 14 negatives per sample, plus one positive, to generate a candidate set of 15 for each query. The negatives are randomly selected from the precomputed negative set. For evaluation, we used all 100 negatives for each query.

5.3.3 Analysis of the Effectiveness of Noise Injection ($\mathbf{S}_1$)

In Tables 5.2, 5.3, and 5.4 we present the results of various noise injection strategies on the two different bi-encoder models E5 and dual bi-encoder and the cross-encoder model across the datasets, MS MARCO, LC-QuAD 2.0, Mintaka, and AIDA.[5] Herein, we also compute the statistical significance of the results. This was computed by comparing per-query MRR and Recall scores of each noise-injection variant against the *No Noise* baseline using a paired two-tailed t-test (Lehmann, 1992). Herein, we see that E5 demonstrates consistent performance improvements with noise injection (Table 5.2). For instance, with cross-entropy loss, the anticorrelated noise using the gradient (Anti-Grad) yields the best MRR and recall on MS MARCO, Mintaka, and AIDA, while the anticorrelated noise using the previous noise term (Anti-Prev) achieves the best results on the LC-QuAD dataset. We see a similar trend considering the pairwise ranking loss as well, whereas Anti-Prev achieves more statistically significant improvements. Therefore, our results indicate the overall effectiveness of noise injection into the Adam optimizer across diverse datasets for the E5 model. The dual bi-encoder model, which contains more parameters than E5, shows greater sensitivity to noise injection (Table 5.3). For instance, we observe that Gaussian noise yields the best MRR and recall in MS MARCO and in AIDA, whereas Anti-Grad performs best on the LC-QuAD and Mintaka datasets. The effectiveness of noise injection can also be

[5] Due to the brevity, we denote the corresponding datasets in the tables as MS MARCO, LC-QuAD, Mintaka, Aida respectively.

Table 5.2: MRR and recall results evaluated on four ranking datasets using the E5 architecture, comparing models trained without noise and with various noise injection strategies. Bolded values indicate the best performance for each dataset. *Significance* columns report statistical significance against the *No Noise* baseline (*: $p < 0.05$, **: $p < 0.01$, n.s.: not significant). LF signifies loss functions: cross entropy (CE) and pairwise loss (PR)

(a) MRR Results

LF	Approach	MS MARCO	LC-QuAD	Mintaka	Aida	Significance
CE	No Noise	0.6423	0.8767	0.2672	0.3049	–
	Gaussian	0.6327	0.8761	0.2675	0.2971	n.s.
	Anti-Grad	**0.6708**	**0.8972**	**0.2704**	**0.3072**	*
	Anti-Prev	0.6534	**0.8972**	0.2651	0.3034	n.s.
PR	No Noise	0.7456	0.8841	0.2948	0.3009	–
	Gaussian	0.7566	0.8896	0.2895	**0.3035**	n.s.
	Anti-Grad	0.7332	0.9000	0.2939	0.3022	*
	Anti-Prev	**0.7582**	**0.9056**	**0.3001**	0.3028	**

(b) Recall Results

LF	Approach	MS MARCO	LC-QuAD	Mintaka	Aida	Significance
CE	No Noise	0.8475	0.8719	0.3658	0.1668	–
	Gaussian	0.8406	0.8725	0.3688	0.1622	n.s.
	Anti-Grad	**0.8658**	0.8835	**0.3808**	**0.1674**	*
	Anti-Prev	0.8565	**0.8836**	0.3628	0.1638	n.s.
PR	No Noise	0.8867	0.8888	0.4016	0.1653	–
	Gaussian	0.8675	0.8841	0.3903	0.1651	n.s.
	Anti-Grad	0.8772	**0.8990**	0.4000	**0.1681**	*
	Anti-Prev	**0.8852**	0.8943	**0.4109**	0.1668	**

observed in both the cross-entropy and pairwise ranking losses. Herein, we see that both Anti-Grad and Gaussian noise injection approaches achieve statistically significant results. Cross-encoder models, on the other hand, already achieve very high recalls and MRRs on the LC-QuAD, Mintaka, and Aida datasets. Therefore, we do not see a substantial improvement over the results after injecting noise into the parameters. However, on the largest dataset, i.e., on MS MARCO, the effect of noise becomes quite prominent. Here, we see that noise injection leads to an almost **~5%** improvement in recall over the no-noise setting. This trend is observed considering both the cross-entropy and pairwise ranking loss. Thus, the findings of the evaluation have led us to the observation that the effectiveness of noise injection depends on the ranking model and the dataset it is trained on. To investigate this further,

Table 5.3: MRR and recall of the Dual Bi-Encoder model. The evaluation setup is the same as in table 5.2

(a) MRR Results

LF	Approach	MS MARCO	LC-QuAD	Mintaka	Aida	Significance
CE	No Noise	0.6479	0.9491	0.4594	0.4414	–
	Gaussian	**0.6712**	0.9481	0.4625	**0.4635**	*
	Anti-Grad	0.6643	**0.9513**	**0.4642**	0.4633	*
	Anti-Prev	0.6555	0.9500	0.4496	0.4553	n.s.
PR	No Noise	0.8001	0.9401	0.4771	0.4615	–
	Gaussian	**0.8331**	0.9432	0.4955	0.4866	**
	Anti-Grad	0.8305	**0.9572**	**0.4994**	**0.4872**	**
	Anti-Prev	0.8220	0.9571	0.4801	0.4785	*

(b) Recall Results

LF	Approach	MS MARCO	LC-QuAD	Mintaka	Aida	Significance
CE	No Noise	0.8329	0.9369	0.5725	0.2794	–
	Gaussian	**0.8529**	0.9363	0.5752	**0.2882**	*
	Anti-Grad	0.8458	**0.9372**	**0.5783**	0.2832	*
	Anti-Prev	0.8406	0.9361	0.5745	0.2828	n.s.
PR	No Noise	0.9019	0.9388	0.5898	0.3001	–
	Gaussian	0.9223	0.9388	0.5883	0.3112	*
	Anti-Grad	0.9189	0.9388	**0.5993**	0.3106	*
	Anti-Prev	**0.9208**	**0.9418**	0.5805	**0.3194**	*

we observe a **~2-3%** performance gain for bi-encoder models and an almost **~5%** gain for the cross-encoder on the MS MARCO dataset. This suggests that introducing noise allows the Adam optimizer to generalize better as the dataset size increases. Furthermore, we observe that the model size, measured by the number of parameters, also affects the noise injection. In particular, we see that the dual bi-encoder, which has more parameters than E5, achieves better generalization. For the cross-encoder model, the results show substantial improvement from noise injection into the optimization process. To this end, we observe that on the MS MARCO dataset, noise injection can lead to an almost **~5%** gain in performance. Note that on other datasets, such as Aida, Mintaka, and LC-QuAD, the noise injection does not significantly improve performance. Nonetheless, on those datasets, the cross-encoder model already achieves very high recalls and MRRs; therefore, the room for further improvement through noise injection is limited. This suggests a diminishing returns effect, where performance gains from regularization techniques such as noise injection are more pronounced when the base model underperforms. In contrast, when the model already

Table 5.4: MRR and recall of the Cross-Encoder model. The evaluation setup is the same as in table 5.2

(a) MRR Results

LF	Approach	MS MARCO	LC-QuAD	Mintaka	Aida	Significance
CE	No Noise	0.0441	0.9731	0.9362	0.9941	–
	Gaussian	0.0532	0.9720	**0.9392**	0.9937	*
	Anti-Grad	**0.0643**	0.9739	0.9368	0.9938	**
	Anti-Prev	0.0614	**0.9749**	0.9364	**0.9956**	**
PR	No Noise	0.0891	0.9881	0.9415	**0.9993**	–
	Gaussian	**0.0922**	0.9817	**0.9511**	0.9944	*
	Anti-Grad	0.0800	0.9800	0.9416	0.9941	n.s.
	Anti-Prev	0.0777	**0.9818**	0.9413	0.9961	n.s.

(b) Recall Results

LF	Approach	MS MARCO	LC-QuAD	Mintaka	Aida	Significance
CE	No Noise	0.1541	**0.9818**	**0.9992**	0.9174	–
	Gaussian	0.1785	0.9816	**0.9992**	0.9168	n.s.
	Anti-Grad	0.1962	0.9812	**0.9992**	0.9158	*
	Anti-Prev	**0.2017**	0.9816	**0.9992**	**0.9183**	**
PR	No Noise	0.1672	0.9900	0.9994	0.9245	–
	Gaussian	0.1809	**0.9922**	0.9992	**0.9308**	*
	Anti-Grad	0.2005	0.9876	0.9992	0.9215	*
	Anti-Prev	**0.2173**	0.9887	**0.9995**	0.9201	**

achieves near-optimal performance, additional regularization contributes only marginal benefits, aligning with existing literature (Arpit et al., 2017; Patrini et al., 2017). Overall, our results indicate that models like dual bi-encoders and cross-encoders benefit more from noise injection when their baseline performance leaves room for improved generalization; otherwise, noise injection does not lead to substantial improvements in recall and MRR.

5.3.4 Comparison of Noise Injection Strategies ($\mathbf{S}_2$)

Considering the Tables 5.2, 5.3, and 5.4, we observe that the anticorrelated noise injection approaches perform better than the Gaussian noise injection approach. Specifically, Anti-Prev often shows better statistical significance across both cross-entropy and pairwise ranking losses. However, Anti-Grad performs on par with Anti-Prev and is most effective when applied to the cross-entropy loss. More specifically, Anti-Prev leverages the noise added in the

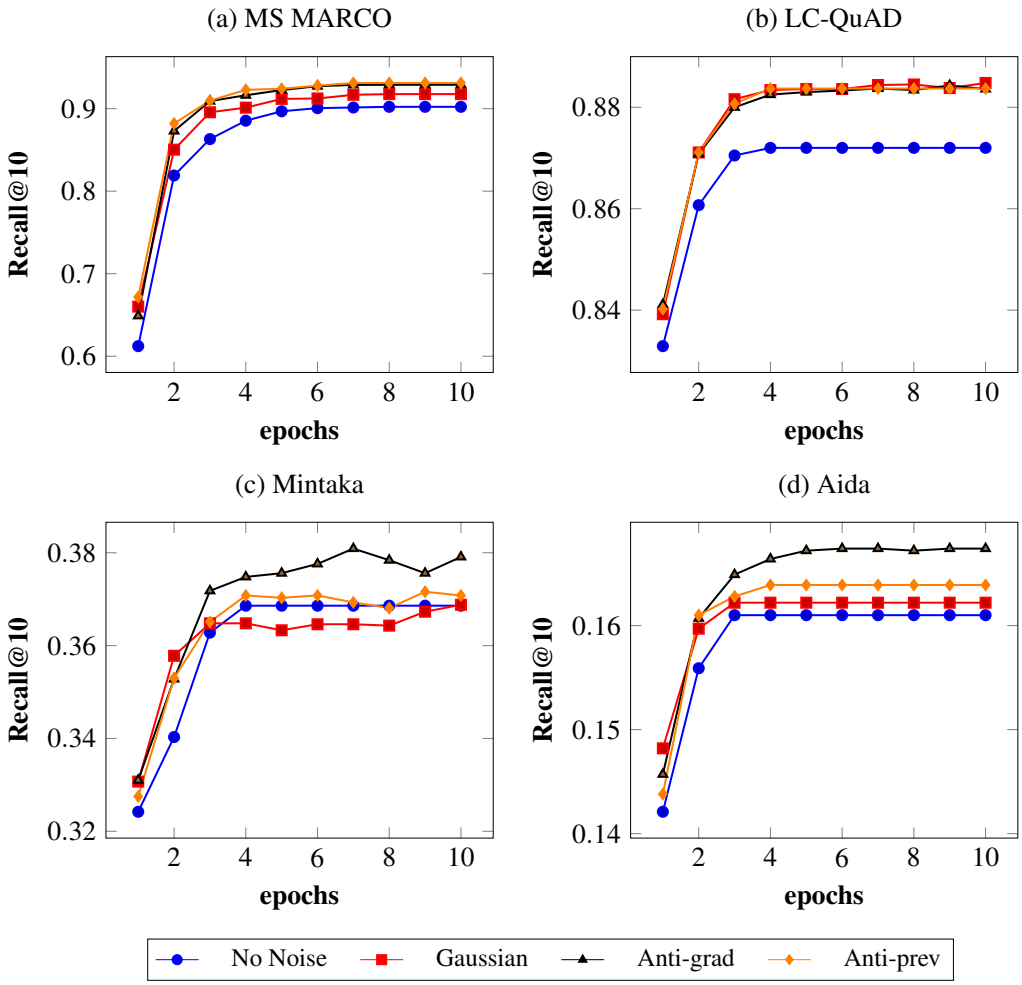


Figure 5.1: Recall@10 over epochs for different datasets using E5-encoder under various noise injection strategies.

previous iteration to inject a new noise term that is anticorrelated. This introduces a form of temporal regularization, creating stable noise injection during training. Unlike independent Gaussian noise or gradient-based noise, this can help in maintaining smooth optimization trajectories, especially in overparameterized models like cross-encoders. Therefore, we observe that in the cross-encoder model, Anti-Prev consistently yields statistically significant results compared to other approaches, considering the MS MARCO dataset. This further indicates that Anti-Prev is particularly effective when the model has already learned strong representations and only needs regularized fine-tuning to avoid overfitting. On the other hand, the effectiveness of Anti-Grad stems from its dynamic adaptation to the optimization landscape. In particular, this can be attributed to the ability to adjust noise based on the loss by (i) reducing noise during steep descents where the gradient magnitude is large, ensuring that the optimizer follows the correct direction for faster convergence, and (ii) increasing

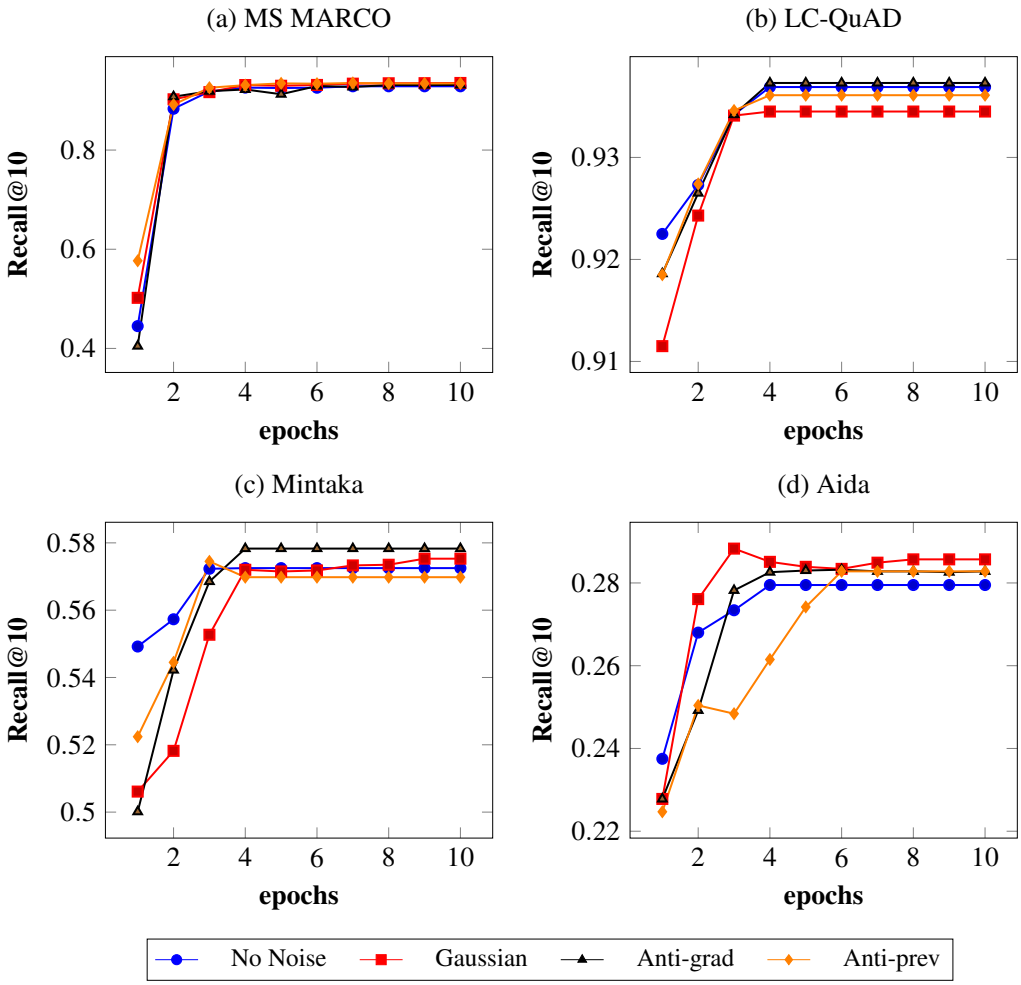


Figure 5.2: Recall@10 over epochs for the dual Bi-encoder model across four datasets under various noise injection strategies.

noise in flatter regions or near convergence, which helps the model escape saddle points or shallow local minima that might otherwise trap traditional optimizers. Therefore, such adaptability makes Anti-Grad perform better than injecting uncorrelated noise. Considering these outcomes, we observe that anticorrelated noise injection approaches, i.e., Anti-Grad and Anti-Prev, perform better than the uncorrelated noise injection approaches, considering both the cross-entropy and pairwise ranking loss. In Section 5.4, we provide a detailed discussion on which noise injection strategy is useful for specific models and datasets.

5.3.5 Influence of Noise Injection on Convergence ($\mathbf{S}_3$)

The Figures 5.1, 5.2, and 5.3 show the stability and convergence behavior of noise injection strategies for E5, dual bi-encoder, and cross-encoder models, considering the cross-entropy

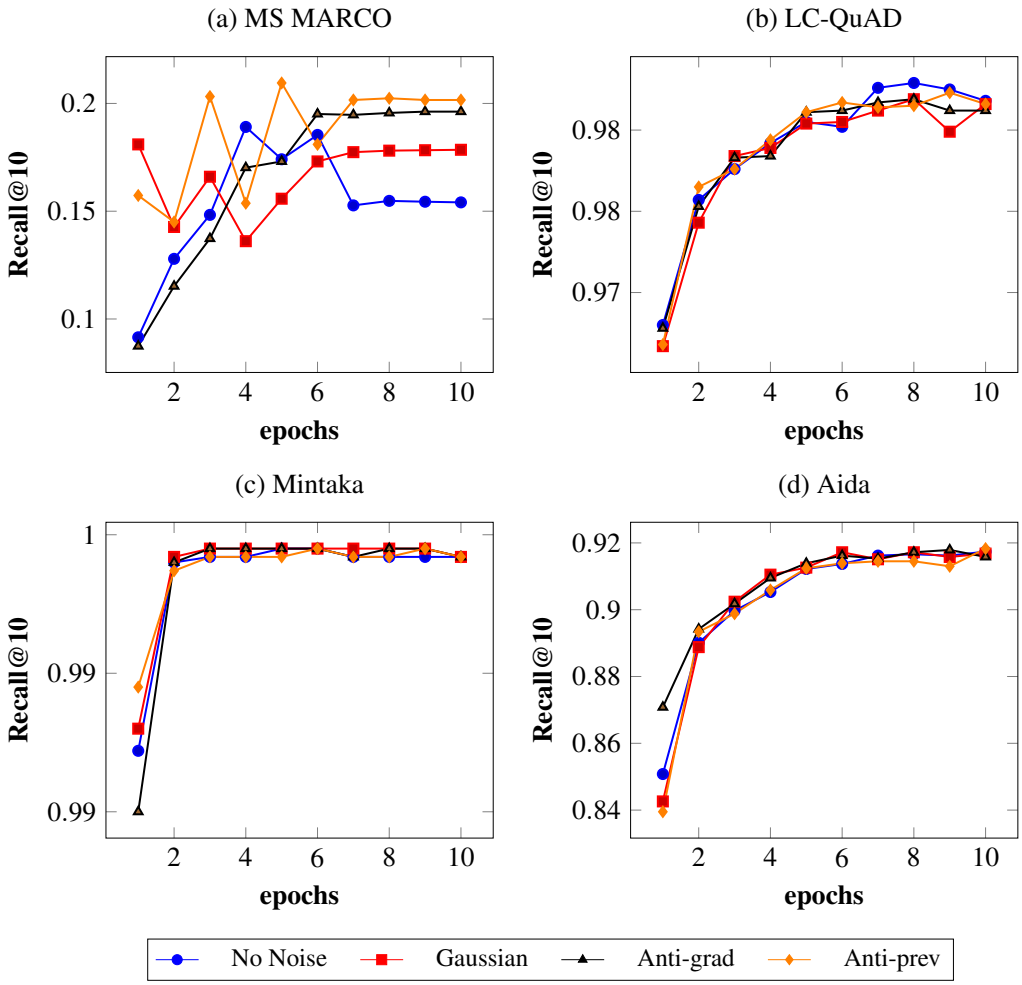


Figure 5.3: Recall@10 over epochs for the Cross-encoder model across four datasets under various noise injection strategies.

loss.[6] Among the bi-encoder models, E5 exhibits increased stability and faster convergence when noise is introduced (Figure 5.1), particularly with the anti-grad strategy, whereas dual bi-encoder, with its greater model complexity, shows varying sensitivity to different noise types (Figure 5.2). In the cross-encoder model, we observe that noise injection leads to faster convergence and helps mitigate overfitting. More specifically, for the E5 model, trained on the largest dataset MS MARCO, we observe that anti-grad noise leads to rapid and steady convergence, achieving peak performance at epoch 5 with minimal fluctuations thereafter (Figure 5.1a). In the dual bi-encoder, Gaussian noise provides the most stable performance on MS MARCO (Figure 5.2a). Without noise, the training requires at least 8-9 epochs to achieve the best results for both ranking models. To this end, we find that the

[6]Note that, when using pairwise ranking loss, we observe a similar trend in the result. Therefore, due to brevity, we only report the results considering cross-entropy loss.

cross-encoder benefits significantly from noise injection. More specifically, as shown in Figure 5.3a, when trained on the MS MARCO dataset, the cross-encoder model achieves higher recall with noise injection and converges faster with the Anti-Prev approach. Most importantly, noise injection approaches can mitigate overfitting, which is prevalent when using the noise-free approach. Specifically, we see that after the sixth epoch, the recall value drops considerably when no noise injection is used; however, when noise injection is used, the model stabilizes and does not overfit. Considering the LC-QuAD dataset, both Anti-Grad and Anti-Prev maintain stable recall and MRR after epoch 4, indicating that anticorrelated noise helps the model converge faster across both bi-encoder models. Apart from these two large ranking datasets, we also see a similar trend in the Mintaka and AIDA datasets. For both of these two datasets, Anti-Grad outperforms all the other approaches by converging within four to five epochs in both E5 and bi-encoder. Note that, since the cross-encoder already achieves a high recalls, as discussed before, the performance gain in terms of faster convergence is not substantial. Nonetheless, our results suggest that an adaptive noise strategy such as Anti-Grad is particularly effective in datasets of moderate complexity, providing both performance gains and faster convergence. Overall, our results show that noise injection accelerates early-stage convergence for the bi-encoder models, yielding noticeable gains in recall across all datasets. For the cross-encoder model, noise injection not only leads to much higher recall and faster convergence, but also helps reduce overfitting.

5.4 Optimal Noise Injection per Model and Dataset

Table 5.5: Best noise injection approach per dataset–model combination.

Model	MS MARCO	LC-QuAD	Mintaka	Aida
E5	Anti-Prev	Anti-Prev	Anti-Prev	Anti-Grad
Bi-Encoder	Gaussian	Anti-Grad	Anti-Grad	Anti-Grad
Cross-Encoder	Gaussian	Anti-Prev	Gaussian	Anti-Prev

When comparing results across loss functions, we find that different noise injection approaches work differently for models and datasets. For the E5 model, Anti-Prev consistently yields the highest MRR on MS MARCO , LC-QuAD, and Mintaka, suggesting that stabilizing updates across training steps helps this architecture maintain ranking consistency, more specifically in structured or moderately noisy datasets. However, in the case of Aida, Anti-Grad performs the best, likely due to its ability to actively counteract misleading gradients in entity-centric retrieval. In the bi-Encoder setting, Gaussian noise is optimal for MS MARCO, indicating that noise injection can help avoid overfitting in large, diverse datasets.

In contrast, Anti-Grad dominates LC-QuAD, Mintaka, and Aida, reflecting the benefits of gradient correction in smaller, more structured collections. For the Cross-Encoder, Gaussian noise performs better on MS MARCO and Mintaka. At the same time, Anti-Prev is superior on LC-QuAD and Aida, where stable, incremental parameter updates appear to preserve fine-grained semantic alignment better.

5.5 Conclusion

In this work, we have studied the use of noise injection in the Adam optimizer to improve the performance and convergence speed of ranking models. By incorporating noise into model parameters during training through different strategies, we demonstrated that noise can act as an effective regularizer, helping the model escape saddle points and converge to flatter minima. Our experimental results across four different ranking datasets and two different ranking frameworks (bi-encoder and cross-encoder) consistently showed that noise-enhanced Adam optimization leads to faster convergence and mitigates the overfitting problem. These findings indicate that noise injection is a promising strategy for improving the training of ranking models in information retrieval.

While our work has provided valuable insights into the benefits of noise injection for ranking, several research directions remain to be explored. For instance, using an adaptive strategy, the level of noise and the hyperparameters (related to the anticorrelated approaches) can be adjusted. More specifically, a Bayesian learning strategy could be used therein to select the optimal noise injection parameters. Moreover, the effect of noise could be further studied to determine an optimal bound on model complexity and dataset size beyond which the effect might diminish. Therefore, we envisage that a notion of balanced perturbation needs to be further studied, which can help to determine optimal noise levels and methods for dynamically adjusting perturbations based on training dynamics, loss curvature, model metrics, and datasets. Finally, we envisage exploring the role of curriculum-based noise scheduling, where noise intensity is adapted to the model's confidence. Additionally, integrating such techniques while considering contrastive learning objectives may offer further improvements.

6 Keyphrase Extraction using Language Models and Knowledge Graphs

This chapter addresses research question $\mathbf{RQ}_1$: *What methods can be used to predict knowledge that is implicit rather than explicitly expressed in questions?*. In this chapter, we first present the framework MultPAX (Zahera et al., 2022) for present and absent keyphrase extraction, which applies entity disambiguation to link present keyphrases to KGs for extracting additional information as absent keyphrases. For this thesis, we will modify this approach to predict absent knowledge through contextual augmentation and by applying the ED approach presented in this chapter to link this absent knowledge to a KG. The author of this thesis co-designed, co-implemented, and co-wrote the corresponding paper.

6.1 Overview

Formulating search queries that return complete yet relevant results remains a common challenge in complex search scenarios (Gabín and Parapar, 2025). To this end, keyphrases are an important feature to enhance the performance of IR systems (Boudin et al., 2020). However, in a KGQA system, queries are natural language questions, so automatic keyphrase extraction can enhance the performance of knowledge extraction from KGs. In the literature, keyphrase extraction is divided into two sub-tasks: (i) detecting present keyphrases (PKE) that appear in a document, and (ii) generating absent keyphrases (AKG) that do not appear in the original document, but are essential for downstream applications (e.g., text summarization, indexing) (Ray Chowdhury et al., 2022). Table 6.1 shows an example of extracting present and absent keyphrases from an input text. The capability to extract absent keyphrases is crucial for a KGQA system, since entities and relations are often mentioned implicitly in questions as already presented in Section 2.4.3.

Consequently, the goal of this chapter is to develop an approach to *extract* present keyphrases for textual input and *generate* relevant absent keyphrases that do not appear in the input text. We describe our MultPAX approach for keyphrase extraction introduced in 2022. MultPAX reduces the effort required to develop a keyphrase model by employing pre-computed

Table 6.1: An example of present and absent keyphrase extraction from Inspec dataset. The predicted present keyphrases are in *italic*, and the absent ones are highlighted in gray.

Input Text	*"This paper shows the importance that management plays in the protection of information and in the planning to handle a security breach when a theft of information happens. Recent thefts of information that have hit major companies have caused concern. These thefts were caused by companies' inability to determine risks associated with the protection of their data, and these companies lack of planning to properly manage a security breach when it occurs." quoted from (Polstra III, 2005)*
Ground-truth Keyphrases	security breach, risk analysis, management issue, theft of information
Predicted Keyphrases	*security breach*, *theft of information*, security management, security risk, data management

resources. In particular, we use *pre-trained language models* to extract present keyphrases and *knowledge graphs* (KGs) to generate absent keyphrases. MultPAX processes of the following pipeline: i) We tokenize an input document into n-gram phrases and embed both (*document and n-gram phrases*) as low-dimensional vectors into one semantic space. Then, we *extract* the top-k phrases that are close to the document's vector as candidates for present keyphrases. ii) We then *link* the extracted present keyphrases to find additional related terms (e.g., synonyms, hypernyms) from external KGs (e.g., DBpedia, BabelNet). For this purpose, we developed a new version of the MAG framework (Moussallem et al., 2017), which is optimized for linking keywords and extracting related terms. iii) Finally, we *rank* all keyphrases (i.e., *present* and *absent*) based on their semantic similarity to the input document. The top-k phrases are returned as the final keyphrases output. In the second step, we adapt this approach for keyphrase extraction from questions to enhance KG resource extraction for KGQA. To evaluate the performance of MultPAX, we conducted several experiments across four benchmark datasets, comparing our system against different approaches. The evaluation results show that our approach significantly outperforms the state-of-the-art baselines with a significance t-test $p < 0.041$ and F1-score up to 0.535.

The main contributions in this paper can be summarized as follows:

- We propose an unsupervised multitask framework that is capable of predicting absent keyphrases based on present keyphrases.

- We leverage knowledge graphs for keyphrase generation without the need to create keyphrase vocabularies or phrase banks.
- We introduce an embedding-based F1 metric that assesses the semantic similarity between generated and ground-truth keyphrases, moving beyond traditional exact-matching scores.
- We conducted several experiments across four benchmark datasets. The evaluation results show that our approach is more accurate compared to state-of-the-art baselines.

6.2 Approach

In this section, we present our approach for extracting present and absent keyphrases. Figure 6.1 depicts the architecture of our MultPAX framework, including three components: i) `Present Keyphrase Extraction` (PKE), ii) `Absent Keyphrase Generation` (AKG), and iii) `Keyphrases Semantic Matching`. The main contribution of the author is the approach for absent keyphrase generation, which is the main focus of this chapter.

6.2.1 Problem Formulation

Let D be an input document with $|S|$ sentences; each sentence $s \in S$ is a sequence of $|s|$ tokens $T = \{t_1, t_2, \cdots, t_{|s|}\}$. Our goal is to build a keyphrase model that not only extracts *present keyphrases* $Y^p = \{y^p_1, y^p_2, \cdots, y^p_{|Y^p|}\}$ but also generates *absent keyphrases* $Y^a = \{y^a_1, y^a_2, \cdots, y^a_{|Y^a|}\}$ that are relevant to D by leveraging knowledge graphs such as DBpedia and BabelNet (Navigli and Ponzetto, 2012).

Following previous works (Gollapalli et al., 2017; Sahrawat et al., 2020), we divide the task of keyphrase extraction into two sub-tasks: `Present Keyphrase Extraction` (PKE) and `Absent Keyphrase Generation` (AKG). Furthermore, we define the computation of final keyphrases as a `Semantic Matching` task. First, we consider PKE as a *ranking* problem, where candidate phrases are extracted and then ranked based on their similarities to the input document (see Section 6.2.2). Second, we formulate AKE as a *linking* problem to infer relevant information from external knowledge graphs. For this task, we employ an unsupervised *entity linker* (Shen et al., 2014) that maps a present keyphrase (Y^p) to its corresponding entity in a knowledge graph (i.e., DBpedia, BabelNet) and then gets relevant terms (e.g., from *dct:subject*, *gold:hypernym* properties) as candidates for

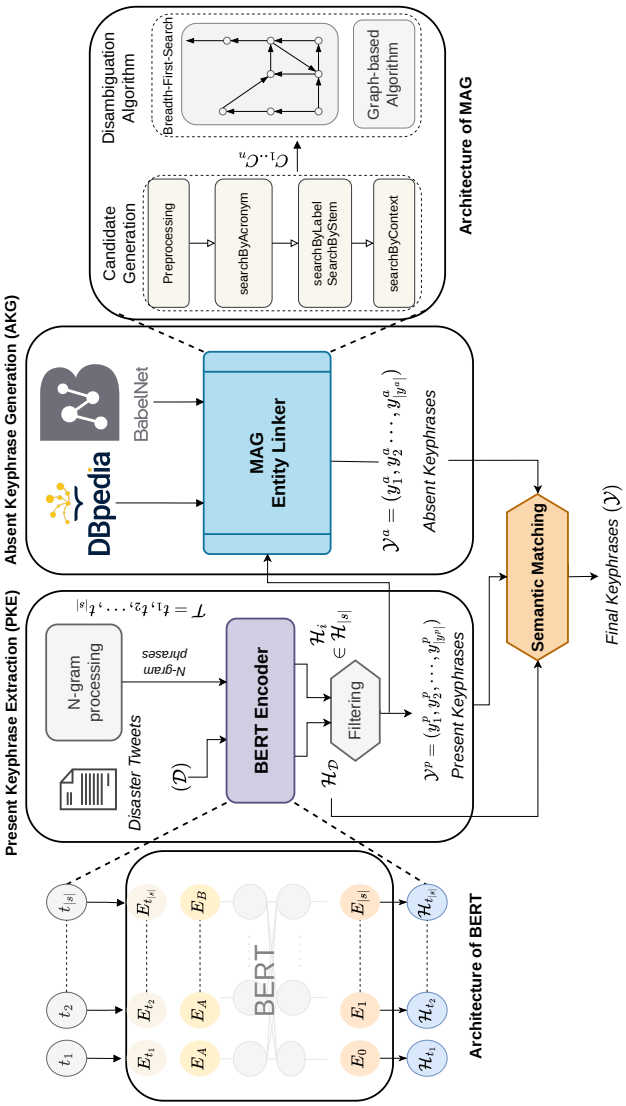


Figure 6.1: The architecture of MultPAX framework with three components: `Present Keyphrase Extraction`, `Absent Keyphrase Generation` and `Semantic Matching`.

absent keyphrases. Finally, all keyphrases $Y^p \cup Y^a$ are ranked based on their similarities to D, and the top-k keyphrases are returned as the final output.

6.2.2 Present Keyphrase Extraction (PKE)

Since the absent keyphrase extraction is based on the present keyphrase extraction, we briefly describe the present keyphrase extraction here. This step first preprocesses the

document to extract n-grams as potential keyphrase candidates. Then BERT is applied to generate embedding vectors for all candidate keyphrases and the input document. Finally, the cosine-distance between all keyphrase embeddings and the document embedding is computed, so that the keyphrases can be ranked according to their similarity with the input document, and the top k keyphrases with the highest similarity are selected as the set of present keyphrases Y^p. For all details, we refer to Zahera et al. (2022).

6.2.3 Keyphrase Linking and Absent Keyphrase Generation (AKG)

To obtain absent keyphrases, we first link all present keyphrases Y^p to a knowledge graph and get additional surface forms (i.e., strings that could be synonyms or alternative names). We consider the DBpedia knowledge graph since it provides surface forms for a wide range of common entities. For entity linking, we follow a similar approach to the MAG framework (Moussallem et al., 2017).

MAG extracts entity links using two steps: candidate generation and candidate disambiguation. In the candidate generation step, MAG aims to find candidate links $(C_1, \ldots, C_n)$ for premarked entities in the search index (Moussallem et al., 2017). To this end, MAG uses acronyms and labels in a knowledge graph to map premarked entity spans from the input text to candidate entities. Furthermore, MAG also relies on the concise bounded description (CBD)[1] of the entities in a knowledge graph by comparing the context of the entity spans in the input document and the CBD of an entity in a knowledge graph (Moussallem et al., 2017). We keep the candidate generation step from MAG and apply it to the extracted keyphrases from the PKE component. In the candidate disambiguation step, MAG generates a local graph including all candidate entities from a knowledge graph using breadth-first search. Then, MAG applies the HITS ranking algorithm (Kleinberg, 1999) to jointly rank the candidate links for all entities in the local graph. HITS ranks the nodes in a directed graph based on incoming and outgoing edges. Authorities are nodes that carry important information, while hubs are nodes that point to many authority nodes. So the authority score of a node n is calculated based on the hub scores of the nodes that have a directed edge to n, while the hub score of n is calculated based on the authority scores of the nodes linked by n (Kleinberg, 1999). Formally, HITS calculates the authority score a_p for the node p as

$$a_p = \sum_{q:(q,p)\in G} h_q. \tag{6.1}$$

[1] https://www.w3.org/Submission/CBD/

where h_q is the hub-score for the node q, given that a directed edge from node q to node p exists in the graph G. The hub-score h_p for a node p is calculated as

$$h_p = \sum_{q:(q,p)\in G} a_q. \tag{6.2}$$

where a_q is the authority-score for a node q, which is linked by node p (Kleinberg, 1999). a_q and h_p are initialized randomly and updated iteratively until convergence.

In contrast to MAG, we not only link present keyphrases, but also extract related terms for each linked keyphrase from a knowledge graph. Furthermore, we extract the top-ranked candidates for each entity and n nodes with the highest authority scores in the local graph, since their surface forms could serve as candidates for absent keyphrases. In our approach, we use BabelNet to find hypernyms for the present keyphrases, in addition to the surface forms from DBpedia.

6.2.4 Keyphrases Semantic Matching

In the last component, we aim to identify top-k relevant keyphrases (*present* and *absent*), we set $k = \{5, 10, 20\}$ in our experiments. We regard this task as a semantic textual similarity (Majumder et al., 2016). To match similarities between a document D and candidate keyphrases, we embed them into one semantic space using a pre-trained embedding model. Then we employ cosine distance to find top-k nearest keyphrases (H_i) to the document's vector H_D and return as final keyphrase predictions. Formally,

$$\text{Cos}(H_i, H_D) = \frac{H_i \cdot H_D}{||H_i|| \times ||H_D||}. \tag{6.3}$$

where H_i donates the embedding vector of candidate keyphrase (*present* y_i^p or *absent* y_i^a), and H_D represents the embedding vector of the input document.

6.3 Experiments

In this chapter, we conduct experiments on the keyphrase extraction task, which is relevant for the following research question: $\mathbf{RQ}_1$: *What methods can be used to predict knowledge that is implicit rather than explicitly expressed in questions?* To investigate how absent knowledge can be predicted from text, we answer the following subquestions:

Table 6.2: Statistics about the datasets (#Doc: number of documents, #Test: size of test set, #Avg. KP: average keyphrase per document, #Ratio%: percentage of absent keyphrase per dataset).

Dataset	#Doc	#Test	Avg. KP	Ratio%
Inspec	2k	500	7.65	37.7%
Krapivin	2.3k	460	3.03	15.3%
SemEval2010	144	100	7.15	11.3%
NUS	211	211	2.71	17.8%

S$_1$. How good does the absent keyphrase generation approach based on the MAG-framework perform against baseline approaches, and are the existing exact-matching metrics (Precision, Recall, and F1-score) suitable for evaluating absent keyphrases?

S$_2$. To what extent does the absent keyphrase generation contribute to the overall performance of MultPAX?

6.3.1 Experimental Setup

Datasets In our experiments, we used four benchmark datasets of English documents, namely, Inspec (Hulth, 2003), SemEval2010 (Kim et al., 2010), Krapivin (Krapivin and Marchese, 2009), and NUS (Nguyen and Kan, 2007). Table 6.2 provides a statistical overview of each dataset, including the total number of documents (#Doc.), the number of documents in the evaluation set (#Test), average keyphrases per document (Avg. KP), and the ratio of absent keyphrases in each dataset (Ratio%).

Baselines We compared our approach against the following baselines for extracting keyphrases:

- **TextRank** (Mihalcea and Tarau, 2004) is an unsupervised approach that constructs a graph representation of a document, where nodes represent phrases and edges are computed based on lexical similarity. Further, TextRank uses the PageRank algorithm to extract present keyphrases.
- **YAKE** (Campos et al., 2020) is a simple unsupervised method that automatically extracts keywords based on statistical features, such as word co-occurrence and frequency.

- **EmbedRank** (Bennani-Smires et al., 2018) is an unsupervised method that employs word embeddings to identify relevant words to a document as candidate keyphrases. Furthermore, EmbedRank utilizes the Maximum Marginal Relevance algorithm to increase the diversity of the extracted keyphrases.
- **Supervised-CopyRNN** (Meng et al., 2017) is a supervised baseline that trains a sequence-to-sequence model with a *copy mechanism* on KP20K dataset (Meng et al., 2017). We used this approach as a baseline for present keyphrases extraction as well as absent keyphrase generation to compare the performance of the copy mechanism.
- **AutoKeyGen** (Shen et al., 2022) is an unsupervised approach that constructs a phrase bank by combining keyphrases from all documents into a corpus. Then, AutoKeyGen considers lexical- and semantic-level similarities for selecting top candidate keyphrases (present and absent) for each input document.

Evaluation Metrics We evaluated our approach using the following metrics: Precision, recall, and F1, as introduced in Section 2.5, between predicted and gold keyphrases. To this end, we select the top-k ranked keyphrases $Y_{:k} = (y_1, \ldots, y_{\min(k,|Y|)})$ and compare with the top-k ranked keyphrases in the ground-truth set. We set $k = \{5, 10\}$ for present keyphrases and $k = \{10, 20\}$ for absent ones in our experiments. Following previous works (Shen et al., 2022; Ye and Wang, 2018), we use the Porter Stemmer from the NLTK library[2] v3.7 to compute exact-matching between the top-k predicted ($Y_{:k}$) and the ground-truth (Y^{gold}) keyphrases. Although the exact-matching metric has been widely used in the literature (Liang et al., 2021), there is still room for improvement in the evaluation of absent keyphrases based on semantic similarity. Hence, we propose a semantic-based matching to evaluate the performance of generated absent keyphrases in Section 6.3.2. To provide an overview of the results of the present keyphrase extraction, Table 6.3 shows the results of the present keyphrase extraction. The results outperform the baseline approaches on most datasets.

Hyperparameters We performed a grid search to optimize the hyperparameters of our approach. We found that the following values yield the best F1 Scores. In the PKE component, we tokenized the input text into phrases of two to four n-grams. Further, we considered the top-10 ranked phrases as candidates for the present keyphrases. The full setup of our experiments is available at the GitHub repository.[3] For the baseline methods, the hyperparameters were set as specified in their original papers. In the MAG framework, we adapted the extraction of common entities to cover a larger set of entity types. In

[2] https://www.nltk.org/index.html
[3] https://github.com/dice-group/MultPAX

Table 6.3: Evaluation results of *present* keyphrases prediction on Inspec, SemEval2010, Krapivin, and NUS datasets. F1@k-scores are reported based on **exact-matching** between the predicted and groundtruth keyphrases. Best results are reported in bold.

(a) Results on Inspec and SemEval2010

Model	Inspec		SemEval2010	
	F1@5	F1@10	F1@5	F1@10
TextRank	0.263	0.279	0.183	0.181
YAKE	0.027	0.038	0.050	0.242
EmbedRank	0.295	0.344	0.108	0.145
Supervised-CopyRNN	0.292	0.336	0.291	**0.296**
AutoKeyGen	0.303	**0.345**	0.187	0.240
MultPAX	**0.371**	0.210	**0.449**	0.255

(b) Results on Krapivin and NUS

Model	Krapivin		NUS	
	F1@5	F1@10	F1@5	F1@10
TextRank	0.148	0.139	0.187	0.195
YAKE	0.013	0.020	0.013	0.020
EmbedRank	0.131	0.138	0.103	0.134
Supervised-CopyRNN	0.302	0.252	0.342	0.317
AutoKeyGen	0.171	0.155	0.218	0.233
MultPAX	**0.384**	**0.334**	**0.535**	**0.344**

addition, we set the other hyperparameter values with the standard configuration of the MAG framework.[4]

6.3.2 Absent Keyphrase Evaluation ($\mathbf{S}_1$)

We conduct further experiments to evaluate the performance of our approach against two baselines (namely, CopyRNN and AutoKeyGen) in generating absent keyphrases. Following previous work (Shen et al., 2022), we use the Recall metric (R@10, R@20) based on *exact-matching* for the performance evaluation as shown in Table 6.4. Since we used the same experimental setup of CopyRNN and AutoKeyGen approaches, we obtained the evaluation results from their papers (Meng et al., 2017; Shen et al., 2022).

[4] https://github.com/dice-group/AGDISTIS/blob/master/src/main/resources/config/agdistis.properties

Table 6.4: *Absent* keyphrases evaluation (in terms of R@10, R@20). All results are reported based on **exact-matching** between the predicted and groundtruth keyphrases, except the last row shows recall results based on **semantic-matching**.

(a) Results on Inspec and SemEval2010

Model	**Inspec**		**SemEval2010**	
	R@10	R@20	R@10	R@20
Supervised-CopyRNN	0.051	0.068	0.049	0.057
AutoKeyGen-Bank	0.015	0.017	0.007	0.009
AutoKeyGen-Full	0.017	0.021	0.010	0.011
$\text{MultPAX}_{\text{exact-Matching}}$	0.079	0.080	–	–
$\text{MultPAX}_{\text{semantic-Matching}}$	0.696	0.584	–	–

(b) Results on Krapivin and NUS

Model	**Krapivin**		**NUS**	
	R@10	R@20	R@10	R@20
Supervised-CopyRNN	0.116	0.142	0.078	0.10
AutoKeyGen-Bank	0.031	0.041	0.021	0.026
AutoKeyGen-Full	0.033	0.054	0.024	0.032
$\text{MultPAX}_{\text{exact-Matching}}$	–	–	0.017	0.017
$\text{MultPAX}_{\text{semantic-Matching}}$	–	–	0.608	0.669

Regarding S_1, we can clearly see that all approaches perform poorly when exact matching between predicted and ground-truth keyphrases is considered. For example, if two keyphrases are semantically similar, e.g., *"disaster relief organization"* and *"crisis responses institute"*, these keyphrases will not be considered as a *match* using the existing metrics. Hence, we found that such metrics are unsuitable for evaluating absent keyphrases. We propose an improved evaluation metric based on the *semantic-matching*. Formally, let Y^a be predicted keyphrases; Y^{gold} is ground-truth keyphrases. We first embed each keyphrase in Y^a and Y^{gold}. Then, we use cosine distance to compute similarities between the embedding of each keyphrase in Y and Y^{gold}. We set a threshold (> 0.5) for similarities scores to consider semantic-matching between Y and Y^{gold}. The last two rows in Table 6.4 show the evaluation results of R@10 and R@20 based on semantic-matching compared to exact-matching in absent keyphrase extraction.

The AutoKeyGen baseline demonstrates competitive performance in generating absent keyphrases on the NUS dataset. However, the keyphrases generated by AutoKeyGen are limited to those from the phrase bank of each dataset. In contrast, our approach leverages

public knowledge graphs (such as DBpedia and BabelNet) to obtain relevant phrases as candidates for absent keyphrases.

Limitations In our experiment, we used the MAG framework to connect present keyphrases to DBpedia knowledge graph (see Section 6.2.3). In the SemEval2010 and Krapivin datasets, we were unable to link present keyphrases due to the lack of coverage for these keyphrases in the DBpedia knowledge graph. That is why the last two rows of Table 6.4 show missing values for these datasets. In our future work, we plan to integrate additional knowledge graphs (e.g., YAGO and Wikidata) to expand entity linking coverage within the MAG framework.

6.3.3 Ablation Study ($\mathbf{S}_2$)

To answer $\mathbf{S}_2$, we analyzed the impact of each component of our framework on the overall performance. For this purpose, we set up four variants of our framework. The first variant MultPAX-PKE was dedicated to only extracting present keyphrases, i.e., no absent keyphrase generation and thus no linking with knowledge graphs. We also created two variants of MultPAX with the purpose of evaluating the generation of absent keyphrases, namely MultPAX-AKE$_{\text{DBpedia}}$ and MultPAX-AKE$_{\text{BabelNet}}$. Furthermore, we configured the MAG framework to link present keyphrases only with DBpedia in case of MultPAX-AKE$_{\text{DBpedia}}$, and only with BabelNet for MultPAX-AKE$_{\text{BabelNet}}$. Finally, we benchmarked the entire framework MultPAX$_{\text{Full}}$ as our fourth variant.

Table 6.5 reports the evaluation results of each component in terms of *semantic-matching* F1@5, and F1@10 on the Inspec dataset, since it contains the highest ratio of absent keyphrases among the benchmark datasets. We can see that the performance of MultPAX-PKE is improved when linking with knowledge graphs, e.g., MultPAX-AKE$_{\text{DBpedia}}$ outperforms MultPAX-PKE by $+0.41$ in F1@10. In addition, we noticed that our approach could retrieve more terms from DBpedia than BabelNet, since DBpedia contains a larger semantic ontology (approximately 3.5 million instances) extracted from Wikipedia information boxes. Finally, our MultPAX-$_{\text{Full}}$ showed an improved performance with F1-scores (0.911 in F1@5, 0.763 in F1@10) when incorporating both knowledge graphs (i.e., DBpedia and BabelNet) compared with individual variants. These findings conclude that each component of MultPAX contributes to the overall performance of our framework and answers our last research question Q_3.

Table 6.5: Ablation Study of MultPAX framework on Inspec dataset. F1@k-scores are reported based on **semantic-matching** between the predicted and Ground-truth keyphrases.

MultPAX-Variant	F1@5	F1@10
MultPAX-PKE	0.892	0.686
MultPAX-AKE$_{\text{BabelNet}}$	0.907	0.701
MultPAX-AKE$_{\text{DBpedia}}$	0.911	0.727
MultPAX$_{\text{Full}}$	**0.911**	**0.763**

6.4 Conclusion

This chapter presents MultPAX framework, a multitask approach for extracting present and absent keyphrases, including three components: i) *Present Keyphrase Extraction*, ii) *Absent Keyphrases Generation*, and iii) *Keyphrases Semantic Matching*. In our approach, we employ a pre-trained language model (Bert) and the knowledge graphs DBpedia and BabelNet for the extraction of absent keyphrases. Our experiments show that KGs proved to be valuable resources for generating keyphrases that are absent, especially in short texts. In our future work, we plan to apply a bootstrapped approach for keyphrase extraction from DBpedia abstracts to find more relevant terms. In particular, we intend to apply MultPAX recursively on the abstracts of DBpedia entities. In addition, we will experiment with other knowledge graphs (e.g., YAGO and Wikidata) to extend the coverage of entity links in the MAG framework.

In the context of this thesis, we will adapt the idea to predict keyphrases for the contextual augmentation of questions, inspired by our contextual augmentation approach for entities from Chapter 4. With the improvements of large, generative language models over the last years, applying generative LLMs for keyphrase prediction became increasingly popular (Maragheh et al., 2023; Song et al., 2024). The benefit of this approach is that instead of predicting scores for n-grams for present keyphrases extraction, generative LLMs are capable of expanding the context of an input text to predict additional information, such as absent keyphrases or alternative terminology (Gabín and Parapar, 2025). Furthermore, we will use the presented MAG-based approach for linking extracted keyphrases from questions and compare it with modern dense-retrieval and autoregressive linking approaches. We will describe the details of this adaptation in Chapter 9.

7 UniQ-Gen: Unified Query Generation across Multiple Knowledge Graphs

This chapter contributes to the following research questions:

- **RQ**$_4$: *Can LLMs be jointly fine-tuned for query generation for multiple KGs, and how do these models perform compared to models trained for single KGs?*
- **RQ**$_6$: *How does the performance of knowledge extraction frameworks influence the performance of query prediction?*

To answer Question 4, we jointly fine-tune T5 on data from two different KGs, namely Wikidata and Freebase, and compare the performance of these jointly trained models with individual models on multiple benchmarks. In the context of Question 6, we investigate to what extent incorporating outputs from knowledge extraction models into the training data improves model performance for end-to-end query generation. The content of this chapter is based on the paper Vollmers et al. (2025b), where the author designed, implemented, and evaluated the corresponding approach and co-wrote the corresponding paper.

7.1 Overview

Large language models (LLMs) have recently shown significant performance in various NLP tasks, including answering questions on knowledge graphs (KGQA) (Tan et al., 2023). These models are often fine-tuned on a domain-specific dataset (e.g., QALD-9) to convert natural text to corresponding logical forms, such as SPARQL queries (Banerjee et al., 2022) or S-Expressions (Ye et al., 2022). However, training or fine-tuning LLMs is a resource-intensive process that requires substantial computational resources, such as GPU hours (Yin et al., 2024). Current approaches typically train or fine-tune an LLM on a single domain-specific dataset or knowledge graph (Banerjee et al., 2022). However, these methods require further tuning when applied to new domains or knowledge graphs. This is due to the fact that knowledge graphs (e.g., Freebase, Wikidata, and DBpedia) exhibit significant variances in data representation. For instance, DBpedia represents *"The Alps Mountains"*

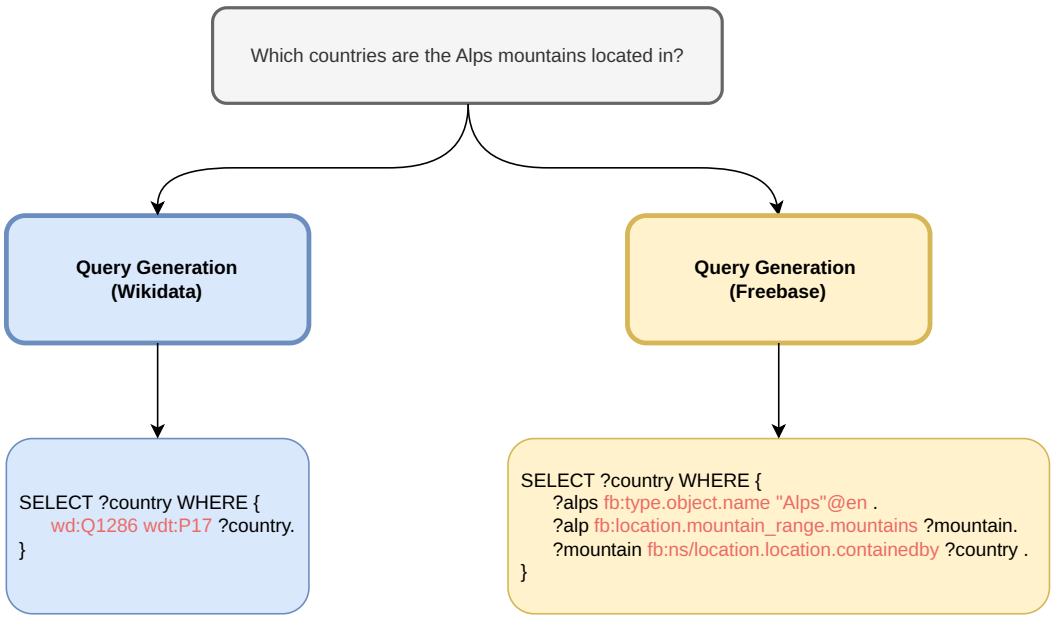


Figure 7.1: An example of two individual models for generating SPARQL queries across different knowledge graphs.

with semantic identifiers like `dbr:Alps`, while Wikidata uses numerical identifiers such as `wd:Q1286`. Furthermore, answering one and the same question on different KGs may require completely different query structures, as presented in Figure 7.1. Moreover, Freebase hierarchically organizes semantic relations between entities, unlike the structures in other knowledge graphs. These differences pose challenges for developing systems compatible with multiple knowledge graphs. Each knowledge graph requires different entity linking and query generation components. Consequently, adapting to these differences requires retraining or fine-tuning LLMs to ensure effective performance.

In this chapter, we propose a unified approach to fine-tune a single large language model for generating SPARQL queries across different knowledge graphs. Our approach involves fine-tuning a single LLM across multiple knowledge graphs rather than training separate models for each graph. To achieve this, we combine training data from multiple knowledge graphs into a single dataset. We hypothesize that joint fine-tuning of a single LLM enables better generalization across different data representations and structures. As a result, a single model for multiple KGs significantly reduces the resource requirements in a productive environment, since only one model needs to be deployed rather than multiple models for different KGs. To evaluate the performance of our approach, we compare models tailored to each knowledge graph with our unified model. In our experiments, we use two different knowledge graphs (Wikidata and Freebase) that differ significantly in the structure of SPARQL queries. Furthermore, we extract relevant information (e.g., entities, relations, types) from the knowledge graph and investigate which information has the greatest impact

on query generation performance. The evaluation results demonstrate that our approach achieves performance comparable to, or even surpassing, single KG models, reducing the need to train or fine-tune a separate model for each knowledge graph. We summarize the main contributions of this chapter as follows:

- We propose a unified approach for generating SPARQL queries for multiple knowledge graphs.
- Our approach achieves performance equivalent to or better than that of individual models (tailored to each KG), eliminating the need for separate training or fine-tuning an LLM for each KG.
- Incorporating relevant information (e.g., entities, relations, and types) within the LLM prompt improves the performance of SPARQL query generation.
- The source code and datasets used in our experiments are publicly available.[1]

7.2 Approach

Figure 7.2 shows an overview of the approach (UniQ-Gen) for generating SPARQL queries for multiple knowledge graphs (e.g., Wikidata and Freebase) using a jointly trained model. We achieve this by fine-tuning the T5 (Raffel et al., 2020) model on a mixed dataset, containing training examples of (natural questions, SPARQL-for-Freebase) and (question, SPARQL-for-Wikidata) pairs. In this way, the fine-tuned T5 model learns how to generate SPARQL queries for both graphs, rather than fine-tuning two separate models for each graph. Accordingly, we reduce the computational and maintenance costs associated with fine-tuning and managing separate models for each knowledge graph. The following sections describe the details of each module in our approach, including `Entity and Relation Linking (ERL)` and `Query Generation`.

7.2.1 Entity and Relation Linking (ERL)

The ERL process involves three main tasks: named entity recognition (NER), entity disambiguation (ED), and entity linking (EL). Named entity recognition identifies and classifies entity spans within the text (Nadeau and Sekine, 2007), while entity disambiguation associates these spans with corresponding entities in the target KG. First, we extract relevant information (e.g., entities, relations, and types) from the input question. Notably, this

[1] https://github.com/dice-group/KATRINA

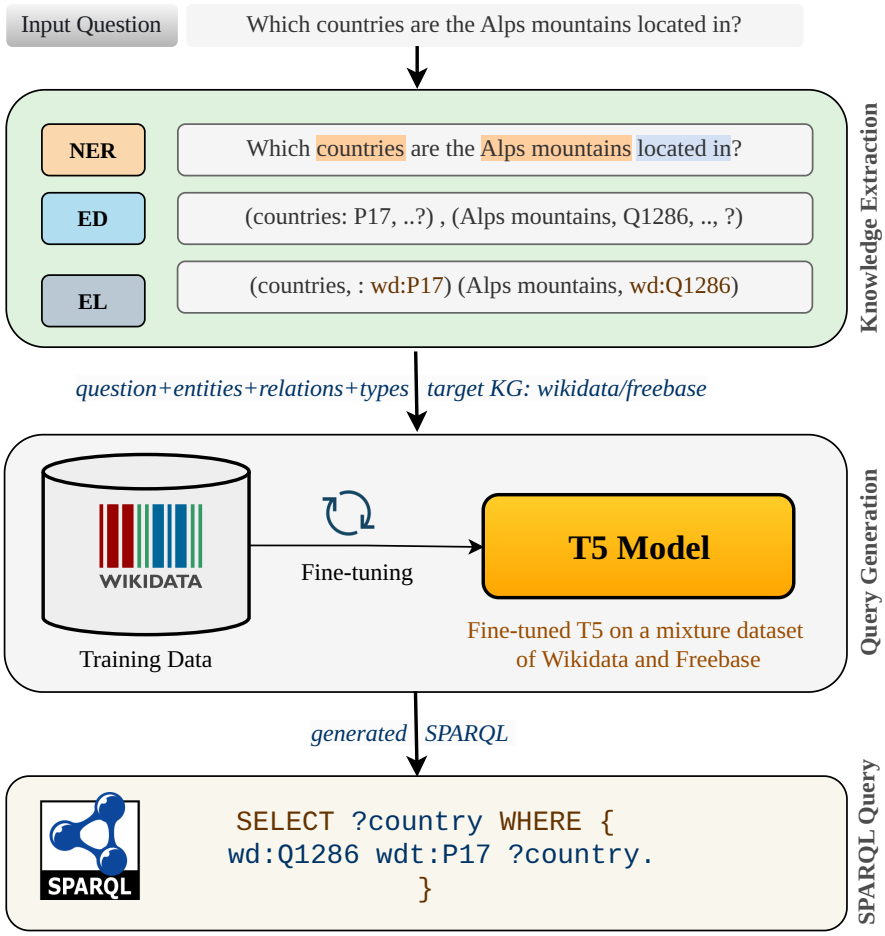


Figure 7.2: Our approach (UniQ-Gen) for generating SPARQL queries using one model for multiple knowledge graphs.

process differs between Wikidata and Freebase due to their distinct structures and the way information is represented.

ERL on Wikidata Many question answering approaches rely on pre-built frameworks, such as DBpedia Spotlight (Mendes et al., 2011), which generally yield satisfactory results, due to the significant resources required to develop an efficient linking system. However, these frameworks often struggle to identify and categorize relations and types. For instance, in a query like *"Which mountains are located in the US?"*, a typical NER framework would only recognize *"US"* as an entity. For accurate query generation, it is essential to also link the term *"mountains"* to the knowledge graph. Additionally, these frameworks usually fail to predict relationships between entities, a critical factor for enhancing QA system accuracy. To address these limitations, we employ the following methods:

Question: Which countries are the Alps mountains located in? [SEP]
Entities: Country: P17, Alps mountains: Q1286; ,[SEP]
Relations: located in: P138 [SEP]
Target Knowledge Graph: Wikidata.

Figure 7.3: An example input for generating SPARQL query with Wikidata.

- *Entity Recognition, Disambiguation, and Linking*: We use Flair (Akbik et al., 2019), a state-of-the-art framework that employs an LSTM network with contextual string embeddings to recognize and classify entity mentions within text. For entity disambiguation, we use GENRE (De Cao et al., 2021), which applies an autoregressive transformer architecture based on the pre-trained BART model with constrained decoding and beam search to predict Wikipedia titles. We link these titles to Wikidata using a dictionary, assuming each entity has a unique label that maps to a single Wikipedia title.

- *Linking Relations, Types, and Additional Entities*: We use a fine-tuned T5 model to link types, relations, and additional entities that are not directly mapped to spans in the input sequence. For example, in the question *"Which is the highest mountain in the US?"*, the relation *"is located in"* is needed but not directly present in the text. We extend target labels to include relations, e.g., *"Who is the spouse of Obama?[SEP]entities: Barack Obama, relations: spouse"*.

We combine the outputs of these methods and use a dictionary to map Wikidata labels to their URIs, assuming each label is unique within the knowledge graph. These outputs are then used as input for the `Query Generation module`, as shown in Figure 7.2. The input includes the question, entities, relations, and the target knowledge graph. All are concatenated using a separation token, *"[SEP]"* as shown in Figure 7.3.

ERL on Freebase Many types and relations in Freebase share identical labels due to its hierarchical ontology, making pre-built methods (e.g., Flair) for ERL in Wikidata inadequate. Therefore, we adapt existing methods for extracting knowledge from Freebase as follows:

- *Entity Recognition, Disambiguation, and Linking*: Our approach aligns with the RNG-KGQA framework (Ye et al., 2022), employing a BERT-NER model to detect entity mentions accurately within the text. For entity disambiguation, we use a pre-trained BERT-based model that leverages relation information linked with each entity, thereby improving the ranking of the target entity. For entity linking, our approach

Table 7.1: Training Samples

Question	Entities	Relations	Target	SPARQL
Is Kevin Costner owner of Fielders Stadium?	wd:Q11930 wd:Q5447154	wdt:P1830	wikidata	ASK
how many hadrons are in the family meson?	physics.hadron, m.04_rh	physics.particle.family	freebase	SELECT COUNT
What periodical literature does Delta Air Lines use as a moutpiece?	wd:Q188920 wd:Q1002697	wdt:P2813 wdt:P31	wikidata	SELECT

matches mentions to surface forms in the Freebase KG, ranks them by popularity, and retains the top 5 candidates. The ranking model employed is a cross-encoder model, as described in Equation 7.1.

- *Type and Relation Linking*: Our approach follows the schema retrieval method from the TIARA Framework (Shu et al., 2022), using a cross-encoder ranker to rank relations and types from the Freebase ontology. The score for question (x) and a schema (c) is computed as:

$$s(x,c) = \text{Linear}(\text{BertCLS}([x;c])) \tag{7.1}$$

where BertCLS represents the *CLS* token from a BERT-encoder (Shu et al., 2022). We use the top-5 relations and types as input for our query generation model.

7.2.2 Query Generation

In this module, we employ the T5 model for query generation, as it has demonstrated promising results for this task (Banerjee et al., 2022; Ye et al., 2022). In particular, we fine-tune the T5 model on diverse training datasets containing SPARQL queries from two knowledge graphs (Wikidata and Freebase). Table 7.1 shows some training examples of the dataset used to fine-tune the T5 model, including examples for question-to-SPARQL$_{(wikidata)}$ and question-to-SPARQL$_{(Freebase)}$. An example includes: an input question, the (entities, relations, and types), a target KG (e.g., Wikidata), and the ground-truth SPARQL query. During the training phase, these samples are combined into a single dataset and shuffled, and the T5 model is trained to generate SPARQL queries from input questions and target KGs. This process involves fine-tuning the model to accurately map the linguistic structures of the questions to the target KG.

Language models often struggle with special tokens such as *"{"* and *"}"*, which are integral to SPARQL queries. To address this issue, we replace these tokens in the target strings as follows: *"{"* with *"_cbo_"* and *"}"* with *"_cbc_"*. We normalize variables by replacing the leading *"?"*-token. For example, a variable like *"?uri"* is replaced with "_var_<id>", or "_result_<id>" if it is a part of the result set. Here, *"<id>"* represents a number, as a query can contain multiple variables. During the T5 model's inference, we revert these substitutions to generate a valid SPARQL query. For variables, we only replace the leading underscore with the *"?"*-token. Note that types are included under the entity tag to shorten the input string, as entities and types are used similarly in SPARQL queries. For Freebase, the approach is the same, except that the string *"target: Freebase"* is appended at the end of the input string.

7.3 Experiments

The experiments in this chapter answer **RQ**$_4$: *Can LLMs be jointly fine-tuned for query generation for multiple KGs, and how do these models perform compared to models trained for single KGs?*, and contribute to **RQ**$_6$: *How does the performance of knowledge extraction frameworks influence the performance of query prediction?*. Concretely, we answer the following subquestions:

- **S**$_1$**:** How well does our unified model perform compared to individual language models, trained on single knowledge graphs, for SPARQL query generation?
- **S**$_2$**:** How does our unified model perform compared to state-of-the-art baselines?
- **S**$_3$**:** What is the impact of incorporating knowledge such as entities, relations, and types on the performance of the query generation models?
- **S**$_4$**:** How does integrating knowledge from different resource extraction frameworks into the training datasets affect the performance of query generation models?

7.3.1 Datasets

In our experiments, we use different benchmark datasets, namely, LC-QuAD 2.0 (Dubey et al., 2019), and QALD-10 (Usbeck et al., 2023) on the Wikidata knowledge graph and GrailQA (Gu et al., 2021) on the Freebase knowledge graph. We briefly describe these datasets as follows:

- **LC-QuAD 2.0** (Dubey et al., 2019) was already introduced in Section 5.3.1 The dataset is divided into a training subset with around 24k questions and a test set with 6k questions.
- **QALD-10** (Usbeck et al., 2023) this dataset is manually annotated with 394 question-query pairs across different languages. For training purposes, it applies updated version of the QALD-9 dataset (Usbeck et al., 2018a), referred to as QALD-9-plus (Perevalov et al., 2022). As the dataset is relatively small, we initially trained our model on the LC-QuAD 2.0 dataset as a pre-trained (i.e., foundation) model, then fine-tuned it on the QALD-9-plus dataset, following the same training strategy as Zhou et al. (2021).
- **GrailQA** (Gu et al., 2021) This dataset is a large, crowdsourced collection for the Freebase KG, containing around 64k questions. The dataset provides not only SPARQL queries but also contains S-expressions as alternative logical representations. The dataset is divided into a training split with 44K questions, a development split with 6k, and a test split with 13k.

7.3.2 Experiment Setup

In our experiments, we used Nvidia H100 GPUs for efficient model training. The T5-base model serves as a foundation model, as it is widely used in query generation research. This further ensures comparability with other methodologies. Each model is trained for a maximum of 50 epochs, with early stopping to mitigate overfitting. For our unified model, we combine the LC-QuAD 2.0 (Gu et al., 2021) and GrailQA (Dubey et al., 2019) datasets. Despite GrailQA containing approximately 20k more questions than LC-QuAD 2.0, our experiments show no significant impact on performance, indicating that data balancing is unnecessary. For the QALD-9-plus dataset, we fine-tuned our pretrained LC-QuAD and Freebase model. We supplemented the training data with an equivalent number of randomly selected entries from the GrailQA dataset to match the volume of the QALD-9-plus dataset.

7.3.3 Evaluation

We evaluated the performance of SPARQL query generation using the GERBIL-QA framework (Usbeck et al., 2019). This framework is well-established, compatible with different

Table 7.2: Comparison of joint and single KG models $\mathbf{S}_1$.

Dataset	Experiment	Precision	Recall	F1	QALD F1
LC-QuAD	Gold Resources Joint	0.88	0.87	0.88	**0.92**
	Gold Resources LC-QuAD	0.88	0.88	**0.89**	**0.92**
	End-to-End Joint	0.47	0.47	0.47	0.62
	End-to-End LC-QuAD	0.42	0.43	0.42	0.59
GrailQA	Gold Resources Joint	0.51	0.59	0.54	0.67
	Gold Resources Grail QA	0.54	0.62	**0.57**	**0.68**
	End-to-End Joint	0.30	0.34	0.31	0.49
	End-to-End Grail QA	0.30	0.34	0,31	0.49
QALD-10	Gold Resources Joint	0.49	0.49	**0.49**	**0.64**
	Gold Resource QALD-10	0.47	0.47	0.47	0.62
	End-to-End Joint	0.44	0.45	0.44	0.60
	End-to-End QALD-10	0.45	0.45	0.45	0.61

benchmark datasets, and provides multiple evaluation metrics, including micro-F1, macro-F1, and QALD F1, which are used in the QALD challenge.[2] We adapted the same evaluation setting of Usbeck et al. (2019), and also include metrics such as macro-precision, macro-recall, macro-F1, and QALD F1. The macro-F1 score is calculated per question and is computed as the arithmetic mean of the per-question scores. For clarity, we refer to macro-precision, macro-recall, and macro-F1, as precision, recall, and F1, respectively. We set up the Virtuoso Triple Store for Freebase following instructions from the GrailQA repository[3], and for Wikidata, we used the official Triple Store.[4] Each query is generated regardless of whether the triple store returns an empty result set. We did not verify the correctness of the generated queries; therefore, improperly formatted queries return an empty result set. Finally, we compiled all outputs into a QALD-formatted JSON file. We submitted it using the *'upload result file'* function in the GERBIL-QA framework to calculate the final results.

7.3.4 Comparison of Unified Model and Single KG models ($\mathbf{S}_1$)

To answer this question, we implement two variants of the model: the first is a unified model that is fine-tuned on a heterogeneous dataset of SPARQL queries for Wikidata and Freebase. The other variants are single models tailored for Freebase and Wikidata, trained only on the training subsets for the respective KGs.

[2] https://www.nliwod.org/challenge
[3] https://github.com/dki-lab/GrailQA
[4] https://query.wikidata.org/

We conducted two experiments using these models. The first one, referred to as the *gold-resource experiment*, involved evaluating the models using high-quality input data derived from the test splits. This evaluation process consists of extracting entities, types, and relations from the test split, then incorporating them as additional information into the model's input. In contrast, the second experiment, referred to as the *end-to-end experiment*, used knowledge from our `ERL` module as direct input to the model. The results are presented in Table 7.2. Our analysis on the GrailQA dataset shows that in the end-to-end configuration, the unified model performs equivalently to the single KG models. Conversely, in the gold-resource setup, the performance disparity between the unified and single KG models is minimal. Similarly, on the LC-QuAD dataset, the end-to-end performance of the unified model surpasses the single KG model. For the GrailQA dataset, the difference in performance between models trained and evaluated using gold resources is negligible. On the QALD-10 dataset, the unified model's performance with gold-resource input slightly outperforms the single KG model. In the end-to-end experiment, the single KG model achieves a 1% higher F-Measure than the unified model, though this difference is marginal, consistent with results from other datasets. Overall, the results show that the unified model performs comparable to that of single KG models across all experiments.

7.3.5 Comparison with Baseline Models ($\mathbf{S}_2$)

Results on Wikipedia Datasets We conducted two experiments on the LC-QuAD 2.0 dataset to compare the performance with state-of-the-art baselines in SPARQL query generation (Table 7.3). First, we evaluated the performance of our system using gold entities and relations as inputs. Remarkably, the performance of our approach aligns with that of Banerjee et al. (2022), which also uses the T5-small model. This similarity in performance can be attributed to the shared inputs, despite our study employing a unified model across multiple KGs. In an end-to-end setup, our unified model outperforms the model by Tan et al. (2023) on the macro-F1 score. For additional comparison, we refer to the results from the KGQA-leaderboard.[5] Furthermore, on the QALD-10 dataset, our approach achieves state-of-the-art results on the QALD F1 measure. Across all Wikipedia datasets, our unified model achieves results comparable to those of single models.

Results on the Freebase (GrailQA) Dataset In the next step, we compare our results with the baseline models on the GrailQA dataset (Table 7.4). It is important to note that the evaluation script used by GrailQA differs from ours, as it assesses queries in S-expressions rather than SPARQL. To evaluate on the Gerbil-QA dataset, we included only systems

[5] https://github.com/KGQA/leaderboard/

Table 7.3: Comparison with baselines on Wikidata datasets $\mathbf{S}_2$.

(a) Results on the QALD-10 Dataset

Approach	QALD F1
Borroto et al. (2021)	0.59
Diefenbach et al. (2020)	0.58
Shivashankar et al. (2022)	0.49
Baramiia et al. (2022)	0.43
Joint Model End-to-End	0.60
Single KG Model End-to-End	**0.61**

(b) End-to-end Results on the LC-QuAD 2.0 Dataset

Approach	F1
GPT-3.5 (Tan et al., 2023)	0.39
ChatGPT (Tan et al., 2023)	0.42
Joint Model End-to-End	**0.46**
Single KG Model End-to-End	0.42

(c) Results on the LC-QuAD Dataset with Gold Knowledge

Approach	QALD F1
Banerjee et al. (2022) (T5 base)	0.91
Banerjee et al. (2022) (T5 small)	**0.92**
Banerjee et al. (2022) (PGN-BERT)	0.86
Joint Model Gold Knowledge	**0.92**
Single KG Model Gold Knowledge	**0.92**

Table 7.4: Comparison with baselines on the GrailQA dataset $\mathbf{S}_2$.

Approach	Precision	Recall	F1	QALD F1
Shu et al. (2022)	0.59	0.71	0.62	0.71
Shu and Yu (2024)	0.59	0.71	0.62	0.71
Gu et al. (2023)	0.64	0.79	**0.68**	**0.72**
Gu and Su (2022)	0.62	0.79	0.67	0.71
Joint Model End-to-End	0.30	0.34	0.31	0.49
Joint Model Gold	0.51	0.59	0.54	0.67

that provide results in a format compatible with QALD. Our findings indicate that the results are not as strong as those achieved by the baseline models. This is mainly because the GrailQA dataset focuses on queries that require detailed knowledge of the Freebase structure, especially its hierarchical ontology. Existing methods usually generate and rank subqueries, enabling learning of the knowledge graph structure. However, these methods are resource-intensive, as they need to generate and rank a large set of queries, resulting in slow processing (Gu and Su, 2022; Shu et al., 2022), which may not be suitable for production environments.

Table 7.5: Comparison of different input data $\mathbf{S}_3$.

Dataset	Experiment	Precision	Recall	F1	QALD F1
LC-QuAD	Baseline	0.37	0.37	0.37	0.53
	Gold Entities & Gold Types	0.70	0.71	0.71	0.81
	Gold Resources	0.88	0.87	**0.88**	**0.92**
GrailQA	Baseline	0.20	0.24	0.21	0.39
	Gold Entities & Gold Types	0.33	0.40	0.35	0.54
	Gold Resources	0.51	0.59	**0.54**	**0.67**

7.3.6 Influence of KG Knowledge on the Model Performance ($\mathbf{S}_3$)

To answer this question, we conducted experiments on extensive datasets, LC-QuAD 2.0 and GrailQA, to ensure that the models are trained on a sufficient number of entities and relations. We carried out three experiments for each knowledge graph: i) The first experiment, referred to as a *baseline* experiment, uses only the question as input. ii) The second experiment includes additional information, such as entities and types, as well as the question as input. iii) The third experiment is referred to as a gold-resource experiment, where entities, types, and relations are included with the question as an input.

Table 7.5 shows the evaluation results of all experiments, indicating that the model's performance improved with the additional information (entities, relations, and types). Specifically, adding entities and types led to a significant performance boost. While relations improve performance, less noticeable than entities. Overall, the LC-QuAD dataset demonstrates better results (F-measure of 0.88), with the additional information. In contrast, the GrailQA dataset reaches an F-measure of only 0.54. Therefore, future research should focus on providing more detailed information, including the structural aspects of knowledge graphs.

7.3.7 Experiments with Different Training Data ($\mathbf{S}_4$)

Typically, training data is enriched with additional information by adding gold-resource entities and relations from the target SPARQL queries. This method often causes the model to duplicate the input information without distinguishing between relevant and irrelevant data. We address this issue by extracting relevant information from our ERL modules and incorporating it into the training data. To investigate the influence of this method on query generation, we conducted several experiments per dataset, training the model once on the dataset's gold-resource information and once with additional knowledge-extraction inputs. Afterward, we conducted the same experiments as in Section 7.3.4, evaluating the models with both gold input and in an end-to-end setup.

Table 7.6: Comparison of different training setups $\mathbf{S}_4$.

Dataset	Experiment	Precision	Recall	F1	QALD F1
LC-QuAD	Gold Resources	0.88	0.87	**0.88**	**0.92**
	Gold Resources inc. ERL	0.84	0.85	0.84	0.90
	End-to-End	0.46	0.46	0.46	0.60
	End-to-End inc. ERL	0.47	0.47	0.47	0.62
GrailQA	Gold Resources	0.33	0.40	0.35	0.54
	Gold Resources inc. ERL	0.51	0.59	**0.54**	**0.67**
	End-to-End	0.12	0.15	0.12	0.26
	End-to-End inc. ERL	0.30	0.34	0.31	0.49
QALD-10	Gold Resources	0.48	0.49	0.48	**0.64**
	Gold Resources inc. ERL	0.49	0.49	**0.49**	**0.64**
	End-to-End model	0.27	0.28	0.28	0.43
	End-to-End inc. ERL	0.44	0.45	0.44	0.60

Our findings presented in Table 7.6 indicate that including relevant information improves the model performance across all datt'asets in the end-to-end setup. However, on the LC-QuAD dataset, the performance improvement is minimal compared to training with gold-standard data. On the GrailQA dataset, we achieve a significant improvement, increasing the F1 QALD score from 0.26 to 0.49. Similarly, on the QALD dataset, the model's performance improved from 0.43 to 0.6. These variations in performance across datasets can be attributed to different ERL methods used to link data. For instance, entity mentions in LC-QuAD closely align with those in the Wikidata, whereas QALD-10 shows greater ambiguity. For example, the question: *"Do the princes William and Harry share the same mother?"*, where the entities are referred to only by their first names. In the evaluation setup, with gold-resource information, we observe a performance improvement only on the GrailQA dataset. This is not surprising, as incorporating gold information can introduce noise into the model inputs.

7.4 Conclusion

This paper presents UniQ-Gen, a unified approach for fine-tuning a joint model to generate SPARQL queries across different knowledge graphs. Our results demonstrate that training a unified model on multiple heterogeneous datasets (e.g., including samples from Wikidata and Freebase) achieves performance comparable to that of single models for individual KGs, eliminating the need for separate models for each graph. Moreover, incorporating additional information such as entities, relations, and types significantly enhances the performance

of query generation models. While there are many effective solutions for entity linking, accurate and efficient relation linking remains a challenge in knowledge graph question answering. However, our one-shot query generation approach lacks the incorporation of structural information about the KG. In our future work, we address this limitation by incorporating structural information (e.g., hierarchical relationships among entities) into our unified model. Furthermore, we will also adapt our approach to handle structural differences between knowledge graphs by integrating KG-specific structural knowledge. In the next chapter, we will compare the fine-tuning approach against a traditional query generation approach 8. Afterward, we will implement different knowledge extraction pipelines based on the frameworks presented in Chapters 4-6. Additionally, we will apply the knowledge extraction approach on Wikidata presented in Section 7.2.1 for the Wikidata KG, and fine-tune models for each knowledge extraction pipeline (Chapter 9).

8 Knowledge Graph Question Answering using Graph-Pattern Isomorphism

This chapter addresses research question $\mathbf{RQ}_5$: *How well can traditional template-based question answering approaches perform on the KGQA task compared to LLM-based approaches?*. To answer this question, we will introduce the traditional template-based QA system TeBaQA as a baseline and evaluate it on the KGQA task, comparing it with the previously implemented fine-tuning-based approach. The content is based on our work Vollmers et al. (2021), where the author co-designed, co-implemented, and evaluated the approach and co-wrote the corresponding paper.

8.1 Overview

Modern state-of-the-art LLMs achieve a significant performance in generating valid logical queries (Banerjee et al., 2022). However, for traditional KGQA systems, generating queries is a challenging problem. To this end, a common approach is to utilize query templates (alias graph patterns) with placeholders for relations and entities. The placeholders are then filled with entities and relations extracted from a given natural language question (Abujabal et al., 2018; Höffner et al., 2017; Unger et al., 2012) to generate a SPARQL query, which is finally executed. Semantic parsing assumes that a template can be constructed or selected to represent the internal structure of a natural language question. Thus, the KGQA task can be reduced to finding a matching template and filling it with entities and relations extracted from the question.

The performance of KGQA systems based on this approach depends heavily on the query templates implemented, which in turn depend on the question's complexity and the KG's topology. Consequently, costly hand-crafted templates designed for a particular KG cannot be easily adapted to a new domain.

In this chapter, we present the TeBaQA KGQA engine. TeBaQA reduces the effort required for manual template generation by learning templates from existing KGQA benchmarks. We rely on learning templates based on isomorphic basic graph patterns.

The goal of TeBaQA is to employ machine learning and feature engineering to learn to classify natural language questions into isomorphic basic graph pattern classes. At execution time, TeBaQA uses this classification to map a question to a basic graph pattern, i.e., a template, which it can fill and augment with semantic information to create the correct SPARQL query.

TeBaQA achieves state-of-the-art performance partially compared to other traditional KGQA systems without any manual effort. In contrast to existing solutions, TeBaQA can be easily ported to a new domain using only benchmark datasets, as evidenced by our evaluation across different KGs and train-test splits. We use a best-effort to work with the data at hand instead of either (i) requiring a resource-intensive dataset creation and annotation process to train deep neural networks or (ii) hand-crafting mapping rules for a particular domain. Our contributions can be summarized as follows:

- We present TeBaQA, a QA engine that learns templates from benchmarks based on isomorphic basic graph patterns.
- We describe a greedy yet effective ranking approach for query templates that aims to identify the best-matching template for a given input query.
- We evaluate TeBaQA on several standard KGQA benchmark datasets and unveil choke points and future research directions.
- The code is publicly available.[1]

8.2 Approach

TeBaQA is based on isomorphic graph patterns that can be extracted across different SPARQL queries and used as templates for our KGQA approach. Figure 8.1 provides an overview of TeBaQA's architecture and its five main stages:

First, all questions run through a `Preprocessing` stage to remove semantically irrelevant words and create a set of meaningful n-grams. The `Graph-Isomorphism Detection and Template Classification` phase uses the training sets to train a classifier based on a natural language question and a SPARQL query by analyzing the basic graph pattern for graph isomorphisms. The main idea is that structurally identical SPARQL queries represent syntactically similar questions. At runtime, a question is classified into a ranked

[1]The code is available at https://github.com/dice-group/TeBaQA and a demo of TeBaQA over encyclopedic data can be found at https://tebaqa.demos.dice-research.org/. We also provide an online appendix which contains more details about our algorithms and their evaluations at https://github.com/dice-group/TeBaQA/blob/master/TeBaQA_appendix.pdf.

list of SPARQL templates. While `Information Extraction`, TeBaQA extracts all critical information, such as entities, relations, and classes, from the question and determines the answer type based on a KG-agnostic set of indexes. In the `Query Building` phase, the extracted information is inserted into the top templates, the SPARQL query type is determined, and query modifiers are added. The resulting SPARQL queries are executed, and their answers are compared with the expected answer type. The subsequent ranking is based on a combination of all information, the natural language question, and the returned answers.

In the following, we present each of these steps in more detail. We use DBpedia (Lehmann et al., 2015a) as a reference KG for the sake of simplicity in our description.

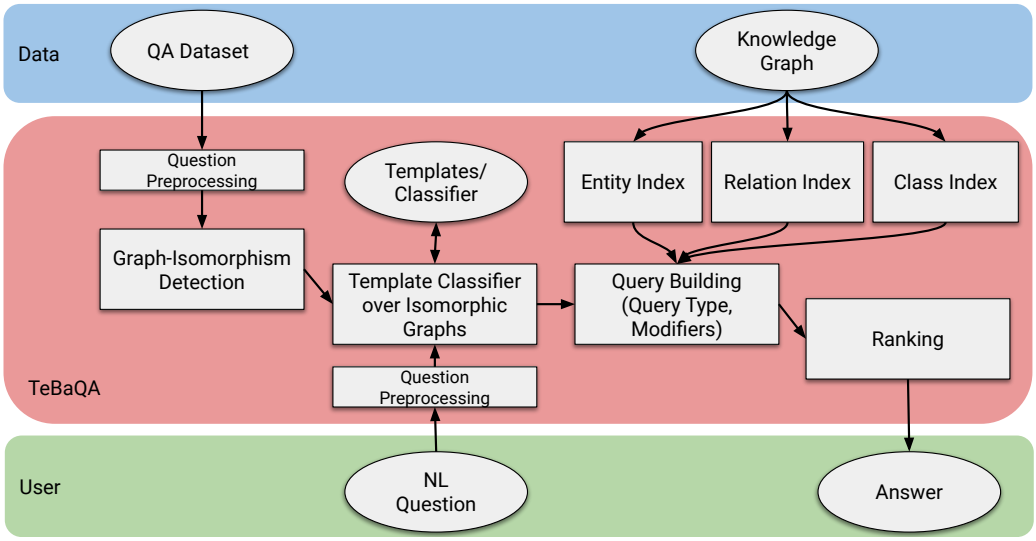


Figure 8.1: The TeBaQA architecture on the running example.

8.2.1 Question Preprocessing

There are often words that do not contribute any information to the answer to natural language questions. Thus, we distinguish semantically relevant and irrelevant n-grams. Irrelevant n-grams can lead to errors that could propagate through the architecture. An example of this is the entity `dbr:The_The`[2]. If the word *"The"* were to be wrongly associated with this entity every time *"the"* occurs in a question, the system's performance would decrease severely. However, irrelevant words are sometimes part of entities, e.g., `dbr:The_Two_Towers`, so we cannot always filter these words. For this reason, we combine up to six neighboring words from the question to n-grams and remove all n-grams

[2] `dbr:` is a prefix which stands for `http://dbpedia.org/resource/`

that contain stop words only. To identify irrelevant words, we provide a stop word list that contains the most common words of a particular language that are highly unlikely to add semantic value to the sentence. Additionally, TeBaQA distinguishes relevant and irrelevant n-grams using part-of-speech (POS) tags. Only n-grams beginning with `JJ`, `NN`, or `VB` POS-tags are considered relevant. After this preprocessing step, TeBaQA maps the remaining n-grams to entities from DBpedia in the information extraction step.

8.2.2 Graph-Isomorphism Detection and Template Classification

TeBaQA classifies a question to determine which isomorphic basic graph pattern (BGP) it belongs to. Since SPARQL is a graph-based query language (Pérez et al., 2009), the structural equality of two SPARQL queries can be determined using an isomorphism. At runtime, TeBaQA can classify incoming questions to find the correct query templates, in which later semantic information can be inserted at runtime.

SPARQL BGP Isomorphism to Create Template Classes Using the training datasets, TeBaQA generates one basic graph pattern for each given question and its corresponding SPARQL query, see Figure 8.2. Subsequently, all *isomorphic* SPARQL queries are grouped into the same class. Now, each class contains semantically distinct natural language questions but structurally similar SPARQL queries.

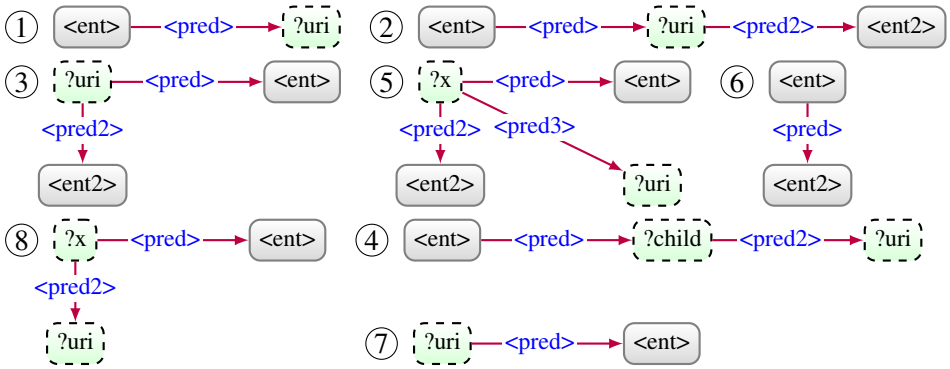


Figure 8.2: All basic graph patterns used as classes for QALD-8 and QALD-9, which later become templates. Note that the depicted templates contain more than five examples in the training dataset. Our running example, "Who was the doctoral advisor of Albert Einstein?" belongs to template (1).

Theorem 1 (Isomorphism of labeled graphs) *Two labeled graphs are isomorphic when a 1:1 relationship and a surjective function are present between the nodes of the graphs, wherein the node labels, edge labels, and neighborhood relationships are preserved by the mapping (Gervasi and Kumar, 2006).*

Question Features and Classification Next, TeBaQA trains a classifier that uses all questions of an isomorphism class as input to calculate features for this class. A feature vector holds all the information required to make a reliable statement about which question belongs to which class. The features can be seen in Table 8.1.

Table 8.1: Features to map a question to an isomorphic basic graph pattern.

Feature	Type	Description
QuestionWord	Nominal	Adds the question word (e.g. Who, What, Give) as a feature.
EntityPerson	Boolean	Checks the named entity tags of the sentence to see if any persons are mentioned in it.
NumberOfToken	Numeric	Stores the number of tokens separated by spaces excluding punctuation.
QueryResourceType	Nominal	Categorizes the question based on a list of subject areas, e.g., film, music, book or city.
Noun	Numeric	Aggregates the number of nouns.
Number	Numeric	Indicates how often numbers occur in the question.
Verb	Numeric	Aggregates the number of verbs.
Adjective	Numeric	Aggregates the number of adjectives.
Comperative	Boolean	Indicates whether comparative adjectives or adverbs are included in the sentence.
TripleCandidates	Numerical	Estimates how many SPARQL triples are needed to answer the question based on the number of verbs, adjectives, and related nouns.

The features can be divided into semantic and syntactic features. *QuestionWord*, *EntityPerson*, and *QueryResourceType* form the group of semantic features and represent particular content aspects of the question, e.g., persons or specific topics that are mentioned in the question. All other features describe the structure of the question.

Note that other features were investigated, but they did not improve the model's recognition rate. We report these features to aid future research in this area: 1) Cultural Categories: Mainly included music and movies, e.g., *"Who is the singer on the album The Dark Side of the Moon?"* and 2) Geographical entities: Questions in which countries or cities occur, as well as where questions, e.g., *"In which country is Mecca located?"*

Using the features above, it is possible to represent the question *"Who was the doctoral advisor of Albert Einstein?"* with the following vector:

```
1   <Who, Person ,8 , dbo:Person ,1 ,0 ,1 ,0 , NoComperative ,1>
```

TeBaQA trains a statistical classifier using the described features extracted from the input question and the isomorphic basic graph patterns as class labels. A feature vector's target class can be determined by generating the basic graph pattern for the corresponding SPARQL query and assigning the class, which represents this pattern. An evaluation can be found in Section 8.3.2.

8.2.3 Information Extraction

TeBaQA identifies entities, classes, and relations to fill the placeholders of a particular SPARQL template. Since questions are shorter than typical texts, semantic entity linking tools such as DBpedia Spotlight (Daiber et al., 2013) or MAG (Moussallem et al., 2017) do not perform well due to the lack of semantic context. For example, in *"Who was the doctoral advisor of Albert Einstein?"*, the word *"Einstein"* has to be linked to `dbr:Albert_Einstein` and not to any other person with that name. For this reason, we apply a KB-agnostic, index-based approach to identify candidate entities, relations, and classes. TeBaQA uses three indexes that are created at runtime.

Entity Index The entity index contains all entities from the target knowledge graph. To map an n-gram from the preprocessing step to an entity, TeBaQA queries against the index's label field. The index contains information about entities, relations, and classes connected to the entity at hand.

Relation Index and Class Index These two indices contain all OWL classes and relations from a KG. The indexes map n-grams to relations and classes in the KB's ontologies.TeBaQA additionally indexes hypernyms and synonyms for all relations and classes.[3]

Consider the question *"Who was the doctoral mentor of Einstein?"*. DBpedia contains only the relation `dbo:doctoralAdvisor`[4] and not `dbp:mentor`[5]. Through the synonym *"advisor"* of *"mentor"*, the relation `dbo:doctoralAdvisor` can be determined. This example highlights the lexical and semantic gap between natural language and knowledge graphs.

[3]The dictionary can be found at `https://github.com/dice-group/NLIWOD/tree/master/qa.annotation/src/main/resources` which was previously used by (Singh et al., 2018)
[4]`dbo:` stands for `http://dbpedia.org/ontology/`
[5]`dbp:` stands for `http://dbpedia.org/property/`

Disambiguation By querying the indexes for an n-gram, we get candidates for entities, relations, and classes, whose labels contain all tokens of the n-gram. Since a candidate's label may contain more tokens than the n-gram, we apply a Levenshtein distance filter of 0.8 to the candidates. All remaining candidates are used to fill a given template.

8.2.4 Query Building

Template Filling To fill the templates, we provide information about connected entities and relations for the found entities from the entity index. For the triples in a template, there are two cases:

1.) The triple contains one placeholder for an entity and one placeholder for a relation. In this case, we resort to only the connected relation information from the entity index. An entity candidate e and a relation candidate p are combined to a triple $< e, p, ?v >$ or $<?v, p, e >$ if the set of connected relations $S(e)$ of the entity e contains p and if the connected n-grams do not contain each other.

2.) The triple contains only one placeholder for a relation p'. This case only occurs when at least one triple in the template matches case 1. We can utilize these triples to generate matching triples for the given triple. Thus, we query the entity index and search for a set of entities $S(e')$ connected with the entity e by the relation p. All connected relations from the entities in $S(e')$ in the set of relation candidates and whose n-grams do not cover the n-grams of e and p are candidates for p'.

Each candidate SPARQL query is checked for consistency with the ontology. In general, there are query patterns that do not contain variables. This case only occurs in ask queries like *"Did Socrates influence Aristotle?"*. We ignore this case to keep simplicity and be aware of the performance impact. To summarize, TeBaQA creates several candidate SPARQL queries per template and thus per question.

Query Modifiers and Query Types To translate a question into a semantically equivalent SPARQL query, it is often not enough to recognize the entities, relations or classes in the question and insert them into a SPARQL query. For example, the question *"How many children did Benjamin Franklin have?"* asks for the number of children and not the concrete list. Thus, we apply a rule-based look-up to add query modifiers and choose a query type. The supported modifiers are *COUNT*, if the question contains keywords like *"How many"* or *"How much"*, *Filter (?x<?y)*, if we identify comparatives and *ORDER BY [ASC (?x) | DESC (?x)] LIMIT 1*, if the question contains superlatives. Additionally, we support *ASK*-type questions when keywords such as *"Is"* or *"Are"* are identified.

Note that the templates and their respective basic graph patterns neither contain information about the query type nor about query modifiers. Thus, TeBaQA generates one SPARQL query per candidate SPARQL query for each recognized cross-product of query type and modifier. The outcome is a list of executable queries for each input question.

8.2.5 Ranking

Since the *conciseness* of an answer plays a decisive role in QA, in contrast to full-text search engines, only the answer that corresponds best to the user's intention should be returned. Thus, all generated SPARQL queries and their corresponding answers are ranked. This ranking is carried out in two steps. First, we filter by 1) the expected answer type of the question in comparison to the actual answer type of the query and by 2) the cardinality of the result set. Second, TeBaQA ranks the quality of the remaining SPARQL queries.

Answer Type and Cardinality Check For certain types of answer sets, only those that match the question's expected answer type are considered for the next ranking step. We empirically analyzed the benchmark datasets and derived a rule-based system for the most common expected answer type and their distinguishing features.

- Temporal questions usually begin with the question word *"When"*, e.g., *"When was the Battle of Gettysburg?"*. TeBaQA expects a date as the answer type.
- Decision questions mostly start with a form of *"Be"*, *"Do"* or *"Have"*. The possible answer type is boolean.
- Questions that begin with *"How much"* or *"How many"* can be answered with numbers. This also includes questions that begin with a combination of the question word *"How"* and a subsequent adjective, such as *"How large is the Empire State Building?"*

If none of the above rules apply to a question, the result set's cardinality is checked. There are two cases: First, when several answers are needed to answer a question fully, consider *"Which ingredients do I need for carrot cake?"*, if only one answer is found for this question, it can be assumed that it is either wrong or incomplete. Second, when there is only one answer to a question, e.g., *"In which UK city are the headquarters of the MI6?"*, an answer consisting of several entities would not be correct.

To recognize which query type (ASK or SELECT) a question belongs to, the first noun or the first compound noun after the question word is checked. If they occur in the singular form, a single answer is needed to answer the question. For the above question *"In which UK city are the headquarters of the MI6?"*, the compound noun would be *"UK city"*. Since both

words occur in the singular, it is assumed that only a single answer is required. If the first noun or group of nouns occurs in the plural, this indicates that the question requires multiple answers. In the question *"Which ingredients do I need for carrot cake?"* the decisive word *"ingredients"* is in the plural.

However, the question type may not be recognized correctly. For instance, if the question is grammatically correct but contains words with identical singular and plural forms, such as *"news"*, we cannot determine the correct answer type. These issues will be tackled in future research.

Once the question type and answer type have been determined, all answers whose type or cardinality do not match the question will be discarded.

Quality Ranking For the remaining SPARQL queries, TeBaQA calculates a *rating* based on the sum of the individual scores of the bindings B and the input question *phrase*. A binding B is the mapping of entities, relations, and classes to placeholders contained in one SPARQL query q. To compute the relatedness factor r, the following factors are taken into account:

- **Annotation Density**: The annotation density measures that the more words from the sentence are linked to an entity, class, or relation, the more likely it is that it corresponds to the intention of the user. For the question *"What is the alma mater of the chancellor of Germany Angela Merkel?"*, one candidate query may apply the binding `dbr:Angela`, while another query applies the binding `dbr:Angela_Merkel`. The former refers only to the word *"Angela"*. The latter refers to two words of the sentence: *"Angela Merkel"* and covers longer parts of the phrase.
- **Syntactic Similarity**: The syntactic similarity is an indicator of how similar an n-gram of the sentence and the associated binding are. For example, in the question *"Who is the author of the interpretation of dreams?"* the n-gram *"the interpretation of dreams"* can be linked with `dbr:The_Interpretation_of_Dreams` or `dbr:Great_Book_of_Interpretation_of_Dreams` among others. The former has a smaller Levenshtein distance and a greater syntactic similarity with the selected n-gram.

We cover both aspects with the following formulas:

$$rating = \sum_{B \in q} r(B, phrase) \tag{8.1}$$

$$r(entity, phrase) = |words(phrase)| - levenshtein\_ratio(label(B), phrase) \tag{8.2}$$

After all entities, classes, and relations used in a query have been evaluated and summed up, the rating is corrected down by 30% if more than 50 results are returned by the query, based on empirical observations in the datasets.

8.3 Experiments

In this chapter, we address $\mathbf{RQ}_5$: *How well can traditional template-based question answering approaches perform for the KGQA task compared to LLM-based approaches?*. We answer the following subquestions:

- $\mathbf{S}_1$: How well is the performance of the template classification approach?
- $\mathbf{S}_2$: How well does the approach perform compared to state-of-the-art traditional approaches and fine-tuning-based approaches?
- $\mathbf{S}_3$: How good is the query generation performance compared to fine-tuning approaches?

8.3.1 Datasets

We performed the evaluation on the 8th and 9th Question Answering over Linked Data challenge training datasets (QALD-8 train (Usbeck et al., 2018b) and QALD-9 train (Usbeck et al., 2018a)), which contain 220 (QALD-8) and 408 (QALD-9) heterogeneous training questions. Additionally, we evaluated on the two LC-QuAD (Dubey et al., 2019; Trivedi et al., 2017) datasets with 4000 train and 1000 test questions and 24.000 train and 6.000 test questions, respectively. Across datasets, the questions vary in complexity, as they include comparatives, superlatives, and temporal aggregations. An example of a simple question is *"How tall is Amazon Eve?"*. A more complex example is *"How many companies were founded in the same year as Google?"*, since it involves temporal aggregation (*"the same year"*). We created separate instances of TeBaQA for each training dataset and evaluated each instance on its corresponding test dataset.

8.3.2 Classification Evaluation ($\mathbf{S}_1$)

For the QALD-8 and QALD-9 datasets, eight classes were identified for each dataset, as shown in Figure 8.2. For LC-QuAD v1 and LC-QuAD v2, TeBaQA identified 17 classes for both datasets. Since the two LC-QuAD datasets were constructed to promote greater

diversity, classification is more challenging. Note, we omitted classes with fewer than five examples in the training dataset. We are aware that we are trading our overall performance for classification accuracy.

To this end, we evaluated a variety of machine learning methods, which required questions to be converted into feature vectors. In particular, we used the QALD-8 and QALD-9 training datasets, along with 10-fold cross-validation, to evaluate the computed models. All duplicate questions and questions without SPARQL queries were removed from the training datasets. We tested multiple machine learning algorithms with the WEKA framework (Eibe et al., 2016) on our training data using 10-fold cross-validation.[6] To achieve comparable results, TeBaQA uses only the standard configuration of the algorithms. The macro-weighted F1 score for one fold in cross-validation is calculated from the classes' F1 scores, weighted according to the size of the class. After that, we calculated the average macro-weighted F1 score across all folds. On the QALD datasets, the algorithm *RandomizableFilteredClassifier* achieves the highest F1 scores of 0.523964 (QALD-8) and 0.528875 (QALD-9), respectively. On LC-QuAD v1, we achieve a template classification F1 score of 0.400464 and 0.425953 on LC-QuAD v2 using a MultilayerPerceptron. Consequently, we use the best-performing classifiers for the end-to-end evaluation.

Similar experiments can be found in the work of Athreya et al. (2020). The authors use a recurrent neural network, i.e., a tree-LSTM, to identify templates in LC-QuAD and achieve an accuracy of 0.828 after manually merging several template classes.

8.3.3 Baseline Experiments ($\mathbf{S}_2$)

For evaluation, we used the fair benchmarking platform GERBIL, as already introduced in the previous chapter, to ensure future reproducibility of the experiments.

Table 8.2 contains the results of selected question answering systems, measured against the QALD-8 and the QALD-9 test benchmarks.[7] We focused on English questions only, as all available QA systems support English at the time of these experiments. The macro values for precision, recall, and F1 score were selected. The evaluation was performed with GERBIL version 0.2.3 if possible. We always report the highest numbers if several papers reported numbers and evaluation with GERBIL was not possible.

On the QALD-8 benchmark, TeBaQA achieved the best results in terms of F1 score by 5% QALD F1 score. Our average time is significantly larger than that of the other reported

[6] https://www.cs.waikato.ac.nz/ml/weka/

[7] The links to our GERBIL-experiments can be found on our Github page: https://github.com/dice-group/TeBaQA/blob/master/README.md

Table 8.2: Results of TeBaQA and other state-of-the-art QA systems for multiple datasets. * indicates F1 score instead of QALD F1 score (Usbeck et al., 2019). ** Numbers are taken from the corresponding papers.

System	KB	Dataset	Precision	Recall	QALD F1	Avg. Time in s
gAnswer (Zou et al., 2014)	DBpedia	QALD-8	0.337	0.354	0.440	4.548
QAnswer (Diefenbach et al., 2020)	DBpedia	QALD-8	0.452	0.480	0.512	0.446
Zheng and Zhang (2019)**	DBpedia	QALD-8	0.459	0.463	* 0.461	–
TeBaQA	DBpedia	QALD-8	0.476	0.488	**0.556**	28.990
Elon (Usbeck et al., 2018a)	DBpedia	QALD-9	0.049	0.053	0.100	0.219
gAnswer (Zou et al., 2014)	DBpedia	QALD-9	0.293	0.327	0.430	3.076
NSQA (Kapanipathi et al., 2020)**	DBpedia	QALD-9	0.314	0.322	*0.453	–
QAnswer (Diefenbach et al., 2020)	DBpedia	QALD-9	0.261	0.267	0.289	0.661
QASystem (Usbeck et al., 2018a)	DBpedia	QALD-9	0.097	0.116	0.200	1.014
Zheng and Zhang (2019)**	DBpedia	QALD-9	0.458	0.471	***0.463**	–
TeBaQA	DBpedia	QALD-9	0.241	0.245	0.374	5.049
NSQA (Kapanipathi et al., 2020)**	DBpedia	LC-QuAD v1	0.382	0.404	*0.383	–
QAMP (Vakulenko et al., 2019)**	DBpedia	LC-QuAD v1	0.250	0.500	*0.330	0.720
QAnswer (Diefenbach et al., 2020) **	DBpedia	LC-QuAD v1	0.590	0.380	***0.460**	1.500
TeBaQA	DBpedia	LC-QuAD v1	0.230	0.229	0.300	36.000
TeBaQA	Wikidata	LC-QuAD v2	0.140	0.136	0.227	–
Fine-tuned T5	Wikidata	LC-QuAD v2	0.420	0.430	0.590	–

systems, since the ranking mechanism is activated and then fires several SPARQL queries after the initial null-retrieving query.

On QALD-9, TeBaQA is in fourth place with a QALD F1 score of 0.37. This implies that TeBaQA achieves comparable or partially better results than other semantic QA systems with a wide margin of possible improvements, as shown in the ablation study. QALD-9 is a more challenging benchmark than QALD-8, as it contains many questions that require complex queries involving more than 2 triples. A more in-depth analysis shows that questions from QALD-9 often require complex templates that are not present in the training queries or receive only limited support in the training set (Gu et al., 2021). This mismatch leads to a high number of misclassifications and explains the limited performance on QALD-9 compared to QALD-8 and, thus, its limited generalization abilities to unseen templates. Although we are not outperforming the state-of-the-art systems on QALD-9, TeBaQA is a novel research avenue with respect to learning from data.

On the LC-QuAD dataset, which contains the most complex questions, TeBaQA achieves an F1 score of 0.30. We ran this benchmark only once in our system and had some errors during runtime. We will further investigate the performance on LC-QuAD in our future research. Note that we ran LC-QuAD only once through the system to test our independence from the dataset, similar to the methodology of Wang et al. (2015).

Finally, we compare the performance of TeBaQA on the Wikidata KG against the fine-tuning approach presented in Section 7 on the LC-QuAD v2 dataset. The results show that the fine-tuning approach outperforms TeBaQA in the end-to-end setup. Note that most current benchmarks use Wikidata rather than DBpedia as the KG, so we use Wikidata as the main KG for query generation in this thesis. Wikidata contains many more triples than DBpedia, which heavily affects the performance of the template-filling step in TeBaQA. Furthermore, relation paths such as *"wdt:P31/wdt:P279"*, which are frequently used in the QALD-9 plus and QALD-10 benchmarks, are not covered in the extraction step for query templates. Due to the relatively slow template-filling step, limited performance on the Wikidata KG, and adaptation issues with relation paths, we decided to use a fine-tuning approach to investigate the impact of knowledge extraction on the query generation performance. However, we compare the performance on the query generation task under the condition of perfect entity linking between TeBaQA and the fine-tuning approach in a ablation study, presented in Section 8.3.4, for a more detailed evaluation.

8.3.4 Ablation Study ($\mathbf{S}_3$)

We conducted an ablation study to identify the modules of TeBaQA's pipeline that most influence end-to-end performance. Since the number of possible entities is roughly a magnitude larger than the number of relations, and for the sake of experimental time, we omitted testing for perfect relation and class linking.

Table 8.3: Ablation study for TeBaQA on the QALD-9 test benchmark.

QA System	Precision	Recall	Avg. Time	QALD F1 score
Perfect Classification	0.205	0.210	4.618	0.337
Perfect Classification + EL	0.407	0.407	0.355	0.578
Perfect Classification + Ranking	0.245	0.257	4.029	0.399
Perfect EL	0.301	0.317	0.713	0.473
Perfect EL + Ranking	0.302	0.320	0.653	0.477
Perfect Ranking	0.258	0.270	6.281	0.405
Perfect Classification + EL + Ranking	0.407	0.407	0.251	0.578
Fine-tuned T5 with Perfect EL	0.335	0.359	-	0.504

For the perfect classification experiment, the QALD F1 score is lower than that of the overall system; see Table 8.2. Investigating the detailed outputs, TeBaQA selects the simpler templates containing only one triple more often than the more complex templates because they have more instances (i.e., support) in the QALD-9 training dataset. In many cases, a simple query returns a result set that is a superset of the target result set, thereby decreasing precision.

When TeBaQA fails to fill the correct complex template with the proper entities, the query result set is often disjoint from the target result set. It is also reasonable that TeBaQA fails to fill the more complex templates due to missing or incorrect entity links.

Still, there is a gap in the system that becomes evident when looking at the perfect classification and ranking. Ranking is activated only if the perfect template, filled with semantic information, returns no answer. That is, TeBaQA fails to find the correct modifiers or needs to circle through other semantic information candidates.

When the perfect classification is combined with perfect entity linking, the results achieve a QALD F1 score of 0.58, which would clearly outperform any other system. The same happens if we add the perfect ranking. The results are the same in both cases because the ranking is rarely triggered, since the perfect template is already filled in correctly.

The strongest single influence is the entity linking part, enabling TeBaQA to jump to 0.47 F1 score. We will tackle this challenging module in future work.

Regarding runtime, failing to find the perfect template and then iterating through the ranking, i.e., querying the SPARQL endpoint often, increases the average time needed significantly.

Compared to the fine-tuning approach with perfect entity links, TeBaQA only outperforms it when the perfect template is selected and perfectly linked entities are used, meaning that predicting queries based on fine-tuning is more effective than template classification

approaches. However, in current LLM-based chain-of-thought approaches for query generation, such as Zahera et al. (2024), introducing the structure of example queries for similar questions remains a common technique to improve the prediction of valid queries.

8.4 Conclusion

We presented TeBaQA, a QA system that learns to map questions to SPARQL template mappings using basic graph pattern isomorphisms. TeBaQA significantly eases the domain/KB adoption process for traditional KGQA systems as it relies only on a benchmark dataset at its core.

While current state-of-the-art fine-tuning approaches outperform classical template-based approaches, introducing the structure of queries from similar questions into LLMs became a core component of LLM-based prompt engineering techniques for query generation (Walter and Bast, 2025; Zahera et al., 2024). In future work, we will also compare unsupervised prompt-based query generation, for example, based on RAG and chain-of-thought pipelines, with fine-tuning-based query generation approaches.

9 Evaluation of ERL for Question Answering over Knowledge Graphs

9.1 Overview

To conclude, we evaluate the introduced techniques for entity and relation linking on the ERL and end-to-end KGQA task. To achieve this, we combine the LLM-based query generation approach introduced in Chapter 7 with the methods for contextual augmentation, dense retrieval, and keyphrase extraction to end-to-end KGQA pipelines. Additionally, we set up baseline approaches based on non-LLM-based systems for further comparison. This chapter addresses the following research questions:

- **RQ**$_3$: *How can LLM-based entity linking approaches improve the performance for extracting KG-knowledge for questions?*
- **RQ**$_6$: *How does the performance of knowledge extraction frameworks influence the performance of query prediction?*

The evaluation consists of two steps: In the first step, we evaluate ERL performance on KGQA datasets. In the second step, we assess performance on end-to-end KGQA using the outputs of each ERL pipeline as inputs to a fine-tuned query generation model based on T5. The content of this chapter is based on the paper Vollmers et al. (2025a), where the author designed, implemented, and evaluated the corresponding approach and co-wrote the corresponding paper.

9.2 Approaches and Pipelines

In this section, we describe the approaches used to obtain results on the ERL and KGQA tasks. We apply two classes of t'approaches: LLM-based methods that apply LLMs at various stages of the ERL task, and non-LLM-based methods. In the following, we describe these approaches in detail, beginning with the non-LLM-based approaches.

9.2.1 Non-LLM-based Approaches

For the class of non-LLM-based methods, we apply three different approaches based on popular EL frameworks. The first approach combines the systems Stanford NER (Finkel et al., 2005), which leverages a linear-chain conditional random field (CRFs) (Lafferty et al., 2001) for NER and MAG (Moussallem et al., 2017) as described in chapter 6 for ED. The second system is DBpedia Spotlight (Mendes et al., 2011), an end-to-end entity linking system for DBpedia that applies precomputed lexicalizations for entity disambiguation. To link the predicted entities to Wikidata, we apply the `owl:same-as` links provided by DBpedia. The last system is Falcon 2.0 (Sakor et al., 2020), which also offers relation linking, using span detection for retrieval and triple-based candidate extraction for disambiguation.

9.2.2 Neural Approaches

To implement neural ranking pipelines, we combine the approaches for contextual augmentation (chapter 4, dense retrieval 5, MAG, and the ERL approach from UniQ-Gen for Wikidata 7.2.1 to different pipelines. The pipelines are implemented with the components presented in the follow-up paragraphs.

Contextual Augmentation We adapt our contextual augmentation approach for questions. Unlike the EL task in chapter 4, it is not necessary to map entities to spans in the input sequence to predict entities and relations for questions. Thus, the NER step can be omitted, and we use the following prompt to expand the context of questions:

LLM Prompt

```
Your task is to help to link Information from Questions
to Knowledge Graphs.  Please generate a list with
all Entities, Relations, and Types for the following
Question.  Please generate one list with all entities.
Do not format the JSON output.{question}
```

After executing the prompt on the LLM, we extract the predicted entities and relations from the response and append all these terms at the end of each question to generate an *augmented question*. However, we found that many of these predicted terms already perfectly match the labels of entities and relations in the KG. So we mapped these terms directly to entity and relation identifiers in Wikidata with a dictionary. To link the remaining terms, we introduce the *augmented question* in the other components of the pipeline.

Dense Retrieval We use the dense retrieval approach trained with noisy optimization (Chapter 5) to retrieve entities and relations from *augmented questions*. To this end, we use the Anti-Grad noise injection strategy, as it has shown the best performance on the LC-QuAD 2 dataset. For the encoder, we choose the E5 model (Wang et al., 2022), as it can be trained faster while still achieving sufficient performance compared to a dual bi-encoder model on LCQuAD. We modify the hard negative sampling strategy to scale training efficiently; specifically, we increase the candidate pool by 100,000 entities after each epoch, starting from an initial set of the same size. In the final two epochs, all entities are used to compute global hard negatives. This follows the intuition that in early epochs, random negatives are sufficient for training, and the difficulty of the ranking task should be increased for each epoch by introducing harder negatives from an expanded subset of the entity corpus. This saves time in the early stages of the training process. For the relations, we fine-tune a separate bi-encoder model, also based on E5. We compute global negatives after each epoch, which remains computationally feasible since Wikidata contains only around 10,000 relations. Both models are trained for 20 epochs on the LC-QuAD v2 (Dubey et al., 2019) training set. At inference time, we extract the top 5 entities and relations from the indices, as our evaluation has shown that this number already provides a sufficient recall (Table 9.1). We implement two versions of this pipeline: one that only applies dense retrieval and one that uses the LLM-based filtering as described in the following. In both cases, we also include the directly mapped entities before the filtering step.

MAG-based Retrieval As an alternative to dense retrieval, we use the MAG Framework (Moussallem et al., 2018) to link the augmented terms to the KG. This process is similar to the keyphrase linking approach presented in section 6. For this, we mark all entities and relations in the *augmented questions* as entities, and extract entities and relations using MAG. For this, we use a Wikidata Index. This index also contains nodes for each relation in the Wikidata KG. By using these within the MAG HITS algorithm to predict hub and authority scores, relations can be treated identically to entities. This enables the adaptation of MAG for relation linking. Consistent with the dense retrieval approach, we evaluate variants with and without filtering. In both cases, we augment the result set with directly linked entities and relations prior to the filtering step.

LLM-based filtering To filter entities and relations, we use a joint prompt to exclude irrelevant entities and relations introduced during augmentation and retrieval. For this, we use the following prompt:

LLM Prompt

```
Your task is to help to link Information from Questions
to Knowledge Graphs. From the following list of
entities: {entities}. And the following list of
relations: {relations}. Which of them are required
to write a SPARQL query to answer the question:
{sentence} Please return as few entities and relations
as possible. Please return only the IDs of the
entities and relations.
```

We apply this strategy instead of a cross-encoder because it does not need any costly fine-tuning. Additionally, the number of required entities and relations is unknown for each question, so it is not possible to set a fixed value for k that fits for each question. We use this strategy in combination with the dense- and MAG-based retrieval approaches described above.

UniQ-Gen Augmentation As an alternative strategy, we adapted the ERL approach of the UniQ-Gen framework for Wikidata (see Section 7.2.1) by fine-tuning the T5 model on *augmented questions*. Similar to the other pipelines, we merge the predictions from the augmented T5 model with those from Flair and Genre. For this approach, filtering is not necessary, since the T5 model already shows a well-balanced performance between precision and recall (see Table 9.1). Furthermore, explicitly adding directly linked entities has been shown to be unnecessary, as the T5 model inherently copies their labels into the output sequence.

9.3 Query Generation

We fine-tune the T5 model for query generation, in the same way as described in chapter 7. Since it has been shown to be beneficial, we also incorporate predicted entities and relations into the training process in addition to gold entities. Thus, we train one model for each pipeline by first fine-tuning it on the LC-QuAD training dataset and subsequently fine-tuning it on QALD-9 plus."

9.4 Experiments

We conducted several experiments on the question answering benchmarks QALD-9 plus, QALD-10, and LC-QuAD 2.0, which were already presented in the chapters before. We computed the micro-precision, micro-recall, and micro-F1 scores for ERL, and the macro-precision, macro-recall, and macro-F1 scores, together with the QALD F1 score, for end-to-end question answering on each dataset. The results are presented in the Tables 9.1 and 9.2. We evaluated four traditional approaches (including their combined result sets) and six neural pipelines by combining the previously introduced components. These experiments directly answer the research questions mentioned in the beginning of this chapter:

- **RQ**$_3$: *How can LLM-based entity linking approaches improve the performance for extracting KG-knowledge for questions?*
- **RQ**$_6$: *How does the performance of knowledge extraction frameworks influence the performance of query prediction?*

9.4.1 Extraction of KG Knowledge for Questions (**RQ**$_3$)

Results on Entities The results show that the neural approaches achieve higher F1 scores and recall. The augmentation approach on the T5 model achieves the highest score on all datasets. The primary advantage of the neural approaches lies in their significantly higher recall, whereas some traditional pipelines maintain competitive precision. This is because traditional approaches are usually capable of predicting entities when the KG labels are similar to those in the query, but struggle to predict labels that are semantically more distant from the query entities. Neural approaches are better at predicting these patterns. Furthermore, the two approaches that apply the T5 framework are well balanced between recall and precision compared to the other approaches. The dense retrieval approach achieves high recall but low precision because it generates a fixed set of five entities, whereas fewer entities are usually sufficient for query generation. The results show that the additional filtering can remove many irrelevant entities while maintaining high recall across all datasets. For the MAG augmentation approach, we observe the same effect, but the results are slightly lower overall. For traditional approaches, we can improve the score by combining their results, but neural approaches still achieve better performance.

Results on Relations Similar to entity linking, the neural approaches outperform the only traditional approach (Falcon) on the relation linking task. Overall, the scores are lower compared to the entity linking performance, due to higher ambiguity and the fact that

Table 9.1: Entity Linking Results on KGQA Datasets

Dataset	Approach	F1	Precision	Recall
QALD 9	DBpedia Spotlight	0.5275	0.6547	0.4417
	Stanford & MAG	0.3746	0.6883	0.2573
	Falcon	0.2945	0.4000	0.2330
	Combined	0.4683	0.4263	0.5194
	UniQ-Gen	0.7170	**0.8051**	0.6463
	UniQ-Gen Augmented	**0.7537**	0.7574	0.7500
	MAG Augmented	0.1932	0.1103	0.7794
	MAG Augmented & Filtered	0.5027	0.3966	0.6827
	Augmented & Dense Retrieval	0.1870	0.1014	**0.8431**
	Augmented & Dense Retrieval & Filtered	0.5301	0.4085	0.7549
QALD 10	DBpedia Spotlight	0.5955	0.6990	0.5187
	Stanford & MAG	0.3962	0.7480	0.2695
	Falcon	0.3251	0.3385	0.3127
	Combined	0.4861	0.4006	0.6182
	UniQ-Gen	0.5702	0.5848	0.5562
	UniQ-Gen Augmented	**0.6921**	**0.7726**	0.6268
	MAG Augmented	0.2416	0.1425	0.7925
	MAG Augmented & Filtered	0.5741	0.4659	0.7478
	Augmented & Dense Retrieval	0.2258	0.1300	**0.8602**
	Augmented & Dense Retrieval & Filtered	0.6032	0.4847	0.7983
LC-QuAD	DBpedia Spotlight	0.5473	0.5303	0.5655
	Stanford & MAG	0.4496	0.7281	0.3252
	Falcon	0.4971	0.6000	0.4243
	Combined	0.5524	0.4454	0.7270
	UniQ-Gen	0.6741	0.6740	0.6742
	UniQ-Gen Augmented	**0.7615**	**0.7713**	0.7519
	MAG Augmented	0.1860	0.1054	0.7866
	MAG Augmented & Filtered	0.5354	0.4156	0.7523
	Augmented & Dense Retrieval	0.1815	0.1014	**0.8642**
	Augmented & Dense Retrieval Filtered	0.5821	0.4561	0.8042

relations are more often implicitly mentioned than entities. To this end, the results show that the dense retrieval approach on augmented sequences in combination with filtering has the best results on the QALD-9 and QALD-10 datasets. On LC-QuAD, the UniQ-Gen approaches achieve the highest performance because, for training the T5 model, the LC-QuAD training data was used, which applies the same patterns to generate question-query pairs as the according test split. For the other two datasets, the test split exhibits a more pronounced difference to the training data; consequently, applying augmentation exerts a greater influence on the final score, as previously noted in Section 4.4.4. Unlike the

performance on entities, the results show a larger difference between recall and precision on the QALD-9 and QALD-10 datasets for the two T5-based approaches.

Table 9.2: Relation Linking Results on KGQA Datasets

Dataset	Approach	F1	Precision	Recall
QALD 9	Falcon	0.2268	0.4536	0.1512
	UniQ-Gen	0.3257	**0.9661**	0.1959
	UniQ-Gen Augmented	**0.5815**	0.6697	0.5139
	MAG Augmented	0.3940	0.3237	0.5035
	MAG Augmented & Filtered	0.4794	0.5520	0.4236
	Augmented & Dense Retrieval	0.3222	0.2143	**0.6495**
	Augmented & Dense Retrieval & Filtered	0.5285	0.5313	0.5258
QALD 10	Falcon	0.2562	0.3762	0.1942
	UniQ-Gen	0.4609	0.4672	0.4548
	UniQ-Gen Augmented	0.4730	0.6015	0.3896
	MAG Augmented	0.4404	0.3764	0.5308
	MAG Augmented & Filtered	0.5209	0.5676	0.4813
	Augmented & Dense Retrieval	0.3365	0.2238	**0.6779**
	Augmented & Dense Retrieval & Filtered	**0.5436**	**0.5285**	0.5597
LC-QuAD	Falcon	0.3929	0.5116	0.3189
	UniQ-Gen	0.8187	0.8223	0.8151
	UniQ-Gen Augmented	**0.8433**	**0.8729**	**0.8157**
	MAG Augmented	0.3675	0.2982	0.4785
	MAG Augmented Filtered	0.4795	0.5273	0.4397
	Augmented & Dense Retrieval	0.3526	0.2303	0.7518
	Augmented & Dense Retrieval & Filtered	0.6106	0.5882	0.6349

9.4.2 Influence of ERL Frameworks on Query Prediction ($\mathbf{RQ}_6$)

Finally, we present the KGQA results for all ERL pipelines. The results in Table 9.3 show that end-to-end pipelines utilizing neural entity linking significantly outperform traditional approaches.[1] The best-performing pipeline varies across datasets. For QALD-9, the original UniQ-Gen pipeline achieves the best performance, matching that of the augmentation and dense retrieval pipeline without filtering. For QALD-10, the combination of augmentation, dense retrieval, and filtering shows the best performance, and on LC-QuAD, the T5-based augmentation approach performs best. In general, superior ERL performance directly translates to higher overall KGQA scores. The results also show that recall exerts a greater influence, as the query generation models are trained on predicted entities, making them

[1] Note that the experiments in this section use an updated version of GERBIL, where the scoring function was modified which leads to different results compared to those in the chapters 7 and 8

Table 9.3: Question Answering Results on KGQA Datasets

Dataset	Approach	Precision	Recall	F1	QALD F1
QALD-9	DBpedia Spotlight	0.16	0.15	0.15	0.26
	Stanford & MAG	0.16	0.15	0.15	0.26
	Falcon	0.16	0.15	0.16	0.27
	Combined	0.17	0.17	0.17	0.28
	UniQ-Gen	0.40	0.40	**0.40**	**0.56**
	UniQ-Gen Augmented	0.36	0.37	0.39	0.50
	MAG Augemented	0.26	0.26	0.27	0.39
	MAG Augemented & Filtered	0.27	0.27	0.28	0.41
	Augmented & Dense Retrieval	0.40	0.40	**0.40**	0.52
	Augmented & DR & Filtered	0.32	0.33	0.32	0.45
QALD-10	DBpedia Spotlight	0.15	0.16	0.16	0,26
	Stanford & MAG	0.11	0.11	0.11	0.20
	Falcon	0.15	0.15	0.15	0.25
	Combined	0.15	0.15	0.15	0.25
	UniQ-Gen	0.27	0.27	0.27	0.40
	UniQ-Gen Augmented	0.22	0.22	0.22	0.33
	MAG Augemented	0.22	0.22	0.22	0.33
	MAG Augemented & Filtered	0.25	0.25	0.26	0.38
	Augmented & Dense Retrieval	0.26	0.27	0.26	0.39
	Augmented & DR & Filtered	0.29	0.31	**0.29**	**0.43**
LC-QuAD	DBpedia Spotlight	0.25	0.25	0.25	0.39
	Stanford & MAG	0.22	0.23	0.22	0.35
	Falcon	0.25	0.25	0.25	0.39
	Combined	0.26	0.26	0.26	0.40
	UniQ-Gen	0.45	0.45	0.45	0.60
	UniQ-Gen Augmented	0.48	0.48	**0.48**	**0.62**
	MAG Augemented	0.35	0.35	0.35	0.50
	MAG Augemented & Filtered	0.32	0.32	0.33	0.48
	Augmented & Dense Retrieval	0.40	0.40	0.40	0.55
	Augmented & DR & Filtered	0.37	0.38	0.37	0.53

robust to noisy entity and relation inputs. Filtering entities and relations in the MAG-based approach and the dense retrieval approach decreases the performance of the end-to-end systems on the LC-QuAD and QALD-9 datasets, while improving results on QALD-10. Both pipelines show low recall and high precision scores; nevertheless, they achieve competitive results on the end-to-end task.

9.5 Conclusion

In this chapter, we have evaluated several traditional and LLM-based ERL systems, incorporating prompt-based augmentation strategies across multiple KGQA benchmarks. Our findings highlight the importance of recall-oriented linking as it has a substantial impact on downstream performance. Furthermore, the contextual knowledge of LLMs can improve the performance on ERL. For future work, we plan to leverage this knowledge to develop more advanced query generation models, incorporating recent techniques such as chain-of-thought prompting and retrieval-augmented generation. Additionally, integrating the structure of knowledge graphs into query generation is crucial for enabling LLMs to select relations more accurately. Future research should therefore focus on more sophisticated methods for embedding this structural information within LLMs.

10 Application: Enhancing Answers Verbalization using Large Language Models

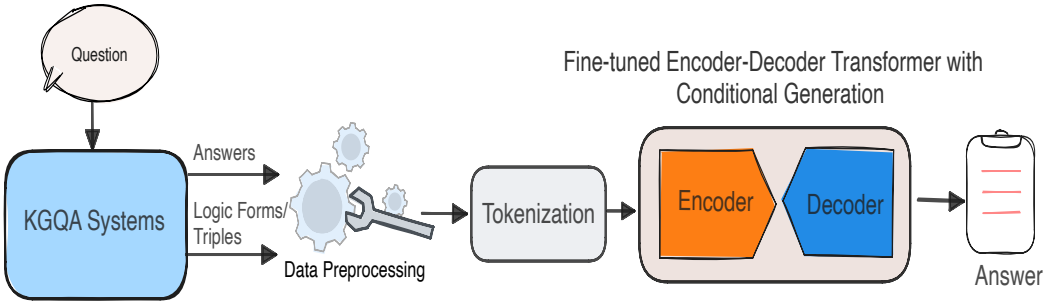
As an application of semantic parsing, we present an LLM-based approach for answer verbalization. In this application, we show that queries can enhance the verbalization of answers in a KGQA system. This chapter is based on the paper Vollmers et al. (2024). The author co-designed, co-implemented, and co-evaluated the approach and co-wrote the corresponding paper.

10.1 Overview

Semantic parsing-based KGQA systems generate answers in the form of single entities, a list of entities, or literals. However, in real-world applications of KGQA systems, such as chatbots or speech assistants, users prefer a more natural form of interaction. Consequently, there is a growing need to provide verbal explanations for the answers generated by KGQA systems. By presenting answers in natural language, verbalization makes information more accessible to a broader audience, including those who may not be familiar with linked data or query languages used by KGQA systems (Kacupaj et al., 2021b, 2022). Current open source KGQA systems typically provide answers without verbalizing them in natural language (Fu et al., 2020; Kacupaj et al., 2022). This lack of verbalization makes interactions with users less natural than with voice assistants such as Siri and Alexa. Few studies have been proposed to address this problem. For example, the VOGUE framework (Kacupaj et al., 2021b) has been designed to generate natural language explanations for visual elements in user interfaces. This allows AI systems to provide verbal descriptions and rationales for graphical components, making the interaction more conversational.

To verbalize answers in KGQA, multiple inputs can be used, including *questions*, and *answers*. KGQA systems can produce various answer types, including single entities, lists of entities, and literals such as numbers or text. Additionally, semantic parsing systems generate a logical query, which can be crucial for answer verbalization. Furthermore, it is possible to extract additional information, such as entity labels, from knowledge graphs. This

Figure 10.1: The architecture of our verbalization model for generating natural language answers.



wide range of input sources and answer types makes it challenging to verbalize answers for KGQA systems. Transfer learning of large language models (LLMs) has shown impressive performance in text generation tasks, including query generation across structured and unstructured inputs (Banerjee et al., 2022). In our study, we focus on fine-tuning different LLMs, such as T5 or BART, to address the challenges of answer verbalization. Furthermore, we experiment with different inputs to assess the impact of logical form and answers on the quality of the verbalized output. We summarize the main contributions of our study as follows:

- Fine-tuning LLMs achieves significant results for answer verbalization in KGQA systems.
- Incorporating additional information in the LLM input, such as logical forms or triples, yields better verbalized answers.
- We provide the implementation of our approach and the datasets used in our experiments on a GitHub repository.[1]

10.2 Approach

Figure 10.1 shows the architecture of our approach, including a fine-tuned encoder-decoder model. Our approach takes a question, its answer, and a query from a KGQA system as input. Further information, such as relevant triples, is included as alternative input. These inputs are preprocessed to clean noisy data and tokenized to generate the token vector for the language model. The following sections describe the preprocessing step and the large language models used in our study.

[1] https://github.com/dice-group/QAAnswerVerbalizer

10.2.1 Preprocessing Step

Our experiments incorporate two different forms of input for the verbalization model. In both versions, we included the answer set and the input question. Additionally, we explore using triples and SPARQL queries as additional inputs. In the case of SPARQL queries, language models often encounter problems with special tokens such as *{* or *}*, which are mapped to unknown tokens. Therefore, it is necessary to replace them with different tokens. In particular, we apply the following replacements:

- A variable such as *"?uri"* is replaced by the string *"var_uri"*.
- *"{"* and *"}"* are replaced by the strings. *"brack_open"* and *"brack_close"* respectively.
- The period *"."* is replaced by the string *"sep_dot"*.
- All SPARQL keywords are converted to lowercase.
- All prefixes are removed.

Listing 10.1: An example of input preprocessing

```
Question:         What is the nationality of Aishath Saffa?

Logical form:     select distinct var_uri where brack_open
Aishath_Saffa nationality var_uri brack_close

Triple(s):        Aishath Saffa nationality Maldives

Verbalization:    Maldives is the nationality of Aishath Saffa.
```

In both approaches (queries and triples), we replace the URIs of entities and relations with their labels from the KG. URIs consist of hashes or numeric identifiers that lack semantic information. For example, we use the string input *"Aishath Saffa"* string in our model for the DBpedia entity `dbr:Aishath_Saffa`. Listing 10.1 shows the output of the preprocessing steps for the example question *"What is the nationality of Aishath Saffa?"*.

10.2.2 Large Language Models

We experimented with three different pretrained models from the literature: T5, BART, and PEGASUS. T5 and BART are already introduced in Section 2.2.4. PEGASUS (Zhang et al., 2019b) is originally designed for generating summaries of natural language texts, which is similar to answer verbalization for KGQA systems. Answer verbalization can be treated as a summarization task, where various inputs such as questions, answers, and logical forms

shall be summarized into a short natural language text. In our experiments, we investigated whether fine-tuning a summarization model can improve answer verbalization performance. We trained two versions of LLMs: one model using triples as input and another model using queries along with answers and input questions. All models were fine-tuned with an equal number of epochs and input samples.[2]

10.3 Experiments

In this section, we describe the setup of our experiments, including datasets, baselines, and evaluation metrics for answering the following additional research questions:

- $\mathbf{A}_1$: Does incorporating structured data from knowledge graphs, such as triples or queries, improve the performance of LLMs in answer verbalization?
- $\mathbf{A}_2$: What types of inputs are most effective for generating natural language answers?

10.3.1 Datasets and Metrics

VQuAnDa (Verbalization QUestion ANswering DAtaset (Kacupaj et al., 2020)) is one of the first datasets that contains natural language verbalizations for questions. It contains 5,000 examples, SPARQL queries for DBpedia, and their verbalizations. The dataset is based on the large-scale complex question answering dataset (LC-QuAD v1 (Trivedi et al., 2017)).

ParaQA (Kacupaj et al., 2021a) is formed using the verbalizations in VQuAnDa (Kacupaj et al., 2020) and contains up to eight paraphrased verbalizations of the answer on the DBpedia KG, along with the question and the SPARQL query. It contains 5,000 examples.

For evaluation, we use the well-established metrics BLEU and METEOR introduced in chapter 2.5, for comparing our approach against the baseline methods.

10.3.2 Results

In this section, we present our results for answering both of our research questions. All results are presented in Table 10.1.

[2]Training setup: https://github.com/dice-group/QAAnswerVerbalizer/blob/main/args.py

Table 10.1: Comparison with baselines. The results of the baselines are taken from the corresponding papers. The inputs to the models: **Q** represents the question, **LF** represents Logical form/query, **T** represents Triples(s), and **H** represents Hybrid, which is a combination of both LF and Q. The BLEU and METEOR scores are reported. The scores are normalized on a scale of $[0, 100]$. The results of the baselines are extracted from Kacupaj et al. (2021b) for the Transformer, BERT, and VOGUE models and from Montella et al. (2022) for the T5 and BART models.

Model	BLEU		METEOR	
	VQuAnDa	ParaQA	VQuAnDa	ParaQA
Transformer (**Q**)	18.37	23.61	56.83	59.64
Transformer (**LF**)	23.18	28.01	60.17	63.75
BERT (**Q**)	22.78	26.12	59.28	62.59
BERT (**LF**)	26.48	30.31	65.92	65.92
VOGUE (**H**)	28.76	32.05	67.21	68.85
T5 (**Q**)	39.07	30.62	67.70	59.81
BART (**Q**)	43.90	35.57	71.92	65.40
PEGASUS (**Q+LF**)	45.97	**50.18**	79.87	80.70
PEGASUS (**Q+T**)	45.26	48.48	80.24	**81.97**
BART (**Q+LF**)	45.43	46.48	78.80	80.21
BART (**Q+T**)	43.02	47.32	78.57	80.98
T5 (**Q+LF**)	**49.25**	47.49	**80.66**	80.26
T5 (**Q+T**)	45.02	45.91	79.55	79.87

Incorporating Structura Data from KGs in LLMs (A_1): The evaluation results show that incorporating structured knowledge, such as triples and queries, improves answer verbalization performance across both datasets. Existing models in the literature use fine-tuned LLMs for answer verbalization. In contrast, our approach includes triples or SPARQL queries in the verbalization process, yielding better performance. Other models, such as VOGUE, also use logical forms, but without using pre-trained LLMs such as T5 or BART, which contributes to the performance margin in the results.

Experiment with Different Forms of Input (A_2): Our findings indicate that verbalizing SPARQL queries rather than triples generally yields superior performance across datasets, except for PARAQA, where the PEGASUS model achieved slightly better results. This marginal difference may come from the fact that queries may contain aggregation functions. These are not present in the KG itself but can improve the verbalization performance.

10.4 Conclusion

In this study, we demonstrated that incorporating structural information into an LLM significantly enhances the quality of answer verbalization in KGQA systems. Furthermore, the results show that logical forms, such as SPARQL queries, often yield better results than introducing triples. Our answer verbalization approach can be used as an extension to any question answering model capable of producing logical forms, such as SPARQL queries. Alternatively, triples can be applied in cases where no queries are available. However, only a few KGQA datasets are available at the moment, mainly on DBpedia KG, which include verbalizations, so extending existing KGQA datasets is necessary to improve verbalizations across other KGs. On the other hand, using current interaction-based LLMs such as LlaMA (Touvron et al., 2023) might also be an alternative for fine-tuning LLMs for verbalization.

11 Conclusion and Future Work

This chapter summarizes our findings in relation to the research questions and outlines potential avenues for future work. We begin by highlighting our main contributions before discussing future research directions.

11.1 Summary

In this thesis, we developed and evaluated neural entity linking approaches for knowledge graph question answering. Most previous EL approaches address the classical entity linking task, which identifies named entity spans in the input text and links these spans to entities in the target KG (De Cao et al., 2021; Wu et al., 2020a; Zhang et al., 2022c). In contrast, entity linking for KGQA imposes distinct requirements. First, it requires the extraction of relations and types in addition to entities–a task typically not covered by classical entity linking systems. Second, many entities, relations, and types are not explicitly mentioned in questions, so they cannot be linked to a text span directly.

Within the KGQA literature, many systems evaluate query generation performance by either assuming perfectly linked entities and relations (Banerjee et al., 2022; Su et al., 2024a) or by relying on existing APIs to extract this information from the KG (Shivashankar et al., 2022) Others, such as Diefenbach et al. (2020), use their own approach for entity and relation linking by mapping n-grams from the question to the KG, which is similar to our traditional approach, described in Chapter 8. However, the influence of these ERL approaches on the performance of KGQA systems remains underexplored.

In this thesis, we developed techniques to address ERL challenges in the context of query generation and evaluated them by analyzing the impact of the ERL performance on the overall performance of a KGQA system. We further discuss our contributions with respect to our research questions in the following sections.

11.1.1 Contextual Augmentation for Entity Linking and Question Answering

Our first contribution is the development of contextual augmentation approaches for entity linking. In Chapter 4, we showed that contextual augmentation can improve the performance of classical entity linking systems, especially on out-of-domain datasets with highly ambiguous entities. Our results further show that contextual augmentation is more effective at expanding predictions of entity mentions than at extracting additional mentions in the NER step, as this can introduce hallucinations, especially in longer texts. These findings directly answer $\mathbf{RQ}_2$: *How can LLM-based augmentation enhance the performance of entity linking systems?*. For applying augmentation to natural language questions, we first observed that it is not necessary to map entities to entity mentions first. This enables predicting knowledge that is only implicitly mentioned in questions. Consequently, we modified our prompt for augmentation to predict relevant KG terms from the input question directly. Our results on entity and relation linking for questions presented in Chapter 9 show that this augmentation strategy improves ERL performance on question answering datasets over Wikidata. These findings contribute to answering $\mathbf{RQ}_1$: *What methods can be used to predict knowledge that is implicit rather than explicitly expressed in questions?* and $\mathbf{RQ}_3$: *How can LLM-based entity linking approaches improve the performance for extracting KG-knowledge for questions?*.

11.1.2 Noisy Optimization in Fine-tuning LMs for Neural Ranking

In KGQA systems, neural ranking approaches are critical for ranking candidate query patterns (Ye et al., 2022) and generating entity linking candidates (Wu et al., 2020a). In Chapter 5, we developed an approach to inject noise within the optimization process of cross- and bi-encoder ranking models for retrieval and ranking. The results show that introducing noise in the optimization process improves the robustness of ranking models. The trained ranking models are used to extract present and absent knowledge from questions, thereby contributing to **RQ1** and **RQ3**. Furthermore, we used these neural ranking approaches in combination with contextual augmentation to answer **RQ6**: *How does the performance of knowledge extraction frameworks influence the performance of query prediction?*.

11.1.3 Keyphrase Extraction using Language Models and Knowledge Graphs

In Chapter 6, we demonstrate that explicit keyphrases extracted from input texts can be linked to KGs using traditional entity linking systems such as MAG (Moussallem et al., 2018). This process allows for the extraction of additional KG terms to facilitate the generation of absent keyphrases. For KGQA, we adapted this approach by combining it with contextual augmentation to extract explicitly and implicitly mentioned KG resources from questions. This method contributes to $\mathbf{RQ}_1$. Our results in Chapter 9 further show that combining MAG with contextual augmentation can improve ERL performance– specifically in terms of recall–when compared to traditional neural ranking approaches. In the end-to-end QA setup, this pipeline achieves better results than all traditional approaches. However, neural disambiguation approaches achieve better results than MAG on both tasks. These findings contribute to $\mathbf{RQ}_2$, $\mathbf{RQ}_3$ and $\mathbf{RQ}_6$.

11.1.4 UniQ-Gen: Unified Query Generation across Multiple Knowledge Graphs

In Chapter 7, we showed how it is possible to fine-tune a single LLM to generate queries across multiple knowledge graphs, achieving performance comparable to that of individually trained models. To show this, we fine-tuned models in a pure query generation setup, assuming gold entities and relations as input. Additionally, we conducted experiments in an end-to-end setup with predicted entities and relations of an ERL system. The results clearly show that in both setups, the joint model performs comparably to the individual model across all datasets and both considered KGs. These findings directly answer $\mathbf{RQ}_4$: *Can LLMs be jointly fine-tuned for query generation for multiple KGs, and how do these models perform compared to models trained for single KGs?*.

In addition, we observed that introducing output from ERL frameworks into the training data for fine-tuning query generation models enhancing end-to-end query generation results. This can be attributed to the fact that LLMs trained exclusively on gold KG resources tend to merely copy the provided entities and relations from the input. This occurs because they have only encountered relevant entities and relations during training. In practice, it is not possible to guarantee that all KG resources extracted by ERL apply to the query prediction step, meaning that query generation models must be robust to irrelevant entities. To this end, introducing predicted knowledge into the training process can solve this problem, at the cost of executing an additional preprocessing step on the training data. These findings contribute to $\mathbf{RQ}_6$.

11.1.5 Knowledge Graph Question Answering using Graph-Pattern Isomorphism

To compare traditional, non-LLM-based query generation approaches with current LLM-based methods, we present our approach, TeBaQA, in Chapter 8. This approach uses graph pattern isomorphism to identify frequent query templates and train a template classifier on input questions from the training split of KGQA benchmarks. At inference time, a template classifier is used to identify a matching query template for each input question. Entities, relations, and types are extracted by extracting n-grams and applying a TF-IDF-based search index. Finally, rule-based query ranking is applied to select the final query.

The results show that this approach achieved state-of-the-art results compared to other traditional approaches on some datasets at the time of its release. However, compared to TeBaQA, LLM-based KGQA pipelines consistently achieve superior results across most query generation tasks. Furthermore, this approach is bound to templates that are well-supported in the training data, whereas the LLM-based approach can learn rare templates due to its few-shot capabilities. However, one key advantage of traditional systems is that less or no training is required compared to LLM-based fine-tuning approaches. The findings of this chapter answer $\mathbf{RQ}_5$: *How well can traditional template-based question answering approaches perform on the KGQA task compared to LLM-based approaches?*

11.1.6 Evaluation of ERL for Question Answering over Knowledge Graphs

We conducted a joint evaluation of neural and traditional entity linking approaches as described in Chapter 9. The results show that neural approaches outperform traditional approaches on all datasets. The main advantage of neural systems is the vast amount of context LLMs have seen during pre-training. Overall, the recall has been shown to be crucial to KGQA system performance, making neural linking an essential component of current KGQA systems. Furthermore, we demonstrated that contextual augmentation not only enhances ERL performance of LLM-based systems on KGQA benchmarks but also improves the efficacy of traditional ED systems like MAG (Moussallem et al., 2018). This chapter answers $\mathbf{RQ}_3$ and $\mathbf{RQ}_4$.

11.1.7 Application: Enhancing Answers Verbalization using Large Language Models

As an application of semantic parsing KGQA systems, we have shown that SPARQL queries and RDF triples can enhance the performance of fine-tuned answer verbalization systems (Chapter 10). Verbalization of answers can improve the user experience of dialogue-based systems, such as chatbots or speech assistants. However, our fine-tuning approach is primarily designed for a low-resource setup, where deploying large LLMs such as GPT is not feasible.

11.2 Future Work

In this thesis, we focused on extracting relevant resources from KGs for semantic parsing-based KGQA systems. Furthermore, we mostly rely on fine-tuning approaches for query prediction. Recent research in NLP has shown that retrieval-augmented generation (RAG), which applies retrieval techniques to inject knowledge from databases or text corpora, can improve the performance of open-domain question answering (Fan et al., 2024), obviating the need for extensive fine-tuning. However, for semantic parsing on KGs, the query generation model must be aware of the underlying KG's structure, which can be addressed to a large extent by providing examples of question-query pairs to an LLM during fine-tuning. Since vanilla LLM models are usually unaware of this structure, it must be presented to the model via prompts, which necessitates extracting additional information beyond entity and relation IDs, such as relevant subgraphs (Han et al., 2025). Generating subgraphs is a complex task in itself, as it is infeasible to decide individually for each node in the KG whether it should be part of the subgraph due to the immense graph size. Thus, subgraph extraction approaches rely on iteratively ranking and expanding paths starting from the entities mentioned in a question (Zhang et al., 2022a). This iterative approach either requires implementing a graph ranking algorithm or applying LLMs for path ranking, both of which are computationally expensive. Consequently, one goal for future work is to develop efficient extraction techniques for graph-RAG pipelines.

Another research direction is to apply deep reinforcement learning on LLMs for KGQA. Recent research has shown that reinforcement learning can improve the reasoning capabilities of LLMs (DeepSeek-AI et al., 2025). Furthermore, GPT models apply this technique to refine their outputs based on human feedback, as mentioned in Section 2.2.4. Following this idea, Zhang and Zhao (2025) have recently developed an approach that applies reinforcement learning on KGs. However, their experiments are limited to the Hits@1-score on entities. Additionally, their approach does not optimize the LLM itself, since it only uses

the LLM to guide the decision process while optimizing smaller Transformer and LSTM models. In the future, it might be interesting to experiment to what extend it is possible to apply reinforcement learning on LLMs to predict SPARQL queries in an unsupervised or semi-supervised manner. One main benefit of this approach might be that datasets such as Mintaka (Sen et al., 2022) can also be leveraged for training, even though they only provide answers and question entities rather than SPARQL queries.

The field of hybrid question answering, which integrates information from diverse sources, including images, text, knowledge graphs, and relational databases, represents a promising avenue for future research. Recent studies have demonstrated the successful retrieval of images from text (Liu et al., 2021b) and the efficacy of the Transformer architecture in computer vision tasks (Dosovitskiy et al., 2021). Furthermore, GPT-4 has proven successful at generating natural language text from images and producing images from textual descriptions. Together with the development of RAG systems for structured and unstructured knowledge, advances in image processing might be used to incorporate information extracted from images into QA systems. However, combining these approaches requires integrating different methods and building complex pipelines. In this context, it might be helpful to implement a scheduling system to determine in advance which knowledge sources should be used to answer a specific user question.

Finally, another research direction is to improve the performance of retrieval systems on knowledge graphs by leveraging graph structures. Currently, text-based dense retrieval approaches mainly rely on textual descriptions of entities (Wu et al., 2020a; Zhang et al., 2022c). However, leveraging the graph structure is challenging, since entities are often included in large sets of triples, making it infeasible to include all of them in a textual description. In particular, training such dense retrieval models requires a set of candidate entities. Another problem is that entities in text are not always directly connected within the KG, so obtaining a subgraph is a difficult task that would require training on its own, described earlier. An option might be to leverage only the distances between candidate entities extracted from the input text, while also incorporating information about traversed edges. However, this leads to a loss of context, as intermediate entities along the path may provide crucial evidence for the retrieval system. In this context, a future goal can be to learn deep representations of paths within the KG by using Graph Transformers (Shehzad et al., 2024).

Bibliography

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2018. Never-ending learning for open-domain question answering over knowledge bases. In Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW, pages 1053–1062.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 4895–4901, Singapore. Association for Computational Linguistics.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pages 54–59.

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of Proceedings of Machine Learning Research, pages 233–242. PMLR.

Ram G. Athreya, Srividya Bansal, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck. 2020. Template-based question answering using recursive neural networks. CoRR, abs/2004.13843.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg. Springer-Verlag.

Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. 1999. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., USA.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Debayan Banerjee, Pranav Ajit Nair, Jivat Neet Kaur, Ricardo Usbeck, and Chris Biemann. 2022. Modern baselines for sparql semantic parsing. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 22. ACM.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

N. Baramiia, A. Rogulina, S. Petrakov, V. Kornilov, and A. Razzhigaev. 2022. Ranking approach to monolingual question answering over knowledge graphs. In Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference (ESWC 2022).

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In Proceedings of the 22nd Conference on Computational Natural Language Learning, pages 221–229.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Nikita Bhutani, Xinyi Zheng, Kun Qian, Yunyao Li, and H. Jagadish. 2020. Answering complex questions by combining information from curated and extracted knowledge bases. In Proceedings of the First Workshop on Natural Language Interfaces, pages 1–10, Online. Association for Computational Linguistics.

Andrzej Bialecki, Rob Muir, and Grant Ingersoll. 2012. Apache lucene 4. In OSIR@SIGIR.

Alexander Bigerl, Lixi Conrads, Charlotte Behning, Mohamed Ahmed Sherif, Muhammad Saleem, and Axel-Cyrille Ngonga Ngomo. 2020. Tentris – A tensor-based triple store. In The Semantic Web – ISWC 2020, pages 56–73, Cham. Springer International Publishing.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5:135–146.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, pages 1247–1250, New York, NY, USA. Association for Computing Machinery.

Manuel Alejandro Borroto, Francesco Ricca, and Bernardo Cuteri. 2021. A system for translating natural language questions into sparql queries with neural networks: Preliminary results. In SEBD 2021: Italian Symposium on Advanced Database Systems, pages 226–234, Aachen, Germany. RWTH Aachen.

Florian Boudin, Ygor Gallina, and Akiko Aizawa. 2020. Keyphrase generation for scientific document retrieval. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1118–1126, Online. Association for Computational Linguistics.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In Proceedings of the Sixth International Joint Conference on Natural Language Processing, pages 543–551.

Arwen V. Bradley and Carlos Alberto Gomez-Uribe. 2021. How can increased randomness in stochastic gradient descent improve generalization? CoRR, abs/2108.09507.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. YAKE! keyword extraction from single documents using multiple local features. Information Sciences, 509:257–289.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann Lecun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. 2019. Entropy-SGD: biasing gradient descent into wide valleys. Journal of Statistical Mechanics: Theory and Experiment, 2019:124018.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2021. FedMatch: Federated learning over heterogeneous question answering data. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21, pages 181–190, New York, NY, USA. Association for Computing Machinery.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022. CorpusBrain: Pre-train a generative retrieval model for knowledge-intensive language tasks. In Proceedings of the 31st ACM International Conference on Information and Knowledge Management, CIKM 22, pages 191–200. ACM.

Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. 2019. Title-guided encoding for keyphrase generation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 6268–6275.

Justin Chiu and Keiji Shinzato. 2022. Cross-encoder data annotation for bi-encoder based product matching. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 161–168, Abu Dhabi, UAE. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Ellen M. Voorhees, and Ian Soboroff. 2021. Trec deep learning track: Reusable test collections in the large data regime. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21, pages 2369–2375, New York, NY, USA. Association for Computing Machinery.

Richard Cyganiak. 2005. A relational algebra for SPARQL.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In Proceedings of the 9th International Conference on Semantic Systems (I-Semantics).

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: fast and memory-efficient exact attention with io-awareness. In Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, pages 933–941. JMLR.org.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing reasoning capability in llms via reinforcement learning.

Hervé Déjean, Stéphane Clinchant, and Thibault Formal. 2024. A thorough comparison of cross-encoders and llms for reranking SPLADE. CoRR, abs/2403.10407.

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. 2015. Analysis of named entity recognition and linking for tweets. Information Processing & Management, 51(2):32–49.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Papa Abdou Karim Karou Diallo and Amal Zouaq. 2025. FRASE: Structured representations for generalizable sparql query generation.

Dennis Diefenbach, Andreas Both, Kamal Singh, and Pierre Maret. 2020. Towards a question answering system over the semantic web. Semantic Web, 11(3):421–439.

Wentao Ding, Jinmao Li, Liangchuan Luo, and Yuzhong Qu. 2024. Enhancing complex question answering over knowledge graphs through evidence pattern retrieval. In Proceedings of the ACM Web Conference 2024, WWW '24, pages 2106–2115, New York, NY, USA. Association for Computing Machinery.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. The Faiss library. IEEE Transactions on Big Data, PP:1–17.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia. In The Semantic Web – ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II, pages 69–78, Berlin, Heidelberg. Springer-Verlag.

Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. 2018. EARL: Joint entity and relation linking for question answering over knowledge graphs. In The Semantic Web – ISWC 2018: 17th International Semantic Web Conference, Monterey,

CA, USA, October 8–12, 2018, Proceedings, Part I, pages 108–126, Berlin, Heidelberg. Springer-Verlag.

Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. 2021. Activation functions in deep learning: A comprehensive survey and benchmark. Neurocomputing, 503:92–108.

Frank Eibe, MA Hall, IH Witten, and JC Pal. 2016. The WEKA workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques, 4.

Vincent Emonet, Jerven Bolleman, Severine Duvaud, Tarcisio Mendes de Farias, and Ana Claudia Sima. 2025. LLM-based sparql query generation from natural language over federated knowledge graphs.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on RAG meeting LLMs: Towards retrieval-augmented large language models. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '24, pages 6491–6501, New York, NY, USA. Association for Computing Machinery.

Zheng Fang, Yanan Cao, Qian Li, Dongjie Zhang, Zhenyu Zhang, and Yanbing Liu. 2019. Joint entity linking with deep reinforcement learning. In The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019, pages 438–447. ACM.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), Ann Arbor, Michigan. Association for Computational Linguistics.

Rand Fishkin. 2017. The state of searcher behavior revealed through 23 remarkable statistics. https://moz.com/blog/state-of-searcher-behavior-revealed. [Online; accessed 05-March-2026].

Hans C. Fogedby. 1994. Langevin equations for continuous time lévy flights. Phys. Rev. E, 50:1657–1660.

Bin Fu, Yunqi Qiu, Chengguang Tang, Yang Li, Haiyang Yu, and Jian Sun. 2020. A survey on complex question answering over knowledge base: Recent advances and challenges. arXiv preprint arXiv:2007.13069.

Jorge Gabín and Javier Parapar. 2025. Leveraging retrieval-augmented generation for keyphrase synonym suggestion. In Advances in Information Retrieval: 47th European Conference on Information Retrieval, ECIR 2025, Lucca, Italy, April 6–10, 2025, Proceedings, Part II, pages 311–327, Berlin, Heidelberg. Springer-Verlag.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, pages 2619–2629. Association for Computational Linguistics.

Jörg Garrel and Jana Mayer. 2023. Artificial intelligence in studies–use of ChatGPT and AI-based tools among students in germany. Humanities and Social Sciences Communications, 10.

O. Gervasi and V. Kumar. 2006. Computational Science and Its Applications - ICCSA 2006: International Conference, Glasgow, UK, May 8-11, 2006, Proceedings. Computational Science and Its Applications: ICCSA 2006. Springer.

Sujatha Das Gollapalli, Xiao-Li Li, and Peng Yang. 2017. Incorporating expert knowledge into keyphrase extraction. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 31.

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond IID: three levels of generalization for question answering on knowledge bases. In Proceedings of the Web Conference 2021, pages 3477–3488. ACM.

Yu Gu and Yu Su. 2022. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In Proceedings of the 29th International Conference on Computational Linguistics, pages 1718–1731, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2022. Semantic models for the first-stage retrieval: A comprehensive review. ACM Trans. Inf. Syst., 40(4).

Ria Hari Gusmita, Rricha Jalota, Daniel Vollmers, Jan Reineke, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck. 2019. QUANT - question answering benchmark curator. In Semantic Systems. The Power of AI and Knowledge Graphs, pages 343–358, Cham. Springer International Publishing.

Kailash A. Hambarde and Hugo Proença. 2023. Information retrieval: Recent advances and beyond. IEEE Access, 11:76581–76604.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qi He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. 2025. Retrieval-augmented generation with graphs (GraphRAG).

Yanchao Hao, Hao Liu, Shizhu He, Kang Liu, and Jun Zhao. 2018. Pattern-revising enhanced simple question answering over knowledge bases. In Proceedings of the 27th International Conference on Computational Linguistics, COLING, pages 3272–3282.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1262–1273, Baltimore, Maryland. Association for Computational Linguistics.

Tom Hedges. 2024. AI in search: Charting changing trends in search engines. https://www.gwi.com/blog/ai-and-search. [Online; accessed 05-March-2026].

Adam Heitzman. 2025. How people search today: A study on evolving search behaviors in 2025. https://www.highervisibility.com/seo/learn/how-people-search/. [Online; accessed 05-March-2026].

Dan Hendrycks and Kevin Gimpel. 2023. Gaussian error linear units (GELUs).

Julia Hirschberg and Christopher D. Manning. 2015. Advances in natural language processing. Science, 349(6245):261–266.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural Comput., 9(8):1735–1780.

Marvin Hofer, Sebastian Hellmann, Milan Dojchinovski, and Johannes Frey. 2020. The new dbpedia release cycle: Increasing agility and efficiency in knowledge extraction workflows. In Semantic Systems. In the Era of Knowledge Graphs: 16th International Conference on Semantic Systems, SEMANTiCS 2020, Amsterdam, The Netherlands, September 7–10, 2020, Proceedings, pages 1–18, Berlin, Heidelberg. Springer-Verlag.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. KORE: keyphrase overlap relatedness for entity disambiguation. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12, pages 545–554, New York, NY, USA. Association for Computing Machinery.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 782–792, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. 2017. Survey on challenges of question answering in the semantic web. Semantic Web, 8(6):895–920.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia Damato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge graphs. ACM Computing Surveys, 54(4):1–37.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. ACM Trans. Inf. Syst., 43(2).

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In Proceedings of the 2003 conference on Empirical methods in natural language processing, pages 216–223.

Md Zahidul Islam, Jixue Liu, Jiuyong Li, Lin Liu, and Wei Kang. 2019. A semantics aware random forest for text classification. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19, pages 1061–1070, New York, NY, USA. Association for Computing Machinery.

Rricha Jalota, Daniel Vollmers, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2021. LAUREN - knowledge graph summarization for question answering. In 2021 IEEE 15th International Conference on Semantic Computing (ICSC), pages 221–226.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Hang Jiang, Sairam Gurajada, Qiuhao Lu, Sumit Neelam, Lucian Popa, Prithviraj Sen, Yunyao Li, and Alexander Gray. 2021. LNN-EL: A neuro-symbolic approach to short-text entity linking. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 775–787, Online. Association for Computational Linguistics.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. 2017. How to escape saddle points efficiently. In Proceedings of the 34th International Conference on Machine Learning, ICML, volume 70 of Proceedings of Machine Learning Research, pages 1724–1732. PMLR.

Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M. Kakade, and Michael I. Jordan. 2021. On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. J. ACM, 68(2).

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 7(3):535–547.

Endri Kacupaj, Barshana Banerjee, Kuldeep Singh, and Jens Lehmann. 2021a. ParaQA: A question answering dataset with paraphrase responses for single-turn conversation.

Endri Kacupaj, Shyamnath Premnadh, Kuldeep Singh, Jens Lehmann, and Maria Maleshkova. 2021b. VOGUE: Answer verbalization through multi-task learning. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 563–579. Springer.

Endri Kacupaj, Kuldeep Singh, Maria Maleshkova, and Jens Lehmann. 2022. An answer verbalization dataset for conversational question answerings over knowledge graphs. arXiv preprint arXiv:2208.06734.

Endri Kacupaj, Hamid Zafar, Jens Lehmann, and Maria Maleshkova. 2020. VQuAnDa: Verbalization question answering dataset. In The Semantic Web, pages 531–547, Cham. Springer International Publishing.

Suvarna G Kanakaraddi and Suvarna S Nandyal. 2018. Survey on parts of speech tagger techniques. In 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT), pages 1–6.

Jenna Kanerva, Filip Ginter, and Tapio Salakoski. 2019. Universal lemmatizer: A sequence-to-sequence model for lemmatizing universal dependencies treebanks. Natural Language Engineering, 27:545–574.

Jakub Kanis and Lucie Skorkovská. 2010. Comparison of different lemmatization approaches through the means of information retrieval performance. In Text, Speech and Dialogue, pages 93–100, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander G. Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois P. S. Luus, Ndivhuwo Makondo, Nandana Mihindukulasooriya, Tahira Naseem, Sumit Neelam, Lucian Popa, Revanth Gangi Reddy, Ryan Riegel, Gaetano Rossiello, Udit Sharma, G. P. Shrivatsa Bhargav, and Mo Yu. 2020. Question answering over knowledge bases by leveraging semantic parsing and neuro-symbolic reasoning. CoRR, abs/2012.01707.

Simon Kemp. 2026. Digital 2025: Global overview report. https://datareportal.com/reports/digital-2025-global-overview-report. [Online; accessed 05-March-2026].

Imed Keraghel, Stanislas Morbieu, and Mohamed Nadif. 2024. Recent advances in named entity recognition: A comprehensive survey and comparative study.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net.

Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. 2023. Natural language processing: state of the art, current trends and challenges. Multimedia Tools and Applications, 82:3713–3744.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 21–26.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient Transformer. In International Conference on Learning Representations.

Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM), 46(5):604–632.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In Proceedings of the 22nd Conference on Computational Natural Language Learning, pages 519–529, Brussels, Belgium. Association for Computational Linguistics.

Liubov Kovriguina, Roman Teucher, Daniil Radyush, and Dmitry Mouromtsev. 2023. SPARQLGEN: One-shot prompt-based approach for SPARQL query generation. In Proceedings of the Posters and Demo Track of the 19th International Conference on Semantic Systems co-located with 19th International Conference on Semantic Systems (SEMANTiCS 2023), Leipzing, Germany, September 20 to 22, 2023, volume 3526 of CEUR Workshop Proceedings. CEUR-WS.org.

Mikalai Krapivin and Maurizio Marchese. 2009. Large dataset for keyphrase extraction.

Frederik Kunstner, Alan Milligan, Robin Yadav, Mark Schmidt, and Alberto Bietti. 2024. Heavy-tailed class imbalance and why adam outperforms gradient descent on language models. In Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Tuan Lai, Heng Ji, and ChengXiang Zhai. 2022. Improving candidate retrieval with entity profile generation for Wikidata entity linking. In Findings of the Association for Computational Linguistics: ACL 2022, pages 3696–3711, Dublin, Ireland. Association for Computational Linguistics.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A survey on complex knowledge base question answering: Methods, challenges and solutions. In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, pages 4483–4491. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

E. L. Lehmann. 1992. Introduction to Student (1908) The Probable Error of a Mean, pages 29–32. Springer New York, New York, NY.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015a. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web, 6(2):167–195.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015b. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web, 6(2):167–195.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. Unsupervised keyphrase extraction by jointly modeling local and global context. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 155–164.

Xiao Ling, Sameer Singh, and Daniel Weld. 2015. Design challenges for entity linking. Transactions of the Association for Computational Linguistics, 3:315–328.

Tianyi Liu, Yan Li, Song Wei, Enlu Zhou, and Tuo Zhao. 2021a. Noisy gradient descent converges to flat minima for nonconvex matrix factorization. In The 24th International Conference on Artificial Intelligence and Statistics, AISTATS, volume 130 of Proceedings of Machine Learning Research, pages 1891–1899. PMLR.

Zheyuan Liu, Cristian Rodriguez-Opazo, Damien Teney, and Stephen Gould. 2021b. Image retrieval on real-life images with pre-trained vision-and-language models. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pages 2105–2114.

Vanessa Lopez, Miriam Fernández, Enrico Motta, and Nico Stieler. 2012. PowerAqua: Supporting users in querying and exploring the semantic web. Semant. Web, 3(3):249–265.

Vanessa Lopez, Enrico Motta, and Victoria Uren. 2006. PowerAqua: fishing the semantic web. In Proceedings of the 3rd European Conference on The Semantic Web: Research and Applications, ESWC'06, pages 393–410, Berlin, Heidelberg. Springer-Verlag.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning LLaMA for multi-stage text retrieval. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24, pages 2421–2425, New York, NY, USA. Association for Computing Machinery.

Mateus Machado and Evandro Ruiz. 2024. Evaluating large language models for the tasks of PoS tagging within the Universal Dependency framework. In Proceedings of the 16th International Conference on Computational Processing of Portuguese - Vol. 1, pages 454–460, Santiago de Compostela, Galicia/Spain. Association for Computational Lingustics.

Goutam Majumder, Partha Pakray, Alexander Gelbukh, and David Pinto. 2016. Semantic textual similarity methods, tools, and applications: A survey. Computación y Sistemas, 20(4):647–665.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Boolean retrieval, pages 1–17. Cambridge University Press.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. 2023. Cross-entropy loss functions: theoretical analysis and applications. In Proceedings of the 40th International Conference on Machine Learning, ICML'23. JMLR.org.

Reza Yousefi Maragheh, Chenhao Fang, Charan Chand Irugu, Parth Parikh, Jason Cho, Jianpeng Xu, Saranyan Sukumar, Malay Patel, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2023. LLM-TAKE: Theme aware keyword extraction using large language models.

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia spotlight: shedding light on the web of documents. In Proceedings of the 7th International Conference on Semantic Systems, I-Semantics '11, pages 1–8, New York, NY, USA. Association for Computing Machinery.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 582–592.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In Proceedings of the 2004 conference on empirical methods in natural language processing, pages 404–411.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In International Conference on Learning Representations.

George A. Miller. 1994. WordNet: A lexical database for English. In Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: A comprehensive review. ACM Comput. Surv., 54(3).

Sebastien Montella, Lina Rojas-Barahona, Frederic Bechet, Johannes Heinecke, and Alexis Nasr. 2022. Transfer learning and masked generation for answer verbalization. In Proceedings of the Workshop on Structured and Unstructured Knowledge Integration (SUKI), pages 47–54, Seattle, USA. Association for Computational Linguistics.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: A unified approach. Transactions of the Association for Computational Linguistics, 2:231–244.

Farhad Mortezapour Shiri, Thinagaran Perumal, and Raihani Mohamed. 2024. A comprehensive overview and comparative analysis on deep learning model. Journal on Artificial Intelligence, 6:301–360.

Diego Moussallem, Ricardo Usbeck, Michael Röder, and Axel-Cyrille Ngonga Ngomo. 2018. Entity linking in 40 languages using MAG. In ESWC, pages 176–181.

Diego Moussallem, Ricardo Usbeck, Michael Röeder, and Axel-Cyrille Ngonga Ngomo. 2017. MAG: A multilingual, knowledge-base agnostic and deterministic entity linking approach. In Proceedings of the 9th Knowledge Capture Conference, K-CAP '17, New York, NY, USA. Association for Computing Machinery.

Isaiah Onando Mulang, Kuldeep Singh, Chaitali Prabhu, Abhishek Nadgeri, Johannes Hoffart, and Jens Lehmann. 2020. Evaluating the impact of knowledge graph context on entity disambiguation models. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM 20, pages 2157–2160. ACM.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. Lingvisticae Investigationes, 30:3–26.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10, pages 807–814, Madison, WI, USA. Omnipress.

Usman Naseem, Imran Razzak, Shah Khalid Khan, and Mukesh Prasad. 2020. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial intelligence, 193:217–250.

Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Pendus, Saswati Dana, Dinesh Garg, Achille Fokoue, G P Shrivatsa Bhargav, Dinesh Khandelwal, Srinivas Ravishankar, Sairam Gurajada, Maria Chang, Rosario Uceda-Sosa, Salim Roukos, Alexander Gray, Guilherme Lima, Ryan Riegel, Francois Luus, and L V Subramaniam. 2022. SYGMA: A system for generalizable and modular question answering over knowledge bases. In Findings of the Association for Computational Linguistics: EMNLP 2022, pages 3866–3879, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yurii E. Nesterov and Vladimir G. Spokoiny. 2017. Random gradient-free minimization of convex functions. Found. Comput. Math., 17(2):527–566.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers, pages 317–326, Berlin, Heidelberg. Springer Berlin Heidelberg.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. CoRR, abs/1901.04085.

Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Darío Garigliotti, and Roberto Navigli. 2015. Open knowledge extraction challenge. In SemWebEval@ESWC.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, ukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, ukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney,

Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. GPT-4 technical report.

Antonio Orvieto, Hans Kersting, Frank Proske, Francis R. Bach, and Aurélien Lucchi. 2022. Anticorrelated noise injection for improved generalization. In International Conference on Machine Learning, ICML, volume 162 of Proceedings of Machine Learning Research, pages 17094–17116. PMLR.

Antonio Orvieto, Anant Raj, Hans Kersting, and Francis R. Bach. 2023. Explicit regularization in overparametrized models via noise injection. In International Conference on Artificial Intelligence and Statistics, 2023.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In Annual Meeting of the Association for Computational Linguistics.

Alberto Parravicini, Rhicheek Patra, Davide B. Bartolini, and Marco D. Santambrogio. 2019. Fast and accurate entity linking via graph embedding. In Proceedings of the 2nd Joint International Workshop on Graph Data Management Experiences & Systems

(GRADES) and Network Data Analytics (NDA), GRADES-NDA'19, New York, NY, USA. Association for Computing Machinery.

Razvan Pascanu, Tomas Mikolov, and Y. Bengio. 2012. On the difficulty of training recurrent neural networks. 30th International Conference on Machine Learning, ICML 2013.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 2233–2241. IEEE Computer Society.

Qiwei Peng, David Weir, Julie Weeds, and Yekun Chai. 2022. Predicate-argument based bi-encoder for paraphrase identification. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5579–5589, Dublin, Ireland. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. 2022. QALD-9-plus: A multilingual dataset for question answering over DBpedia and Wikidata translated by native speakers.

Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. 2009. Semantics and complexity of SPARQL. ACM Trans. Database Syst., 34(3).

Laura Perez-Beltrachini, Parag Jain, Emilio Monti, and Mirella Lapata. 2023. Semantic parsing for conversational question answering over knowledge graphs. In Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2507–2522, Dubrovnik, Croatia. Association for Computational Linguistics.

Svetlana Pestryakova, Daniel Vollmers, Mohamed Ahmed Sherif, Stefan Heindorf, Muhammad Saleem, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2022. CovidPubGraph: A fair knowledge graph of covid-19 publications. Scientific Data.

Georgios Petasis, Frantz Vichot, Francis Wolinski, Georgios Paliouras, Vangelis Karkaletsis, and Constantine D. Spyropoulos. 2001. Using machine learning to maintain rule-based named-entity recognition and classification systems. In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, ACL '01, pages 426–433, USA. Association for Computational Linguistics.

Robert M Polstra III. 2005. A case study on how to manage the theft of information. In Proceedings of the 2nd annual conference on Information security curriculum development, pages 135–138.

Boris Teodorovich Polyak and Aleksandr Borisovich Tsybakov. 1990. Optimal order of accuracy of search algorithms in stochastic optimization. Problemy Peredachi Informatsii, 26(2):45–53.

David M. W. Powers. 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. Journal of Machine Learning Technologies, 2(1):37–63.

Umair Qudus, Michael Röder, Daniel Vollmers, and Axel-Cyrille Ngonga Ngomo. 2024. ExPrompt: Augmenting prompts using examples as modern baseline for stance classification. In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM '24, pages 3994–3999, New York, NY, USA. Association for Computing Machinery.

L. Rabiner and B. Juang. 1986. An introduction to hidden markov models. IEEE ASSP Magazine, 3(1):4–16.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67.

Jonathan Raiman and Olivier Raiman. 2018. Deeptype: Multilingual entity linking by neural type system evolution.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. 2017. Swish: a self-gated activation function. arXiv: Neural and Evolutionary Computing.

Juan Enrique Ramos. 2003. Using TF-IDF to determine word relevance in document queries.

Jishnu Ray Chowdhury, Seo Yeon Park, Tuhin Kundu, and Cornelia Caragea. 2022. KP-DROP: Improving absent keyphrase generation. In Findings of the Association for Computational Linguistics: EMNLP 2022, pages 4853–4870, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 3980–3990. Association for Computational Linguistics.

Herbert E. Robbins. 1951. A stochastic approximation method. Annals of Mathematical Statistics, 22:400–407.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994, volume 500-225 of NIST Special Publication, pages 109–126. National Institute of Standards and Technology (NIST).

Michael Röder, Ricardo Usbeck, Sebastian Hellmann, Daniel Gerber, and Andreas Both. 2014. N^3 - a collection of datasets for named entity recognition and disambiguation in the NLP interchange format. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), pages 3529–3533, Reykjavik, Iceland. European Language Resources Association (ELRA).

Michael Röder, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. 2018. GERBIL - benchmarking named entity recognition and linking consistently. Semantic Web, 9(5):605–625.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. Nature, 323:533–536.

Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. 2020. Keyphrase extraction as sequence labeling using contextualized embeddings. In European Conference on Information Retrieval, pages 328–335. Springer.

Ahmad Sakor, Kuldeep Singh, Anery Patel, and Maria-Esther Vidal. 2020. Falcon 2.0: An entity and relation linking tool over wikidata. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management, CIKM' 20, pages 3141–3148, New York, NY, USA. Association for Computing Machinery.

Amri Samir and Zenkouar Lahbib. 2018. Stemming and lemmatization for information retrieval systems in Amazigh language. In Big Data, Cloud and Applications, pages 222–233, Cham. Springer International Publishing.

Jürgen Schmidhuber. 2014. Deep learning in neural networks: An overview. Neural networks : the official journal of the International Neural Network Society, 61:85–117.

Robin M. Schmidt. 2019. Recurrent neural networks (RNNs): A gentle introduction and overview.

Idan Schwartz. 2021. Ensemble of MRR and NDCG models for visual dialog.

Priyanka Sen, Alham Fikri Aji, and Amir Saffari. 2022. Mintaka: A complex, natural, and multilingual dataset for end-to-end question answering. In Proceedings of the 29th International Conference on Computational Linguistics, pages 1604–1619, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann. 2022. Neural entity linking: A survey of models based on deep learning. Semantic Web, 13(3):527–570.

Arnab Sharma, Daniel Vollmers, and Axel-Cyrille Ngonga Ngomo. 2025. Calibrating language models for neural ranking under noisy supervision with relaxed labels. In Proceedings of the 2nd Workshop on Uncertainty-Aware NLP (UncertaiNLP 2025), pages 259–272, Suzhou, China. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. CoRR, abs/1911.02150.

Noam Shazeer. 2020. GLU variants improve transformer.

Ahsan Shehzad, Feng Xia, Shagufta Abid, Ciyuan Peng, Shuo Yu, Dongyu Zhang, and Karin Verspoor. 2024. Graph transformers: A survey.

Saeedeh Shekarpour and Sören Auer. 2014. "SINA: semantic interpretation of user queries for question answering on interlinked data. SIGWEB Newsl., 2014.

Wei Shen, Jianyong Wang, and Jiawei Han. 2014. Entity linking with a knowledge base: Issues, techniques, and solutions. IEEE Transactions on Knowledge and Data Engineering, 27(2):443–460.

Xianjie Shen, Yinghan Wang, Rui Meng, and Jingbo Shang. 2022. Unsupervised deep keyphrase generation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, pages 11303–11311.

Liang Shi, Zhengju Tang, Nan Zhang, Xiaotong Zhang, and Zhi Yang. 2025. A survey on employing large language models for text-to-SQL tasks. ACM Comput. Surv., 58(2).

K. Shivashankar, K. Benmaarouf, and N. Steinmetz. 2022. From graph to graph: AMR to SPARQL. In Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference (ESWC 2022).

Yiheng Shu and Zhiwei Yu. 2024. Distribution shifts are bottlenecks: Extensive evaluation for grounding language models to knowledge bases. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop, pages 71–88, St. Julian's, Malta. Association for Computational Linguistics.

Yiheng Shu, Zhiwei Yu, Yuhan Li, Börje F. Karlsson, Tingting Ma, Yuzhong Qu, and Chin-Yew Lin. 2022. TIARA: Multi-grained retrieval for robust question answering over large knowledge bases.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 5464–5472. AAAI Press.

Kuldeep Singh, Andreas Both, Arun Sethupat Radhakrishna, and Saeedeh Shekarpour. 2018. Frankenstein: A platform enabling reuse of question answering components. In ESWC, volume 10843 of Lecture Notes in Computer Science, pages 624–638. Springer.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. Transactions of the Association for Computational Linguistics, 11:1–17.

Samuel L. Smith, Erich Elsen, and Soham De. 2020. On the generalization benefit of noise in stochastic gradient descent. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, Proceedings of Machine Learning Research.

Mingyang Song, Xuelian Geng, Songfang Yao, Shilong Lu, Yi Feng, and Liping Jing. 2024. Large language models as zero-shot keyphrase extractors: A preliminary empirical study.

Nadine Steinmetz and Harald Sack. 2013. Semantic multimedia information retrieval based on contextual descriptions. In The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings, volume 7882 of Lecture Notes in Computer Science, pages 382–396. Springer.

Chang Su, Jiexing Qi, He Yan, Kai Zou, and Zhouhan Lin. 2024a. Enhancing SPARQL generation by triplet-order-sensitive pre-training. In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM '24, pages 4061–4065, New York, NY, USA. Association for Computing Machinery.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024b. RoFormer: Enhanced transformer with rotary position embedding. Neurocomput., 568(C).

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 14918–14937, Singapore. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. 2021. A survey on neural speech synthesis.

Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Can ChatGPT replace traditional kbqa models? An in-depth analysis of the question answering performance of the GPT LLM family.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. In Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Ann Taylor, Mitchell Marcus, and Beatrice Santorini. 2003. The Penn Treebank: An Overview, pages 5–22. Springer Netherlands, Dordrecht.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito,

David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikua, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models. ArXiv, abs/2302.13971.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. LC-QuAD: A corpus for complex question answering over knowledge graphs. In International Semantic Web Conference, pages 210–218. Springer.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over RDF data. In Proceedings of the 21st World Wide Web Conference 2012, WWW, pages 639–648.

Ricardo Usbeck, Ria Hari Gusmita, Axel-Cyrille Ngonga Ngomo, and Muhammad Saleem. 2018a. 9th challenge on question answering over linked data (QALD-9). In Joint proceedings of (SemDeep-4) and (NLIWOD-4), volume 2241 of CEUR Workshop Proceedings, pages 58–64.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Felix Conrads, Michael Röder, and Giulio Napolitano. 2018b. 8th challenge on question answering over linked data (QALD-8). In Joint proceedings of (SemDeep-4) and (NLIWOD-4), volume 2241 of CEUR Workshop Proceedings, pages 51–57.

Ricardo Usbeck, Michael Röder, Michael Hoffmann, Felix Conrads, Jonathan Huthmann, Axel-Cyrille Ngonga Ngomo, Christian Demmler, and Christina Unger. 2019. Benchmarking Question Answering Systems. Semantic Web, 10(2):293–304.

Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, Muhammad Saleem, and Andreas Both. 2023. QALD-10 the 10th challenge on question answering over linked data. Under review in the Semantic Web Journal.

Svitlana Vakulenko, Javier David Fernandez Garcia, Axel Polleres, Maarten de Rijke, and Michael Cochez. 2019. Message passing for complex question answering over knowledge graphs. In CIKM, pages 1431–1440. ACM.

Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. 2020. REL: An entity linker standing on the shoulders of giants. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, pages 2197–2200, New York, NY, USA. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, pages 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Tim Verdonck, Bart Baesens, María Óskarsdóttir, and Seppe vanden Broucke. 2021. Special issue on feature engineering editorial. Mach. Learn., 113(7):3917–3928.

Daniel Vollmers, Rricha Jalota, Diego Moussallem, Hardik Topiwala, Axel-Cyrille Ngonga Ngomo, and Ricardo Usbeck. 2021. Knowledge graph question answering using graph-pattern isomorphism, pages 103–117. IOS Press.

Daniel Vollmers, Arnab Sharma, and Axel-Cyrille Ngonga Ngomo. 2026. Evaluating noisy optimization in finetuning lms for neural ranking. In NLDB 2026. NLDB 2026.

Daniel Vollmers, Parth Sharma, Hamada M. Zahera, and Axel-Cyrille Ngonga Ngomo. 2024. Enhancing answers verbalization using large language models. In Proceedings of the 20th International Conference on Semantic Systems, 17–19 September 2024, Amsterdam, The Netherlands. SEMANTICS 2024.

Daniel Vollmers, René Speck, Hamada M. Zahera, and Axel-Cyrille Ngonga Ngomo. 2025a. Evaluation of entity and relation linking for question answering over knowledge graphs. In Proceedings of the 13th Knowledge Capture Conference 2025, K-CAP '25, pages 211–214, New York, NY, USA. Association for Computing Machinery.

Daniel Vollmers, Nikit Srivastava, Hamada M. Zahera, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2025b. UniQ-Gen: Unified query generation across multiple knowledge graphs. In Knowledge Engineering and Knowledge Management, pages 174–189, Cham. Springer Nature Switzerland.

Daniel Vollmers, Hamada Zahera, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. 2025c. Contextual augmentation for entity linking using large language models. In Proceedings of the 31st International Conference on Computational Linguistics, pages 8535–8545, Abu Dhabi, UAE. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. Commun. ACM, 57(10):78–85.

Denny Vrandečić, Lydia Pintscher, and Markus Krötzsch. 2023. Wikidata: The making of. In Companion Proceedings of the ACM Web Conference 2023, WWW '23 Companion, pages 615–624, New York, NY, USA. Association for Computing Machinery.

Sebastian Walter and Hannah Bast. 2025. GRASP: Generic reasoning and sparql generation across knowledge graphs. In The Semantic Web – ISWC 2025: 24th International Semantic Web Conference, Nara, Japan, November 2–6, 2025, Proceedings, Part I, pages 271–289, Berlin, Heidelberg. Springer-Verlag.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. CoRR, abs/2212.03533.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, Guoyin Wang, and Chen Guo. 2025. GPT-NER: Named entity recognition via large language models. In Findings of the Association for Computational Linguistics: NAACL 2025, pages 4257–4275, Albuquerque, New Mexico. Association for Computational Linguistics.

Yue Wang, Jing Li, Hou Pong Chan, Irwin King, Michael R Lyu, and Shuming Shi. 2019. Topic-aware neural keyphrase generation for social media language. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2516–2526.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In ACL, pages 1332–1342.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020a. Scalable zero-shot entity linking with dense entity retrieval. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6397–6407, Online. Association for Computational Linguistics.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020b. Scalable zero-shot entity linking with dense entity retrieval. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, pages 6397–6407. Association for Computational Linguistics.

Peiyun Wu, Xiaowang Zhang, and Zhiyong Feng. 2019. A survey of question answering over knowledge base. In Knowledge Graph and Semantic Computing: Knowledge Computing and Language Understanding: 4th China Conference, CCKS 2019, Hangzhou, China, August 24–27, 2019, Revised Selected Papers 4, pages 86–97. Springer.

Amy Xin, Yunjia Qi, Zijun Yao, Fangwei Zhu, Kaisheng Zeng, Xu Bin, Lei Hou, and Juanzi Li. 2024. LLMAEL: Large language models are good context augmenters for entity linking.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In Proceedings of the 27th International Conference on Computational Linguistics, pages 2145–2158, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: deep contextualized entity representations with entity-aware self-attention. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, pages 6442–6454. Association for Computational Linguistics.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4142–4153.

Jiacheng Ye, Ruijian Cai, Tao Gui, and Qi Zhang. 2021. Heterogeneous graph neural networks for keyphrase generation. arXiv preprint arXiv:2109.04703.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RnG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

Junjie Yin, Jiahao Dong, Yingheng Wang, Christopher De Sa, and Volodymyr Kuleshov. 2024. ModuLoRA: Finetuning 2-bit LLMs on consumer GPUs by integrating with modular quantizers. TMLR.

Hamada M. Zahera, Manzoor Ali, Mohamed Ahmed Sherif, Diego Moussallem, and Axel Ngonga. 2024. Generating sparql from natural language using chain-of-thoughts prompting. In International Conference on Semantic Systems.

Hamada M. Zahera, Daniel Vollmers, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. 2022. MultPAX: Keyphrase extraction using language models and knowledge graphs. In The Semantic Web – ISWC 2022, pages 303–318, Cham. Springer International Publishing.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George E. Dahl, and Christopher J. Shallue. 2019a. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS 2019.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022a. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5773–5784, Dublin, Ireland. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019b. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization.

Kai Zhang, Yu Wang, Hongyi Wang, Lifu Huang, Carl Yang, Xun Chen, and Lichao Sun. 2022b. Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation. In Findings of the Association for Computational Linguistics: EMNLP 2022, pages 613–621, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Qin Zhang, Shangsi Chen, Dongkuan Xu, Qingqing Cao, Xiaojun Chen, Trevor Cohn, and Meng Fang. 2023. A survey for efficient open domain question answering. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 14447–14465, Toronto, Canada. Association for Computational Linguistics.

Wen Zhang, Long Jin, Yushan Zhu, Jiaoyan Chen, Zhiwei Huang, Junjie Wang, Yin Hua, Lei Liang, and Huajun Chen. 2025. TrustUQA: A trustful framework for unified structured data question answering. Proceedings of the AAAI Conference on Artificial Intelligence, 39:25931–25939.

Wenzheng Zhang, Wenyue Hua, and Karl Stratos. 2022c. EntQA: Entity linking as question answering. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net.

Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint model for question answering over multiple knowledge bases. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1).

Zhenyu Zhang, Xiaobo Sind, Tingwen Liu, Zheng Fang, and Quangang Li. 2020. Joint entity linking and relation extraction with neural networks for knowledge base population. In 2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020, pages 1–8. IEEE.

Zhiqiang Zhang and Wen Zhao. 2025. A collaborative reasoning framework powered by reinforcement learning and large language models for complex questions answering over knowledge graph. In Proceedings of the 31st International Conference on Computational Linguistics, pages 10672–10684, Abu Dhabi, UAE. Association for Computational Linguistics.

Jing Zhao, Junwei Bao, Yifan Wang, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2021. SGG: Learning to select, guide, and generate for keyphrase generation. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5717–5726.

Weiguo Zheng and Mei Zhang. 2019. Question answering over knowledge graphs via structural query patterns. CoRR, abs/1910.09760.

Mo Zhou, Tianyi Liu, Yan Li, Dachao Lin, Enlu Zhou, and Tuo Zhao. 2019. Toward understanding the importance of noise in training neural networks. In Proceedings of the 36th International Conference on Machine Learning, ICML, Proceedings of Machine Learning Research. PMLR.

Yucheng Zhou, Xiubo Geng, Tao Shen, Wenqiang Zhang, and Daxin Jiang. 2021. Improving zero-shot cross-lingual transfer for multilingual question answering over knowledge graph. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5822–5834, Online. Association for Computational Linguistics.

Xiangrong Zhu, Guangyao Li, and Wei Hu. 2023. Heterogeneous federated knowledge graph embedding learning and unlearning. In Proceedings of the ACM Web Conference 2023, WWW '23, pages 2444–2454, New York, NY, USA. Association for Computing Machinery.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2025. Large language models for information retrieval: A survey. ACM Trans. Inf. Syst. Just Accepted.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. Efficient RAG: Efficient retriever for multi-hop question answering. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 3392–3411, Miami, Florida, USA. Association for Computational Linguistics.

Lei Zou, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao. 2014. Natural language question answering over RDF: a graph data driven approach. In International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014, pages 313–324.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. 2023. A formal perspective on byte-pair encoding. In Findings of the Association for Computational Linguistics: ACL 2023, pages 598–614, Toronto, Canada. Association for Computational Linguistics.

Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. 2016. DoSeR - a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings. In The Semantic Web. Latest Advances and New Domains, pages 182–198, Cham. Springer International Publishing.