Conversational Question Answering over Knowledge Graphs based on Semantic Retrieval and Large Language Model

Haowen Ji

Master of Science
Data Science
School of Informatics
University of Edinburgh
2024

Abstract

Conversational Question Answering over Knowledge Graphs (KGQA) is a critical task that focuses on developing systems capable of understanding and responding to user questions within an ongoing dialogue context. The answers are exclusively based on Knowledge Graphs (KGs), ensuring precise and reliable responses when user questions are correctly interpreted and executed. However, KGQA presents unique challenges due to large subgraphs extracted from the knowledge graph, which can contain thousands of triples per user question, making it difficult for semantic parsing models to work effectively.

This paper proposes to improve a two-stage approach that integrates retrieval techniques to truncate subgraphs, ensuring their quality and reducing their size. We explore the effectiveness of LLMs in generating SPARQL queries based on these truncated subgraphs. Experiments on a stratified sample of the SPICE dataset reveal that the retrieval method effectively captures relevant information for simple questions, while the accumulated subgraph approach improves performance for indirect questions. Fine-tuned LLMs demonstrate strong ability to extract information from context and supplement it to the current question, and generate SPARQL queries. Performance variability across question types highlights the challenges associated with complex reasoning tasks. Future research should focus on improving retrieval methods, developing dynamic subgraph truncation techniques, enhancing LLM performance, and incorporating answer-sensitive knowledge generation. This work paves the way for more efficient, accurate, and scalable conversational KGQA systems.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Haowen Ji*)

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Laura Perez-Beltrachini, for her invaluable support and guidance throughout this project. Her insightful instruction, thoughtful direction, and consistent communication through numerous emails and weekly meetings have been instrumental in shaping this work. I deeply appreciate her patience and kindness, which have greatly contributed to the success of this endeavor.

Table of Contents

Chapter 1

Introduction

Conversational Question Answering over Knowledge Graphs (KGQA) is a critical task in natural language processing, focusing on developing systems capable of understanding and responding to user questions within an ongoing dialogue context. As the answers are exclusively based on Knowledge Graphs (KGs), the QA system will deliver responses that are both precise and reliable if it can correctly interpret and execute user questions. This distinguishes it from non-KG-based QA systems, which typically generate answers based on patterns learned from their inherent training data [5], potentially resulting in less accurate or contextually appropriate responses.

In a Conversational KGQA system based on semantic parsing, the user questions are translated into a formal query and executed against a KG to obtain responses. Perez et al. [15] introduced the *SPICE* dataset, which integrates conversational turns with semantic parsing, i.e., it contains translations of user natural language questions into SPARQL queries over the Wikidata KG. Figure 1.1 shows an example extracted from the dataset illustrating the dataset format and how the translation works. To translate from user questions into SPARQL most of the existing approaches proceed in two stages [7, 13, 15]. First, Named Entity Recognition (NER) and Named Entity Linking (NEL) [8] are used to identify entities in each conversation turn. NER is locating and classifying named entities (such as names of people, organizations, locations, etc.) in the question text (e.g., *Ferrara*), while NEL involves mapping these identified entity mentions to their corresponding entity symbols in a KG (e.g., Q13362). In the example in Figure 1.1, the named entity cannot be found in the current turn T_{curret}, because the *Ferrara* entity is been referred to as the 'administrative territory'. The CQA system should utilise the information from T_1 and T_2 to solve the co-reference problem and resolve 'administrative territory' to *Ferrara* and Q13362 as the KG entity.

Then, from these entity symbols, a KG subgraph is extracted for each question constructed by taking these entities' neighbourhoods in the KG. Note that a KG can be seen as a set of triples of the form (subject, relation, object).[1] For instance, (*Soroca*, *sister-town*, *Ferrara*) is a triple in the KG where *Soroca* is the named entity appearing in the subject position, *Ferrara* is the object entity, and *sister-town* is the relation. Given that a KG such as Wikidata contains thousands of relations and types and millions of entities and relationships, these steps enable the semantic parsing model to work on relevant KG portions. In the second stage, a sequence-to-sequence model will predict SPARQL queries from input user questions using only the elements in the subgraph as target vocabulary (in addition to SPARQL keywords). Thus, taking a subgraph permits improving the accuracy and efficiency of the semantic parsing process.

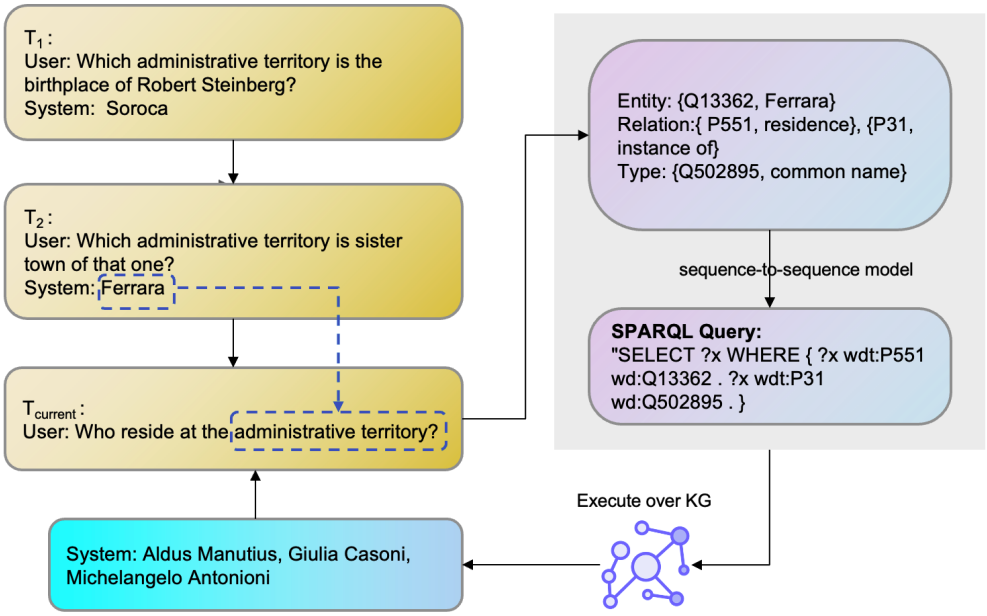

Figure 1.1: An illustration of the conversational question answering application

However, the extracted subgraphs can still be quite large even if we only extracted the linked named entities and their neighbours, sometimes encompassing up to 4,000 triples per user question, which exceeds the input size limits of the BERT model. To address this, Perez et al. [15] addressed this issue by applying a method that randomly linearises and truncates the extracted subgraphs. This truncation approach can lead to the loss of valuable information.

Additionally, their approach uses a BERT-based model to translate natural language

[1] https://en.wikipedia.org/wiki/Resource_Description_Framework

questions and KG triples into executable SPARQL queries. Recent advancements in Large Language Models (LLMs) have shown promise in this area, with Schneider et al. [19] demonstrating the efficiency of fine-tuned open-source models in generating SPARQL queries. However, their approach relies on the set of gold triples as input for SPARQL query generation, which is typically very small (only contains around two triples or less) and not available at test time when users enter new questions. KG subgraphs are often very large with thousands of triples. Even with the application of the retrieval methods to truncate the subgraph, it can still be much larger than the set of gold triples. Consequently, an LLM prompt including the entire subgraph plus the conversation history will be very long, making it difficult to get the appropriate information from the context and slowing down the execution.

Therefore, the goal of this project is to improve the two-stage approach for conversational KGQA. The primary objectives include:

- **Truncating the Excessively Large Subgraph:** The project seeks to enhance the performance and efficiency of the conversational KGQA system by utilising retrieval-based methods to extract the best top-k KG triples.
- **Exploring the Use of LLMs:** The project aims to explore the use of LLMs for semantic parsing. We compare the performance of the LLM-based parser against the BERT-based model in [15] over the extracted subgraph. Then we compare the LLM-based paser with the entire subgraph versus the truncated subgraphs.

Given these two objectives, this project has the following contributions:

- Implement and test a full pipeline for translating natural language questions to SPARQL queries with LLMs, assessing the effectiveness of the LLM in generating SPARQL queries.
- Implement and evaluate a retrieval method to truncate extracted KG subgraphs. Many experiments are done around it (accumulate the subgraph, fine-tune the embedding).
- Evaluation of LLM in SPARQL generation, exploring the effect of fine-tuning and different input subgraph sizes.

The rest of the thesis is structured as follows: Chapter 2 introduces the related work on datasets, approaches, and challenges in KGQA. Chapter 3 presents the two-stage methodology, consisting of subgraph truncation using a retrieval model and SPARQL

query generation using an LLM. Chapter 4 describes the experiments, including setup, metrics, data preparation, and results for both the retrieval stage and LLM-based query generation, comparing the proposed approach with state-of-the-art methods. Chapter 5 concludes the thesis by summarizing the main findings, contributions, and future work directions, such as improving retrieval methods, developing dynamic subgraph truncation techniques, enhancing LLM performance, adapting fine-tuning for retrieval, and incorporating answer-sensitive knowledge generation.

Chapter 2

Related Work

2.1 Datasets for KGQA

For the conversational KG-QA task, specialized factoid QA datasets are essential [4]. These datasets, tailored for responses based on factual data within KGs, address contextual complexities inherent in such tasks. Among these, the *CSQA* dataset [17] stands out due to its integration of complex queries necessitating both quantitative and logical reasoning.

SPICE [15] enhances the *CSQA* dialogues [17] by incorporating executable SPARQL queries over the Wikidata KG. *SPICE* includes 197,000 multi-turn conversational instances between users and an assistant agent, each linked to SPARQL queries that yield answers from a static Wikidata snapshot. This dataset ensures comprehensive semantic and answer-based evaluation of QA performance across dialogues. It also tackles diverse complex question types, including logical, quantitative, comparative, and basic reasoning, thereby providing a robust platform for developing and testing knowledge-driven conversational QA agents. Thus the *SPICE* dataset is adopted as the main dataset to run experiments in this project.

To better understand the performance of the conversational KGQA system, it is essential to consider the different types of questions present in the *SPICE* dataset. The SPARQL queries in *SPICE* can be broadly categorized into two main types: simple questions and reasoning questions [15]. Simple questions are direct factoid questions that seek information about one or more entities. For example, ”Which party is Michel Phlipponneau affiliated with?” is a simple question. The simple questions have three types: simple questions, simple questions (coreferenced), and simple questions (ellipsis).

On the other hand, reasoning questions are more complex, demanding the use

Reasoning	**Type**	**Containing**	**Example**
Logical	Union	Single Relation	Which rivers flow through India or China?
	Intersection		Which rivers flow through India and China?
	Difference		Which rivers flow through India but not China?
	Any of the above	Multiple Relations	Which river flows through India but does not originate in Himalayas?
Verification	Boolean	Single/Multiple entities	Does Ganga flow through India ?
Quantitative	Count	Single entity type	How many rivers flow through India ?
		Mult. entity type	How many rivers and lakes does India have ?
		Logical operators	How many rivers flow through India and/or/but not China?
	Min/Max	Single entity type	Which river flows through maximum number of countries ?
		Mult. entity type	Which country has maximum number of rivers and lakes combined ?
	Atleast/Atmost /Approx./ Equal	Single entity type	Which rivers flow through at least N countries ?
		Mult. entity type	Which country has at least N rivers and lakes combined ?
	Count over Atleast /Atmost / Approx./Equal	Single entity type	How many rivers flow through at least N countries?
		Mult. entity type	How many countries have at least N rivers and lakes combined ?
Comparative	More/Less	Single entity type	Which countries have more number of rivers than India ?
		Mult. entity type	Which countries have more rivers and lakes than India ?
	Count over More/Less	Single entity type	How many countries have more number of rivers than India ?
		Mult. entity type	How many countries have more rivers and lakes than India ?

Figure 2.1: Types of questions in both *CSQA*[17] and *SPICE*[15], from [17]

of numerical and logical operators. Figure 2.1, adapted from the *CSQA* paper [17], illustrates the various types of reasoning questions in the *SPICE* dataset, along with example questions for each type. These question types include comparative reasoning, logical reasoning, quantitative reasoning, and verification questions. In particular, we split quantitative reasoning into two sub-types: quantitative reasoning (all) and quantitative reasoning (count), following the scheme of *SPICE*. This distinction is made because the count type has different SPARQL query results than the other forms of quantitative reasoning. The count type only involves counting, while the quantitative reasoning (all) type encompasses all other forms of quantitative reasoning except counting. The same distinction is made for comparative reasoning.

Furthermore, to investigate the problem of incomplete information, we further split each question type into direct and indirect categories. Direct questions explicitly express all the information necessary to answer the question within the current turn, while indirect questions are incomplete, potentially involving phenomena such as co-reference, ellipsis, or other implicit information. To answer indirect questions, the

information from the current turn alone is insufficient; we need to refer back to the conversational history. In particular, we look at simple questions split into three types: simple question (direct), which only includes direct questions, and simple question (co-reference) and simple question (ellipsis), which only includes indirect questions. By examining the system's performance across these different question types, we can gain valuable insights into its strengths and weaknesses in handling various forms of reasoning.

2.2 Approaches to KGQA

2.2.1 Retrieval Based Approaches

The retrieval-based method, such as [17], relies heavily on embeddings and similarity measures to find answers directly from the knowledge graph (KG) in response to user questions. The approach involves encoding both the questions and the KG entities into a shared vector space where the similarity between them can be computed. The retrieval process essentially matches the encoded query to the most similar KG entities, aiming to find the correct answers.

This method works effectively for straightforward queries where the answer can be directly retrieved based on similarity. However, it encounters significant challenges when dealing with more complex queries that require operations like aggregation (e.g., counting, and finding maximum values). Since the approach is designed around retrieval, it lacks the inherent capability to perform such operations, requiring additional ad-hoc programming to handle these cases. This limitation underscores the need for more sophisticated methods capable of executing these operations directly, rather than relying solely on similarity-based retrieval.

Thus, while effective for direct retrieval tasks, this method falls short in scenarios requiring aggregation, reflecting the need for more advanced techniques capable of performing such operations natively within the KGQA framework.

2.2.2 Semantic Parsing Approaches

Semantic parsing methods for KG question answering aim to convert natural language queries into executable logical forms, such as SPARQL queries, which can be used to retrieve answers from KGs [7, 13, 15, 16].

The process typically begins with a question understanding module that performs linguistic analysis to determine the semantics and syntax necessary for parsing the input question, like recognising named entities. This module may also incorporate conversational context to enhance the representation of the question, thereby improving the accuracy of the subsequent parsing. Once the linguistic analysis is complete, the information gathered is used to extract from the KG. This step involves retrieving a subgraph from the full KG, which contains relevant information centred around the entities mentioned in the query. However, a significant challenge here is that these subgraphs can be quite large due to the numerous neighbouring nodes connected to any given entity. The next stage is handled by a logical parsing module, which converts elements in the extracted subgraph and the corresponding semantic information into an executable logical form, such as SPARQL. The logical form is executed on the KB to accurately obtain the final answers.

The effectiveness of the final SPARQL query heavily depends on the quality of the subgraph. If the target elements are not included in the subgraph, the model cannot generate a correct SPARQL query, leading to incorrect or incomplete answers. This challenge is particularly pronounced in conversational question answering, where questions in later turns often require information from previous turns. Incorporating relevant information from earlier turns into the current subgraph is a complex task that can significantly impact the system's performance.

Conversely, if the subgraph contains too much irrelevant information, it can overwhelm the model, resulting in poorer performance. This also challenges the input limitations of the SPARQL generation model, where large and complex inputs may exceed processing capacities and reduce the accuracy of the final query.

2.2.3 LLM-based Approaches

Recent work by Schneider et al. [19] explores the use of large language models (LLMs) to generate SPARQL queries in the context of semantic parsing. Their approach involves feeding the LLM a natural language question along with a gold subgraph (normally only contains two or fewer triples), which is an optimally small and targeted subset of the KG containing only the necessary elements to answer the query. Their study demonstrates that LLMs can effectively generate accurate SPARQL queries and that performance can be further improved through few-shot prompting and fine-tuning techniques.

However, this method's reliance on a gold subgraph poses challenges in real-world

scenarios, where such precise subgraphs are not available. In practice, the subgraph derived from natural language inputs is often larger and can introduce noise and complicate the LLM parsing task. This raises concerns about the model's robustness when dealing with more complex, unfiltered subgraphs, showing the need for further research to evaluate LLM performance in these more challenging, realistic conditions.

To address the limitations of purely retrieval-based methods in handling questions that require aggregation and logical operations, this thesis adopts a semantic parsing approach as the baseline. However, previous work in semantic parsing has faced challenges related to large and noisy subgraphs extracted from knowledge graphs, which can hinder the performance of the parsing models. Inspired by the success of retrieval methods in identifying relevant information, we propose integrating retrieval techniques to truncate and refine the subgraph, ensuring its quality and reducing its size to a manageable level. In the context of conversational QA, where preserving information from the conversational history is crucial, we also investigate the impact of accumulating the subgraph across each turn. Furthermore, we explore the effectiveness of LLMs in generating SPARQL queries based on these truncated subgraphs, leveraging their advanced language understanding and generation capabilities.

Chapter 3

Methodology

This chapter presents a comprehensive description of our proposed approach for conversational semantic parsing over KGs. We begin with a formal problem definition, followed by a detailed exposition of our system architecture and its constituent modules.

3.1 Formal Problem Definition

We address the semantic parsing task within a multi-turn question-answering dialogue system. Our objective is to accurately interpret user questions and generate appropriate formal queries for execution against a knowledge graph. Let $\mathcal{D} = (d_1, d_2, \ldots, d_{|\mathcal{D}|})$ represent a sequence of dialogue turns, where each turn d_t represents a user-system interaction. Each interaction consists of a user question x_t and the corresponding system answer a_t. The conversation context c_t for the current turn t is defined as the set of all previous interactions $d_i : i < t$. Given the previous interaction d_t, its associated context c_t, and the current user question x_t, our primary objective is to generate a SPARQL query y_t, which should accurately represent the semantic intent of x_t and, when executed against the knowledge graph $\mathcal{K}$, produce the appropriate system response.

3.2 Semantic Parser Architecture

The semantic parsing approach [10, 15] that we will improve employs a two-stage architecture as illustrated in Figure 3.1.

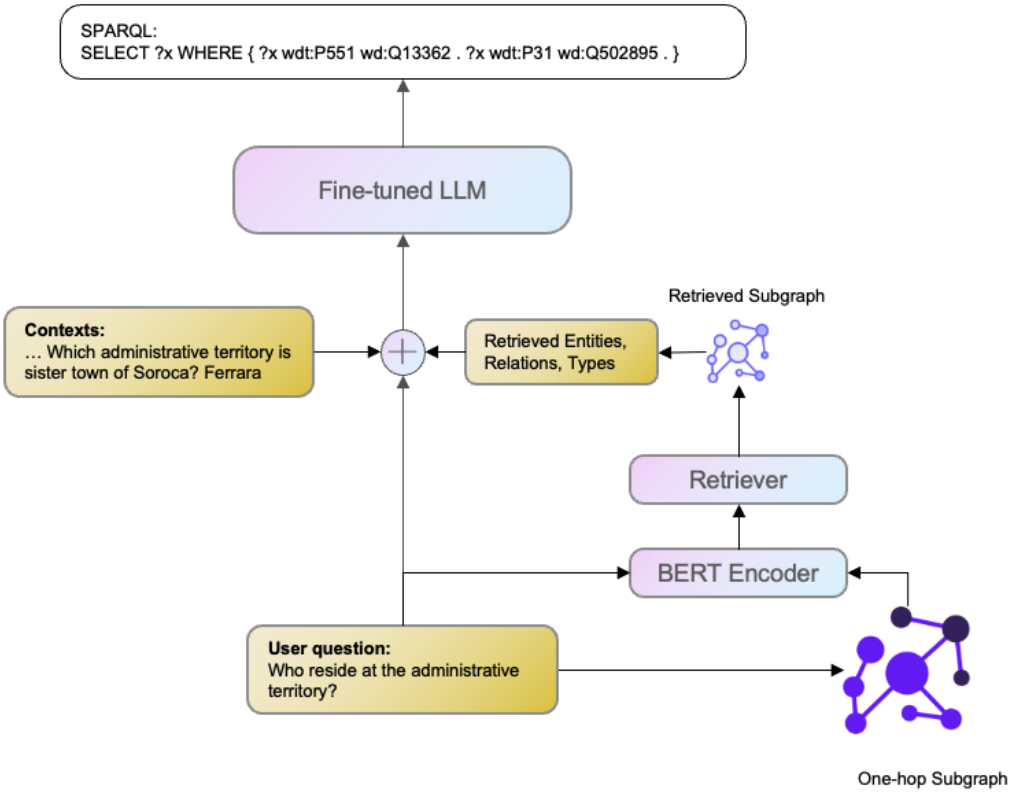

Figure 3.1: Conversational KGQA system architecture

Stage 1: subgraph Extraction, Truncation

From the current user question, the system first extracts the subgraph from the complete KG. Let $\mathcal{K} = (\mathcal{V}, \mathcal{E})$ denote the complete KG, where $\mathcal{V}$ represents the set of vertices and $\mathcal{E}$ the set of edges. Thus, a KG encompasses a set of triples (v, e, v') with $v, v' \in \mathcal{V}$ and $e \in \mathcal{E}$. The initial stage of the system performs the following operations:

1. **Subgraph Extraction[10, 15]**: A subgraph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ is extracted from $\mathcal{K}$ by extracting the one-hop neighborhood for each named entity (determined via NER/NEL) in the current user question x_t. Formally,

$$\mathcal{G}_s = (v, e, v') \in \mathcal{G} \mid v, v' \in \mathcal{V}_r \tag{3.1}$$

   where $\mathcal{V}_r \subset \mathcal{V}$ is the set of named entities and one-hop neighbor types.

2. **subgraph Truncation (contribution in this work)**: We propose a retrieval (i.e., ranking) method $\mathcal{R}$ to truncate $\mathcal{G}_s$ to k most relevant triples. Let $\mathcal{G}_t$ denote the truncated subgraph:

$$\mathcal{G}_t = \mathcal{R}(\mathcal{G}_s, k) \tag{3.2}$$

Stage 2: SPARQL Query Generation

The second stage utilizes a LLM Φ to generate SPARQL queries. As illustrated in Figure 3.1, given the conversation history c_t, current user question x_t, and truncated subgraph $\mathcal{G}_t$, the LLM generates a SPARQL query y_t:

$$y_t = \Phi(c_t, x_t, \mathcal{G}_t) \tag{3.3}$$

3.3 Retrieval Model

To address the challenges posed by large subgraphs, we propose a retrieval method that truncates the subgraph based on semantic similarity with the natural language question. We use a sentence encoder model E to encode both the user question x_t and triples t_i in the KG subgraph $\mathcal{G}_s$. The similarity between the question and each triple in $\mathcal{G}_s$ is computed as:

$$\text{sim}(x_t, t_i) = \cos(E(x_t), E(t_i)) \tag{3.4}$$

where cos denotes the cosine similarity function, $E(x_t)$ is the embedding of the user question, and $E(t_i)$ is the embedding of a KG triple in $\mathcal{G}_s$. To obtain triple embeddings, we convert KG triples t_i into a string of the form '*subject, relation, object*' before passing them through the encoder.

Currently, truncation is applied to the subgraph $\mathcal{G}_s$, which is smaller than the complete KG $\mathcal{K}$ and may be influenced by preceding NER and NEL steps (if these steps are inaccurate, then the extracted subgraph will also be inaccurate). Future work will involve applying retrieval directly to the entire KG, and one method [1] is discussed in detail in Section 5.

We utilize DistilBERT [18] as our sentence encoder due to its efficiency and effective balance between performance and computational cost. DistilBERT is a streamlined version of the BERT model, created through knowledge distillation. This process reduces the model size by approximately 40% and accelerates runtime by about 60%, while preserving over 95% of BERT's performance. This reduction in model size and computational demands makes DistilBERT particularly suitable for environments with limited resources. In conjunction with DistilBERT, we employ the FAISS [6] (Facebook AI Similarity Search) library to perform similarity searches using cosine similarity. FAISS employs efficient indexing to organise and quickly access vectors, reducing the search space. By utilizing FAISS, we benefit from its scalability and speed in retrieving relevant sentence embeddings, enhancing the overall efficiency of our system.

3.4 Large Language Model for Query Generation

The truncated subgraph $\mathcal{G}_t$ obtained from the retrieval stage is used in conjunction with an LLM to generate SPARQL queries. We formulate the query generation task as follows. Given the conversation history c_t, current question x_t, and truncated subgraph $\mathcal{G}_t$, we construct a prompt P:

$$P = f_{\text{prompt}}(c_t, x_t, \mathcal{G}_t) \tag{3.5}$$

where f_{prompt} is a function that formats the inputs into a suitable prompt for the LLM. The SPARQL query y_t is then generated by the LLM Φ:

$$y_t = \Phi(P) \tag{3.6}$$

We employ a fine-tuned version of an LLM for this task, optimizing it for SPARQL query generation in the context of conversational question answering over knowledge graphs.

The model selected for generating SPARQL queries from the truncated subgraph $\mathcal{G}_t$ is the LLaMA 3-8b-instruct model [1]. This choice is driven by several considerations. Firstly, the task involves fine-tuning, making it essential to select an open-source model that can be run locally to ensure flexibility and control over the training process. Secondly, due to the high computational cost associated with running large models, a smaller, more efficient version is preferred. The instruct variant of the LLaMA 3-8b model is particularly well-suited for this task because it is fine-tuned to follow complex instructions and interpret detailed prompts with high precision. Its enhanced capability to understand and execute nuanced instructions enables it to effectively transform the formatted prompt P into the desired SPARQL query y_t.

By integrating our retrieval-based subgraph truncation with the fine-tuned LLM, we establish a comprehensive pipeline for conversational semantic parsing, addressing the challenges in QA systems over large-scale KGs.

[1] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

Chapter 4

Experiments and Performance Evaluation

4.1 Experiment Settings

All experiments were conducted on an NVIDIA A10G GPU, providing robust hardware acceleration for deep learning tasks. This GPU has 24 GB of memory, which significantly enhances the performance of model training and inference, especially for computationally intensive operations like generating SPARQL queries through LLM.

4.2 Evaluation Metrics

Our project methodology encompasses two distinct stages, necessitating different evaluation metrics. For the initial retrieval stage, we employ two primary metrics: **Hit Rate** and **Set Recall**. Hit Rate quantifies the model's ability to precisely retrieve the gold KG triples within the top-k retrieved results. Specifically, a retrieval is considered a "hit" if all the gold triples are included in the top k triples.

To assess the model's performance in identifying relevant knowledge graph elements, we utilize set recall. This metric is calculated as follows:

$$\text{Set Recall} = \frac{|\text{Set(Topk Triples)} \cap \text{Set(Gold Triples)}|}{|\text{Set(Gold Triples)}|} \tag{4.1}$$

where triples are decomposed into their constituent elements. This metric provides insights into the model's capacity to retrieve pertinent KG elements, even if the exact triple structure is not preserved.

For the second stage, which involves the actual generation of SPARQL queries, we employ the **Exact Match** (EM) metric. This evaluation criterion assesses the model's ability to produce SPARQL queries that precisely match the gold queries.[1]

4.3 Data Preparation

The *SPICE* dataset [15] forms the foundation of our experimental work. This dataset encompasses a diverse range of conversational interactions, each consisting of a sequence of user questions $[x_1, x_2, ..., x_t]$, contextual information, corresponding SPARQL queries, and system responses.

4.3.1 Data Pre-processing

For the KG subgraph extraction we use the scripts provided by [15] (steps 1-3 are taken from [15]). For our proposed retrieval step, our required pre-processing aims to transform the raw subgraph from step (3) into a format appropriate for triple retrieval (steps 4-5). The pipeline comprises several key steps:

1. **NER:** A NER technique identifies and extracts named entities from user questions (e.g., *Ferrara*). We use the String Match method from [15].

2. **NEL:** The identified entities are then linked to KG entity identifies (IDs), e.g., Q13362.

3. **Subgraph Extraction:** A subgraph by extracting one-hop neighbors of the linked entities within the KG.

4. **Entity Labeling:** A local SQLite database is utilized to contain KG entity IDs with their corresponding natural language descriptions (i.e., labels).

5. **Subgraph Formatting for Retrieval:** The subgraph extraction process yields three distinct subgraph types: *local_subgraph, local_subgraph_nel, and type_subgraph*. To enhance the completeness of our knowledge representation, we merge the *local_subgraph* and *local_subgraph_nel* into a comprehensive subgraph structure. The resulting subgraph is structured as follows:

[1] Note that there exists the execution-based F1 metric for this stage (i.e., execute the predicted SPARQL query and compare the obtained results with the gold answers). But this metric requires the installation of a SPARQL server and execution of the queries so we leave this metric for future work.

```
{
  "subgraph": {
    "entity id":{
        "subject":{
            "relation id":{"type id"}
        }
        "object":{
            "relation id": {"type id"}
        }
    }
  }
}
```

The extracted subgraph in our study adopts a triple structure of the form (entity, relation, type), different from the conventional (entity, relation, entity) representation. For instance, instead of a triple like (Paris, is-capital, France), the subgraph extraction produces (Paris, is-capital, Country). This design choice aligns more closely with the required structure of SPARQL queries, and it also potentially facilitates efficient query processing by including the type (Country) instead of enumerating all the concrete entities from the KG that are countries.

For the retrieval step proposed in this work, we concatenate labels for entities, relations, and types in a triple. For subject-type triples, we format triples as the following string *'type, relation, entity'*, while object-type triples are formatted as *'entity, relation, type'*. To address the specific requirements of Verification and Quantitative Reasoning question types, we format the corresponding subgraphs as the string *'entity, relation, entity'*, as these question categories uniquely involve (entity, relation, entity) triples.

The *type_subgraph*, which exclusively contains type and relation information, is formatted as the string *'type, relation'* and *'relation, type'*.

Throughout the subgraph formatting process, we prioritize data cleaning and normalization. Entity, type, and relation IDs are systematically converted to natural language labels using our SQLite database (step 4 above).

4.3.2 Dataset Analysis

The *SPICE* dataset is big with many dialogues and interactions. Thus, to run experiments faster, we created a stratified sample of 2,000 dialogues from the *SPICE* dataset. This subset maintains the distribution of question types present in the full dataset, ensuring a representative sample for our analyses. Recognizing the potential for incomplete information in individual conversation turns due to phenomena such as co-reference and ellipsis, we investigated two approaches to subgraph construction:

1. **Current Turn Subgraph:** This approach considers only the triples related to the current conversational turn.

2. **Accumulated Subgraph:** This method aggregates triples across all previous turns in the conversation, providing a more comprehensive context.

Note that, these two subgraphs are extracted via NER/NEL entities. The subgraphs constructed using these methods appear to be relatively small in terms of the number of triples. This can be attributed to the accuracy limitations of the NER/NEL system employed. However, the size and information richness of these subgraphs can be expanded by considering not just a single ID for each named entity during the NEL step, but instead taking the top-n IDs or even all IDs associated with each entity.

To assess the efficacy of subgraph truncation, we conduct an analysis to determine an upper bound, i.e., before truncation, whether the gold triples corresponding to each user question x_t are present within the extracted subgraphs $\mathcal{G}_s$ to begin with. We define a "Hit@all" as the presence of the gold triples within the subgraph $\mathcal{G}_s$. Table 4.1 presents a detailed comparison of these two subgraph construction methods across various question types and categories. The columns in Table 4.1 include Hit@all, Set Recall, Average(Avg) subgraph size, and Standard Deviation (Std) of the subgraph size for current and accumulated triples approaches.

The comparative analysis of current turn subgraphs and accumulated subgraphs reveals several notable trends. Accumulated subgraphs consistently exhibit higher Set Recall values across nearly all question types and categories, indicating that accumulating triples over multiple turns enables the subgraph to capture a more comprehensive set of relevant KG elements to carry out SPARQL query generation. This improvement is particularly evident for indirect question types, such as Simple Question (Coreference) and Logical reasoning (indirect). Moreover, for reasoning types such as Logical Reasoning, Quantitative Reasoning, and Comparative Reasoning, the accumulated subgraphs

Question Type	Category	Current Triples				Accumulated Triples				# of Q
		Hit@all	Set Recall	Avg	Std	Hit@all	Set Recall	Avg	Std	
Simple Question (Direct)	Direct	0.5338	0.5920	70	180	0.5439	0.6595	243	497	5838
Simple Question (Coreference)	Indirect	0.0396	0.0348	7	40	0.0632	0.3250	183	424	3482
Simple Question (Ellipsis)	Indirect	0.4986	0.5511	6	21	0.5188	0.6467	246	467	735
Logical Reasoning	Direct	0.4211	0.6603	80	157	0.4333	0.7280	371	547	824
	Indirect	0.0000	0.3873	3	4	0.3846	0.6275	132	351	52
Verification (Boolean)	Direct	0.8275	0.9779	152	421	0.8275	0.9850	433	787	1519
Quantitative Reasoning (All)	Direct	0.0000	0.0000	1	13	0.0469	0.3475	265	475	1216
Quantitative Reasoning (Count)	Direct	0.1965	0.2722	87	220	0.2185	0.4887	381	602	1730
	Indirect	0.3472	0.3746	7	44	0.3673	0.5787	368	609	648
Comparative Reasoning (All)	Direct	0.1450	0.2850	113	208	0.2257	0.5229	427	662	731
	Indirect	0.1116	0.2553	18	104	0.2744	0.5462	412	645	475
Comparative Reasoning (Count)	Direct	0.1325	0.2875	111	204	0.2265	0.5142	416	556	702
	Indirect	0.0688	0.2553	18	104	0.2538	0.5313	412	566	465

Table 4.1: Comparison of current turn subgraph and accumulated subgraph characteristics

show substantial improvements in both Hit@all and Set Recall metrics. This suggests that accumulating information is particularly beneficial for questions requiring multi-step reasoning or comparison. The superior performance of accumulated subgraphs can be attributed to their ability to retain relevant information from previous turns, providing a more comprehensive context for parsing the current question. The implications of these two subgraph construction methods on retrieval performance are further explored in Section 4.4.

However, it is important to note that the average number of triples in the accumulated subgraphs is consistently higher than in the current turn subgraphs, often by a factor of 3-4 times or more. While this increased context richness can be beneficial for question answering, it comes at the cost of increased computational complexity. This trade-off between information comprehensiveness and efficiency warrants careful consideration in the design and implementation of KGQA systems.

4.4 Results of retrieval

This section presents and analyzes the results of our retrieval model. We evaluate the model's performance across various question types and categories, using metrics introduced at Section4.2 for different subgraph construction approaches.

Experimental Setup We conducted experiments using a subset of the *SPICE* dataset. The retrieval model, based on DistilBERT, was tasked with truncating subgraphs to retrieve the most relevant triples for each user question. We use **Hit@k** to denote the proportion of queries where the correct triple is retrieved within the top k results, and **Recall@k** to denote the proportion of relevant KB symbols retrieved within the top k results, and we also count the number of the questions for each question type as **# of Q**

Results Analysis We examined the top 5, 20, and 100 triples for the experiment. The result is in Table 4.2.

Question Type	Category	Current Triples				Accumulated Triples				# of Q
		Hit@5	Hit@20	Hit@100	Hit@All	Hit@5	Hit@20	Hit@100	Hit@All	
Simple Question (Direct)	Direct	0.4629	0.5094	0.5257	*0.5338*	0.4468	0.5086	**0.5259**	***0.5439***	5838
Simple Question (Coreference)	Indirect	0.0213	0.0362	0.0385	*0.0396*	0.0260	0.0366	**0.0488**	***0.0632***	3482
Simple Question (Ellipsis)	Indirect	0.4220	0.4856	0.4960	*0.4986*	0.3865	0.4831	**0.4997**	***0.5188***	735
Logical Reasoning	Direct	0.2694	0.3956	**0.4138**	*0.4211*	0.2184	0.3811	0.4126	***0.4333***	824
	Indirect	0.0000	0.0000	0.0000	*0.0000*	0.0385	0.2692	**0.3462**	***0.3846***	52
Verification (Boolean)	Direct	0.0566	**0.0612**	**0.0612**	***0.8275***	0.0566	0.0599	**0.0612**	***0.8275***	1519
Quantitative Reasoning (All)	Direct	0.0000	0.0000	0.0000	*0.0000*	0.0049	0.0115	**0.0288**	***0.0469***	1216
Quantitative Reasoning (Count)	Direct	0.1208	0.1642	0.1821	*0.1965*	0.1127	0.1613	**0.1884**	***0.2185***	1730
	Indirect	0.2577	0.3272	**0.3395**	*0.3472*	0.2130	0.3102	0.3364	***0.3673***	648
Comparative Reasoning (All)	Direct	0.0219	0.0971	0.1245	*0.1450*	0.0260	0.1053	**0.1696**	***0.2257***	731
	Indirect	0.0379	0.0779	0.0842	*0.1116*	0.0042	0.0442	**0.1221**	***0.2947***	475
Comparative Reasoning (Count)	Direct	0.0185	0.0755	0.1068	*0.1325*	0.0214	0.0855	**0.1538**	***0.2265***	702
	Indirect	0.0194	0.0495	0.0559	0.0688	0.0086	0.0430	**0.1011**	***0.2538***	465

a) hit rate comparison between current triples and accumulated triples

Question Type	Category	Current Triples				Accumulated Triples				# of Q
		Rec@5	Rec@20	Rec@100	Rec@All	Rec@5	Rec@20	Rec@100	Rec@All	
Simple Question (Direct)	Direct	0.5541	0.5804	0.5882	*0.5918*	0.5586	0.6123	**0.6383**	***0.6598***	5838
Simple Question (Coreference)	Indirect	0.0267	0.0342	0.0354	*0.0361*	0.1878	0.2512	**0.3037**	***0.3350***	3482
Simple Question (Ellipsis)	Indirect	0.4917	0.5366	0.5435	*0.5439*	0.5080	0.6119	**0.6507**	***0.6751***	735
Logical Reasoning	Direct	0.6049	0.6504	0.6578	*0.6603*	0.5879	0.6729	**0.6967**	***0.7332***	824
	Indirect	0.3333	0.3873	0.3873	*0.3873*	0.4167	0.5588	**0.5980**	***0.6275***	52
Verification (Boolean)	Direct	0.3364	0.3568	0.3601	*0.9779*	0.3477	0.3952	**0.4207**	***0.9850***	1519
Quantitative Reasoning (All)	Direct	0.0007	0.0010	0.0010	*0.0010*	0.1773	0.2527	**0.3087**	***0.3475***	1216
Quantitative Reasoning (Count)	Direct	0.2318	0.2561	0.2663	*0.2722*	0.3094	0.3862	**0.4372**	***0.4887***	1730
	Indirect	0.3260	0.3648	0.3712	*0.3746*	0.3584	0.4798	**0.5393**	***0.5787***	648
Comparative Reasoning (All)	Direct	0.2152	0.2590	0.2763	*0.2875*	0.2974	0.4031	**0.4725**	***0.5229***	731
	Indirect	0.1706	0.2209	0.2495	*0.2553*	0.2330	0.3762	**0.4965**	***0.5646***	475
Comparative Reasoning (Count)	Direct	0.2094	0.2541	0.2738	*0.2838*	0.2957	0.3907	**0.4647**	***0.5142***	702
	Indirect	0.1409	0.1883	0.2008	*0.2047*	0.2146	0.3522	**0.4595**	***0.5313***	465

b) set recall comparison between current triples and accumulated triples, Rec represent Set Recall

Table 4.2: Comparison between current triples subgraph and accumulated triples subgraph. The Hit@All and Recall@all are italicized, and the best value in the All are bold; the best value among the experiments are also bold

Effectiveness of the Retrieval Method The retrieval method exhibits strong performance in terms of both hit rate and set recall for simple questions. In the ”Simple Question (Direct)” category, the current triples subgraph achieves a Hit@5 of 0.4629 and a Hit@20 of 0.5094, indicating that the correct answer is retrieved within the top 5 and 20 triples for a substantial portion of the questions. Moreover, the Set Recall@5 and Set Recall@20 values for this category are 0.5541 and 0.5804, respectively, suggesting that the necessary elements for answering the question are often found within the top retrieved triples. Similarly, the ”Simple Question (Ellipsis)” category shows impressive results, with Hit@5 and Hit@20 values of 0.4220 and 0.4856, and Set Recall@5 and Set Recall@20 values of 0.4917 and 0.5366, respectively. These results are close to their upper bounds (Hit@All, Set Recall@All), demonstrating the effectiveness of the retrieval method in capturing both the correct answer and the necessary elements for simple questions.

For more complex reasoning questions, such as those in the ”Logical Reasoning” and ”Comparative Reasoning” categories, the hit rate performance is lower at top 5 and top 20 but improves at top 100. This suggests that the gold answer may not always receive a high ranking in the retrieval process. However, the set recall values remain high for these questions, often approaching the upper bound (Set Recall@All) within the top 20 retrieved triples. For instance, in the ”Comparative Reasoning (Count)” category for indirect questions, the Set Recall@20 is 0.1883, close to the Set Recall@All of 0.2047. This indicates that while the exact correct triple may not always be retrieved in the truncated subgraph, the necessary elements for answering the question are still present within the subgraph.

Based on these findings, we can conclude that the retrieval method is effective for the subgraph truncation task. However, to further enhance the search and ensure that the gold triples are ranked at the top, improvements in the embedding or search method may be necessary. To address this, we conducted experiments on fine-tuning the embedding with the aim of improving performance. However, the results did not surpass those obtained using the off-the-shelf embedding model. As a result, we have included the fine-tuning experiments in the Appendix for reference.

Effectiveness and Limitations of the Accumulated Subgraph The accumulated subgraph approach demonstrates significant improvements in performance for many indirect questions. In the ”Simple Question (Coreference)” category, the Set Recall@100 increases from 0.0354 for the current triples subgraph to 0.3037 for the accumulated

subgraph. The ”Quantitative Reasoning (Count)” category for indirect questions also shows a substantial improvement, with the Set Recall@100 increasing from 0.3712 to 0.5393. These results suggest that the accumulated subgraph, when combined with retrieval, can effectively capture more information from previous turns and mitigate the issue of incomplete (implicit) information as the conversation progresses.

However, it is noteworthy that for some question types, the hit rate for the accumulated subgraph is lower compared to the current triples subgraph. This can be attributed to the increased size of the accumulated subgraph, which makes it more challenging to accurately search for specific information. Nevertheless, the set recall values for the accumulated triples method consistently surpass those of the current triples method across all question types and categories, indicating that even if the exact correct triple is not retrieved, the accumulated subgraph still contains the necessary KB elements to formulate the SPARQL query.

The comparison between the two types of subgraphs reflects the advantages of the accumulated approach. As the conversation progresses, the accumulated subgraph can provide a more comprehensive knowledge base for answering questions by incorporating information from previous turns. However, it is important to note that as the triples accumulate over turns, the subgraph can become very large towards the end of the conversation. This poses challenges in terms of efficiently searching for relevant information within the expanded subgraph. To address these challenges, we could apply our truncation method for dynamic truncation that can maintain a manageable size of the subgraph throughout the conversation.

Evaluating Retrieval Performance with Gold Entities Although the retrieval method has proven to be effective, it is important to acknowledge that the subgraphs extracted using NER and NEL techniques often lack a significant number of gold triples, as shown in Table 4.1. This limitation can hinder the accurate evaluation of the retrieval method’s performance.

To assess the retrieval method’s effectiveness in isolation, we decided to conduct an evaluation using a KG subgraph $\mathcal{G}_s$ extracted via gold entities (i.e., use the dataset annotated entities instead of NER/NEL). By using gold entities, we ensure that the subgraph contains all the necessary entities referenced in the conversation, including those from previous turns (and their corresponding neighboods). This approach allows us to evaluate the retrieval performance under the best possible conditions, eliminating the impact of missing entities on the retrieval process.

However, it is important to note that using gold entities is not realistic in real-world scenarios or at test time. In practical applications, we do not have access to gold entities, as they are not readily available during the conversation. Instead, the system must rely on the entities extracted using NER and NEL techniques or other techniques. One direction of the future work is to directly extract triples from KG without the NER and NEL methods [1], which is dicussed in Section 5.

Table 4.3 presents the results of applying the retrieval methods to the subgraphs visa gold entities. In this table, we use percentage values to represent the recall, distinguishing it from the hit rates

Question Type	Category	Hit@5	Rec@5 (%)	Hit@20	Rec@20 (%)	Hit@100	Rec@100 (%)	Hit@All	Rec@All (%)
Simple Question (Direct)	Direct	0.7575	90.38	0.8974	96.87	**0.9692**	**99.15**	0.9981	100.00
Simple Question (Coreference)	Indirect	0.1176	42.30	0.2680	67.98	**0.5798**	**91.22**	0.8548	100.00
Simple Question (Ellipsis)	Indirect	0.6303	79.38	0.8385	93.04	**0.9435**	**98.02**	1.0000	100.00
Logical Reasoning	Direct	0.5364	87.94	0.9175	98.60	**0.9964**	**99.95**	1.0000	100.00
	Indirect	0.1154	67.65	0.4231	95.10	**0.8846**	**98.04**	1.0000	100.00
Verification (Boolean)	Direct	0.4240	79.99	0.6162	92.79	**0.7156**	**97.91**	0.8282	99.77
Quantitative Reasoning (All)	Direct	0.0321	45.20	0.1209	65.18	**0.3248**	**81.97**	0.7064	92.29
Quantitative Reasoning (Count)	Direct	0.2988	66.00	0.4821	82.39	**0.6636**	**93.58**	0.8786	98.53
	Indirect	0.4877	71.58	0.7207	88.20	**0.8827**	**97.00**	1.0000	100.00
Comparative Reasoning (All)	Direct	0.0547	55.35	0.2367	78.15	**0.5458**	**92.07**	0.9207	98.39
	Indirect	0.0295	36.79	0.0968	63.02	**0.3811**	**86.25**	0.7558	95.74
Comparative Reasoning (Count)	Direct	0.0527	53.35	0.1980	75.95	**0.5570**	**92.51**	0.9274	98.38
	Indirect	0.0516	38.38	0.1312	61.29	**0.3269**	**78.80**	0.5634	87.16

Table 4.3: Retreival Performance on the Subgraph visa Gold Entities

This results of the retriever in isolation with subgraph via gold entities show that performance is low when user questions have underspecified inforation (ellipsis/coreference) or are long and complex. Also, is important to point out that although the Hit Rate is moderate, the Set Recall looks better and indicates that the KG elements are somehow present.

The results show that the retriever's performance varies depending on the question type and complexity when using subgraphs with gold entities. For simple direct questions, the retriever achieves high hit rates and recall values, with Hit@100 reaching 0.9692 and Rec@100 at 99.15%. However, the performance decreases for questions with underspecified information, such as those involving coreference or ellipsis. For example, in the "Simple Question (Coreference)" category, the Hit@100 is 0.5798, and the Rec@100 is 91.22%, indicating that the retriever struggles to find the exact correct triples for these question types.

Similarly, the retriever's performance is lower for long and complex questions, such as those in the "Comparative Reasoning" and "Quantitative Reasoning" categories. For instance, in the "Comparative Reasoning (All)" category for indirect questions, the

Hit@100 is only 0.3811, while the Rec@100 is 86.25%. This shows that the retriever faces challenges in accurately identifying the relevant triples for these complex question types.

It is important to note that although the hit rates may be moderate in some cases, the recall values are generally higher, indicating that the necessary KG elements are present in the subgraphs. For example, in the ”Logical Reasoning” category for indirect questions, the Hit@100 is 0.8846, but the Rec@100 is 98.04%, suggesting that the subgraph contains the required information to answer the question, even if the exact correct triple is not always ranked at the top k.

We adopted the top 20 truncated subgraphs for further experiments with LLM SPARQL generation. We want to explore LLM performance with the different input subgraph sizes.

4.5 Large Language Model Approach

4.5.1 Model Selection and Configuration

In this study, we explore the application of LLMs to convert natural language questions and KG triples into executable SPARQL queries. Our approach builds upon recent advancements in LLM capabilities, as demonstrated by Schneider et al. [19], who showed the efficiency of fine-tuned open-source models in SPARQL query generation.

Model Architecture We selected the Llama3 architecture as our base model due to its robust performance in various natural language processing tasks, as discussed in Section3.4. The Llama3 model’s architecture, characterized by its extensive parameterization, multi-head self-attention mechanisms, and deep transformer layers, provides a suitable foundation for our task of translating complex semantic structures into structured SPARQL queries. Its ability to capture intricate language patterns and maintain contextual coherence across long sequences [20] makes it particularly effective for handling the requirements of our SPARQL generation task.

Input Strategies To comprehensively evaluate the model’s performance under different input conditions, we explored three distinct subgraph strategies:

- **Gold subgraph**: Given in the *SPICE* dataset, it contains exactly the set of triples needed to build the SPARQL query.

- **Full subgraph (extracted via gold entity annotations)**: It includes the entire extracted subgraph via gold entity annotations, which can be extensive, containing up to 4000 triples for complex questions (as shown in Table 4.3).
- **Truncated subgraph (top-20 triples)**: To test the performance of LLMs with respect to the KG subgraph sizes, we choose to use the subgraph via gold entities, and truncate it to top-20 triples.

Model Variants and Fine-tuning To thoroughly investigate the impact of the different input subgraphs and task-specific training, we developed and evaluated four distinct model variants:

- **Base Llama3**: The unmodified Llama3 model, serves as a baseline to assess the effectiveness of fine-tuning.
- **Llama3-Gold**: Fine-tuned on gold subgraph inputs, optimizing for concise and highly relevant semantic information.
- **Llama3-Full**: Adapted to process full subgraph inputs, potentially capturing more comprehensive semantic context at the cost of increased computational demands.
- **Llama3-Top20**: Specialized for the top-20 triple representation, balancing information richness and computational efficiency.

4.5.2 Prompt Engineering

Prompt Techniques In the realm of LLM-based query generation, two primary prompting techniques are commonly employed: zero-shot and few-shot learning [3]. Zero-shot learning requires the model to perform a task without any task-specific examples, relying solely on its pre-trained knowledge and the given instructions. In contrast, few-shot learning provides the model with a small number of examples to guide its understanding of the task.

While few-shot learning has been shown to improve the performance of base language models [2], it is not as effective when applied to fine-tuned models. Our preliminary experiments revealed that using few-shot prompts with fine-tuned models actually led to a decrease in performance. This observation is consistent with the findings of [14], who noted that fine-tuned models may not benefit from few-shot

examples, as they have already acquired task-specific knowledge during the fine-tuning process.

Furthermore, incorporating few-shot examples in the prompt can substantially increase the token count, especially for complex queries like comparative tasks. This increased input size can potentially cause issues related to context window limitations and higher computational costs [12].

Considering that our models are fine-tuned for specific tasks, using few-shot prompts would be counterproductive. The fine-tuning process has already equipped the models with the necessary task-specific knowledge. Therefore, we chose to employ a zero-shot approach in our experimental setting, prioritizing efficiency and leveraging the specialized knowledge acquired during fine-tuning.

Prompt Structure Our zero-shot prompt was carefully designed to provide clear instructions and context for the SPARQL query generation task. The prompt structure is as follows:

> **System**: Construct a SPARQL query based on the 'Input question:' using the provided 'Entities:', 'Relations:', and 'Types:'. Refer to the 'Conversation history:' to address any issues with incomplete information, such as coreferences or ellipses. Ensure the query is compatible with the Wikidata knowledge graph, utilizing only ids and excluding labels. Prefixes like 'wdt' and 'wd' are already defined, so do not redefine them. Use '?x' as the variable in the query, and do not include any language tags or additional comments. Only return the SPARQL query.
>
> **User**: Conversation history: {conversation history}
> Input question: {question}
> Entities: {entities}
> Relations: {relations}
> Types: {types}

This prompt structure, totalling 156 tokens in length, was designed to optimize the balance between providing comprehensive instructions and minimizing input size.

Input Optimization To enhance the model's performance and mitigate potential issues with input size limitations, we implemented several optimization strategies: 1. **Conversation History**: The inclusion of previous conversation turns provides crucial context for resolving ambiguities, such as co-references or ellipses, that may be present in the current question; 2. **Structured Triple Decomposition**: Rather than inputting full KG subgraphs, we decomposed the structured triples into three distinct categories:

entities, relations, and types. This approach not only reduces input size but also leverages the LLM's inherent ability for logical reasoning and structure comprehension; 3. **Identifier-Label Pairing**: Each item in the entities, relations, and types lists is presented in an {id, label} format. This pairing allows the model to utilize both the unique identifiers required for SPARQL query construction and the human-readable labels that may aid in semantic understanding.

This optimized input structure aims to strike a balance between providing comprehensive information and maintaining a manageable input size, thereby enabling the model to generate accurate SPARQL queries efficiently.

4.5.3 Model Fine-tuning

To optimize our LLM for SPARQL query generation, we conducted a comprehensive fine-tuning process using three distinct datasets derived from the *SPICE* dataset, each comprising 2,964 samples that maintained the original data distribution.

Fine-tuning Configuration We employed the Low-Rank Adaptation (LoRA) technique [9] for efficient fine-tuning of the Llama-3-8b-Instruct base model, a variant of the LLaMA 3 architecture optimized for instruction-following tasks. Key configuration details include:

- Maximum Sequence Length: 5,000 tokens, chosen to accommodate the potentially large input sizes from full subgraphs.
- LoRA Rank (r): 32, the number of trainable parameters that balance adaptability and computational efficiency.
- Learning Rate: 2e-4, with a linear scheduler to promote stable convergence.
- Batch Size: 5 per device with 4 gradient accumulation steps, effectively creating a larger virtual batch size to improve training stability.

These parameters were carefully selected to optimize the model's ability to learn complex tasks while managing computational resources effectively.

Training Process and Analysis We fine-tuned three separate models, each corresponding to a different input representation strategy: gold subgraph, full subgraph, and

top-20 triples. The training process was monitored using both training and validation loss metrics, with evaluations conducted every 20 steps over three epochs.

Figure 4.1 illustrates the training and validation loss curves for each model variant, revealing significant differences in learning dynamics across the three input representation strategies.

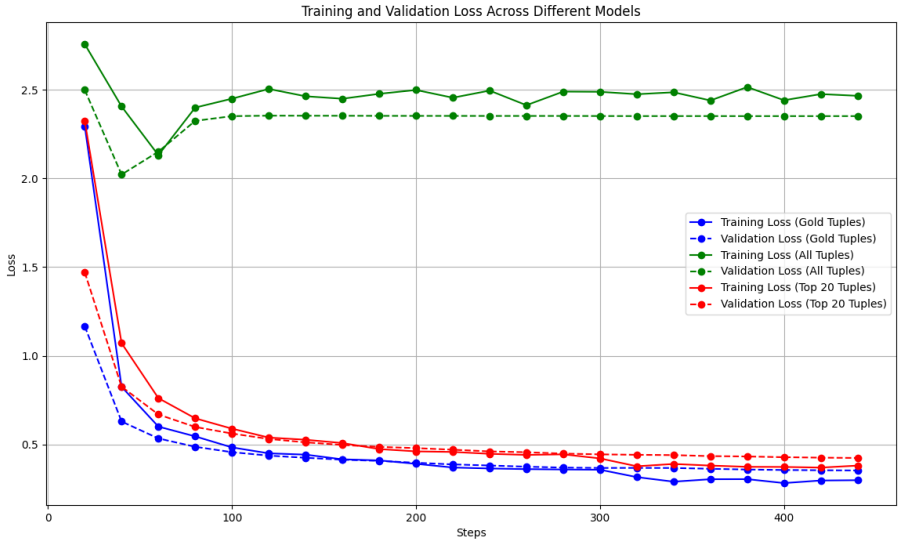

Figure 4.1: Training and Validation Loss Curves for Different Input Representations

The gold subgraph model demonstrated the most rapid convergence and achieved the lowest final loss values (training: 0.298900, validation: 0.353659). This superior performance can be attributed to the concise and highly relevant information provided by gold subgraphs, allowing the model to quickly learn the essential patterns for accurate SPARQL query generation. The steep initial decline in both training and validation loss suggests that the model efficiently captured the core relationships between natural language questions and their corresponding SPARQL representations.

In contrast, the full subgraph model struggled to converge, with both training and validation losses remaining high throughout the process (final training: 2.466000, validation: 2.351645). This poor performance indicates that the sheer volume of information in full subgraphs, which can include up to 4,000 triples, overwhelmed the model's capacity to extract relevant patterns for query generation. The consistently high loss values suggest that the model may have been unable to effectively distinguish between crucial and information within the expansive subgraphs, leading to suboptimal query formulation.

The top-20 triples model presented an intriguing middle ground. While not converging as rapidly as the gold subgraph model, it showed steady improvement throughout training, achieving final loss values (training: 0.381200, validation: 0.424139) that were significantly closer to the gold subgraph model than to the full subgraph model. This performance indicates that the top-20 triples selection strategy effectively balances information richness and model tractability, providing sufficient context for query generation without overwhelming the model's learning capacity.

These divergent outcomes reflect the importance of input sizes in fine-tuning LLMs for specialized tasks like SPARQL query generation. The superior performance of the gold subgraph model highlights the value of concise, relevant input in facilitating efficient learning. Conversely, the full subgraph model's poor convergence demonstrates that an abundance of information can be detrimental, potentially obscuring the essential patterns necessary for effective query generation.

The respectable performance of the top-20 triples model suggests a promising direction for practical applications. It indicates that the retrieval method over large knowledge graphs can yield effective results, potentially offering a scalable approach for interfacing LLMs with extensive knowledge bases without sacrificing query generation accuracy.

4.5.4 LLM Performance Analysis

Table 4.4 presents the Exact Match (EM) accuracy for our LLM experiments across various question types and categories. The table compares the performance of three model variants: M_G (gold subgraphs), M_A (full subgraphs), and M_{20} (top 20 triples) and their fine-tuned version: FM_G (fine-tuned with gold subgraph), FM_A (fine-tuned with full subgraph), and FM_{20} (fine-tuned with top 20 triples). For each variant, we report the EM accuracy (M) for the base model and the EM accuracy for the fine-tuned model (FM), along with the average input triples amount (Size), which corresponds to the subgraph size, and the number of questions in each question type ($\#ofQ$)

Results Analysis by Inputs The results reveal a landscape of model performance across different question types and input representations. Notably, all base models (M_G, M_A, and M_{20}) consistently achieved 0% EM accuracy across all question types, which is accorded with the results in [19], as shown in Table 4.5. This outcome suggests that the base models, which were fine-tuned for chat instructions, struggle to generate

Question Type	Category	M_G	FM_G	$Size_G$	M_A	FM_A	$Size_A$	M_{20}	FM_{20}	$Size_{20}$	$\#ofQ$
Simple Question (Direct)	Direct	0	0.9368	310	0	0	2351	0	0.7371	453	5855
Simple Question (Coreference)	Direct	0	0.5823	357	0	0	3846	0	0.2701	525	881
	Indirect	0	0.9289	293	0	0	2954	0	0.4722	438	2785
Simple Question (Ellipsis)	Direct	0	0.9850	319	0	0	2555	0	0.7421	483	667
	Indirect	0	1.0000	346	0	0	3097	0	0.3235	469	68
Verification (Boolean)	Direct	0	0.2159	347	0	0	3212	0	0.0704	525	1519
	Indirect	-	-	-	-	-	-	-	-	-	-
Logical Reasoning	Direct	0	0.8711	378	0	0	4080	0	0.4992	518	1280
	Indirect	0	0.5616	325	0	0	1711	0	0.3014	473	73
Quantitative Reasoning	Direct	0	0.5901	368	0	0	3563	0	0.3602	513	2757
	Indirect	0	0.8457	363	0	0	2226	0	0.6096	508	648
Comparative Reasoning	Direct	0	0.3990	407	0	0	3194	0	0.2358	541	1213
	Indirect	0	0.4351	415	0	0	2015	0	0.2319	579	940
Overall	-	0	0.7410	336	0	0	2923	0	0.4813	489	18686

Table 4.4: Exact Match (EM) accuracy for LLM experiments. The table compares the base and fine-tuned (F) performance of three model variants: M_G, M_A, and M_{20}, along with the average number of input triples (Size) and number of questions in each question type (# of Q).

syntactically correct SPARQL queries, a task that is quite different from their original training objective. This finding suggests that pre-trained LLMs still require targeted fine-tuning to bridge the gap between semantic understanding and the generation of syntactically correct SPARQL queries, which requires specialized knowledge and skills that are not inherently present in the base models.

However, a markedly different picture emerges when examining the performance of the fine-tuned models (FM_G, FM_A, and FM_{20}). The fine-tuned gold subgraph model (FM_G) significantly outperformed the other variants across most question types, with particularly strong results for simpler questions and those involving coreference or ellipsis. For instance, in the ”Simple Question” category, M_{FG} achieved an impressive 94.95% accuracy for direct questions and 66.56% for indirect questions. This stark improvement demonstrates the critical role of fine-tuning in enabling the model to generate accurate SPARQL queries.

In contrast, the fine-tuned full subgraph model (M_{FA}) consistently showed 0% accuracy, aligning with our training observations where this model struggled to converge due to information overload. After investigating some responses from this model, we found that it frequently gives responses that is no relevant to SPARQL generation, like ”Do you mean ...”. This under-performance is likely related to the significantly longer input lengths for this model – often 2-3 times longer than the other variants.

The fine-tuned top-20 triples model (FM_{20}) did not perform as well as the gold subgraph model but still achieved commendable results. Although its overall performance was lower than that of the gold subgraph model, FM_{20} demonstrated significant potential, particularly in handling simpler question types. For instance, it achieved an accuracy of 74.79% for direct simple questions, indicating that the strategy of selecting the most relevant triples may offer a viable approach for balancing information richness and model performance in real-world scenarios where gold subgraphs are not available.

The performance difference between FM_{20} and FM_A also underscores the critical importance of input quality over quantity in this task. This observation aligns with our training results and reinforces the idea that carefully curated input is crucial for effective query generation. Thus, it proves that efficient methods for subgraph retrieval and truncation are promising. The retrieval approach could serve as a baseline for developing more sophisticated subgraph selection methods that balance information content with model tractability.

Results Analysis by Question Types Surprisingly, in some cases, the FM_G and FM_{20} models have better performance on indirect questions than direct questions. When an input indirect question does not provide enough information for generating the SPARQL query, it indicates that the LLM is extracting information from the context and supplementing it to the current question. This demonstrates the effectiveness of the LLM in understanding and utilizing contextual information.

Performance variability across question types offers further insights into the models' capabilities. Simple questions and those involving basic linguistic phenomena, such as coreference and ellipsis, generally achieved higher accuracy. In contrast, more complex reasoning tasks proved more challenging. For instance, comparative reasoning questions saw notably lower accuracies (39.90% for direct and 43.51% for indirect questions with FM_G), highlighting the increased difficulty in translating complex logical structures into query form.

It is worth noting that the SPARQL queries for complex questions tend to be significantly longer and more intricate compared to those for simple questions. The increased length and complexity of these queries make them more susceptible to syntactical errors during the translation process. This observation further emphasizes the challenges associated with handling complex reasoning tasks and the need for robust models capable of generating accurate and well-formed SPARQL queries in such scenarios.

Results Comparison with Other Methods We compare our results with those from the study by [19], which explores the effectiveness of fine-tuned LLMs in SPARQL generation with gold subgraphs, as shown in Table 4.5. Both studies demonstrate that fine-tuned models (LoRA-7B and FM_G) achieve better performance compared to their base counterparts (LLaMA-7B and M_G), highlighting the effectiveness of fine-tuning in improving the models' ability to generate accurate SPARQL queries. While the LoRA-7B model achieves impressive results on simple questions and logical reasoning tasks, it struggles with comparative reasoning, obtaining 0% exact match scores. In contrast, our FM_G model achieves 39.90% and 43.51% exact match scores on direct and indirect comparative reasoning questions, respectively, suggesting that our fine-tuned model has a better grasp of comparative reasoning than the LoRA-7B model. Moreover, our model fine-tuned on truncated subgraphs via gold entities, FM_{20}, still maintains 23.58% and 23.19% EM rates for comparative reasoning tasks.

It is important to note that the study by [19] used a larger training dataset of 30,000 samples, while we used only 3,000 samples due to computational constraints. This difference in training data size may contribute to the performance discrepancies observed between the two studies. The larger training dataset used by [19] likely enabled their models to learn from a more diverse set of examples, potentially leading to better generalization and performance in certain question types. Despite the differences in training data size, the ability of our FM_G and FM_{20} models to handle comparative reasoning questions demonstrates the effectiveness of our fine-tuning approach. Further research is needed to investigate the impact of training data size on model performance. To make the system more practical, additional research on how LLMs perform without gold subgraphs, relying only on truncated subgraphs obtained through Named Entity Recognition (NER) and Named Entity Linking (NEL), is necessary to explore the feasibility and scalability of the approach in real-world scenarios.

Question Type	LLaMA-7B EM	LoRA-7B EM
Simple Question (Direct)	0.000	0.970
Simple Question (Coreference)	0.000	0.867
Simple Question (Ellipsis)	0.000	0.754
Logical Reasoning (All)	0.000	0.926
Verification (Boolean) (All)	0.000	0.851
Comparative Reasoning (All)	0.000	0.000
Quantitative Reasoning (Count) (All)	0.000	0.561

Table 4.5: Zero-Shot Exact Match (EM) Scores for LLaMA-7B and LoRA-7B from the Study of [19]

We also compares our results with the $BertSP_S$ and $BertSP_A$ models from [15], as

shown in Table 4.6. Our FM_G model, which has access to gold subgraphs, achieves a higher overall performance (74.10% EM) compared to the $BertSP_S$ model (70.96% EM). This suggests that our fine-tuned LLM approach is more effective in generating accurate SPARQL queries compared to the BERT-based model when both have access to gold subgraphs. Furthermore, our FM_G model also outperforms the BertSPS model in several question types, such as simple questions (direct and ellipsis) and comparative reasoning.

Question Type	**BertSPS EM**	**BertSPA EM**
Clarification	77.69	76.58
Logical Reasoning (All)	66.89	28.61
Quantitative Reasoning (All)	66.40	59.01
Comparative Reasoning (All)	73.80	39.37
Simple Question (Coreferenced)	69.87	58.83
Simple Question (Direct)	80.69	58.71
Simple Question (Ellipsis)	71.67	50.90
Verification (Boolean)	62.62	24.90
Quantitative Reasoning (Count)	73.20	48.44
Comparative Reasoning (Count)	66.79	40.67
Overall	70.96	48.60

Table 4.6: Exact Match (EM) Scores for BertSPS and BertSPA from [15]

On the other hand, the $BertSP_A$ model, which relies on NER and NEL for entity identification, exhibits a significant drop in performance compared to $BertSP_S$, with an overall EM score of 48.60%. This performance gap highlights the challenges associated with using NER and NEL for entity identification in SPARQL generation tasks, and the amount of triples might influence the performance. Similarly, our FM_{20} model, which is fine-tuned on truncated subgraphs obtained through gold entities, achieves an overall EM score of 48.13%, comparable to BertSPA's performance. This suggests that both approaches still face challenges in dealing with full subgraphs or truncated subgraphs.

It is worth noting that the BertSPS and BertSPA models demonstrate higher performance in logical reasoning and verification (Boolean) question types compared to our models. This indicates that the BERT-based models may have an advantage in handling these specific reasoning tasks.

The comparison with the BertSPS and BertSPA models highlights the strengths of our fine-tuned LLM approach, particularly when given access to gold subgraphs. While the BERT-based models excel in certain question types, our FM_G model demonstrates overall superior performance, especially in comparative reasoning and simple questions.

Further research is needed to investigate the strengths and weaknesses of various approaches and to develop models that can effectively handle a wide range of question types and reasoning tasks, even in the absence of gold subgraphs.

Chapter 5

Conclusions

In this project, we aim at improving the two-stage approach for conversational KGQA by addressing the challenges associated with large subgraphs extracted from KGs. Our proposed methodology involved integrating retrieval techniques to truncate and refine the subgraphs, ensuring their quality and reducing their size to a manageable level. Additionally, we explored the effectiveness of Large Language Models (LLMs) in generating SPARQL queries based on these truncated subgraphs.

Our experiments, conducted on a stratified sample of the SPICE dataset, yielded several key findings. First, the retrieval method proved effective in truncating subgraphs and capturing relevant information for simple questions. However, its performance was lower for complex reasoning questions, indicating the need for improvements in the embedding or search method. Second, the accumulated subgraph approach demonstrated significant improvements in performance for indirect questions by incorporating information from previous turns, mitigating the issue of incomplete information as the conversation progressed. Third, fine-tuned LLMs significantly outperformed their base counterparts in generating accurate SPARQL queries. The fine-tuned gold subgraph model (FM_G) exhibited the best performance, particularly for simpler questions and those involving coreference or ellipsis. The fine-tuned top-20 triples model (FM_20) showed commendable results, indicating the potential of selecting the most relevant triples for balancing information richness and model performance. Fourth, LLMs demonstrated the ability to extract information from context and supplement it to the current question, leading to better performance on indirect questions compared to direct questions in some cases. Finally, performance variability across question types highlighted the challenges associated with handling complex reasoning tasks and the need for robust models capable of generating accurate and well-formed SPARQL queries in

such scenarios.

Future Work Our findings open up several directions for future research in conversational KGQA.

- First, improving retrieval methods should be a key focus. One promising approach is to utilise a graph-structured representation to aggregate information about a question and its context, as demonstrated by [11]. By injecting graph embeddings directly into the LLM and learning them end-to-end, the LLM's reasoning capabilities can be enhanced. Additionally, incorporating a memory module to track and update past evidence as the conversation evolves can provide robustness against noise and retrieval errors. Another direction for improvement is to explore direct fact retrieval methods, such as the DiFaR framework proposed by Ba. This approach bypasses the need for NER/NEL by directly retrieving facts from the KG based on their representational similarities with the input text. By embedding all facts in the knowledge graph onto a dense embedding space and using a reranker to contextualize the input text and the fact jointly, the DiFaR framework has shown significant improvements over the traditional three-step approach. Investigating the integration of such direct fact retrieval methods into our conversational KGQA system could enhance its scalability and practicality in real-world scenarios.

- Second, developing dynamic subgraph truncation techniques that can maintain a manageable size of the subgraph throughout the conversation while preserving crucial information is an important area for future research. As conversations progress and triples accumulate, the subgraph can become very large, posing challenges in efficiently searching for relevant information.

- Third, further investigation into the impact of training data size on LLM performance and the development of models that can effectively handle a wide range of question types and reasoning tasks, even in the absence of gold subgraphs, is necessary. Exploring the strengths and weaknesses of various LLM architectures and fine-tuning strategies could lead to more robust and versatile models for conversational KGQA.

- Finally, incorporating answer-sensitive KG-to-Text methods, such as the Retrieve-Rewrite-Answer framework proposed by Wu et al. (2023), could enhance the

performance of LLMs in conversational KGQA. Their approach transforms KG knowledge into well-textualized statements that are most informative for KGQA. By generating textual knowledge that is specifically tailored to the question at hand, LLMs can be provided with more targeted and relevant information for generating accurate SPARQL queries. Integrating answer-sensitive knowledge generation into our conversational KGQA system could potentially help bridge the gap between the structured knowledge in the KG and the natural language understanding capabilities of LLMs. By providing LLMs with well-formed, informative textual representations of the relevant KG knowledge, we can leverage their strong language understanding and generation capabilities to produce more accurate and contextually appropriate SPARQL queries, ultimately improving the overall performance of the conversational KGQA system.

Bibliography

[1] Jinheon Baek, Alham Fikri Aji, Jens Lehmann, and Sung Ju Hwang. Direct fact retrieval from knowledge graphs without entity linking. *arXiv preprint arXiv:2305.12416*, 2023.

[2] Tom B Brown. Language models are few-shot learners. *arXiv preprint ArXiv:2005.14165*, 2020.

[3] Jiaoyan Chen, Yuxia Geng, Zhuo Chen, Jeff Z Pan, Yuan He, Wen Zhang, Ian Horrocks, and Huajun Chen. Zero-shot and few-shot learning with knowledge graphs: A comprehensive survey. *Proceedings of the IEEE*, 111(6):653–685, 2023.

[4] Philipp Christmann, Rishiraj Saha Roy, Abdalghani Abujabal, Jyotsna Singh, and Gerhard Weikum. Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 729–738, 2019.

[5] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

[7] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488, New York, NY, USA, 2021. Association for Computing Machinery.

[8] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. Evaluating entity linking with wikipedia. *Artificial Intelligence*, 194:130–150, 2013. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

[9] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[10] Parag Jain and Mirella Lapata. Conversational semantic parsing using dynamic context graphs. *arXiv preprint arXiv:2305.06164*, 2023.

[11] Parag Jain and Mirella Lapata. Integrating large language models with graph-based reasoning for conversational question answering. *arXiv preprint arXiv:2407.09506*, 2024.

[12] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[13] Pierre Marion, Pawel Nowak, and Francesco Piccinno. Structured context and high-coverage grammar for conversational question answering over knowledge graphs. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8813–8829, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[14] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

[15] Laura Perez-Beltrachini, Parag Jain, Emilio Monti, and Mirella Lapata. Semantic parsing for conversational question answering over knowledge graphs. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2507–2522, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[16] Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392, 2014.

[17] Amrita Saha, Vardaan Pahuja, Mitesh Khapra, Karthik Sankaranarayanan, and Sarath Chandar. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In Sheila McIlraith and Kilian Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 705–713, New Orleans, Louisiana, USA, 2018. AAAI Press.

[18] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[19] Phillip Schneider, Manuel Klettner, Kristiina Jokinen, Elena Simperl, and Florian Matthes. Evaluating large language models in semantic parsing for conversational question answering over knowledge graphs. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, pages 807–814. INSTICC, SciTePress, 2024.

[20] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Appendix A

Retrieval model fine-tune

To enhance the performance of our retrieval model, we fine-tuned a DistilBERT model for the specific task of question-triple matching. As the final results do not surpass the DistilBert embedding model, we put it in the Appendix for reference.

We framed the task as a binary classification task, where the model learns to distinguish between relevant (positive) and irrelevant (negative) question-triple pairs.

For the training dataset, we concatenated user questions with their corresponding gold triples to create positive samples. Negative samples were generated by pairing questions with randomly selected, irrelevant triples. The final dataset comprised 62,384 examples, with an equal distribution of positive and negative samples.

We utilized the DistilBERT-base-uncased model and tokenizer from the Hugging Face Transformers library. The model was fine-tuned using the following configuration:

- Batch size: 16 (per device)
- Learning rate: 2e-5 with linear warm-up
- Number of epochs: 3
- Optimizer: AdamW with weight decay of 0.1
- Mixed precision training (FP16)
- Gradient accumulation steps: 2

The training process employed early stopping with a patience of 3 and a threshold of 0.0001. We used a linear learning rate scheduler with warm-up steps. During the training, the training logs reveal a steady decrease in both training and validation loss over the course of the fine-tuning process. The training loss decreased from an initial

value of 0.1829 to 0.0385 by the 3900th step, while the validation loss reduced from 0.1568 to 0.0856, indicating that the model was learning to distinguish between relevant and irrelevant question-triple pairs.

Question Type	**Category**	**Hit@20**		**Hit@20 (Fine-tuned)**	
		Hit rate	**Recall**	**Hit rate**	**Recall**
Simple Question (Direct)	Direct	0.9298	0.9696	0.3347	0.6227
	Indirect	0.5669	0.9043	0.0573	0.4944
Simple Question (Coreferenced)	Direct	0.3144	0.6384	0.0840	0.4123
	Indirect	0.2526	0.6934	0.1065	0.5521
Quantitative Reasoning (All)	Direct	0.1209	0.6518	0.0173	0.5112
Logical Reasoning (All)	Direct	0.9175	0.9860	0.0400	0.4530
	Indirect	0.4231	0.9510	0.1731	0.6863
Comparative Reasoning (All)	Direct	0.2367	0.7815	0.0356	0.5803
	Indirect	0.0968	0.6302	0.0632	0.5691
Verification (Boolean) (All)	Direct	0.6162	0.9279	0.0513	0.4703
Quantitative Reasoning (Count) (All)	Direct	0.4821	0.8239	0.0659	0.5458
	Indirect	0.7207	0.8820	0.4691	0.7380
Simple Question (Ellipsis)	Direct	0.8966	0.9493	0.3178	0.6129
	Indirect	0.3235	0.6716	0.2059	0.7206
Comparative Reasoning (Count) (All)	Direct	0.1980	0.7595	0.0342	0.5948
	Indirect	0.1312	0.6129	0.0817	0.5161

Table A.1: Comparison of Hit@20 (Comma) vs Hit@20 (Fine-tuned) Models

However, when we compare the performance of the fine-tuned model with the base DistilBERT model (Table 1), we observe a significant drop in performance across all question types and categories. For instance, in the Simple Question (Direct) category, the hit rate at 20 decreased from 0.9298 to 0.3347 for direct questions, and from 0.5669 to 0.0573 for indirect questions.

The performance degradation can be primarily attributed to two key factors. First, there's a significant task mismatch between the fine-tuning objective and the actual application. While the model was trained on a binary classification task with individual query-triple pairs, the real-world application involves retrieving relevant triples from large subgraphs containing thousands of triples. This disparity in scale and complexity between training and application environments likely contributed to the model's poor generalization. Second, the data quality, particularly in the generation of negative samples, may not adequately represent the challenges faced in the actual retrieval task. With entities potentially associated with numerous triples in the application, the

training data might not have captured the nuanced discrimination required in real-world scenarios.

For future work, we propose adapting the fine-tuning process to better align with the actual retrieval task. This could involve training on larger contexts that include multiple triples per query, more closely mimicking the structure of the subgraphs encountered in the application. Additionally, improving the negative sampling strategy to generate more challenging and diverse examples, especially considering the many-to-one relationship between triples and entities, could enhance the model's discriminative capabilities in complex, real-world knowledge graphs.