ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

DYNAMIC FEW-SHOT LEARNING FOR KNOWLEDGE GRAPH QUESTION ANSWERING

CANDIDATE
Jacopo D'Abramo

SUPERVISOR
Prof. Paolo Torroni

CO-SUPERVISOR
Andrea Zugarini

Academic year 2023-2024

Session 2nd

Abstract

The advent of large language models has opened up new avenues for Question Answering over Knowledge Graphs (KGQA), a field that has witnessed significant advancements due to their capabilities. Despite these advancements, large language models are not inherently designed for query generation, necessitating the development of fine-tuning solutions or ad-hoc architectures, which, while effective in certain scenarios, exhibit limitations in generalizing across diverse domains.

In response to these challenges, this thesis presents a groundbreaking method known as Dynamic Few-Shot Learning (DFSL). DFSL leverages the principles of in-context learning combined with semantic similarity measures to create a versatile and robust framework for KGQA.

An extensive evaluation across multiple benchmark datasets and architecture configurations, reveals not only the superior performance of DFSL in comparison to existing state-of-the-art methods but also exhibits adaptability against different KGs. This highlights the potential of DFSL to serve as a universally applicable solution for KGQA.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The growth of the Semantic Web has transformed the traditional, document-based web into an intelligent system that integrates data and web content into a more structured environment. This revolutionary change enables software agents to autonomously perform tasks for users, thus enhancing the web's utility and efficiency. Central to the Semantic Web are ontologies, which represent the information in a machine-processable structure. Ontologies are data models used to represent the semantics of domain concepts through classes (entities) and relationships (properties). These ontologies define the schema of a domain without including specific information about individual instances. When data instances are inserted into these ontological terms, they form what is known as a Knowledge Graph (KG) [1].

The creation and storage of vast amounts of structured knowledge have resulted in the development of massive Knowledge Graphs such as Wikidata [2], DBpedia [3], and FreeBase [4]. For instance, Wikidata alone contains over 109 million items [1], showcasing the extensive scale of these KGs. However, as the size of KGs has expanded, extracting relevant information from them has become increasingly challenging. This challenge has given rise to the field of Knowledge Graph Question Answering (KGQA), which aims to answer natural language questions posed over those massive KGs.

[1] https://www.wikidata.org/wiki/Wikidata:Statistics

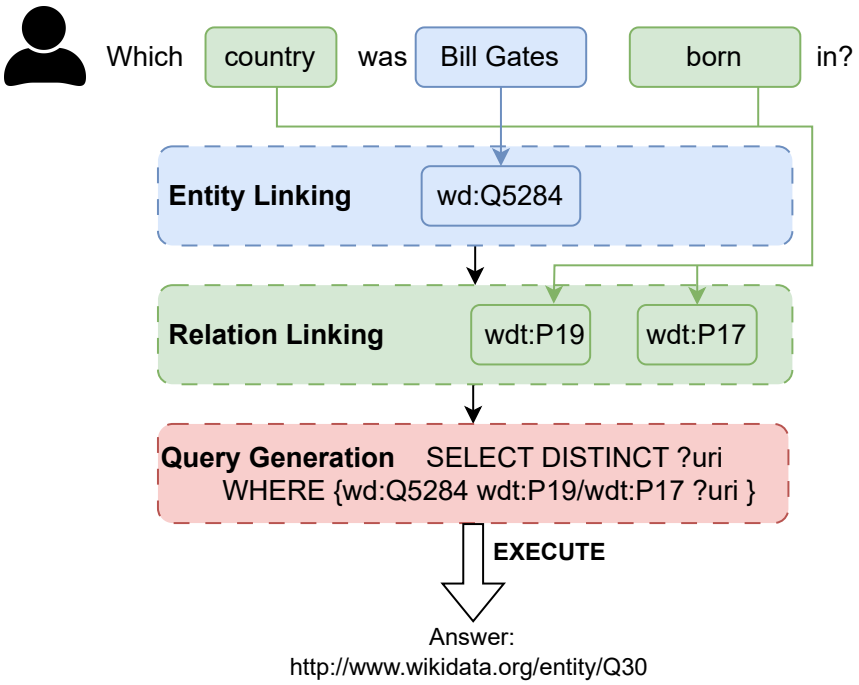

Figure 1.1: A typical KGQA system

A typical KGQA system is composed of three main components. Firstly, the objects of interest in a natural language question (NLQ) are detected and linked to the KG in a process known as entity linking (EL). Secondly, the relationship between these objects is identified and connected to the KG in a step called relation linking (RL). Finally, a formal query, usually in SPARQL, is generated using the linked entities and relations, and executed on the KG to retrieve the answer [5]. This final step, known as Query Generation (QG), is the primary focus of this thesis. Figure 1.1 illustrates this KGQA pipeline.

State-of-the-art approaches to SPARQL query generation from natural language questions are primarily based on fine-tuning language models such as T5 [6] or utilizing ad-hoc architectures that leverage large language models (LLMs) and dependency trees [7]. These methodologies have demonstrated substantial success, significantly outperforming traditional approaches like rule-based or template-based methods. However, despite their advancements, these state-of-the-art techniques exhibit certain limitations in terms of

flexibility and scalability. Fine-tuning language models can be computationally expensive due to the large size of datasets and the limited availability of GPUs. Additionally, these models often struggle to handle out-of-domain distributions effectively.

This thesis introduces a novel approach to KGQA by leveraging in-context learning with Large Language Models. The central hypothesis of this research is that: *a significant number of errors in current systems can be mitigated by making more effective use of the examples within the training set*. To this end, the proposed methodology, termed Dynamic Few-Shot Learning (DFSL), employs semantic search to retrieve similar questions from the training set and enrich the prompt accordingly. This approach not only eliminates the need for fine-tuning models but also has the potential to establish a general system for KGQA. This general system would feature a storage of examples that are not limited to the training set of a specific dataset but could encompass a broader repository of examples.

To evaluate the performance and robustness of DFSL, experiments were conducted on two widely-used Knowledge Bases, DBpedia and Wikidata, utilizing four publicly available datasets: QALD-9 (based on DBpedia), and QALD-9 plus, QALD-10, and LC-QuAD 2.0 (based on Wikidata). As backbones, three state-of-the-art LLMs were employed: Mixtral 8x7B, Llama-3 70B, and CodeLlama 70B. The experimental results demonstrate that the proposed model achieves new state-of-the-art results, with significant advantages in terms of speed and efficiency. Additionally, ablation studies were conducted to assess the effectiveness of the approach in the absence of gold information from the Entity Linking (EL) and Relation Linking (RL) modules.

Overall, this thesis presents a significant advancement in the field of KGQA by addressing the limitations of current approaches and proposing a more flexible and scalable solution.

Chapter 2

Background

In the following section, an overview of large language models (LLMs) will be provided, explaining the transformer-based architecture that underpins them. Additionally, the discussion will detail the concepts behind the three main LLMs used in this thesis: the Sparse Mixture of Experts, which forms the basis of the Mixstral 8x7 model, and the Llama architecture, which is foundational for Llama3 and CodeLlama.

2.1 Large Language Model

Large Language Models (LLMs) represent a groundbreaking advancement in the field of artificial intelligence, particularly within natural language processing (NLP). These models are designed to comprehend, generate, and manipulate human language with an unprecedented level of sophistication and accuracy [8]. The development and deployment of LLMs have revolutionized the way machines interact with human language, enabling a wide range of applications that were previously thought to be the exclusive domain of human intelligence.

The journey towards the development of LLMs has been marked by significant milestones in the evolution of NLP. Initially, language processing tasks were handled using rule-based systems. These models were more flexible

and could handle a wider range of language patterns, but they required a lot of computer power and large datasets to work well. In the early 1970s, the field of artificial intelligence saw the introduction of Hidden Markov Models (HMMs) [9], and Conditional Random Fields (CRFs) [10].

The advent of deep learning, particularly the use of sequence-to-sequence neural networks, marked a transformative phase in NLP. Recurrent Neural Networks (RNNs) [11] and their variants, such as Long Short-Term Memory (LSTM) [12] networks and Gated Recurrent Units (GRUs) [13], have collectively shaped the landscape of natural language processing and deep learning, opening up new possibilities for understanding and processing human language. They enabled the processing of sequential data and improved the handling of context in language . However, these models still faced challenges with long-range dependencies and computational efficiency.

Ultimately, the introduction of the *Transformer* architecture addressed many of above-mentioned shortcomings, leading to significant improvements in performance. This models, introduced in the 2017 paper 'Attention is All You Need' [14], brought parallel sequence processing and the ability to capture extensive dependencies across large sequences.

The Transformer model discards the recurrence used in RNNs and LSTMs, instead relying entirely on a mechanism called *self-attention*. Self-attention allows the model to weigh the importance of different words in a sequence when encoding a word, thereby capturing contextual relationships more effectively. The architecture consists of an encoder and a decoder, each composed of multiple layers of self-attention and feed-forward neural networks. The transformer architecture is shown in Figure 2.1

In the encoder, each layer has two main components: a **multi-head self-attention mechanism** and a **position-wise fully connected feed-forward network**. The multi-head attention mechanism enables the model to focus on different parts of the input sequence simultaneously. **Positional encodings** are added to the input embeddings to retain information about the position of each

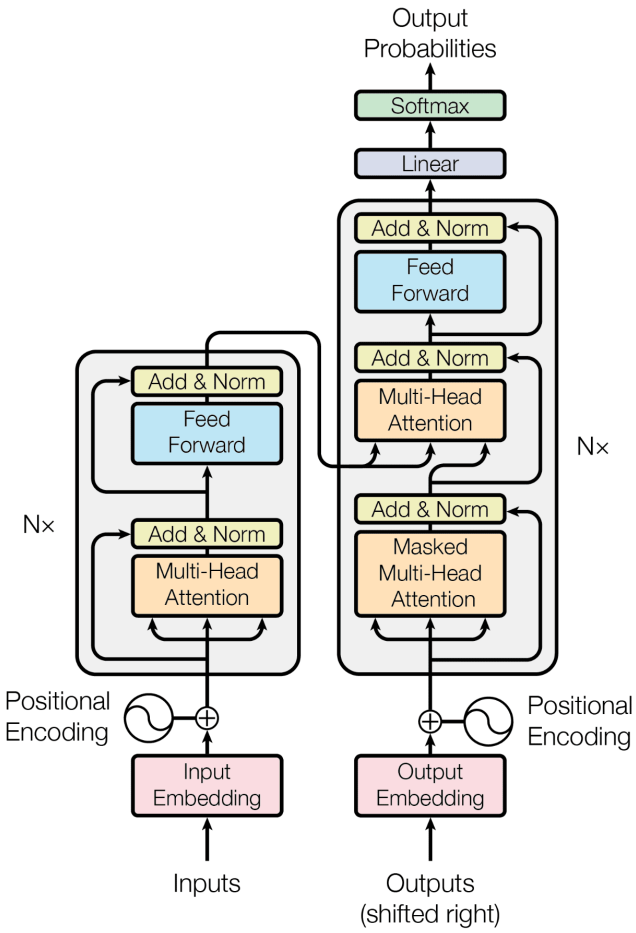

Figure 2.1: Transformer architecture

word in the sequence.

The decoder also has these components but includes an additional layer of attention that attends to the output of the encoder stack. This helps the decoder to generate sequences by focusing on relevant parts of the input sequence and its previously generated outputs.

One of the key innovations of the Transformer is the self-attention mechanism which computes dependencies between all words in the sequence simultaneously, as opposed to the sequential nature of RNNs. The self-attention mechanism operates on an input sequence, where each word is first embedded into a high-dimensional space. Given an input sequence of n words, represented as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, the self-attention mechanism computes the

attention as follows:

1. **Linear Transformations:** The input embeddings are linearly transformed to create three distinct vectors for each word: the *query* vector $\mathbf{Q}$, the *key* vector $\mathbf{K}$, and the *value* vector $\mathbf{V}$. These are computed using learned weight matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V$$

2. **Scaled Dot-Product Attention:** The attention scores are calculated by taking the dot product of the query vector $\mathbf{Q}$ with the key vector $\mathbf{K}^\top$ of all words, followed by scaling by the square root of the dimension d_k of the key vectors. This is then passed through a softmax function to obtain the attention weights:

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

3. **Output Representation:** The attention weights are used to create a weighted sum of the value vectors $\mathbf{V}$. This weighted sum represents the contextualized embedding for each word, incorporating information from the entire sequence.

To allow the model to attend to different parts of the sequence from multiple perspectives, the Transformer employs *multi-head attention*. Instead of computing a single attention function, the mechanism projects the queries, keys, and values into multiple subspaces, applies the attention function in each subspace (referred to as a head), and concatenates the results:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)\mathbf{W}_O$$

where each head is computed as:

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_{Q_i}, \mathbf{K}\mathbf{W}_{K_i}, \mathbf{V}\mathbf{W}_{V_i})$$

Here, $\mathbf{W}_{Q_i}$, $\mathbf{W}_{K_i}$, $\mathbf{W}_{V_i}$ are the projection matrices for the i-th head, and $\mathbf{W}_O$ is the output projection matrix.

The Transformer model revolutionized NLP by introducing a more efficient, scalable, and interpretable architecture that excels in capturing dependencies across long sequences, laying the foundation for numerous state-of-the-art models such as BERT [15], GPT3 [16] and T5 [17] .

2.1.1 Sparse Mixture of Experts

The Sparse Mixture of Experts (SMoE) architecture addresses the limitations of traditional dense models, which, despite their power, suffer from high computational costs and scalability issues.

At the heart of the SMoE architecture are multiple sub-models, referred to as ”experts.” Each expert is trained to specialize in different aspects of the data, allowing the model to handle a variety of tasks with greater precision. For example, the Mixtral 8x7B model utilizes eight experts, each of which focuses on distinct features or patterns within the data

A key feature of SMoE is its sparse activation mechanism. Unlike traditional models that activate all parameters during inference, SMoE selectively engages only a subset of the experts. This selective activation is managed by a small, trainable network known as a router. The router dynamically assigns input tokens to the most relevant experts based on the characteristics of the data. The router’s ability to direct tokens to specific experts ensures that only the necessary parameters are utilized during each forward pass. This approach not only optimizes resource usage but also allows the model to handle larger contexts and more complex tasks without a proportional increase in computation. An advantage of the SMoE architecture is its scalability. The

model's effective parameter usage per inference remains manageable, even with a high total parameter count. The selective activation mechanism also contributes to the model's enhanced performance. By engaging only the most relevant experts, the SMoE model can process inputs more effectively. This is particularly evident in benchmarks where Mixtral 8x7B [18] has demonstrated superior performance over models like Meta's LLaMA 2 70B [19] and OpenAI's GPT-3.5.

2.1.2 LLaMA Architecture

LLaMA (Large Language Model Architecture) [20], a LLM developed by Meta AI, has advanced through several updates, currently up to LLaMA 3. Despite these advancements, all versions are based on the core LLaMA architecture, which employs a transformer-based optimized version (see figure 2.2).

One of the significant advancements in LLaMA is the introduction of Grouped Query Attention (GQA), which improves inference scalability by allowing the model to handle larger context windows more efficiently.

The architecture of LLaMA models also includes pre-normalization using RMSNorm [21] and SwiGLU activation functions. RMSNorm helps stabilize training by normalizing the inputs to each layer, while SwiGLU provides non-linear transformations that improve model expressiveness and performance. Another notable feature is the use of rotary positional embeddings (RoPE), which encode positional information in a way that enhances the model's ability to handle long-range dependencies within the input data.

LLaMA models utilize a tokenizer based on Byte Pair Encoding (BPE), implemented through SentencePiece. This tokenizer splits input text into subword units, allowing the model to handle rare words and out-of-vocabulary terms more effectively. The vocabulary size for LLaMA models is 32k.

In terms of model sizes, Llama 1 was originally released with 4 different

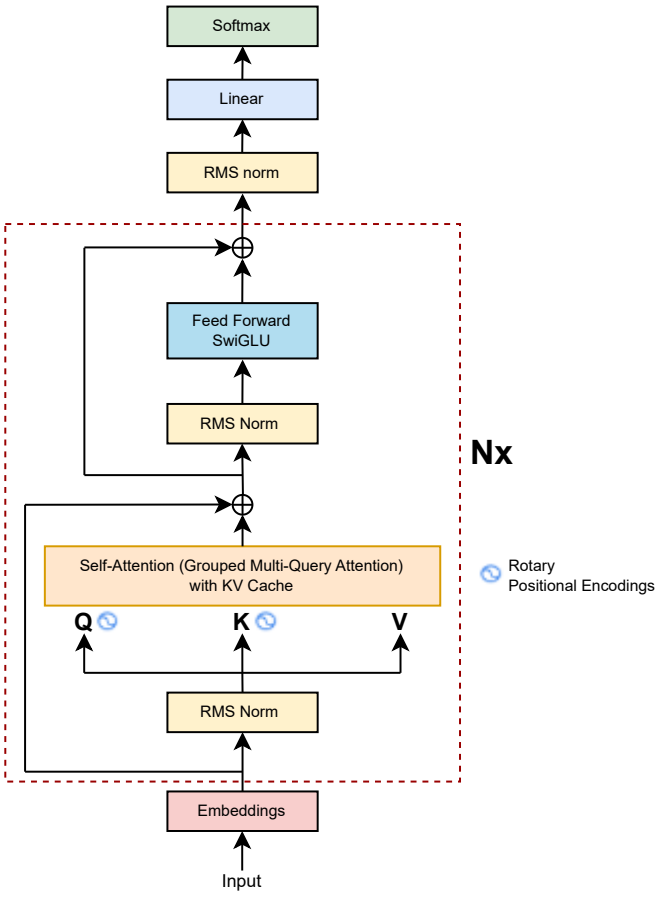

Figure 2.2: LLaMa architecture

variants with parameters 6.7B, 13B, 32.5B and 65.2B. The number of heads in the multi head attention in each of these variants are 32, 40, 52 and 64 respectively as opposed to transformer which had 8 heads in the multi head attention. The most recent version of LLaMa, namely LLaMa 3 has been released in 2 versions:

- **LLaMa 3 8B**, an 8-billion parameter model with a knowledge cutoff of March 2023
- **LLaMa 3 70B**, a 70-billion-parameter model with a knowledge cutoff of December 2023

To unlock Llama 3's full potential for chat and dialogue applications, Meta introduced a new approach aligned with the concept of instruction fine-tuning.

This method combines supervised fine-tuning (SFT), rejection sampling, proximal policy optimization (PPO), and direct preference optimization (DPO).

Chapter 3

Related Works

The following section explores the literature concerning the evolution of methods in the field of Knowledge Graph Question Answering. It examines the major features that influence In-Context Learning and delves into the cognate domain of text-to-SPARQL, namely Text-To-SQL.

3.1 Knowledge Graph Question Answering

3.1.1 Early approaches

Early research in this area focused on manually crafting queries to test the inference capabilities of ontology systems [22, 23]. A notable development was SPLODGE, a tool designed to generate benchmark SPARQL queries for Linked Open Data (LOD). SPLODGE systematically combines smaller query patterns based on predefined characteristics such as structure and data source count, with constants chosen randomly to ensure diversity. A verification step ensures that the generated queries meet specific constraints [24]. However, these manual or semi-manual approaches have limitations when adapting to large-scale knowledge bases like WikiData and DBpedia.

3.1.2 Template-Based approaches

Template-filling approaches have been extensively explored in Knowledge Graph Question Answering (KGQA). The core idea involves generating templates of SPARQL queries and filling in the slots using defined heuristics to produce the final SPARQL query. One method relies on parsing the question to create a SPARQL template that mirrors the internal structure of the question, then instantiates the template using statistical entity identification and predicate detection [25]. Another approach introduces a pipeline paradigm, recognizing the query intention structure inherent in natural language questions and mapping the involved semantic items into existing knowledge bases to instantiate these structures [26]. The QUICK approach [27] identifies keywords to generate an initial query template and refines it incrementally to create the final query. Additionally, a semantic parsing model has been presented that learns to generate SPARQL queries from question-answer pairs.

One of the primary challenges with template-based approaches is the labor-intensive nature of template creation. Typically, templates are constructed either manually or through a semi-automated process, both of which are time-consuming. Furthermore, query templates are inherently specific to the knowledge graph being utilized, meaning any changes to the underlying graph structure may necessitate a complete overhaul of the entire set of templates.

3.1.3 Pretrained Language Models for Text-to-SPARQL

The recent advancements in pretrained language models (PLMs), have significantly enhanced performance in the Knowledge Graph Question Answering task. These improvements have been achieved through fine-tuning models, developing ad-hoc architectures, and prompting LLMs. One notable contribution in this domain utilizes a relation-aware attention decoder and a pointer network encoder [28]. This model incorporates three distinct scaled dot-product attention mechanisms to generate SPARQL queries that effectively

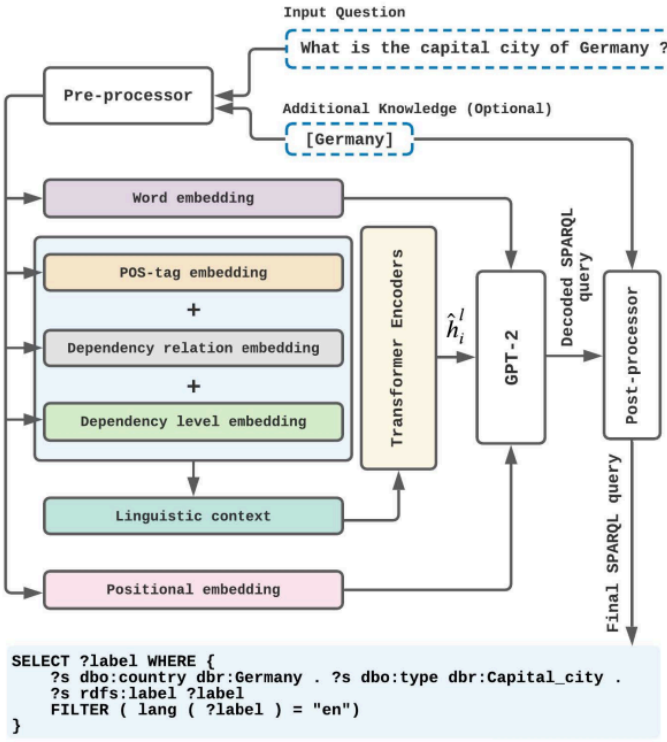

Figure 3.1: SGPT architecture

capture entity, relation, and keyword representations.

Further explorations involved experimenting with various models, including T5 [17], BART [29], and Pointer Generation Networks [30], to assess their effectiveness in KGQA tasks [5].

Another significant model employs a stack of transformer encoders to extract linguistic features from questions and uses GPT-2 as a decoder [7]. This model preprocesses input sequences through five distinct embedding layers to capture various properties of the input: word embeddings for token-level information, POS-tag embeddings for part-of-speech tagging, dependency relation embeddings for encoding relationships between word pairs, dependency level embeddings for information about token children, and positional embeddings for absolute position information (see Figure 3.1). Word and positional embeddings are utilized in the decoder, while the other embeddings are used in

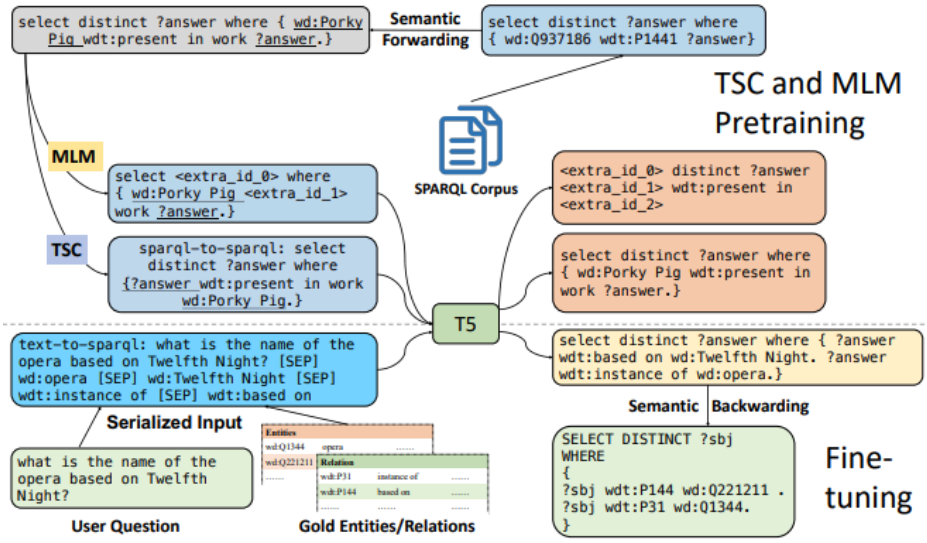

Figure 3.2: TSET architecture

the encoder. Despite its comprehensive design, this model is limited in capturing connections among entities and relations in the knowledge graph, leading to errors in generating accurate triple sequences for SPARQL queries.

A one-shot generative approach has also been proposed, where the prompt is augmented with a knowledge graph fragment necessary for constructing the query and an example of a question-subgraph query [31].

To address this, a fine-tuned T5 model with a pretraining stage called Triplet Structure Correction (TSC) was developed [6] (see Figure 3.2). In this stage, the model randomly swaps the subject, predicate, and object positions in SPARQL query triples with a certain probability, enhancing the model's understanding of triple order. Additionally, in the pretraining stage, they implement masked language modeling (MLM) by randomly masking specific tokens in an input sequence and allowing the model to predict the original masked tokens from their context. Following the pretraining phase, the model undergoes fine-tuning for the SPARQL query generation task. This approach has established state-of-the-art performance on major KGQA datasets.

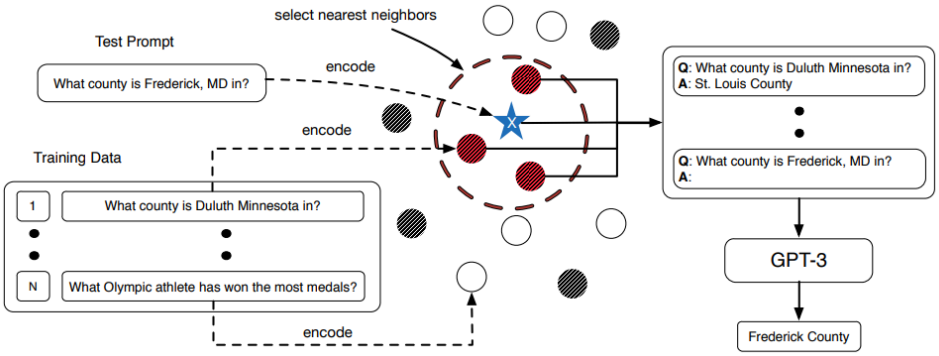

Figure 3.3: KATE architecture

3.2 In-Context Learning

In-Context Learning (ICL) is a paradigm where learning is facilitated through analogy. In this approach, a few demonstration examples are provided to form a contextual prompt, typically expressed in natural language. The query question is then appended to this prompt context, creating an input that is processed by the language model to generate predictions. Unlike supervised learning, which necessitates a training phase involving backward gradients to update model parameters, ICL does not involve parameter adjustments. Instead, the model is designed to discern patterns within the provided demonstrations and make predictions accordingly [32]. Research has indicated that the efficacy of ICL significantly depends on the characteristics of the demonstration surface, including the selection, formatting, and sequencing of the examples [33].

3.2.1 Demonstrations selection

The selection of demonstrations aims to identify samples that serve as effective examples for ICL. KATE, an unsupervised retriever, employs k-nearest neighbors and distance metrics, such as L2 distance and cosine similarity, to select in-context examples for tasks like sentiment analysis, table-to-text generation, and question answering [34] (see Figure 3.3).

The inclusion of diverse demonstrations in prompts for compositional semantic parsing tasks has been shown to enhance structural coverage in target utterances [35]. Using output scores of LLMs as unsupervised metrics has shown effectiveness in demonstration selection; in particular, the best subset permutation of kNN examples was selected based on the code length for data transmission to compress label y given x and C [36]. Additionally, selecting examples based on perplexity, specifically lower perplexity, has been found to be an effective strategy [37].

3.2.2 Demonstrations formatting

In addition to directly selecting examples from training data, recent research has focused on utilizing large language models to reformat the representation of existing demonstrations. One approach involves generating demonstrations directly from LLMs, thereby reducing reliance on external demonstration data [38]. Structured Prompting has been proposed to encode demonstration examples separately with special positional embeddings, which are then provided to the test examples using a rescaled attention mechanism [39]. Other approaches diverge by modifying the latent representation of demonstrations [40].

3.2.3 Demonstrations ordering

Ordering the selected demonstration examples is also a crucial aspect. One approach is to arrange examples based on their similarity to the input, with the closest example positioned as the first demonstration [34]. Another method uses global and local entropy metrics, which have been shown to correlate positively with in-context learning performance, to optimize demonstration ordering [41]. Furthermore, an alternative strategy involves ranking demonstrations from simple to complex, thereby progressively increasing the complexity of examples during the inference process [42].

3.3 Text-To-SQL

A cognate domain, text-to-SQL, focuses on the translation of natural language questions to SQL queries. In this area, research has demonstrated a zero-shot and few-shot approach using simple prompts, achieving lower results compared to fine-tuned approaches with models such as GPT-3 [16] and CODEX [43]. Various strategies for selecting examples based on similarities and dissimilarities have been introduced [44]. For instance, selecting similar questions with the same difficulty level and dissimilar questions using k-means clustering to obtain diverse examples close to each centroid has shown promising results.

More recently, an automatic chain-of-thought [45] approach has been proposed [46], where question slices are matched with all possible table and column names to identify the most relevant ones for a given question. This approach utilizes models such as GPT-3.5 and GPT-4.

Despite the similarities between text-to-SQL and text-to-SPARQL, the methods developed for the former are not directly applicable to the latter. The primary difference lies in the data models: text-to-SQL deals with a relatively small-sized set of tables and columns, whereas text-to-SPARQL operates within a domain modeled by a large-scale, semi-structured knowledge graph. This fundamental difference necessitates distinct approaches and strategies for effectively handling text-to-SPARQL tasks.

Chapter 4

Methodology

The following section defines the task to be tackled and provides an explanation of the proposed approach, detailing the function of all its components.

4.1 Problem formulation

Knowledge Graph Question Answering (KGQA) encompasses two main components: a knowledge graph and a question answering system. A knowledge graph is a structured representation of facts, comprising entities, relationships, and semantic descriptions [47]. Formally, a knowledge graph $\mathcal{G}$ is defined as $\mathcal{G} := (\mathcal{E}, \mathcal{R}, \mathcal{F})$, where $\mathcal{E}$ represents *entities*, $\mathcal{R}$ represents *relations*, and $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ are *facts*. Entities can be real-world objects or abstract concepts, relationships denote the connections between entities, and semantic descriptions of entities and their relationships include types and properties with well-defined meanings [47] (see Figure 4.1) . Knowledge is expressed in factual triples in the form of <*head, relation, tail*> or <*subject, predicate, object*> under the Resource Description Framework (RDF) [1]. For example, a fact can be represented as *(Albert Einstein, WinnerOf, Nobel Prize)*.

On the other hand, Question Answering (QA) systems can be seen as an

[1] `https://www.w3.org/TR/rdf-concepts/`

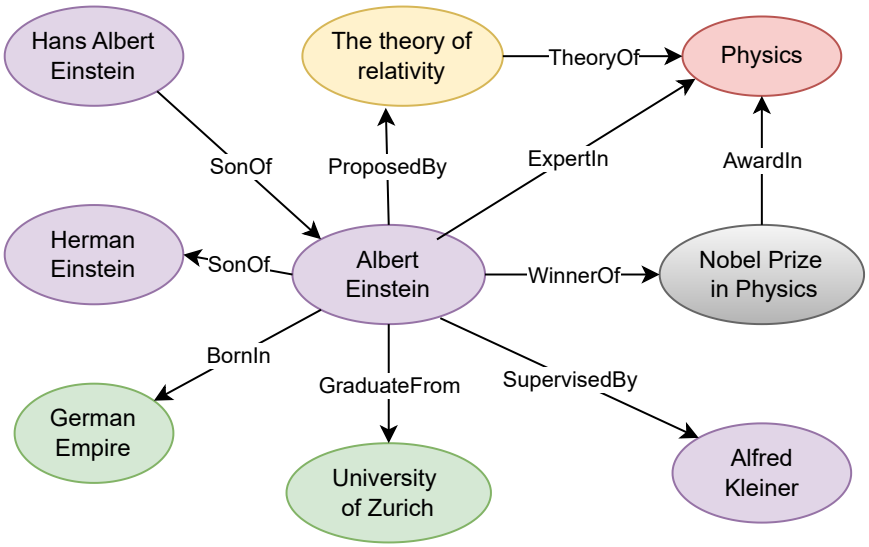

Figure 4.1: An example of a Knowledge Graph (KG)

extension of search engines. They aim to automatically provide users with precise answers to questions posed in natural language, instead of simply returning a ranked list of relevant sources based on a set of keywords. The architecture of QA systems depends significantly on the type of underlying knowledge sources. If the objective is to extract answers from plain text, traditional information retrieval techniques combined with machine comprehension methods are commonly used. Conversely, if data is represented as a graph, methods relying on graph processing and SPARQL query generation are employed to extract the required information [48].

Thus, the KGQA task can be framed as a **text-to-SPARQL** task where a question q must be translated into a SPARQL query s_q to be executed on $\mathcal{G}$ by a SPARQL engine, to return a (possibly empty) answer a. According to the official definition [2] a SPARQL query s_q can be formally considered as a tuple $\langle GP, DS, SM, R_i \rangle$ where GP is a graph pattern, DS is an RDF dataset, SM is a set of solution modifiers (*ORDER, PROJECTION, DISTINCT, OFFSET, LIMIT*), R is a result form (*SELECT, CONSTRUCT, DESCRIBE, ASK*). The

[2] `https://www.w3.org/2001/sw/DataAccess/rq23/defns`

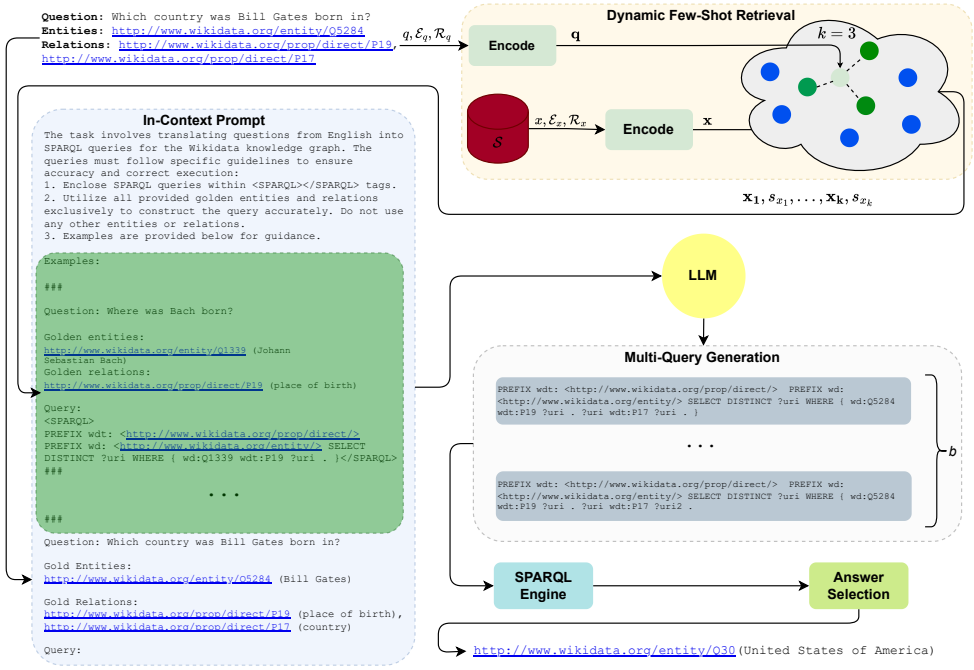

Figure 4.2: Sketch of DFSL. Given a question, its entities and its relations, k-most similar examples are retrieved from a text-to-SPARQL collection S and injected into the in-context prompt. Then, the LLM generates one or more queries that are all executed by a SPARQL engine. An answer selection strategy identifies which response to pick.

entities and relations in q, denoted as $\mathcal{E}_q$ and $\mathcal{R}_q$, may be, and usually are, extracted from q before generating s_q. Hence, query generation can be tackled as a conditional text generation problem given $q, \mathcal{E}_q$ and $\mathcal{R}_q$. Within the scope of ICL, P_θ is a pre-trained LLM and the conditional input $\mathcal{E}_q, \mathcal{R}_q, q$ is combined with other contextual information C, such as additional instructions, guidelines, constraints and demonstrations, all expressed in natural language text. Accordingly, the generated query is:

$$s_q = \arg\max_s P_\theta(s|C, \mathcal{E}_q, \mathcal{R}_q, q). \tag{4.1}$$

4.2 Dynamic Few-Shot Retrieval

The selection and sequencing of demonstrations in the prompt significantly influence performance in the field of In-Context Learning (ICL), as demonstrated in Section 3.2.Typically, without a strategic approach, the examples included in the prompt are predetermined, representative instances of the task at hand, chosen manually during the prompt design phase. However, this hand-picking process presents several challenges:

- **Scalability**: As the number of classes increases, ensuring that the predetermined examples are representative of each class becomes increasingly difficult. This limitation can restrict the model's applicability to larger and more complex datasets.
- **Limited Variability Within Classes**: Predetermined examples may not cover the full spectrum of possible variations within a class. Consequently, the model may struggle to generalize well to new instances.
- **Effort and Time Requirements**: Hand-picking examples demands significant effort and time, making it an inefficient approach.

To address these issues, a dynamic retrieval method for selecting relevant examples based on their similarity to the input question is employed. Inspired by KATE [34], the approach employed leverages the similarity between a question q and a set of previously answered text-to-SPARQL examples collected in a storage $\mathcal{S}$ (see Figure 4.2). Each stored example consists of a tuple containing a question x, its entities $\mathcal{E}_x$ and relations $\mathcal{R}_x$, and the associated SPARQL query s_x. To capture the semantic features of the question, entities, and relations, these components are mapped onto a vector representation $e_q \in \mathbb{R}^d$ using a sentence encoder. The input for the encoder-only language model (LM) is formed by concatenating the question, entities, and relations into a single sequence $\boldsymbol{q} := [q, \mathcal{E}_q, \mathcal{R}_q]$.

Likewise, we encode each example $x \in \mathcal{S}$ into a vector $e_x \in \mathbb{R}^d$ and then compute the similarity between the target question and the storage:

$$score(\boldsymbol{q}, \boldsymbol{x}) = sim(e_q, e_x), \forall x \in \mathcal{S}, \tag{4.2}$$

where sim is a similarity function. Based on the computed scores, the k-most similar examples from $\mathcal{S}$ are retrieved and included as demonstrations in the in-context prompt. This dynamic approach ensures that the examples are relevant and representative, improving the model's performance and generalization capabilities.

4.3 In-Context Prompt

The in-context prompt employed in Dynamic Few-Shot Learning (DFSL) comprises three distinct parts:

- **Task Description**: This section instructs the Language Model (LLM) using a numbered list of guidelines. These guidelines provide detailed instructions on the output format and the available information. This part is highlighted in Figure 4.3 with a blue block.
- **Retrieved Demonstrations**: Highlighted in Figure 4.3 with a green block, this section includes the top-k retrieved demonstrations. Each demonstration consists of a question, its entities and relations, denoted as *gold* entities/relations, all paired with their SPARQL query delimited by `<SPARQL></SPARQL>` tags. Each individual example within this section is separated by the `###` symbol.
- **Input Question**: This final section presents the input question along with its associated gold entities and relations. This part is highlighted in Figure 4.3 with a red block.

The answer generated by the LLM, prompted with the above structure, is

Figure 4.3: DFSL Prompt

parsed to extract the text enclosed within the <SPARQL></SPARQL> tags. The resulting query, denoted as s_q is executed by a SPARQL engine on $\mathcal{G}$ to yield the answer to q. We call our approach Dynamic Few-Shot Learning (DFSL).

4.4 Multi-Query Generation

One of the primary challenges faced by Large Language Models (LLMs) in SPARQL query generation is the identification of the subject and object in a relation, a problem known as the triple-flip error [6]. This error arises when

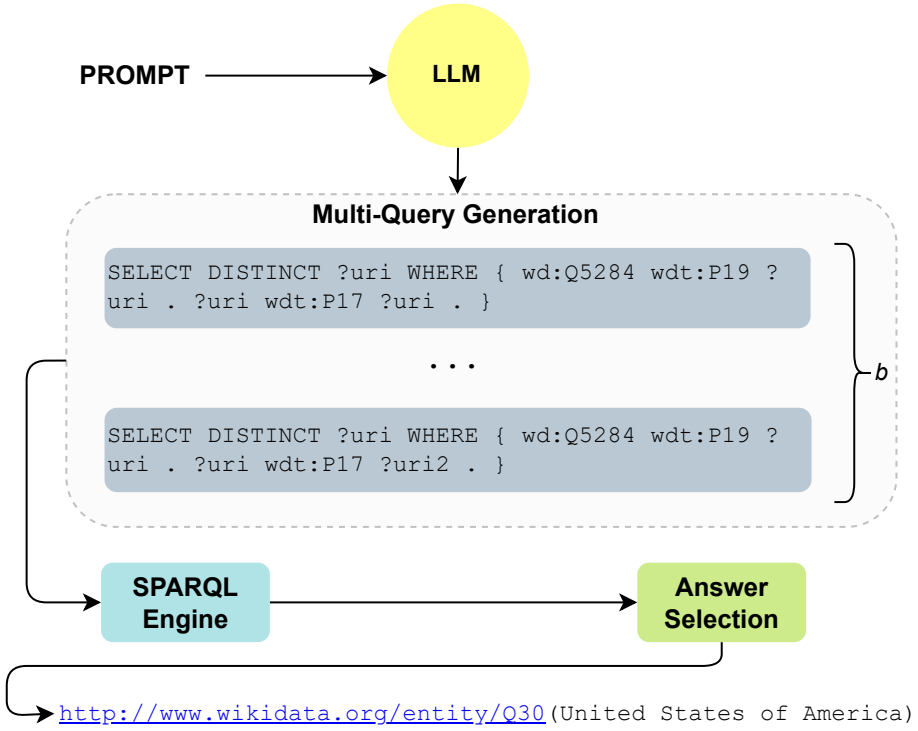

Figure 4.4: Dynamic Few Shot Learning Multi-Query

LLMs incorrectly swap the subject with the object in the query, often seemingly at random. Addressing this issue without resorting to fine-tuning is challenging. One potential solution is to extract a relevant subgraph based on the given question; however, this method has significant drawbacks. It is inefficient for multi-hop queries due to the large volume of facts that need to be extracted, and it can result in exceedingly large prompts.

The Dynamic Few-Shot Learning (DFSL) methodology offers a way to mitigate this issue by including similar examples in the in-context prompt, which helps clarify subject-object roles. To further reduce triple-flip errors, the generation of multiple SPARQL queries is proposed, as depicted in Figure 4.4. This can be achieved either by explicitly instructing the model in the prompt to produce multiple queries that swap the subject and object in different queries (DFSL-MQP), or by retaining all final hypotheses generated during beam search. The uncertainty of the model in correctly identifying subjects and objects is likely reflected in the beam search process, thus considering

both possible triple-ordering hypotheses plausible.

Thus, instead of just returning the most probable sequence s according to Equation 4.1, all b queries $\{s_{q,1}, \ldots, s_{q,b}\}$ formulated by beam search are retained. This approach, referred to as DFSL-MQ, denote such a multi-query extension of DFSL.

4.4.1 Query Selection

Executing multiple queries inevitably results in multiple possible answers. Therefore, it is essential to define an answer selection criterion. Two heuristics were designed for this purpose: Largest Set (LS) and First Set (FS).

The LS heuristic executes all the b queries, obtaining with each query $s_{q,j}$ a (possibly empty) answer set $\mathcal{A}_j$. LS then selects the largest answer set among $\{\mathcal{A}_1, \ldots, \mathcal{A}_b\}$, formally defined as:

$$\mathcal{A} = \arg\max_{\mathcal{A}_i}(|\mathcal{A}_1|, \ldots, |\mathcal{A}_b|), \tag{4.3}$$

In cases of ties, the first largest set is chosen. The rationale behind LS is that incorrect candidates will likely produce empty results. However, LS can be misled by under-constrained queries that return many irrelevant instances. The FS heuristic, on the other hand, adheres to the natural ordering of the beams by selecting the first query that yields a non-empty answer set. This method leverages the inherent prioritization in the beam search process to identify the most relevant answers efficiently.

Chapter 5

Experiments and Evaluation

This section studies the effects of each component involved in the DFSL approach. DFSL and its extension DFSL-MQ are evaluated on four KGQA datasets. The investigation considers different backbones and compares them with multiple baselines and state-of-the-art solutions.

5.1 Datasets

To assess the flexibility and robustness of the approach, it was evaluated on four heterogeneous KGQA benchmark datasets based on two different Knowledge Graphs: Wikidata and DBpedia.

QALD-9 is a dataset from the Question Answering over Linked Data (QALD) challenge series. It comprises 408 training questions and 150 test questions. Unlike other KGQA benchmarks, the SPARQL queries are intended for a DBpedia Knowledge Graph, hence it is referred to as QALD-9 DB.

QALD-9 plus extends QALD-9 with new languages and transfers SPARQL queries from DBpedia to Wikidata. Despite some queries not being portable to Wikidata due to the absence of corresponding information, it still comprises 371 training questions and 136 test questions. Only English questions are considered in the experiments.

QALD-10 is the latest dataset in the QALD series, designed to increase

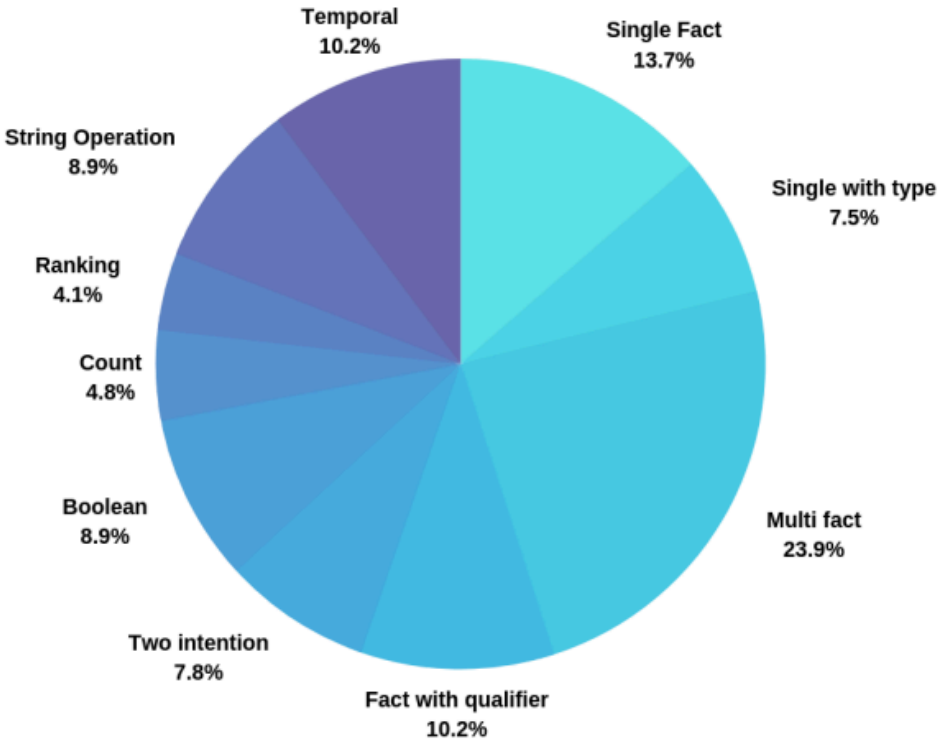

Figure 5.1: LC-QUAD 2.0 questions distribution

Modifier	Q10-WD Test	Q9-Plus-DB Train	Q9-Plus-DB Test	Q9-Plus-WD Train	Q9-Plus-WD Test
COUNT*	126	57	33	32	18
LIMIT	17	39	11	43	12
ORDER BY	17	36	11	43	12
FILTER	74	31	17	30	13
ASK	60	37	4	36	3
UNION	5	29	17	10	6
OFFSET	3	1	0	2	0
GROUP BY	95	19	11	12	2
HAVING*	1	3	2	2	1
YEAR*	43	6	10	20	4
NOW*	1	3	2	1	1

Table 5.1: QALD Datasets statistics

the complexity of gold SPARQL queries. It consists of 412 training questions extracted from QALD-9 plus Wikidata. The test set, created from scratch, includes 394 test questions that express real-world information needs. Test questions significantly differ from those in the training set.

Table 5.1 provides a detailed statistical overview of the QALD datasets, highlighting the distribution of various SPARQL query modifiers across training and test sets.

LC-QUAD 2.0 is a large-scale dataset grounded in Wikidata. It consists of 30,226 simple and complex questions: 24,180 in the training set, and 6,046 in the test set. Questions are diverse, including single-fact and multi-fact, boolean, count, and other query types, the distribution of the queries in the dataset is shown in the diagram in Figure 5.1. LC-QuAD 2.0 allows for the evaluation of DFSL performance against a large text-to-SPARQL dataset.

5.2 Backbones

The DFSL architecture is built upon three large language models: Mixstral 8x7B, LLaMA-3 70B, and CodeLLaMA 70B.

Mixstral 8x7B [18] is based on the Sparse Mixture of Experts (SMoE) architecture. It utilizes eight ”experts” with only two experts activated per token, significantly reducing computational load. This model contains 46.7 billion parameters, making it the smallest backbone used in this work. Due to its SMoE architecture, fewer than 13 billion parameters are active at each inference step, enhancing its efficiency.

LLaMA-3 70B is an LLM developed on the LLaMA architecture. It has been trained on 15 trillion tokens, representing a 650% increase from its predecessor, LLaMA 2 [19]. The vocabulary of LLaMA-3 70B has expanded to 128,256 tokens, up from 32,000 in LLaMA 2. Additionally, the training data includes four times more coding data compared to LLaMA 2, improving its capabilities. As of this writing, LLaMA-3 70B is one of the best-performing open-weights LLMs available.

CodeLLaMA 70B [49] is initialized from LLaMA-2 70B and is a specialized version fine-tuned on 1 trillion tokens of code-heavy data. This specialization makes CodeLLaMA particularly suitable for SPARQL query generation

5.3 Baselines

To evaluate the architecture, several baselines were implemented, allowing an analysis of the impact of the dynamic retrieval of examples versus simple and straightforward solutions for this task (details on the prompts used for each baseline are provided in Appendix).

- **Plain Questions**: The first baseline assesses the model's inherent memory stored in its parameters. In this approach, the LLM is provided with a task description, minor formatting guidelines, and the question q, and it is required to generate a SPARQL query. No in-context examples, entities, or relations associated with q are provided. This baseline aims to evaluate the LLM's general SPARQL query generation capabilities without additional contextual information.

- **Zero-Shot Learning**: The second baseline investigates the effect of excluding dynamically retrieved examples, providing only the question q along with its gold entities and relations. The DFSL prompt, as illustrated in Figure 4.3, remains unchanged except for the removal of the green block containing the demonstrations and minor adjustments in the guidelines within the Instruction Block to account for the absence of examples.

- **Few-Shot Learning**: Following the zero-shot approach, a few-shot learning method is tested. This involves feeding a single set of k manually selected examples, which are used for all questions in the test set.

These examples are chosen based on the dataset to maximize diversity and cover various types of queries.

- **Multi Query Prompting (DFSL-MQP)**: The final baseline provides an alternative to the multi-query generation method (DFSL-MQ) based on beam search. This naive multi-query prompting strategy entails asking the model to generate multiple queries to answer the question. To facilitate the creation of inverted subject-object queries and address triple-flip errors, the prompt explicitly requests the model to produce such SPARQL queries. Answer selection employs LS and FS heuristics, similar to DFSL-MQ.

5.4 Experimental Setup

5.4.1 Implementation

The implementation of the proposed method leverages the advanced language understanding capabilities of LLMs for the English language. Consequently, the experiments are restricted to English questions. Each dataset's training set functions as a storage for retrieving the k most similar examples using DFSL. The specific tuning of k is discussed in the subsequent section. This approach, while currently dataset-specific, can be generalized for non-dataset-specific storage, thereby enhancing both efficiency and flexibility.

Examples, which consist of questions paired with their corresponding gold entities and gold relations, are encoded using a sentence transformer model, all-mpnet-base-v2 [1]. The similarity score is computed using the cosine similarity function:

$$\text{cosine similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

[1] `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`

The advantages of utilizing the cosine similarity function include:

- **Scale Invariance**: The function is independent of the magnitude of the vectors, thus accommodating richer and higher-dimensional embeddings without issues.
- **Efficiency**: Cosine similarity computation is computationally efficient as it primarily involves dot products and vector norms.

Inference is conducted using beam search in two configurations: in DFSL with a beam size b of 3, and in DFSL-MQ with a beam size of 10. All experiments were executed on a cluster with four NVIDIA A100 GPUs.

5.4.2 Number of Few-shot Examples

An initial experiment was conducted to analyze the effect of the k parameter, which represents the number of few-shot examples, prior to exploring different architectural variants. To balance context length and computational efficiency, k was restricted to the values $1, 3, 5, 7$, and the DFSL approach with the LLaMA 3 70B backbone was evaluated across the four datasets.

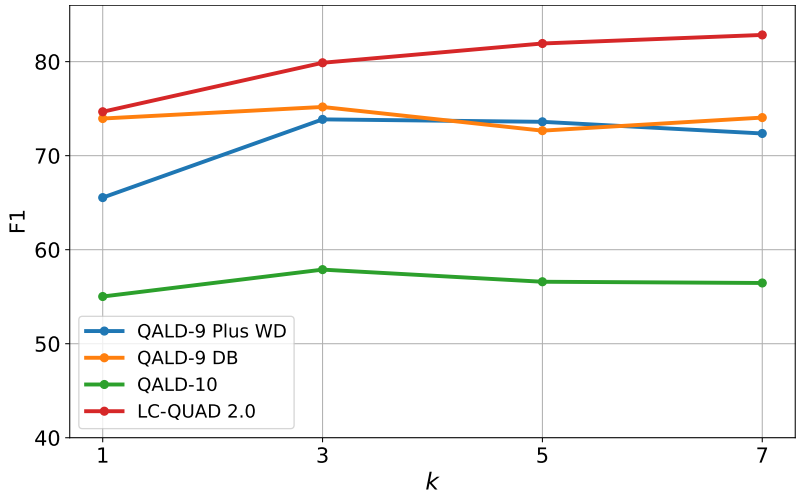

Figure 5.2: Impact of the number of in-context examples on the four benchmarks.

The results in Figure 5.2 indicate that values of k greater than one perform comparably well on smaller benchmarks. However, for LC-QUAD 2.0, which

contains approximately 25,000 examples as storage, increasing k appears to be beneficial. This improvement can be attributed to the higher likelihood of finding similar examples in larger datasets as k increases. Consequently, $k = 5$ was chosen for all subsequent experiments as it provides a balanced trade-off across all datasets.

5.4.3 Prompt

The default template for the prompt used in our experiments is depicted in Figure 4.3. The golden entities and relations are represented in their full Internationalized Resource Identifier (IRI) form rather than a contracted form. For example, the city of *Munich* is represented as *http://www.wikidata.org/entity/q1726* in Wikidata and *http://dbpedia.org/resource/Munich* in DBpedia. Additionally, for the Wikidata experiment prompts, both entities and relations are provided with their associated labels, such as *http://www.wikidata.org/entity/q38111 (Leonardo DiCaprio)*. This inclusion of labels helps the model better understand the context of the entities and relations involved.

However, minor modifications are necessary in certain scenarios. For instance, when conducting experiments on the DBpedia knowledge graph, the reference to Wikidata in the initial text segment (blue-block) is substituted with DBpedia. In studies examining the absence of gold information, all references to gold entities and relations are omitted from the prompt in accordance with the specific ablation. Detailed descriptions of each prompt variation used in our experiments are provided in the Appendix.

There are no differences in the prompt layout when performing few-shot learning baseline experiments. In zero-shot learning, the in-context examples and any references to them are removed, while all other aspects remain unchanged.

5.4.4 Evaluation metric

In our experiment, we utilize the standard F1 score to evaluate the accuracy between the ground truth answer and the answer derived from the predicted SPARQL queries. Assume $\mathcal{A}_p$ is the predicted answer and $\mathcal{A}_q$ is the ground truth. Let $\mathcal{P}$ represent the precision, defined as the ratio of the number of correct results obtained from the predicted query to the total number of results. Similarly, let $\mathcal{R}$ represent the recall, defined as the ratio of the number of correct results obtained from the predicted query to the total number of correct results in the ground truth query.

The formal mathematical definition of the F1 score is the harmonic mean of precision and recall, expressed as:

$$F1 = 2 \cdot \frac{\mathrm{P} \cdot \mathrm{R}}{\mathrm{P} + \mathrm{R}} \tag{5.1}$$

When both the predicted and ground truth queries return an empty set, an F1 score of 1 is assigned. The overall F1 score for the test set is calculated as the average F1 score for each example in the test set.

5.5 Results

A comprehensive evaluation was conducted to assess the performance of the proposed approach in various contexts. Initially, the impact of the primary approach was compared against baseline methods. Subsequently, the differences between the two architectures, DFSL and DFSL-MQ, were analyzed, including a comparison with the Multi-Query Prompt-based baseline. Finally, the results were compared with state-of-the-art approaches to establish the effectiveness of the proposed methods.

5.5.1 Impact of Dynamic Examples

To evaluate the impact of dynamically retrieving few-shot examples, a comparison was conducted between Dynamic Few-Shot Learning (DFSL) on various backbones against Zero-Shot and Few-Shot Learning baselines. The results are detailed in Table 5.2 for Wikidata and in Table 5.3 for DBpedia.

Regarding backbones, Llama 3 consistently surpasses Mixtral and CodeLlama in the zero-shot learning scenario. In the few-shot learning context, the performance of Llama 3 and CodeLlama is generally comparable. The strong zero-shot performance of Llama 3 may be attributed to data contamination; however, further investigation is required to confirm this hypothesis.

Approach	Backbone	QALD-9 Plus	QALD-10	LC-QUAD 2.0
Zero-shot	Mixtral 7x8	49.90	33.76	40.66
Few-shot		54.80 (+**4.90**)	50.26 (+**16.50**)	61.04 (+**20.38**)
DFSL		71.75 (+**21.85**)	49.90 (+**16.14**)	81.81 (+**41.15**)
Zero-shot	Llama-3 70B	63.01	**58.31**	54.21
Few-shot		67.69 (+**4.68**)	51.28 (**-7.03**)	68.52 (+**14.31**)
DFSL		73.60 (+**10.59**)	56.59 (**-1.72**)	81.93 (+**27.72**)
Zero-shot	CodeLlama 70B	45.94	33.36	38.40
Few-shot		64.49 (+**18.55**)	57.38 (+**24.02**)	64.46 (+**26.06**)
DFSL		**76.59** (+**30.65**)	57.69 (+**24.33**)	**85.45** (+**47.05**)

Table 5.2: Comparison between zero-shot, few-shot and DFSL with different backbones on Wikidata Datasets. Absolute F1 gains with respect to the naive zero-shot approach are reported between parenthesis.

Both few-shot learning and DFSL generally demonstrate substantial improvements over the zero-shot baseline across all backbones and datasets. An exception is observed with Llama 3 on the QALD-10 dataset. Notably, DFSL exhibits significant enhancements in F1 scores compared to the few-shot learning baseline, particularly in the LC-QUAD 2.0, QALD-9 Plus, and QALD-9 DB datasets, with F1 score increases reaching up to 21 absolute points.

In contrast, for QALD-10, where the test set distribution differs from the

Approach	Backbone	QALD-9 DB
Zero-shot	Mixtral 7x8	65.73
Few-shot		63.86 (**-1.87**)
DFSL		72.74 (+**7.01**)
Zero-shot	Llama-3 70B	70.49
Few-shot		68.84 (**-1.65**)
DFSL		72.66 (+**2.17**)
Zero-shot	CodeLlama 70B	66.43
Few-shot		72.67 (+**6.24**)
DFSL		ì**75.14** (+**8.71**)

Table 5.3: Comparison between zero-shot, few-shot and DFSL with different backbones on DBpedia Dataset. Absolute F1 gains with respect to the naive zero-shot approach are reported between parenthesis.

training set, there are no significant differences between DFSL and the standard few-shot learning approach. This suggests that DFSL provides limited benefits when the storage contains unrelated examples.

Overall, DFSL with CodeLlama achieved the highest performance across all configurations. Consequently, CodeLlama is selected as the backbone for subsequent DFSL experiments.

5.5.2 Impact of Multi-Query Generation

This section examines the performance of DFSL-MQ, the multi-query extension of DFSL. Both answer selection strategies, Largest Set (LS) and First Set (FS), were assessed and compared against the standard DFSL and the multi-query prompting baseline described in Section 5.3. The comprehensive results are presented in Table 5.4.

The analysis reveals that the advantage of utilizing multiple queries is not guaranteed. Specifically, the multi-query prompting baseline exhibited inferior performance in three out of four datasets compared to the single-query DFSL, regardless of the answer selection strategy implemented. In contrast, DFSL-MQ generally provided performance improvements. Both LS and FS heuristics were effective when hypotheses were generated from the beams.

Approach	QALD-9 Plus	QALD-10	LC-QUAD 2.0	QALD-9 DB
DFSL	76.59	57.69	85.45	75.14
DFSL-MQP$_{LS}$	73.67	58.85	85.06	73.25
DFSL-MQP$_{FS}$	74.40	58.34	85.38	73.92
DFSL-MQ$_{LS}$	83.21	60.48	85.54	72.06
DFSL-MQ$_{FS}$	**84.45** (+**7.86**)	**62.20** (+**4.51**)	**89.10** (+**3.65**)	**77.89** (+**2.75**)

Table 5.4: Multi-query Generation: comparing DFSL-MQ with DFSL and Multi-query prompting baselines. Absolute F1 gains with respect to DFSL are reported for the best performing configuration.

Furthermore, FS consistently surpassed LS, showing notable improvements in the QALD-9 DB dataset.

5.5.3 In-context Learning vs Fine-tuning

Prior sections have focused on evaluating DFSL within the context of In-Context Learning approaches. In Table 5.5 and in Table 5.6, however, the performance of DFSL is compared against state-of-the-art models that have been specifically trained and fine-tuned for downstream KGQA tasks.

Approach	QALD-9 Plus	QALD-10	LC-QUAD 2.0
Plain Question	0.08	0.02	12.00
BART [5]	-	-	64.00
PGN-BERT-BERT [5]	-	-	86.00
SGPT [7]	-	-	89.04
TSET-small [6]	72.86	47.15	94.00
TSET-base [6]	75.85	51.37	**95.00**
Zero-shot Learning	45.94	33.36	38.40
Few-shot Learning	64.49	57.38	64.46
DFSL	76.59	57.69	85.45
DFSL-MQ beam FS	**84.45** (+**8.60**)	**62.20** (+**10.83**)	89.10 (-**5.90**)

Table 5.5: DFSL and ICL approaches vs state-of-the-art fine-tuned models on Wikidata Benchmark.

Remarkably, without any training, DFSL-MQ surpasses current state-of-the-art methods in three out of four benchmarks: QALD-9 Plus, QALD-10, and QALD-9 DB, even in the single-query DFSL configuration.

Approach	QALD-9 DB
Plain Question	16.42
SGPT [7]	67.82
Zero-shot Learning	66.43
Few-shot Learning	72.67
DFSL	75.14
DFSL-MQ beam FS	**77.89** (+**10.07**)

Table 5.6: DFSL and ICL approaches vs state-of-the-art fine-tuned models on DBpedia Benchmark.

However, DFSL-MQ does not achieve state-of-the-art results in LC-QUAD 2.0, the dataset most susceptible to triple-flip errors. This indicates that while multi-query generation mitigates this issue to some extent, it does not completely resolve the problem.

5.6 Qualitative Analysis

The qualitative analysis of the results encompasses several key aspects. Firstly, it demonstrates how errors made by traditional few-shot learning are addressed by the DFSL approach. Following this, a comparative qualitative analysis between the two heuristics employed is presented. Lastly, the analysis illustrates how the top-performing DFSL-MQ approach resolves triple flip errors committed by the DFSL method.

5.6.1 DFSL vs Few-Shot Learning

The qualitative comparison between Dynamic Few-Shot Learning (DFSL) and standard Few-Shot Learning in Table 5.7, Table 5.8, and Table 5.9 highlights the benefits of DFSL in mitigating errors through the use of similar in-context examples.

In Table 5.7, the traditional Few-Shot Learning approach misplaces entities and fails to capture the complete relationship, resulting in an inaccurate

Question	`Who is the daughter of Robert Kennedy married to?`
Target Query	`SELECT DISTINCT ?uri WHERE { wd:Q25310 wdt:P40 ?daughter . ?daughter wdt:P21 wd:Q6581072 . ?daughter wdt:P26 ?uri .}`
Few-Shot	`SELECT DISTINCT ?uri WHERE { ?uri wdt:P40 wd:Q25310 ; wdt:P21 wd:Q6581072 ; wdt:P26 ?spouse .}`
DFSL	`SELECT DISTINCT ?uri WHERE { wd:Q25310 wdt:P40 ?child . ?child wdt:P21 wd:Q6581072 . ?child wdt:P26 ?uri . }`
Similar IC Example	`SELECT DISTINCT ?uri WHERE { wd:Q43247 wdt:P40 ?child . ?child wdt:P26 ?uri . }`

Table 5.7: A qualitative comparison between DFSL and Few-shot Learning for the query "Who is the daughter of Robert Kennedy married to?".

query. DFSL, by leveraging a similar example, correctly identifies the hierarchical relationship between Robert Kennedy, his daughter, and her spouse, thus constructing a precise query. For the query in Table 5.8, Few-Shot

Question	`Which countries are connected by the Rhine?`
Target Query	`SELECT DISTINCT ?uri WHERE { wd:Q584 wdt:P17 ?uri . ?uri wdt:P31 wd:Q6256 .}`
Few-Shot	`SELECT DISTINCT ?uri WHERE { ?uri wdt:P31 wd:Q6256 ; wdt:P17 wd:Q584 .}`
DFSL	`SELECT DISTINCT ?uri WHERE { wd:Q584 wdt:P17 ?uri . ?uri wdt:P31 wd:Q6256 . }`
Similar IC Example	`SELECT DISTINCT ?res WHERE { wd:Q3392 wdt:P885/wdt:P17 ?res . }`

Table 5.8: A qualitative comparison between DFSL and Few-shot Learning for the query "Which countries are connected by the Rhine?".

Learning misplaces the entity representing the Rhine (`wd:Q584`), leading to an incorrect query. DFSL, however, retrieves a relevant example involving the triple `wd:Q3392 wdt:P885/wdt:P17 ?res` from a similar example, correctly positioning the Rhine entity. In Table 5.9 the Few-Shot approach incorrectly places the ?capital variable, resulting in an inaccurate query that doesn't list capitals correctly. DFSL, by using a similar example, accurately identifies the relationship between African countries and their capitals, positioning ?uri, the variable associated to the capitals, in the correct position.

Question	Give me the capitals of all countries in Africa.
Target Query	SELECT DISTINCT ?uri WHERE { ?country wdt:P31 wd:Q6256 . ?country wdt:P30 wd:Q15 . ?country wdt:P36 ?uri .}
Few-Shot	SELECT DISTINCT ?uri WHERE { ?uri wdt:P31 wd:Q6256 ; wdt:P30 wd:Q15 ; wdt:P36 ?capital .}
DFSL	SELECT DISTINCT ?uri WHERE { ?country wdt:P31 wd:Q6256 ; wdt:P30 wd:Q15 . ?country wdt:P36 ?uri .}
Similar IC Example	SELECT DISTINCT ?uri WHERE { wd:Q5451 wdt:P17 ?country . ?country wdt:P36 ?uri . }

Table 5.9: A qualitative comparison between DFSL and Few-shot Learning for the query ”Give me the capitals of all countries in Africa.”.

5.6.2 First Set vs Large Set heuristics

Table 5.10 showcase errors caused by employing LS answer selection heuristic. Notably, by choosing larger sets,are choosen queries that are often relegated to latter positions in the beam hypotheses, which tend to be more general, thus more prone to returning a greater number of results. On the other hand FS return queries placed in the first positions which are the most likely to be correct.

Question	Target Query	Heuristic	Predicted Query	Beam
What is manufactured NEC PC-9800 series whose sector is electronics?	SELECT ?answer WHERE { wd:Q183505 wdt:P176 ?answer . ?answer wdt:P452 wd:Q11650 }	LS	SELECT ?answer WHERE { wd:Q183505 wdt:P176 ?X . ?X wdt:P452 wd:Q11650 . ?X wdt:P31 ?answer}	7
		FS	select distinct ?obj where { wd:Q183505 wdt:P176 ?obj . ?obj wdt:P452 wd:Q11650 }	4
What is the enthalpy of vaporization for potassium hydroxide?	select distinct ?answer where { wd:Q132298 wdt:P2116 ?answer}	LS	ASK WHERE { wd:Q14982 wdt:P2116 ?obj filter(?obj = 37400) }	3
		FS	select distinct ?answer where { wd:Q132298 wdt:P2116 ?answer}	1
Which is the rural city of Fukushim?	SELECT ?answer WHERE { wd:Q161176 wdt:P131 ?answer . ?answer wdt:P150 wd:Q1347240}	LS	SELECT ?answer WHERE { wd:Q161176 wdt:P131 ?X . ?X wdt:P150 ?answer}	3
		FS	SELECT ?answer WHERE { wd:Q161176 wdt:P131 ?answer . ?answer wdt:P150 wd:Q1347240}	1

Table 5.10: Qualitative comparison between different answer selection strategies in DFSL-MQ.

5.6.3 DFSL vs DFSL-MQ

Table 5.11, 5.12 and 5.13 present examples of triple flip errors with the DFSL architecture. The DFSL architecture returns a single query, the one which is the most likely to be correct according to the model but also affected by triple flip error. In contrast, the DFSL-MultiQuery (DFSL-MQ) variant returns multiple queries based on the hypotheses generated by beam search and thus exploring a range of potential answers and choose the most appropriate one. In

Question	`Who is the enclave within of Montreal?`
Target Query	`select distinct ?sbj where { ?sbj wdt:P501 wd:Q340 . ?sbj wdt:P31 wd:Q171441 }`
DFSL	`select distinct ?obj where { wd:Q340 wdt:P501 ?obj . ?obj wdt:P31 wd:Q171441 }`
DFSL-MQ	`select distinct ?sbj where { ?sbj wdt:P501 wd:Q340 . ?sbj wdt:P31 wd:Q171441 }`

Table 5.11: Some triple-flip errors that are solved by DFSL-MQ for the query ”Who is the enclave within of Montreal?”.

Table 5.11, the DFSL model incorrectly places the entity `wd:Q340` (Montreal) as the subject of the triple with the relation `wdt:P501` (enclave within). Table

Question	`The trachea is of what anatomical branch?`
Target Query	`select distinct ?answer where { ?answer wdt:P3261 wd:Q175449}`
DFSL	`select distinct ?answer where { wd:Q175449 wdt:P3261 ?answer}`
DFSL-MQ	`select distinct ?answer where { ?answer wdt:P3261 wd:Q175449}`

Table 5.12: Some triple-flip errors that are solved by DFSL-MQ for the query ”The trachea is of what anatomical branch?”.

5.12 shows an instance of a 1-hop query where the DFSL model erroneously places the entity `wd:Q175449`. Finally, in Table 5.13, the DFSL model incorrectly positions the entity `wd:Q34266` (Russian Empire) as the object of the triple. The DFSL-MQ approach, leveraging the beam search hypothesis, correctly places the entity.

Question	`What revolution caused the destruction of the Russian Empire?`
Target Query	`select distinct ?obj where { wd:Q34266 wdt:P770 ?obj . ?obj wdt:P31 wd:Q10931 }`
DFSL	`select distinct ?sbj where { ?sbj wdt:P770 wd:Q34266 . ?sbj wdt:P31 wd:Q10931 }`
DFSL-MQ	`select distinct ?obj where { wd:Q34266 wdt:P770 ?obj . ?obj wdt:P31 wd:Q10931 }`

Table 5.13: Some triple-flip errors that are solved by DFSL-MQ for the query ”What revolution caused the destruction of the Russian Empire?”.

5.7 Ablation studies

5.7.1 Different Example Encoding

As described in Section 4.2, the embeddings were computed by concatenating the textual input, which consists of the question along with its corresponding list of gold entities and relations, and utilizing a Sentence Transformer model to obtain the relative embeddings. This section examines the impact of incorporating this additional information compared to a baseline where only the question is used to generate embeddings.

In Figure 5.3, the performance of the DFSL, represented by orange bars, is compared with the baseline approach, represented by blue bars, across all datasets and model backbones. The results demonstrate that, with the exception of the LLaMA 3 70B model on the QALD9 DB dataset, the DFSL embedding strategy augmented with gold entities and relations consistently yields higher performance than the baseline.

The enhanced performance can be attributed to the model’s ability to retrieve examples not solely based on semantically similar questions, but also those that contain similar entities and relations. This leads to the inclusion of SPARQL query examples in the prompt that closely align with the test set’s golden SPARQL query, thereby improving the model’s overall efficacy.

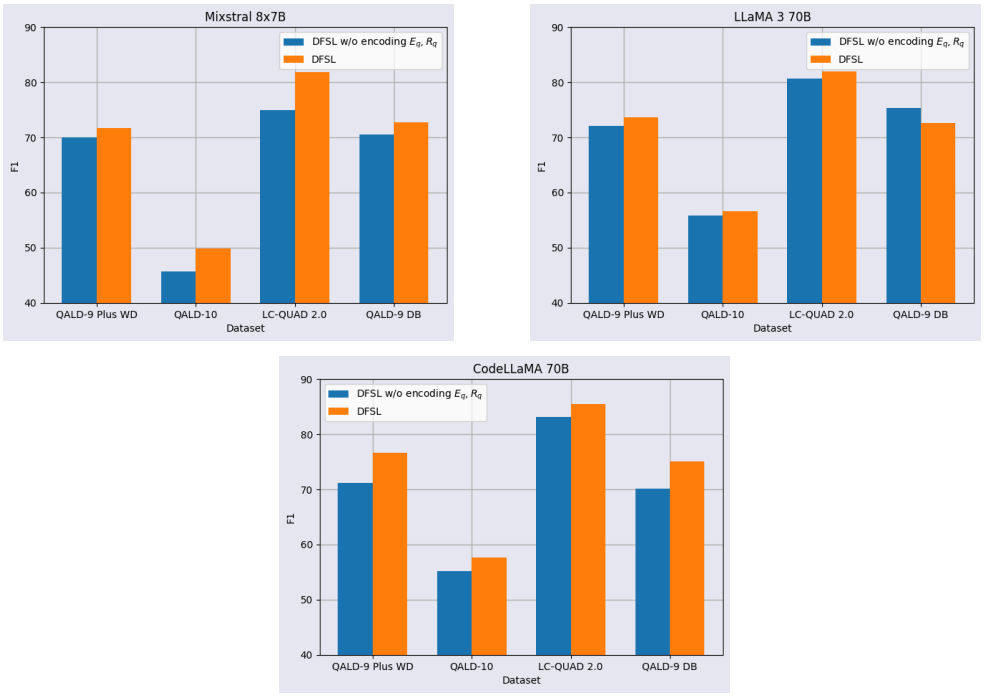

Figure 5.3: Comparison of Embeddings: DFSL (in orange) encoding that incorporates question, entities and relations versus an embedding solely based on the question q (in blue).

5.7.2 Absence of gold Information

In knowledge graph question answering (KGQA), the generation of SPARQL queries typically relies not only on the question itself but also on the associated entities and relations. This information is often pre-identified, as SPARQL query generation is usually the final step in a pipeline where entities and relations are extracted in earlier stages. However, this information may not always be available. Therefore, it is crucial to evaluate the approach in scenarios where some or all of this information is missing. Specifically, the model is assessed by omitting either the entities ($\mathcal{E}_q$) or the relations ($\mathcal{R}_q$), and also when both are completely absent. The absence of information is maintained throughout the entire process. For instance, when entities are removed, they are excluded from both storage and the prompt. The embeddings used for retrieval are computed by encoding an input without any entities concatenated in $\boldsymbol{q}$, resulting in $\boldsymbol{q} = [q, \mathcal{R}_q]$. he impact of this absence of information is evalu-

Approach	**QALD-9 DB**
Plain Question	16.42
DFSL	75.14
DFSL w/o $\mathcal{R}_q$	56.62 (-18.56)
DFSL w/o $\mathcal{E}_q$	60.92 (-14.22)
DFSL w/o $\mathcal{E}_q, \mathcal{R}_q$	49.59 (-25.55)

Table 5.14: DFSL in the absence of entities and/or relations.

ated on the QALD-9 DB dataset. The results, as presented in Table 5.14, indicate a significant drop in performance of the LLM when the complete knowledge of the entities and relations required for generating the query is missing. Nevertheless, even in scenarios where no such information is provided (DFSL without $\mathcal{E}_q$ and $\mathcal{R}_q$), the incorporation of dynamic demonstrations remains crucial. This approach results in an absolute F1 score increase of over 33 points compared to the baseline where only the question is provided.

Chapter 6

Conclusion and Future work

This thesis presents Dynamic Few-Shot Learning (DFSL), a novel approach to Knowledge Graph Question Answering (KGQA). The method focuses on leveraging large language models (LLMs) to generate SPARQL queries, which are subsequently executed on prominent knowledge graphs such as Wikidata and DBpedia. The core innovation of DFSL lies in its strategy of retrieving relevant examples from the training dataset using semantic search within a dense vector space. These examples enrich the final prompt provided to the LLM, resulting in a significant improvement in the quality of the generated SPARQL queries.

Extensive evaluations were conducted on various datasets that differ in query complexity, size, and underlying knowledge graphs. Four publicly available datasets were utilized, with three based on Wikidata and one on DBpedia. To assess the effectiveness of the approach, three different LLMs were employed: LLaMA 3 70B, CodeLLaMA 2, and Mistral 8x7B. The results indicate that DFSL consistently outperforms both standard in-context learning techniques and state-of-the-art models fine-tuned for the downstream task across almost all datasets, with only one exception.

Additionally, a variant of DFSL, namely DFSL-Multi Query (DFSL-MQ), was proposed. This variant leverages the hypotheses generated by beam search to return multiple answers to a question, thereby addressing a known issue in

KGQA, the triple-flip error. DFSL-MQ achieves the highest possible performance, demonstrating a significant positive gap compared to current state-of-the-art approaches.

Further, an in-depth evaluation of DFSL was performed through ablation studies. These studies examined the impact of various hyperparameters, including the number of top-k examples retrieved, different backbone models, embedding methods, answer selection strategies, and the inclusion or exclusion of entity and relation information associated with a question. The findings underscore the robustness and versatility of DFSL, highlighting its potential as a superior method for KGQA tasks.

Future Work

This research represents a significant milestone in the task of Knowledge Graph Question Answering (KGQA) through the use of Language Model (LLM) prompting. However, several avenues for future work can further enhance the system's performance and reliability:

Experiment with Different Languages: The experiments conducted in this study were based solely on English-language questions, where LLMs are known to perform optimally. To validate the robustness and generalizability of the proposed methodology, it is essential to evaluate the approach using datasets available in multiple languages. This will help to ensure the system's applicability and performance across diverse linguistic contexts.

Addressing Data Contamination: Given the extensive pre-training of LLMs on vast portions of the web, there is a risk of unintended data contamination. Future research should investigate mechanisms to identify and mitigate the impact of such contamination on model performance. This could involve developing techniques to detect and handle instances where the model's training

data overlaps with the evaluation datasets.

Exploring Smaller Models: The current study focused exclusively on LLMs with a large number of parameters, neglecting the potential of smaller models. Future investigations should explore the behavior and performance of smaller models, which could offer benefits in terms of computational efficiency and resource utilization without significantly compromising accuracy.

Diverse Example Encoding Strategies: In this research, the investigation into example encoding was limited to the type of text encoded (i.e., only the question or the question along with its entities and relations). Future work should explore various embedding models, similarity criteria, and input concatenation strategies. This could include experimenting with different embedding techniques, optimizing similarity measures, and testing alternative ways of structuring the input to the LLM.

Bibliography

[1] M. Yahya, J. G. Breslin, and M. I. Ali. Semantic web and knowledge graphs for industry 4.0. *Applied Sciences*, 11(11), 2021. ISSN: 2076-3417. URL: https://www.mdpi.com/2076-3417/11/11/5110.

[2] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher. From freebase to wikidata: the great migration. In *Proceedings of the 25th international conference on world wide web*, pages 1419–1428, 2016.

[3] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, and C. Bizer. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6, January 2014. DOI: 10.3233/SW-140134.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 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, Vancouver, Canada. Association for Computing Machinery, 2008. ISBN: 9781605581026. DOI: 10.1145/1376616.1376746. URL: https://doi.org/10.1145/1376616.1376746.

[5] D. Banerjee, P. A. Nair, J. N. Kaur, R. Usbeck, and C. Biemann. 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, July 2022. DOI: 10.1145/

3477495.3531841. URL: http://dx.doi.org/10.1145/3477495.3531841.

[6] J. Qi, C. Su, Z. Guo, L. Wu, Z. Shen, L. Fu, X. Wang, and C. Zhou. Enhancing sparql query generation for knowledge base question answering systems by learning to correct triplets. *Applied Sciences*, 14(4), 2024. ISSN: 2076-3417. URL: https://www.mdpi.com/2076-3417/14/4/1521.

[7] M. R. A. H. Rony, U. Kumar, R. Teucher, L. Kovriguina, and J. Lehmann. Sgpt: a generative approach for sparql query generation from natural language questions. *IEEE Access*, 10:70712–70723, 2022. DOI: 10.1109/ACCESS.2022.3188714.

[8] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018. DOI: 10.1109/MCI.2018.2840738.

[9] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. DOI: 10.1109/5.18626.

[10] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc., 2001. ISBN: 1558607781.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In 1986. URL: https://api.semanticscholar.org/CorpusID:62245742.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, December 1997. DOI: 10.1162/neco.1997.9.8.1735.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. arXiv: 1412.3555 [cs.NE]. URL: https://arxiv.org/abs/1412.3555.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: pre-training of deep bidirectional transformers for language understanding, 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[16] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.

[17] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL: http://jmlr.org/papers/v21/20-074.html.

[18] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, G.

Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mixtral of experts, 2024. arXiv: 2401.04088 [cs.LG].

[19] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: open foundation and fine-tuned chat models, 2023. arXiv: 2307.09288 [cs.CL]. URL: https://arxiv.org/abs/2307.09288.

[20] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: open and efficient foundation language models, 2023. arXiv: 2302.13971 [cs.CL].

[21] B. Zhang and R. Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

[22] Y. Guo, Z. Pan, and J. Heflin. Lubm: a benchmark for owl knowledge base systems. *Journal of Web Semantics*, 3(2):158–182, 2005. ISSN: 1570-8268. DOI: https://doi.org/10.1016/j.websem.2005.06.005. URL: https://www.sciencedirect.com/science/article/pii/S1570826805000132. Selcted Papers from the International Semantic Web Conference, 2004.

[23] A. Owens, N. Gibbins, and m. Schraefel. Effective benchmarking for rdf stores using synthetic data, May 2008.

[24] O. Görlitz, M. Thimm, and S. Staab. Splodge: systematic generation of sparql benchmark queries for linked open data. In P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, and E. Blomqvist, editors, *The Semantic Web – ISWC 2012*, pages 116–132, Berlin, Heidelberg. Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-35176-1.

[25] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over rdf data. *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web*, April 2012. DOI: 10.1145/2187836.2187923.

[26] K. Xu, S. Zhang, Y. Feng, and D. Zhao. Answering natural language questions via phrasal semantic parsing. In C. Zong, J.-Y. Nie, D. Zhao, and Y. Feng, editors, *Natural Language Processing and Chinese Computing*, pages 333–344, Berlin, Heidelberg. Springer Berlin Heidelberg, 2014.

[27] G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl. From keywords to semantic queries—incremental query construction on the semantic web. *Journal of Web Semantics*, 7(3):166–176, 2009. ISSN: 1570-8268. DOI: https://doi.org/10.1016/j.websem.2009.07.005. URL: https://www.sciencedirect.com/science/article/pii/S1570826809000250. The Web of Data.

[28] J. Zou, M. Yang, L. Zhang, Y. Xu, Q. Pan, F. Jiang, R. Qin, S. Wang, Y. He, S. Huang, and Z. Zhao. A chinese multi-type complex questions answering dataset over wikidata, 2021. arXiv: 2111.06086 [cs.CL].

[29] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019. arXiv: 1910.13461. URL: http://arxiv.org/abs/1910.13461.

[30] A. See, P. J. Liu, and C. D. Manning. Get to the point: summarization with pointer-generator networks, 2017. arXiv: 1704.04368 [cs.CL].

[31] D. Pliukhin, D. Radyush, L. Kovriguina, and D. Mouromtsev. Improving subgraph extraction algorithms for one-shot sparql query generation with large language models. In *Scholarly-QALD-23: Scholarly QALD Challenge at The 22nd International Semantic Web Conference (ISWC 2023)(Athens, Greece*, volume 3592, pages 1–10, 2023.

[32] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, L. Li, and Z. Sui. A survey on in-context learning, 2023. arXiv: 2301.00234 [cs.CL].

[33] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh. Calibrate before use: improving few-shot performance of language models. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR, 18–24 Jul 2021. URL: https://proceedings.mlr.press/v139/zhao21c.html.

[34] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. What makes good in-context examples for GPT-3? In E. Agirre, M. Apidianaki, and I. Vulić, editors, *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and

Online. Association for Computational Linguistics, May 2022. DOI: 10.18653/v1/2022.deelio-1.10. URL: https://aclanthology.org/2022.deelio-1.10.

[35] I. Levy, B. Bogin, and J. Berant. Diverse demonstrations improve in-context compositional generalization, 2023. arXiv: 2212.06800 [cs.CL].

[36] X. Li and X. Qiu. Finding support examples for in-context learning, 2023. arXiv: 2302.13539 [cs.CL]. URL: https://arxiv.org/abs/2302.13539.

[37] H. Gonen, S. Iyer, T. Blevins, N. A. Smith, and L. Zettlemoyer. Demystifying prompts in language models via perplexity estimation, 2022. arXiv: 2212.04037 [cs.CL].

[38] H. J. Kim, H. Cho, J. Kim, T. Kim, K. M. Yoo, and S.-g. Lee. Self-generated in-context learning: leveraging auto-regressive language models as a demonstration generator, 2022. arXiv: 2206.08082 [cs.CL].

[39] Y. Hao, Y. Sun, L. Dong, Z. Han, Y. Gu, and F. Wei. Structured prompting: scaling in-context learning to 1,000 examples, 2022. arXiv: 2212.06713 [cs.CL]. URL: https://arxiv.org/abs/2212.06713.

[40] S. Liu, H. Ye, L. Xing, and J. Zou. In-context vectors: making in context learning more effective and controllable through latent space steering, 2024. arXiv: 2311.06668 [cs.LG]. URL: https://arxiv.org/abs/2311.06668.

[41] Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp. Fantastically ordered prompts and where to find them: overcoming few-shot prompt order sensitivity. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics, May 2022. DOI: 10.18653/v1/2022.acl-long.556. URL: https://aclanthology.org/2022.acl-long.556.

[42] Y. Liu, J. Liu, X. Shi, Q. Cheng, Y. Huang, and W. Lu. Let's learn step by step: enhancing in-context learning ability with curriculum learning, 2024. arXiv: 2402.10738 [cs.CL]. URL: https://arxiv.org/abs/2402.10738.

[43] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code, 2021. arXiv: 2107.03374 [cs.LG].

[44] L. Nan, Y. Zhao, W. Zou, N. Ri, J. Tae, E. Zhang, A. Cohan, and D. Radev. Enhancing few-shot text-to-sql capabilities of large language models: a study on prompt design strategies, 2023. arXiv: 2305.12586 [cs.CL].

[45] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. arXiv: 2201.11903 [cs.CL].

[46] H. Zhang, R. Cao, L. Chen, H. Xu, and K. Yu. ACT-SQL: in-context learning for text-to-SQL with automatically-generated chain-of-thought. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532, Singapore. Association for Computational Linguistics, December 2023. DOI: 10.18653/v1/2023.findings-emnlp.227. URL: https://aclanthology.org/2023.findings-emnlp.227.

[47] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu. A survey on knowledge graphs: representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2022. DOI: 10.1109/TNNLS.2021.3070843.

[48] E. Dimitrakis, K. Sgontzos, and Y. Tzitzikas. A survey on question answering systems over linked data and documents. *Journal of Intelligent Information Systems*, 55:1–27, October 2020. DOI: 10.1007/s10844-019-00584-7.

[49] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. Bhatt, C. C. Ferrer, A. Grattafiori, W. Xiong, A. Défossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve. Code llama: open foundation models for code, 2024. arXiv: 2308.12950 [cs.CL].

Appendix A

Appendix: Prompt Examples

```
The task involves translating questions from English into SPARQL queries for the Dbpedia knowledge
graph. These queries must follow specific guidelines to ensure accuracy and correct execution:
1. Enclose SPARQL queries within <SPARQL></SPARQL> tags.
2. Utilize all provided golden entities and relations exclusively to construct the query
accurately.
3. Examples are provided below for guidance.

Examples:

###
Question: What is the timezone in San Pedro de Atacama?
Gold Entities:
http://dbpedia.org/resource/San_Pedro_de_Atacama
Gold Relations:
http://dbpedia.org/ontology/timeZone

Query:
<SPARQL>
PREFIX dbo: <http://dbpedia.org/ontology/> PREFIX res: <http://dbpedia.org/resource/> SELECT
DISTINCT ?uri WHERE { res:San_Pedro_de_Atacama dbo:timeZone ?uri }
</SPARQL>
###
Question: Which U.S. states are in the same time zone as Utah?
Gold Entities:
http://dbpedia.org/resource/Utah http://dbpedia.org/class/yago/WikicatStatesOfTheUnitedStates
http://dbpedia.org/resource/Utah
Gold Relations:
http://dbpedia.org/property/timezone http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://dbpedia.org/property/timezone

Query:
<SPARQL>
SELECT DISTINCT ?uri WHERE { <http://dbpedia.org/resource/Utah>
<http://dbpedia.org/property/timezone> ?x . ?uri a
<http://dbpedia.org/class/yago/WikicatStatesOfTheUnitedStates> ;
<http://dbpedia.org/property/timezone> ?x FILTER ( ?uri != <http://dbpedia.org/resource/Utah> ) }
</SPARQL>
###
Question: Which U.S. states are in the same timezone as Utah?
Gold Entities:
http://dbpedia.org/resource/Utah http://dbpedia.org/class/yago/WikicatStatesOfTheUnitedStates
http://dbpedia.org/resource/Utah
Gold Relations:
http://dbpedia.org/property/timezone http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://dbpedia.org/property/timezone

Query:
<SPARQL>
PREFIX yago: <http://dbpedia.org/class/yago/> PREFIX res: <http://dbpedia.org/resource/> PREFIX
dbp: <http://dbpedia.org/property/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?uri WHERE { res:Utah dbp:timezone ?x . ?uri rdf:type
yago:WikicatStatesOfTheUnitedStates ; dbp:timezone ?x FILTER ( ?uri != res:Utah ) }
</SPARQL>
###
Question: In which time zone is Rome?
Gold Entities:
http://dbpedia.org/resource/Rome
Gold Relations:
http://dbpedia.org/ontology/timeZone

Query:
<SPARQL>
PREFIX dbo: <http://dbpedia.org/ontology/> PREFIX res: <http://dbpedia.org/resource/> SELECT
DISTINCT ?uri WHERE { res:Rome dbo:timeZone ?uri }
</SPARQL>
###
Question: Which city has the oldest running metro?
Gold Entities:
http://dbpedia.org/resource/Rapid_transit http://dbpedia.org/ontology/City
Gold Relations:
http://dbpedia.org/ontology/type http://dbpedia.org/ontology/openingYear
http://dbpedia.org/ontology/location http://www.w3.org/1999/02/22-rdf-syntax-ns#type

Query:
<SPARQL>
PREFIX dbo: <http://dbpedia.org/ontology/> PREFIX dbr: <http://dbpedia.org/resource/> PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#> SELECT ?loc WHERE { ?uri dbo:type dbr:Rapid_transit ;
dbo:openingYear ?date ; dbo:location ?loc . ?loc rdf:type dbo:City } ORDER BY ASC(?date) LIMIT 1
</SPARQL>
###
Question: What is the time zone of Salt Lake City?
Entities:
http://dbpedia.org/resource/Salt_Lake_City
Relations:
http://dbpedia.org/ontology/timeZone

Query:
```

Figure A.1: DFSL prompt DBpedia

```
The task involves translating questions from English into SPARQL queries for the Wikidata knowledge
graph. The queries must follow specific guidelines to ensure accuracy and correct execution:
1. Enclose SPARQL queries within <SPARQL></SPARQL> tags.
2. Utilize all provided golden entities and relations exclusively to construct the query
accurately. Do not use any other entities or relations.
3. Examples are provided below for guidance.

Examples:

###
Question: Which U.S. states are in the same time zone as Utah?
Gold Entities:
http://www.wikidata.org/entity/Q829 (Utah), http://www.wikidata.org/entity/Q35657 (U.S. state),
http://www.wikidata.org/entity/Q829 (Utah),
Gold Relations:
http://www.wikidata.org/prop/direct/P421 (located in time zone),
http://www.wikidata.org/prop/direct/P31 (instance of), http://www.wikidata.org/prop/direct/P421
(located in time zone),

Query:
<SPARQL>
PREFIX wdt: <http://www.wikidata.org/prop/direct/> PREFIX wd: <http://www.wikidata.org/entity/>
SELECT DISTINCT ?uri WHERE { wd:Q829 wdt:P421 ?timezone . ?uri wdt:P31 wd:Q35657 ; wdt:P421 ?
timezone . FILTER(?uri != wd:Q829) }
</SPARQL>
###
Question: Which U.S. states are in the same timezone as Utah?
Gold Entities:
http://www.wikidata.org/entity/Q829 (Utah), http://www.wikidata.org/entity/Q35657 (U.S. state),
http://www.wikidata.org/entity/Q829 (Utah),
Gold Relations:
http://www.wikidata.org/prop/direct/P421 (located in time zone),
http://www.wikidata.org/prop/direct/P31 (instance of), http://www.wikidata.org/prop/direct/P421
(located in time zone),

Query:
<SPARQL>
PREFIX wdt: <http://www.wikidata.org/prop/direct/> PREFIX wd: <http://www.wikidata.org/entity/>
SELECT DISTINCT ?uri WHERE { wd:Q829 wdt:P421 ?timezone . ?uri wdt:P31 wd:Q35657 ; wdt:P421 ?
timezone . FILTER(?uri != wd:Q829) }
</SPARQL>
###
Question: What is the timezone in San Pedro de Atacama?
Gold Entities:
http://www.wikidata.org/entity/Q187893 (San Pedro de Atacama),
Gold Relations:
http://www.wikidata.org/prop/direct/P421 (located in time zone),

Query:
<SPARQL>
 SELECT DISTINCT ?o1 WHERE { <http://www.wikidata.org/entity/Q187893>
<http://www.wikidata.org/prop/direct/P421>  ?o1 .  }
</SPARQL>
###
Question: In which time zone is Rome?
Gold Entities:
http://www.wikidata.org/entity/Q220 (Rome),
Gold Relations:
http://www.wikidata.org/prop/direct/P421 (located in time zone),

Query:
<SPARQL>
SELECT DISTINCT ?uri WHERE {  <http://www.wikidata.org/entity/Q220>
<http://www.wikidata.org/prop/direct/P421> ?uri }
</SPARQL>
###
Question: Which city has the oldest running metro?
Gold Entities:
http://www.wikidata.org/entity/Q5503 (rapid transit),
Gold Relations:
http://www.wikidata.org/prop/direct/P31 (instance of), http://www.wikidata.org/prop/direct/P571
(inception), http://www.wikidata.org/prop/direct/P131 (located in the administrative territorial
entity),

Query:
<SPARQL>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>  PREFIX wd: <http://www.wikidata.org/entity/>
SELECT DISTINCT ?uri WHERE { ?metro wdt:P31 wd:Q5503 ; wdt:P571 ?inception ; wdt:P131 ?uri . }
ORDER BY ?inception LIMIT 1
</SPARQL>
###
Question: What is the time zone of Salt Lake City?
Entities:
http://www.wikidata.org/entity/Q23337 (Salt Lake City),
Relations:
http://www.wikidata.org/prop/direct/P421 (located in time zone),

Query:
```

Figure A.2: DFSL prompt Wikidata

```
The task involves translating a question from English into  SPARQL queries for the Wikidata
knowledge graph. The queries must follow specific guidelines to ensure accuracy and correct
execution:
1. Enclose SPARQL queires within <SPARQL></SPARQL> tags.
2. Utilize all provided golden entities and relations exclusively to construct the query. Do not
use any other entities or relations.
3. you can generate multiple queries for the same question.
Question: What is the time zone of Salt Lake City?

Entities:
http://www.wikidata.org/entity/Q23337 (Salt Lake City),

Relations:
http://www.wikidata.org/prop/direct/P421 (located in time zone),

Query:
```

Figure A.3: Zero-Shot prompt Wikidata

```
The task involves translating a question from English into SPARQL queries for the DBpedia knowledge
graph. The queries must follow specific guidelines to ensure accuracy and correct execution:
1. Enclose SPARQL queries within <SPARQL></SPARQL> tags.
2. Utilize all provided golden entities and relations exclusively to construct the query. Do not
use any other entities or relations.
3. you can generate multiple queries for the same question.
Question: What is the time zone of Salt Lake City?

Entities:
http://dbpedia.org/resource/Salt_Lake_City

Relations:
http://dbpedia.org/ontology/timeZone

Query:
```

Figure A.4: Zero-Shot prompt DBpedia

```
The task involves translating a question from English into  SPARQL queries for the Wikidata
knowledge graph. The queries must follow a specific guideline to ensure accuracy and correct
execution:
1. Enclose SPARQL queries within <SPARQL> </SPARQL> tags.

Question: Who killed John Lennon?

Query:
```

Figure A.5: No-shot prompt Wikidata

```
The task involves translating a question from English into SPARQL queries for the DBpedia knowledge
graph. The queries must follow a specific guideline to ensure accuracy and correct execution:
1. Enclose SPARQL queries within <SPARQL> </SPARQL> tags.


Question: Who killed John Lennon?

Query:
```

Figure A.6: No-shot prompt DBpedia

```
The task involves translating questions from English into SPARQL queries for the Dbpedia knowledge
graph. These queries must follow specific guidelines to ensure accuracy and correct execution:
1. Enclose SPARQL queries within <SPARQL></SPARQL> tags.
2. Utilize all provided golden entities and relations exclusively to construct the query
accurately.
3. Examples are provided below for guidance.

Examples:

###
Question: Where did Hillel Slovak die?
Entities:
http://dbpedia.org/resource/Hillel_Slovak
Relations :
http://dbpedia.org/ontology/deathPlace

Query:
<SPARQL>
SELECT DISTINCT ?uri WHERE { <http://dbpedia.org/resource/Hillel_Slovak>
<http://dbpedia.org/ontology/deathPlace> ?uri }
</SPARQL>
###
Question: Who is the mayor of Paris?
Entities:
http://dbpedia.org/resource/Paris
Relations:
http://dbpedia.org/ontology/mayor

Query:
<SPARQL>
SELECT DISTINCT ?uri WHERE { <http://dbpedia.org/resource/Paris>
<http://dbpedia.org/ontology/mayor> ?uri }
</SPARQL>
###
Question: Give me the grandchildren of Elvis Presley.
Entities:
http://dbpedia.org/resource/Elvis_Presley
Relations:
http://dbpedia.org/ontology/child http://dbpedia.org/property/children

Query:
<SPARQL>
SELECT DISTINCT ?uri WHERE { <http://dbpedia.org/resource/Elvis_Presley>
<http://dbpedia.org/ontology/child> ?child . ?child <http://dbpedia.org/property/children> ?uri }
</SPARQL>
###
Question: How many inhabitants does the largest city in Canada have?
Entities:
http://dbpedia.org/resource/Canada
Relations :
http://dbpedia.org/ontology/largestCity http://dbpedia.org/ontology/populationTotal

Query:
<SPARQL>
PREFIX dbo: <http://dbpedia.org/ontology/> PREFIX res: <http://dbpedia.org/resource/> SELECT
DISTINCT ?num WHERE { res:Canada dbo:largestCity ?city . ?city dbo:populationTotal ?num }
</SPARQL>
###
Question: How many theories did Albert Einstein come up with?
Entities:
http://dbpedia.org/resource/Albert_Einstein
Relations:
http://dbpedia.org/ontology/knownFor

Query:
<SPARQL>
PREFIX dbo: <http://dbpedia.org/ontology/> PREFIX dbr: <http://dbpedia.org/resource/> SELECT
(COUNT(?uri) AS ?count) WHERE { dbr:Albert_Einstein dbo:knownFor ?uri }
</SPARQL>
###
Question: What is the time zone of Salt Lake City?
Entities:
http://dbpedia.org/resource/Salt_Lake_City
Relations:
http://dbpedia.org/ontology/timeZone

Query:
```

Figure A.7: Few-Shot prompt DBpedia

```
The task involves translating questions from English into SPARQL queries for the Wikidata knowledge
graph. These queries must follow specific guidelines to ensure accuracy and correct execution:
1. Enclose SPARQL queries within <SPARQL></SPARQL> tags.
2. Utilize all provided golden entities and relations exclusively to construct the query
accurately. Multiple relations can be combined within a single triple using a forward slash (/).
For example: ?spouse wdt:P19/wdt:P17 wd:Q183
3. Examples are provided below for guidance.

Examples:

###
Question: Give me all companies in Munich.
Entities:
http://www.wikidata.org/entity/q4830453 (business), http://www.wikidata.org/entity/q1726 (Munich)
Relations:
http://www.wikidata.org/prop/direct/p279 (subclass of), http://www.wikidata.org/prop/direct/p31
(instance of), http://www.wikidata.org/prop/direct/p159 (headquarters location)

Query:
<SPARQL>
PREFIX wdt: <http://www.wikidata.org/prop/direct/> PREFIX wd: <http://www.wikidata.org/entity/>
SELECT DISTINCT ?uri WHERE { ?type wdt:P279* wd:Q4830453 . ?uri wdt:P31 ?type ; wdt:P159 wd:Q1726 .
}
</SPARQL>
###
Question: Was Marc Chagall a jew?
Entities:
http://www.wikidata.org/entity/q93284 (Marc Chagall), http://www.wikidata.org/entity/q7325 (Jewish
people)
Relations:
http://www.wikidata.org/prop/direct/p172 (ethnic group)

Query:
<SPARQL>
PREFIX wdt: <http://www.wikidata.org/prop/direct/> PREFIX wd: <http://www.wikidata.org/entity/> ASK
WHERE { wd:Q93284 wdt:P172 wd:Q7325 . }
</SPARQL>
###
Question: How many films did Leonardo DiCaprio star in?
Entities:
http://www.wikidata.org/entity/q11424 (film), http://www.wikidata.org/entity/q38111 (Leonardo
DiCaprio)
Relations:
http://www.wikidata.org/prop/direct/p31 (instance of), http://www.wikidata.org/prop/direct/p161
(cast member)

Query:
<SPARQL>
PREFIX wdt: <http://www.wikidata.org/prop/direct/> PREFIX wd: <http://www.wikidata.org/entity/>
SELECT (COUNT(DISTINCT ?uri) AS ?c) WHERE { ?uri wdt:P31 wd:Q11424 ; wdt:P161 wd:Q38111 . }
</SPARQL>
###
Question: Give me all libraries established earlier than 1400.
Entities:
http://www.wikidata.org/entity/q7075 (library)
Relations:
http://www.wikidata.org/prop/direct/p31 (instance of), http://www.wikidata.org/prop/direct/p571
(inception)

Query:
<SPARQL>
PREFIX wdt: <http://www.wikidata.org/prop/direct/> PREFIX wd: <http://www.wikidata.org/entity/>
SELECT DISTINCT ?uri WHERE { ?uri wdt:P31 wd:Q7075 ; wdt:P571 ?date . FILTER (YEAR(?date) < 1400 )
}
</SPARQL>
###
Question: Is Christian Bale starring in Batman Begins?
Entities:
http://www.wikidata.org/entity/q166262 (Batman Begins), http://www.wikidata.org/entity/q45772
(Christian Bale)
Relations:
http://www.wikidata.org/prop/direct/p161 (cast member)
###
Question: What is the time zone of Salt Lake City?
Entities:
http://www.wikidata.org/entity/Q23337 (Salt Lake City),
Relations:
http://www.wikidata.org/prop/direct/P421 (located in time zone),

Query:
```

Figure A.8: Few-Shot prompt Wikidata

Acknowledgements

I would like to express my deepest gratitude to my family, whose unwavering support and encouragement have been the cornerstone of my journey throughout these two years. To my girlfriend, your patience, understanding, and constant belief in me have meant the world. Lastly, to all my friends, thank you for always being there. This accomplishment would not have been possible without you all by my side.