# Investigating Large Language Models for Text-to-SPARQL Generation

**Jacopo D'Abramo**
Amazon Spain, Spain[*]
jdabramo@amazon.com

**Andrea Zugarini**
expert.ai, Italy
azugarini@expert.ai

**Paolo Torroni**
Department of Computer Science and Engineering, University of Bologna, Italy
paolo.torroni@unibo.it

## Abstract

Large Language Models (LLMs) have demonstrated strong capabilities in code generation, such as translating natural language questions into SQL queries. However, state-of-the-art solutions often involve a costly fine-tuning step. In this study, we extensively evaluate In-Context Learning (ICL) solutions for text-to-SPARQL generation with different architectures and configurations, based on methods for retrieving relevant demonstrations for few-shot prompting and working with multiple generated hypotheses. In this way, we demonstrate that LLMs can formulate SPARQL queries achieving state-of-the-art results on several Knowledge Graph Question Answering (KGQA) benchmark datasets without fine-tuning.

## 1 Introduction

The advent of Large Language Models (LLMs) has significantly advanced the field of Natural Language Processing (NLP), with particular success in the domain of code generation (Chen et al., 2021; Rozière et al., 2024).

At the same time, the growing complexity and scale of Knowledge Graphs (Pellissier Tanon et al., 2016; Lehmann et al., 2014; Bollacker et al., 2008) highlighted the need for robust and accurate mechanisms to query such data stores, for instance, within Knowledge Graph Question Answering (KGQA) pipelines (Li et al., 2023; Nie et al., 2024).

In this paper, we carry out an extensive evaluation of LLM-based In-Context Learning (ICL) for text-to-SPARQL generation. Text-to-SPARQL is a crucial component of many KGQA systems, that typically make use of different modules to assembly the query, such as Entity and Relation Linking. To this end, we define a simple and modular approach to address the text-to-SPARQL task without fine-tuning. The evaluation is focussed on the following key aspects: (1) the influence of various In-Context Learning strategies on the quality of the generated queries; (2) the impact of different state-of-the-art model backbones, varying in architecture, size, and training data; (3) the potential of beam search to generate multiple query candidates, thereby enhancing the results;(4) a comparison between ICL and specialized models fine-tuned for the task. The code is publicly available at https://github.com/jacopodabramo/DFSL.

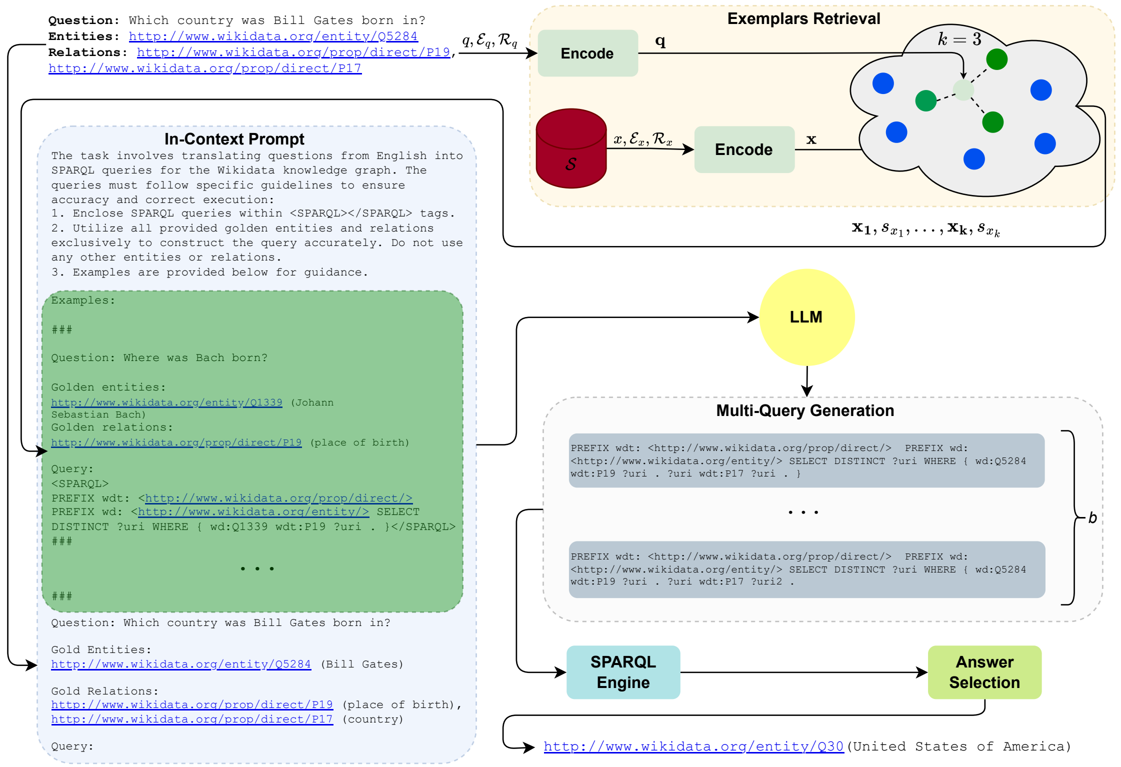
In the interest of reproducibility, as backbones, we use three state-of-the-art open-weight LLMs: Mixtral 8x7B, Llama-3 70B, and CodeLlama 70B. We run experiments on two widely-used Knowledge Bases, DBpedia and Wikidata, using four publicly available datasets: QALD-9, based on DBpedia, and QALD-9 plus, QALD-10 and LC-QuAD 2.0, based on Wikidata.

Our experimental results demonstrate that LLMs In-Context Learning solutions achieve state-of-the-art results, without the need of any fine-tuning. The injection of demonstrations similar to the input question into the prompt combined with the generation of multiple query candidates directly from beam search hypotheses, yield the best results, exceeding in most of the benchmarks state-of-the-art models fine-tuned for the task. Finally, we also run ablation studies to gauge the effectiveness of the approach without gold information from the EL and RL modules.

## 2 Related work

We first provide an overview of most related In-Context-Learning approaches. Then, we discuss text-to-SPARQL methods, including KGQA systems that typically make use of text-to-SPARQL techniques to tackle the problem.

[*]Work done while being at expert.ai.



```
Question: Which country was Bill Gates born in?
Entities: http://www.wikidata.org/entity/Q5284
Relations: http://www.wikidata.org/prop/direct/P19,
http://www.wikidata.org/prop/direct/P17
```

```
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: Where was Bach born?

Golden entities:
http://www.wikidata.org/entity/Q1339 (Johann
Sebastian Bach)
Golden relations:
http://www.wikidata.org/prop/direct/P19 (place of birth)

Query:
<SPARQL>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/> SELECT
DISTINCT ?uri WHERE { wd:Q1339 wdt:P19 ?uri . }</SPARQL>
###

                    . . .

###
Question: Which country was Bill Gates born in?

Gold Entities:
http://www.wikidata.org/entity/Q5284 (Bill Gates)

Gold Relations:
http://www.wikidata.org/prop/direct/P19 (place of birth),
http://www.wikidata.org/prop/direct/P17 (country)

Query:
```

```
PREFIX wdt: <http://www.wikidata.org/prop/direct/>  PREFIX wd:
<http://www.wikidata.org/entity/> SELECT DISTINCT ?uri WHERE { wd:Q5284
wdt:P19 ?uri . ?uri wdt:P17 ?uri . }
```

```
PREFIX wdt: <http://www.wikidata.org/prop/direct/>  PREFIX wd:
<http://www.wikidata.org/entity/> SELECT DISTINCT ?uri WHERE { wd:Q5284
wdt:P19 ?uri . ?uri wdt:P17 ?uri2 .
```

```
http://www.wikidata.org/entity/Q30(United States of America)
```

Figure 1: Sketch of the ICL approach. Given a question, its entities and its relations, $k$-most similar examples are retrieved from a text-to-SPARQL collection $\mathcal{S}$ 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.

### 2.1 In-Context Learning

ICL is a paradigm that leverages reasoning through analogies. A task description, question, and demonstration context are usually concatenated to create a prompt, which is then input into an LLM for prediction. Unlike fine-tuning, ICL performs predictions without gradient updates (Dong et al., 2023). Few-Shot Learning is a type of ICL where the demonstration context includes a few examples. Owing to the effectiveness of ICL and the obvious advantage of building systems that don't need domain-specific training, a great deal of research and engineering efforts have been devoted to designing suitable prompts. ICL has been successfully applied to many NLP problems, including QA (Chada and Natarajan, 2021; Chen et al., 2023).

Some studies have also focused on the selection of in-context examples. In particular, Liu et al. (2022) developed KATE, an unsupervised retriever that utilizes k-nearest neighbors and distance metrics (e.g., L2 distance and cosine similarity) to select in-context examples for tasks such as sentiment analysis, table-to-text generation, and question answering. Levy et al. (2023) explored the incorporation of diverse demonstrations into prompts for compositional semantic parsing task, demonstrating that such diversity leads to better structural coverage in target utterances. Kim et al. (2022) leveraged the generative capabilities of pre-trained language models to generate demonstrations for each class in downstream tasks, conditioned on test inputs and class information. Gonen et al. (2022) found that selecting examples based on perplexity, in particular lower perplexity, is an effective strategy.

Similar principles have been adopted in code generation tasks (Cheng et al., 2022), including text-to-SQL (Cheng et al., 2022; Nan et al., 2023; Zhang et al., 2023; Wei et al., 2023) and KGQA (Li et al., 2023). In the same vein, our study investigates ICL strategies to address text-to-SPARQL. However, while all these approaches are based on proprietary LLMs, such as GPT-3, GPT-4 (Brown et al., 2020) and Codex (Chen et al., 2021), we focus on open-weight LLMs.

### 2.2 Text-to-SPARQL

With the recent wave of decoder-based LLMs such as GPT (Brown et al., 2020), Mixtral (Jiang et al., 2024), and Llama (Touvron et al., 2023), generative AI was also used to translate questions into SPARQL queries. Notably, Zou et al. (2021) introduced a text-to-SPARQL model that leverages a relation-aware attention decoder and a pointer network encoder, incorporating three separate scaled dot-product attention mechanisms to generate SPARQL queries that capture entity, relation, and keyword representations. Banerjee et al. (2022) experimented with various models, including T5 (Raffel et al., 2020), BART (Lewis et al., 2019), and Pointer Generation Networks (See et al., 2017), to explore their efficacy in KGQA tasks. Rony et al. (2022)'s SGPT employs a stack of transformer encoders to extract linguistic features from the natural question and GPT-2 as a decoder. However, this architecture is limited by its inability to capture connections among entities and relations in the underlying knowledge graph, leading to errors in generating triple sequences in the final SPARQL queries. Pliukhin et al. (2023) presented a one-shot generative approach, where the prompt is augmented with a KG fragment required to construct the query and a question-subgraph query example.

Despite promising results, these architectures are prone to systematic errors. One such error, the so-called "triple-flip", refers to the reversal of subject and object positions in the generated SPARQL triples, yielding wrong, often empty answers. Qi et al. (2024) addressed this issue by developing TSET, a fine-tuned T5 model with a pre-training stage called Triplet Structure Correction.

All these works propose dedicated architectures or training objectives designed for the task at hand. This requires fine-tuning, that may be expensive in terms of resources, thus limiting the choice of backbones to specialize. We take a different approach and investigate ICL solutions that do not require any fine-tuning.

Text-to-SPARQL methods are typically evaluated in KGQA tasks, and they all share a similar pipeline, where entities and relations are given or extracted from other modules and the goal of the model is to translate a natural language question, associated with its entities and relations, into the SPARQL query.

**KGQA.** Being text-to-SPARQL an important ingredient in KGQA, many KGQA approaches are inherently related with our work. Early research in KG query generation was rule-based (Guo et al., 2005; Owens et al., 2008), template-based (Zenz et al., 2009; Unger et al., 2012; Görlitz et al., 2012) or search-based. However, manual or semi-manual approaches hit scalability issues with KGs like WikiData and DBpedia. Nowadays, research follows two main streams: information-retrieval based methods and Text-to-SPARQL approaches. The former generally require identifying sub-graphs relevant to the natural question. They include divide-and-conquer (Kim et al., 2023), fact retrieval based on linked entities (Baek et al., 2023), more complex methods involving hops, relation predictions, and triple sampling (Wu et al., 2023), or Evidence Pattern Retrieval (EPR) through structural dependency modeling (Ding et al., 2024). Conversely, solutions based on text-to-SPARQL typically use ICL approaches to build a query draft in a logical form that is then refined and converted into a formal SPARQL query by means of different strategies. KB-BINDER (Li et al., 2023) leverages LLMs for generating preliminary logical forms and refines them using a lexicon-based similarity search, achieving notable performance on several KBQA datasets without customized heuristics for specific knowledge bases. In (Nie et al., 2024) authors propose converting logical form generation into Python function call sequences, reducing format errors and enhancing performance in zero-shot and few-shot settings, establishing new state-of-the-art results on multiple datasets. In the same spirit, other methods generate natural language questions starting from SPARQL queries instead (Li et al., 2024; Liang et al., 2023).

Our study departs from the KGQA works listed above in several ways. First, it focuses on the text-to-SPARQL task starting from a natural question and gold entities/relations, whereas KB-BINDER and other methods address a different task: the production of a logical form starting from a natural question, without gold entities/relations. The SPARQL query can be constructed afterward, and entities/relations can be predicted based on the logical form, which is what KB-BINDER does; nevertheless, the task is a different one. For this reason, we only consider text-to-SPARQL benchmarks with gold entities/relations, as in (Banerjee et al., 2022; Rony et al., 2022; Qi et al., 2024). Another difference is the choice of the KG: instead of freebase, a project closed in 2016, we adopt Wikidata, an ongoing project with 1.57 billion semantic triples to date.[1]

[1]https://grafana.wikimedia.org/d/000000175/

## 3 Method

Given a collection of natural language questions $\mathcal{Q}$ and a knowledge graph $\mathcal{G} := (\mathcal{E}, \mathcal{R}, \mathcal{F})$, where $\mathcal{E}$ are *entities*, $\mathcal{R}$ are *relations*, and $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ are *facts*, KGQA is the problem of answering questions in $\mathcal{Q}$ based on $\mathcal{G}$. KGQA 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 $a$. The 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, $P_\theta$ 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 via natural language text. Accordingly, the generated query is:

$$s_q = \arg\max_s P_\theta(s|C, \mathcal{E}_q, \mathcal{R}_q, q). \qquad (1)$$

### 3.1 Exemplary Demonstrations Retrieval

In few-shot ICL, the choice of demonstrations to inject in the prompt can significantly affect performance. Usually, few-shot examples are predetermined representative instances of the task, handpicked during prompt design. Conversely, we aim to retrieve good examples dynamically, based on their relevance to the input question. Inspired by Liu et al. (2022) and Li et al. (2023), we adopt a retrieval approach based on the similarity between a question $q$ and a set of previously answered text-to-SPARQL examples collected in a storage $\mathcal{S}$ (see Figure 1), where each example is a tuple including a question $x$, its entities $\mathcal{E}_x$ and relations $\mathcal{R}_x$, and the associated SPARQL query $s_x$. Differently from (Li et al., 2023), we encode examples with dense representations instead of BM25. Moreover, beside the question itself, we also encode its entities and relations, i.e. $\langle q, \mathcal{E}_q, \mathcal{R}_q \rangle$ are mapped onto a vector representation $e_q \in \mathbb{R}^d$ using a sentence encoder. To properly feed such information to an encoder-only LM, we concatenate question, entities and relations in a single input 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:

$$score(\boldsymbol{q}, \boldsymbol{x}) = sim(e_q, e_x), \forall x \in \mathcal{S}, \qquad (2)$$

where $sim$ is a similarity function. Based on such a scoring, we retrieve the $k$-most similar examples $\mathcal{S}$ and include them as demonstrations in the in-context prompt. From now on, we refer to this exemplary demonstration retrieval strategy as DFSL, standing for Dynamic Few-Shot Learning.

### 3.2 In-Context Prompt

The in-context prompt has three parts. The first is the task description, instructing the LLM with a numbered list of guidelines on the output format and on the available information. The second, highlighted in Figure 1 with a green block, contains the $k$ 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. The ### symbol delimits each example. The last part is the question, associated with its gold entities and relations. The answer returned by the LLM prompted as such is then parsed to extract the generated text enclosed in `<SPARQL></SPARQL>` tags. The resulting query $s_q$ is executed by a SPARQL engine on $\mathcal{G}$ to yield the answer to $q$.

### 3.3 Multi-Query Generation

A typical challenge faced by LLMs in SPARQL query generation is the understanding of what is the subject and what is the object of a relation, an information the model does not have. LLMs often end up in swapping the subject with the object in the query, almost choosing one way or the other randomly. This problem is called triple-flip error (Qi et al., 2024). Thanks to ICL, this issue may be alleviated whenever there are similar demonstrations in the in-context prompt that clarify the subject-object roles. To further reduce triple-flip errors, we propose the generation of multiple SPARQL queries by retaining all the final hypotheses generated during beam search. The model uncertainty in placing subject and object is likely to be reflected in the beam search exploration. Intuitively, both triple-ordering hypotheses are considered plausible by the model. Thus, instead of just returning the most probable sequence $s$ according to Equation 1, we keep the whole $b$ queries $\{s_{q,1}, \ldots, s_{q,b}\}$ formulated by beam search.

wikidata-datamodel-statements?orgId=1&refresh=30m

| Approach | Backbone | QALD-9 Plus | QALD-10 | LC-QUAD 2.0 | QALD-9 DB |
|---|---|---|---|---|---|
| Zero-shot Learning | | 49.90 | 33.76 | 40.66 | 65.73 |
| Few-shot Learning | Mixtral 7x8 | 54.80 (**+4.90**) | 50.26 (**+16.50**) | 61.04 (**+20.38**) | 63.86 (**-1.87**) |
| DFSL | | 71.75 (**+21.85**) | 49.90 (**+16.14**) | 81.81 (**+41.15**) | 72.74 (**+7.01**) |
| Zero-shot Learning | | 63.01 | **58.31** | 54.21 | 70.49 |
| Few-shot Learning | Llama-3 70B | 67.69 (**+4.68**) | 51.28 (**-7.03**) | 68.52 (**+14.31**) | 68.84 (**-1.65**) |
| DFSL | | 73.60 (**+10.59**) | 56.59 (**-1.72**) | 81.93 (**+27.72**) | 72.66 (**+2.17**) |
| Zero-shot Learning | | 45.94 | 33.36 | 38.40 | 66.43 |
| Few-shot Learning | CodeLlama 70B | 64.49 (**+18.55**) | 57.38 (**+24.02**) | 64.46 (**+26.06**) | 72.67 (**+6.24**) |
| DFSL | | **76.59** (**+30.65**) | 57.69 (**+24.33**) | **85.45** (**+47.05**) | **75.14** (**+8.71**) |

Table 1: ICL techniques comparison on different backbones. Absolute F1 gains with respect to zero-shot are reported between parenthesis.

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

Table 2: 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.

**Answer Selection.** Executing multiple queries inevitably leads to multiple possible answers. Therefore, we must define an answer selection criterion. We designed two heuristics: Largest Set (LS) and First Set (FS). LS executes all the $b$ queries, obtaining with each query $s_{q,j}$ a (possibly empty) answer set $\mathcal{A}_j$. LS then selects, among $\{\mathcal{A}_1, \dots, \mathcal{A}_b\}$, the largest one[2], i.e:

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

the rationale being that incorrect candidates will likely have empty results. However, LS can be misled into selecting answers from under-constrained queries that return many irrelevant instances. FS adheres to the natural beams ordering by selecting the first query that yields a non-empty answer set.

## 4 Experiments

In this section, we conduct the investigation of LLMs capabilities in text-to-SPARQL query generation. KGQA serves as a benchmark task to measure the quality of the generated queries.

### 4.1 Datasets

To make our analysis more robust, we evaluate models and methods on four heterogeneous KGQA benchmarks based on two different Knowledge Graphs (Wikidata, DBpedia).

[2]In case of ties, we take the first largest set.

**QALD-9 DB.** QALD-9 (Ngomo, 2018) is a dataset from the Question Answering over Linked Data (QALD) challenge series. It comprises 408 training questions and 150 test questions. Unlike the other KGQA benchmarks, the SPARQL queries are meant for a DBpedia Knowledge Graph. We refer to it as QALD-9 DB to emphasize that.

**QALD-9 plus.** QALD-9 plus extends QALD-9 on new languages and transfers SPARQL queries from DBpedia to Wikidata. Although some queries were not portable to Wikidata due to the absence of corresponding information, it still comprises 371 training questions and 136 test questions. In our experiments, we only consider English questions.

**QALD-10.** QALD-10 (Usbeck et al., 2023) is the latest dataset in the QALD series, designed to increase the complexity of gold SPARQL queries. It consists of 412 training questions extracted from QALD-9 plus Wikidata. The test set was created from scratch, comprising 394 test questions that express real-world information needs. Test questions significantly differ from those in training.

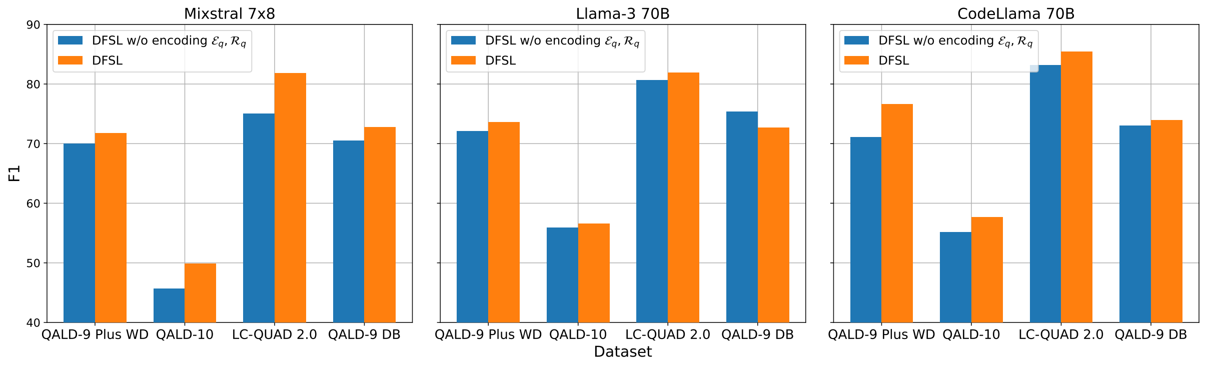


Figure 2: Comparison of Embeddings: DFSL (in orange) encoding that incorporates question, entities and relations versus an embedding solely based on the question $q$ (in blue).

**LC-QuAD 2.0.** LC-QuAD 2.0 (Dubey et al., 2019) is a large-scale dataset grounded on Wikidata. It consists of 30,226 simple and complex questions: 24,180 in training, and 6,046 in test. Questions are diverse. They include single- and multi-fact, boolean, count, and other query types. With such a large and diverse text-to-SPARQL storage, LC-QuAD 2.0 allows us to gauge the benefits of retrieving similar exemplary demonstrations.

### 4.2 Backbones

**Mixtral 8x7B.** Based on the Sparse Mixture of Experts (SMoE) architecture (Fedus et al., 2022), Mixtral 8x7B (Jiang et al., 2024) is a 46.7B parameters model. Among the backbones adopted in this paper, Mixtral is the smallest. Moreover, thanks to the characteristics of its SMoE architecture, less than 13B are active at each inference step, making Mixtral particularly efficient.

**Llama-3 70B.** Built upon the Llama architecture (Touvron et al., 2023), Llama-3 70B has been trained on 15T tokens, a 650% increase from its predecessor, Llama 2. At the time of writing, Llama-3 70B is one of the best-performing open-weights LLMs available.

**CodeLlama 70B.** Initialized from Llama2 70B, CodeLlama (Rozière et al., 2024) is a specialized version fine-tuned on 1T tokens of code-heavy data. Therefore, we expect CodeLlama to be particularly suitable for SPARQL query generation.

### 4.3 Baselines

**Plain Question.** This is a naive baseline where we feed an LLM only with the task description and the question $q$. Without in-context examples nor any entity or relation associated with $q$, the LLM can only rely on its parameter memory.

**Zero-Shot Learning.** Here we do not provide any demonstrative example in the prompt. However, unlike the plain question baseline, we do inject golden entities and relations into the prompt. With reference to Figure 1, the In-Context prompt remains the same but without the green-like block containing the demonstrations.

**Few-Shot Learning.** The prompt is filled with a single set of $k$ manually selected examples, used for all the questions in the test set. The examples were chosen to maximize diversity and cover different kinds of queries[3].

**Multi Query Prompting (DFSL-MQP).** As an alternative to our proposed multi-query generation (DFSL-MQ), this baseline consists in a naive multi-query prompting strategy. Essentially, we ask the model to generate more queries to answer the question. To ease the creation of inverted subject-object queries that can solve triple-flip errors, we extend the prompt to explicitly ask the model to produce this kind of SPARQL queries. Answer selection uses LS and FS heuristics, like with DFSL-MQ.

### 4.4 Experimental Setup

**Implementation.** In our experiments, the training set of each dataset serves as storage for the retrieval of the $k$ most similar examples (see the next paragraph for details on $k$ tuning) with DFSL. Examples are encoded with a sentence transformer[4], all-mpnet-base-v2[5], and $sim$ is defined as the cosine similarity. Inference is performed via beam search in all ICL approaches, where $b$ is set to 3,

[3]The chosen examples and more details are provided in Appendix B.
[4]https://www.sbert.net/index.html
[5]huggingface.co/sentence-transformers/all-mpnet-base-v2

and DFSL-MQ, where $b$ is set to 10. All the experiments were run on a cluster of 4 NVIDIA A100 GPUs.

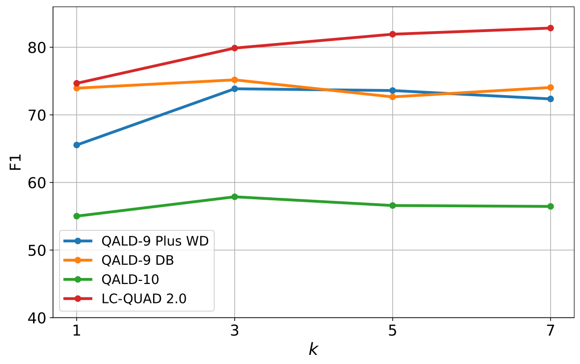


Figure 3: Impact of the number of in-context examples on the four benchmarks.

**Number of Few-shot Examples.** We first analyzed how the number of few-shot examples $k$ retrieved by DFSL affects the performance. We chose among $k = \{1, 3, 5, 7\}$ and evaluated DFSL with Llama 3 70B backbone on the four datasets. The results shown in Figure 3 suggest that values of $k$ greater than one perform comparably well on smaller benchmarks, while on LC-QUAD 2.0, where there are about 25 thousands examples as storage, increasing $k$ seems to be beneficial. This may be due to the increased likelihood of finding similar examples in larger datasets as $k$ grows. We set $k = 5$ for all the forthcoming experiments, which is a good trade-off across all the datasets.

**Prompt.** The prompt illustrated in Figure 1 constitutes the default template in our experimentations. However, slight variations are required in certain cases. For example, when running experiments on DBpedia knowledge graph, we replace the Wikidata reference with DBpedia in the first text segment. When we study the absence of gold information instead, we remove all the references to gold entities/relations (according to the ablation) from the entire prompt. There are no differences in the prompts layout when running few-shot-learning baseline experiments. In zero-shot learning, only the in-context examples any reference to them are removed, all else being equal.

**Evaluation metric.** We follow a standard F1 score evaluation in KGQA benchmarks. The F1 is computed between the answer set returned by the target SPARQL query and the predicted one. When both the queries return an empty set, we assign an F1 score of 1. The F1 scores of all the examples are then averaged.

## 4.5 Results

**In-Context Learning.** To measure how ICL techniques affect the generation of SPARQL queries, we compare Zero-Shot, Few-Shot Learning and DFSL on three different LLMs. Results are outlined in Table 1. Both few-shot learning and DFSL generally yield substantial gains with respect to zero-shot baseline on all the backbones and datasets. An exception occurs in QALD-10 with Llama-3. Notably, when comparing DFSL and Few-shot Learning baseline, we can see how examples selection approach improves F1 scores by a large margin in LC-QUAD 2.0, QALD-9 Plus and QALD-9 DB, with F1 increasing up to 21 absolute points[6]. In QALD-10 instead, where the test set has a different distribution from its training, there are no significant differences between DFSL and the standard few-shot learning approach. Indeed, an approach like DFSL brings little benefits when the storage only contains unrelated examples.

**Backbones Comparison.** In terms of backbones, Llama-3 consistently outperforms both Mixtral and CodeLlama in zero-shot learning scenario, whereas in few-shot, results are generally comparable between Llama-3 and CodeLlama. Such a strong Llama-3 zero-shot performance may be caused by some sort of data contamination, however we leave such an investigation for future works. Overall, DFSL with CodeLlama achieved the greatest performance with respect to all the other configurations. Therefore, we adopt CodeLlama as our backbone in the following experiments.

**Impact of Multi-Query Generation.** Here we investigate DFSL-MQ, the multi-query approach extending DFSL. We evaluate both answer selection strategies, LS and FS, and compare them against the plain DFSL and the multi-query prompting baseline described in Section 4.3. All the results are outlined in Table 2. Having multiple queries is not necessarily beneficial. Indeed, the multi-query prompting baseline under-performs in three datasets out of four with respect to (single query) DFSL, regardless of the answer selection method adopted. On the contrary, DFSL-MQ proves to positively increase results. Both Largest Set and First Set heuristics are effective when the

[6]Some qualitative examples illustrate the benefits of DFSL over few-shot learning in Appendix A (see Table 6).

| **Approach** | **QALD-9 Plus** | **QALD-10** | **LC-QUAD 2.0** | **QALD-9 DB** |
|---|---|---|---|---|
| Plain Question | 0.08 | 0.02 | 12.00 | 16.42 |
| BART (Banerjee et al., 2022) | - | - | 64.00 | - |
| PGN-BERT-BERT (Banerjee et al., 2022) | - | - | 86.00 | - |
| SGPT (Rony et al., 2022) | - | - | 89.04 | 67.82 |
| TSET-small (Qi et al., 2024) | 72.86 | 47.15 | 94.00 | - |
| TSET-base (Qi et al., 2024) | 75.85 | 51.37 | **95.00** | - |
| Zero-shot Learning | 45.94 | 33.36 | 38.40 | 66.43 |
| Few-shot Learning | 64.49 | 57.38 | 64.46 | 72.67 |
| DFSL | 76.59 | 57.69 | 85.45 | 75.14 |
| DFSL-MQ beam FS | **84.45** (**+8.60**) | **62.20** (**+10.83**) | 89.10 (**-5.90**) | **77.89** (**+10.07**) |

Table 3: DFSL and ICL approaches vs state-of-the-art fine-tuned models.

hypotheses come from the beams. Furthermore, FS consistently outperforms LS, even by substantial margins in QALD-9 DB. This shows that exploiting the information coming from beam search hypotheses is a promising strategy to obtain more query candidates.

**In-context Learning vs Fine-tuning.** Up to this point, we have assessed In-Context Learning approaches. In Table 3 instead, we compare them against state-of-the-art models trained and/or fine-tuned for specific downstream KGQA datasets. Without any training, DFSL-MQ outperforms current state-of-the-art approaches in three out of four benchmarks, namely QALD-9 Plus, QALD-10 and QALD-9 DB, even with the single query DFSL setup. DFSL-MQ does not obtain state-of-the-art results in LC-QUAD 2.0, the dataset mostly affected by triple-flip errors. This means that multi-query generation only alleviates the issue, but does not solve it entirely.

## 4.6 Ablation studies

**Different Example Encoding.** As described in Section 3.1, to compute the embeddings we concatenated the textual input made of the question and its list of entities and relations. Here, we gauge the impact of this additional information on DFSL performance. In Figure 2 we compare it, with a variant where only the natural language question $q$ is embedded, without any additional data concatenated. The evaluation carried out in all the benchmarks and with all the backbones, demonstrates that such information improves the quality of the generated queries.

**Absence of gold information.** In KGQA, text-to-SPARQL generation usually relies not only on the question itself, but also on entities and relations associated to it. Here we assess DFSL when either the entities $\mathcal{E}_q$ or the relations $\mathcal{R}_q$, or both are missing. The information is removed throughout the entire process. For example, when removing entities, we discard them from both the storage and the prompt. Even the embeddings for the retrieval are computed by encoding an input without any entity concatenated in $\boldsymbol{q}$, i.e. becoming $\boldsymbol{q} = [q, \mathcal{R}_q]$. We report this on both Wikidata and DBpedia KGs. Results outlined in Table 4, clearly show how the knowledge about entities and the relations is essential for generating the query, indeed without performance drop significantly. Nonetheless, even in the case where no information is given (DFSL w/o $\mathcal{E}_q, \mathcal{R}_q$), the presence of dynamic demonstrations drastically help, yielding respectively a 33+ and 25+ absolute F1 increase compared to plain question baseline in Table 3.

| | QALD 9 DB | QALD 9 Plus |
|---|---|---|
| DFSL | 75.14 | 76.59 |
| w/o Rq | 56.62 | 49.47 |
| w/o Eq | 60.92 | 31.83 |
| w/o Eq, Rq | 49.59 | 26.16 |

Table 4: DFSL in the absence of entities and/or relations.

## 5 Conclusion

In this paper, we investigated the use of out-of-the-box Large Language Models for text-to-SPARQL generation. We carried out an extensive evaluation of several backbones and configurations on four KGQA benchmarks. By leveraging different In-Context Learning (ICL) approaches, we have shown that LLMs can effectively generate SPARQL queries. When demonstrations similar to the input question are injected into the prompt, LLMs achieve performance exceeding state-of-the-art models fine-tuned on the downstream task. The generation of multiple SPARQL query hypotheses from beam search candidates enhances the performance further, even with a simple query selection criterion.

Future work will focus on extending the investigation to multiple languages and (possibly private) KGs.

## Limitations

We recognize some limitations in our work. Our experiments are all on English-based datasets, where notoriously LLMs are better performing. Moreover, the massive pre-training of those LLMs on a vast portion of the Web, may expose those models to unintended data contamination, a phenomenon already observed in similar domains (Ranaldi et al., 2024). Experiments only focused on LLMs with large number of parameters, without investigating the behaviour of smaller models. To encode examples, we limited the investigation to what kind of text to encode (just the question, or the question and its entities and relations), without exploring different embedding models, similarity criteria or other input concatenation strategies. We leave these investigations to future work.

## Acknowledgments

This work was supported by the Cineca consortium, and it was partially funded by:

- PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 8 "Pervasive AI", funded by the European Commission under the NextGeneration EU programme.

- PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - Spoke 2 "FAIR - Future Artificial Intelligence Research" - "ReSpiRA - REplicabilità, SPIegabilità e Ragionamento", funded by the European Commission under the NextGeneration EU programme, CUP B43D22000900004 [7];

- "MAESTRO - Mitigare le Allucinazioni dei Large Language Models: ESTRazione di informazioni Ottimizzate" a project funded by Provincia Autonoma di Trento with the Lp 6/99 Art. 5: ricerca e sviluppo, PAT/RFS067-05/06/2024-0428372, CUP: C79J23001170001 [8].

[7]ReSPiRA: https://www.opencup.gov.it/portale/web/opencup/home/progetto/-/cup/B43D22000900004

[8]MAESTRO: https://www.opencup.gov.it/portale/web/opencup/home/progetto/-/cup/C79J23001170001

## References

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *Preprint*, arXiv:2306.04136.

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

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Rakesh Chada and Pradeep Natarajan. 2021. Fewshotqa: A simple framework for few-shot learning of question answering tasks using pre-trained text-to-text models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6081–6090.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N.

Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Xiusi Chen, Yu Zhang, Jinliang Deng, Jyun-Yu Jiang, and Wei Wang. 2023. Gotta: generative few-shot question answering by prompt-based cloze data augmentation. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pages 909–917. SIAM.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. 2022. Binding language models in symbolic languages. *arXiv preprint arXiv:2210.02875*.

Wentao Ding, Jinmao Li, Liangchuan Luo, and Yuzhong Qu. 2024. Enhancing complex question answering over knowledge graphs through evidence pattern retrieval. *Preprint*, arXiv:2402.02175.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning. *Preprint*, arXiv:2301.00234.

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. page 69–78, Berlin, Heidelberg. Springer-Verlag.

William Fedus, Jeff Dean, and Barret Zoph. 2022. A review of sparse expert models in deep learning. *Preprint*, arXiv:2209.01667.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A. Smith, and Luke Zettlemoyer. 2022. Demystifying prompts in language models via perplexity estimation. *Preprint*, arXiv:2212.04037.

Olaf Görlitz, Matthias Thimm, and Steffen Staab. 2012. Splodge: Systematic generation of sparql benchmark queries for linked open data. In *The Semantic Web – ISWC 2012*, pages 116–132, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. Lubm: a benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3:158–182.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Hyuhng Joon Kim, Hyunsoo Cho, Junyeob Kim, Taeuk Kim, Kang Min Yoo, and Sang goo Lee. 2022. Self-generated in-context learning: Leveraging autoregressive language models as a demonstration generator. *Preprint*, arXiv:2206.08082.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models. *Preprint*, arXiv:2310.11220.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. 2014. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6.

Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization. *Preprint*, arXiv:2212.06800.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18608–18616.

Yuanyuan Liang, Jianing Wang, Hanlun Zhu, Lei Wang, Weining Qian, and Yunshi Lan. 2023. Prompting large language models with chain-of-thought for few-shot knowledge base question generation. *arXiv preprint arXiv:2310.08395*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *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.

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies. *Preprint*, arXiv:2305.12586.

Ngonga Ngomo. 2018. 9th challenge on question answering over linked data (qald-9). *language*, 7(1):58–64.

Zhijie Nie, Richong Zhang, Zhongyuan Wang, and Xudong Liu. 2024. Code-style in-context learning for knowledge-based question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18833–18841.

Alisdair Owens, Nick Gibbins, and m.c Schraefel. 2008. Effective benchmarking for rdf stores using synthetic data.

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From freebase to wikidata: The great migration. In *Proceedings of the 25th international conference on world wide web*, pages 1419–1428.

Dmitrii Pliukhin, Daniil Radyush, Liubov Kovriguina, and Dmitry Mouromtsev. 2023. 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.

Jiexing Qi, Chang Su, Zhixin Guo, Luyuen Wu, Zanwei Shen, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. 2024. Enhancing sparql query generation for knowledge base question answering systems by learning to correct triplets. *Applied Sciences*, 14(4).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Federico Ranaldi, Elena Sofia Ruzzetti, Dario Onorati, Leonardo Ranaldi, Cristina Giannone, Andrea Favalli, Raniero Romagnoli, and Fabio Massimo Zanzotto. 2024. Investigating the impact of data contamination of large language models in text-to-sql translation. *arXiv preprint arXiv:2402.08100*.

Md Rashad Al Hasan Rony, Uttam Kumar, Roman Teucher, Liubov Kovriguina, and Jens Lehmann. 2022. Sgpt: A generative approach for sparql query generation from natural language questions. *IEEE Access*, 10:70712–70723.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *Preprint*, arXiv:1704.04368.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web*.

Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, et al. 2023. Qald-10–the 10th challenge on question answering over linked data. *Semantic Web*, (Preprint):1–15.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Yike Wu, Nan Hu, Sheng Bi, Guilin Qi, Jie Ren, Anhuan Xie, and Wei Song. 2023. Retrieve-rewrite-answer: A kg-to-text enhanced llms framework for knowledge graph question answering. *Preprint*, arXiv:2309.11206.

Gideon Zenz, Xuan Zhou, Enrico Minack, Wolf Siberski, and Wolfgang Nejdl. 2009. From keywords to semantic queries—incremental query construction on the semantic web. *Journal of Web Semantics*, 7(3):166–176. The Web of Data.

Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023. ACT-SQL: In-context learning for text-to-SQL with automatically-generated chain-of-thought. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3501–3532, Singapore. Association for Computational Linguistics.

Jianyun Zou, Min Yang, Lichao Zhang, Yechen Xu, Qifan Pan, Fengqing Jiang, Ran Qin, Shushu Wang, Yifan He, Songfang Huang, and Zhou Zhao. 2021. A chinese multi-type complex questions answering dataset over wikidata. *Preprint*, arXiv:2111.06086.

## A Qualitative Analysis

In this appendix we provide some qualitative analyses of DFSL and DFSL-MQ. First of all, we report some examples in Table 6 that highlight the benefits from introducing similar examples with DFSL with respect to standard few-shot learning approach.

Then, we show some examples in Table 7 where the multi-query approach solves triple-flip errors. In Table 5 instead, we showcase errors caused by employing LS answer selection heuristic. Notably, by choosing larger sets, LS sometimes selects 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.

## B Few-shot Learning Examples

We report in Figure 4 the examples selected for the Few-shot learning baseline prompt. The five examples were chosen to be the most representative of the training set, including queries of different kind and structure, such as ASK, COUNT and SELECT.

| Question | Target Query | Answer Selection | 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: Qualitative comparison between different answer selection strategies in DFSL-MQ.

| Question | Target Query | Approach | Predicted Query | Similar In-context Example |
|---|---|---|---|---|
| Who is the daughter of Robert Kennedy married to? | `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 . }` | `SELECT DISTINCT ?uri WHERE { wd:Q43247 wdt:P40 ?child . ?child wdt:P26 ?uri . }` |
| Which countries are connected by the Rhine? | `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 . }` | SELECT DISTINCT ?res WHERE { wd:Q3392 wdt:P885/wdt:P17 ?res . } |
| Give me the capitals of all countries in Africa. | `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 . ?country wdt:P30 wd:Q15 . ?country wdt:P36 ?uri . }` | SELECT DISTINCT ?uri WHERE { wd:Q5451 wdt:P17 ?country . ?country wdt:P36 ?uri . } |

Table 6: A qualitative comparison between DFSL and Few-shot Learning. Thanks to the similar in-context examples retrieval, the LLMs generates the correct query.

| Question | Target Query | Approach | Predicted Query |
|---|---|---|---|
| Who is the enclave within of Montreal? | `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 }` |
| The trachea is of what anatomical branch? | `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 { ?an swer wdt:P3261 wd:Q175449}` |
| What revolution caused the destruction of the Russian Empire? | `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 7: Some triple-flip errors that are solved by DFSL-MQ.

```
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)

Query:
<SPARQL>
PREFIX wdt: <http://www.wikidata.org/prop/direct/> PREFIX wd: <http://www.wikidata.org/entity/> ASK WHERE { wd:Q166262 wdt:P161 wd:Q45772 }
</SPARQL>
```

Figure 4: Examples injected in the Few-shot-learning prompt.