FIRESPARQL: A LLM-Based Framework for SPARQL Query Generation over Scholarly Knowledge Graphs

Xueli Pan[a], Victor de Boer[b] and Jacco van Ossenbruggen[c]
Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands

Keywords: Scholarly Knowledge Graph, SPARQL Query Generation, Finetuning LLM.

Abstract: Question answering over Scholarly Knowledge Graphs (SKGs) remains a challenging task due to the complexity of scholarly content and the intricate structure of these graphs. Large Language Model (LLM) approaches could be used to translate natural language questions (NLQs) into SPARQL queries; however, these LLM-based approaches struggle with SPARQL query generation due to limited exposure to SKG-specific content and the underlying schema. We identified two main types of errors in the LLM-generated SPARQL queries: (i) structural inconsistencies, such as missing or redundant triples in the queries, and (ii) semantic inaccuracies, where incorrect entities or properties are shown in the queries despite a correct query structure. To address these issues, we propose FIRESPARQL, a modular framework that supports fine-tuned LLMs as a core component, with optional context provided via retrieval-augmented generation (RAG) and a SPARQL query correction layer. We evaluate the framework on the SciQA Benchmark using various configurations (zero-shot, zero-shot with RAG, one-shot, fine-tuning, and fine-tuning with RAG) and compare the performance with baseline and state-of-the-art approaches. We measure query accuracy using BLEU and ROUGE metrics, and execution result accuracy using relaxed exact match(RelaxedEM), with respect to the gold standards containing the NLQs, SPARQL queries, and the results of the queries. Experimental results demonstrate that fine-tuning achieves the highest overall performance, reaching 0.90 ROUGE-L for query accuracy and 0.85 RelaxedEM for result accuracy on the test set.

1 INTRODUCTION

Question Answering (QA) over Knowledge Graphs (KGs), which allows users to query structured data using natural language, has gained considerable attention (Huang et al., 2019; Omar et al., 2023). The task of QA over KGs usually takes a natural language question (NLQ) as an input and translates it into formal queries, typically SPARQL, that retrieve precise answers from the underlying KG. Previous studies in this domain have been centered around large-scale and encyclopedic KGs such as DBpedia, Freebase, and Wikidata. In these generic KGs, QA systems benefit from extensive community resources, well-documented schema, and relatively simple entity-relation structures. Recently, the emergence of large language models (LLMs) has inspired a growing body of research exploring their potential to address the task of QA over KGs (Wang et al., 2024b; Omar et al., 2023) and benchmarked on datasets such as LC-QuAD (Trivedi et al., 2017), QALD (Perevalov et al., 2022), and WebQuestions (Berant et al., 2013).

However, applying these techniques to QA to Scholarly Knowledge Graphs (SKGs) presents significant challenges due to the intricate nature of scholarly data and the complex structure of SKGs (Jiang et al., 2023; Pliukhin et al., 2023; Taffa and Usbeck, 2023). Unlike encyclopedic KGs, SKGs capture domain-specific and technical content such as research contributions, research problems, methodologies, datasets, and evaluation, which are often represented in complex ontological structures. Several studies have investigated the potential of using LLMs for this task, exploring optimization techniques such as zero-shot learning, few-shot learning, and fine-tuning (Taffa and Usbeck, 2023; Lehmann et al., 2024). Despite the improvements of LLMs on the task of QA over SKGs, LLMs face limitations when handling KG-specific parsing due to their lack of direct access to entities within the KGs and insufficient understanding of the ontological schema, partic-

[a] https://orcid.org/0000-0002-3736-7047
[b] https://orcid.org/0000-0001-9079-039X
[c] https://orcid.org/0000-0002-7748-4715

Pan, X., de Boer, V. and van Ossenbruggen, J.
FIRESPARQL: A LLM-Based Framework for SPARQL Query Generation over Scholarly Knowledge Graphs.
DOI: 10.5220/0013774000004000

In *Proceedings of the 17th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2025) - Volume 1: KDIR*, pages 123-134

ularly for low-resource SKGs like the Open Research Knowledge Graph (ORKG)(Auer et al., 2023).

Insights from our pilot experiment revealed two major categories of errors LLMs tend to make in this task: (i) Structural inconsistencies, where generated SPARQL queries contain missing or redundant triples, and (ii) Semantic inaccuracies, where queries reference incorrect entities or properties, despite following the correct structural form. To address these limitations, we propose FIRESPARQL, a LLM-based modular framework for SPARQL query generation over SKGs. At its core, FIRESPARQL supports FIne-tuned LLMs adapted to the SKG domain and offers relevant context provided via REtrieval-augmented generation (RAG) and a lightweight SPARQL correction layer. These components are designed to improve both the structural and semantic accuracy of the generated queries.

We investigate the effectiveness of this framework using the SciQA Benchmark (Auer et al., 2023), comparing multiple configurations, including zero-shot, one-shot, and fine-tuned models with and without RAG, against baselines and state-of-the-art methods. We assess performance based on BLEU and ROUGE scores for SPARQL query accuracy, and use a relaxed Exact Match metric to evaluate the execution accuracy of the returned query results. Our findings demonstrate that domain-specific fine-tuning yields the most consistent and robust performance, significantly enhancing both query accuracy and execution result accuracy. Notably, the best-performance configuration is fine-tuned LLaMA3-8B-Instruct with 15 training epochs, achieving 0.77, 0.91, 0.86, 0.90, and 0.85 on BLEU-4, ROUGE-1, ROUGE-2, ROUGE-L, and RelaxedEM(all), respectively. However, our experiments reveal that incorporating RAG into either the zero-shot or fine-tuned model does not yield further improvements and can even degrade performance.

The main contributions of this paper are threefold: (1) We identify and systematically categorize the common error types in LLM-generated SPARQL queries for QA over SKGs, distinguishing between structural inconsistencies and semantic inaccuracies. (2) We propose FIRESPARQL, a modular framework for SPARQL query generation that integrates a core fine-tuned LLM with an optional RAG module and a lightweight SPARQL correction layer. (3) We conduct comprehensive experiments on the SciQA Benchmark under multiple configurations, including zero-shot, one-shot, fine-tuning, and their RAG-augmented variants, benchmarking against baselines and state-of-the-art methods using different model sizes and training epochs.

All resources and codes are available in our GitHub repository [1]. For reproducibility, we have released the best-performing fine-tuned model—LLaMA-3-8B-Instruct trained for 15 epochs—on Hugging Face. [2].

2 RELATED WORK

2.1 Traditional Methods for QA over KGs

Before the emergence of LLMs, QA over KGs is primarily addressed through knowledge graph embedding(KGE), neural network modeling, and reinforcement learning (RL). These methods typically relied on modeling the structure of the KG and carefully engineered the features of the KG for entity linking, relation prediction, and path ranking. KGE-based approaches transform entities and relations into low-dimensional vector spaces to support efficient reasoning. Huang et al. (Huang et al., 2019) propose the KEQA framework for answering the most common types of questions by jointly recovering the question's head entity, predicate, and tail entity representations in the KG embedding spaces. Graph neural networks (GNNs) have also been leveraged to reason over the structure of KGs. Yasunaga et al. (Yasunaga et al., 2021) introduce QA-GNN, leveraging joint reasoning, where the QA context and KG are connected to form a joint graph and mutually update their representation through GNNs. Some studies emphasized RL to navigate the KG and identify answer paths. Hai et al. (Cui et al., 2023) propose AR2N, an interpretable reasoning method based on adversarial RL for multi-hop KGQA, to address the issue of spurious paths. AR2N consists of an answer generator and a path discriminator, which could effectively distinguish whether the reasoning chain is correct or not.

2.2 QA over Generic KGs Such as DBpedia and Wikidata

QA over generic Knowledge Graphs (KGs), such as DBpedia and Wikidata, has been extensively studied and serves as the foundation for many advances in the field. Early systems primarily focused on semantic parsing and graph-based reasoning (Berant et al., 2013; Trivedi et al., 2017). More recently,

[1] https://github.com/sherry-pan/FIRESPARQL.git

[2] https://huggingface.co/Sherry791/Meta-Llama-3-8B-Instruct-ft4sciqa

attention has shifted to neural and LLM-based approaches. Bustamante and Takeda (Bustamante and Takeda, 2024) explore the use of entity-aware pretrained GPT models for SPARQL generation, showing notable improvements in handling KG-specific structures. Similarly, Hovcevar and Kenda (Hovcevar and Kenda, 2024) integrate LLMs with KGs in industrial settings to support natural language interfaces. Meyer et al. (Meyer et al., 2024) assess the SPARQL generation capabilities of various LLMs, pointing out both strengths and limitations in terms of structural correctness and execution accuracy. Other studies such as Kakalis and Kefalidis (Kakalis and Kefalidis, 2024) focus on domain-specific extensions, like GeoSPARQL, and propose techniques for automatic URI injection. All in all, these works contribute a rich landscape of methods for mapping natural language questions to structured SPARQL queries across diverse knowledge bases.

2.3 SPARQL Generation for QA over SKGs

QA over SKGs, such as the Open Research Knowledge Graph (ORKG), has gained attention due to its potential to support scientific exploration and knowledge discovery. Unlike generic knowledge graphs, SKGs capture fine-grained scholarly information, which introduces additional complexity in terms of schema diversity and domain-specific terminology. With the advent of LLMs, there has been a surge of interest in applying these models on the task of QA over SKGs. Lehmann et al. (Lehmann et al., 2024) conduct a comprehensive evaluation on the effectiveness of LLMs on the SciQA benchmark, demonstrating the models' strengths in generating fluent SPARQL queries but also noting common pitfalls such as entity disambiguation and schema misalignment. Taffa and Usbeck (Taffa and Usbeck, 2023) specifically focus on adapting LLMs to the scholarly setting, emphasizing the need for domain-specific prompts and training data. Meanwhile, Pliukhin et al. (Pliukhin et al., 2023) explore fine-tuning strategies that improve LLM performance in one-shot scenarios. These studies collectively suggest that while LLMs offer promising capabilities, their effectiveness in the scholarly domain hinges on adaptation through fine-tuning, prompt engineering, and schema-aware correction mechanisms.

3 ERROR TYPE ANALYSIS ON GENERATED SPARQL QUERIES

Despite the improvements of LLMs on QA over SKGs, LLMs face limitations when handling KG-specific parsing. The experimental results conducted by Sören Auer et al.(Auer et al., 2023) showed that only 63 out of 100 handcrafted questions could be answered by ChatGPT, of which only 14 answers were correct. To better understand why LLMs fail to generate the correct SPARQL query to a NLQ, we conduct a pilot experiment on using ChatGPT(GPT-4) with a random one-shot example to generate SPARQL queries for 30 handcrafted questions in the SciQA benchmark datasets.

Insights from this pilot experiment revealed two major categories of errors LLMs tend to make in this task: semantic inaccuracies and structural inconsistencies. Semantic inaccuracies occur when LLMs fail to link the correct properties and entities in ORKG, despite generating SPARQL queries with the correct structure. Our observations reveal that LLMs tend to rely on the example provided in the one-shot learning process to generate the correct structure for a certain type of question, but often struggle with linking the correct properties and entities because LLMs do not learn the content of the underlying KG. Structural inconsistencies arise due to LLMs' lack of an ontological schema of the underlying KG, leading to errors in query structure, such as missing or abundant links (triples), despite correctly linking to the mentioned entities or properties.

Figure 1 shows the example of the semantic inaccuracies and structural inconsistencies problem with the generated SPARQL queries in our pilot study. In the example of the semantic inaccuracies problem, ChatGPT failed to link the correct property orkgp:P15687; instead, it linked to a wrong property orkgp:P7101. In the example of the structural inconsistencies problem, the SPARQL query generated by ChatGPT directly links Contribution to Metrics and fails to detect the correct schema of the ORKG where Contribution and Metric are connected via Evaluation.

4 METHODOLOGY

As we mentioned in Section 3, generating executable and semantically accurate SPARQL queries over SKGs using LLMs remains a challenging task due to two main types of errors: semantic inaccuracies and

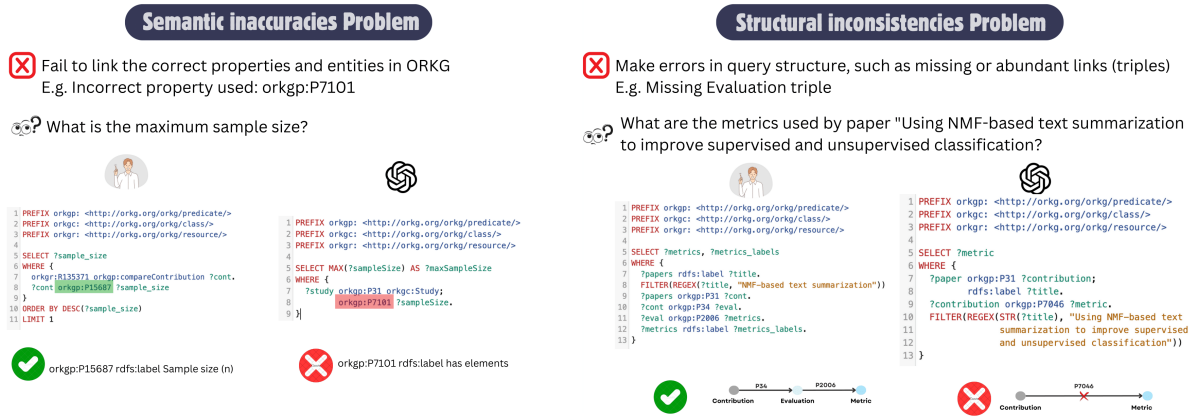

Figure 1: Examples of semantic inaccuracies and structural inconsistencies problem with the generated SPARQL queries.

structural inconsistencies. To address these issues, we propose FIRESPARQL, a modular framework designed to improve both the semantic accuracy and the structural consistency of generated SPARQL queries. The framework consists of three core components: (1) Fine-tuned LLMs, (2) Retrieval-Augmented Generation (RAG), and (3) SPARQL correction. The final SPARQL queries are evaluated at both the query accuracy and execution result accuracy using ground truth comparisons. An overview of the framework and the evaluation setup is shown in Figure 2.

4.1 Fine-Tuning

At the core of FIRESPARQL is a fine-tuning module applying Low-Rank Adaptation (LoRA) (Hu et al., 2021) for parameter-efficient fine-tuning. Unlike full-parameter fine-tuning, LoRA freezes the pre-trained model weights and injects trainable low-rank decomposition matrices into each layer of the Transformer architecture. This approach significantly reduces the number of trainable parameters required for downstream tasks while maintaining strong performance. Fine-tuning LLMs has proven effective in scientific knowledge extraction (Muralidharan et al., 2024) and KG construction (Ghanem and Cruz, 2024; Wang et al., 2024a).

To address structural inconsistencies in generated SPARQL queries, often arising from a limited understanding of the SKG schema, we fine-tune LLMs so that the ontology and structural patterns of the underlying SKG are implicitly captured during fine-tuning. The fine-tuning data can include the ontology descriptions, RDF triples, or task-specific labeled examples. In our implementation, we use NLQ-SPARQL query pairs as training data, which implicitly encode the structure and vocabulary of the target SKG. This results in a fine-tuned LLM capable of generating syntactically correct and semantically meaningful SPARQL queries directly from natural language questions.

We further investigate the impact of training epochs on fine-tuning performance. The number of epochs determines how many times the model iterates over the training data, directly influencing its ability to capture domain-specific patterns. The prompt template for SPARQL generation is shown in Listing 1.

Listing 1: Prompt template for SPARQL generation.

```
The Open Research Knowledge Graph (
    ORKG) is a semantic knowledge
    graph designed to represent,
    compare, and retrieve scholarly
    contributions. Given a natural
    language question in English,
    your task is to generate the
    corresponding SPARQL query to
    this question. The generated
    SPARQL query should be able to
    query the ORKG, getting correct
    answer to the input question.
Give me only the SPARQL query, no
    other text.
Input question: {input question}
Output SPARQL query:
```

4.2 RAG

RAG (Lewis et al., 2020) has been proposed to enable LLMs access to external and domain-specific knowledge for knowledge-intensive NLP tasks, which could be a promising way to address the issue of semantic inaccuracies, where generated SPARQL queries fail to link to the correct properties or entities. These inaccuracies often stem from the model's limited expo-

sure to SKGs or ambiguous entity/property mentions in the input question. Therefore, we propose an optional RAG module in the framework to enhance the model's contextual understanding of the underlying SKGs. Given an NLQ, relevant context is retrieved from the SKG in the form of candidate entities, properties, or subgraphs. A prompt template then incorporates this contextual information alongside the input question, which is passed to the finetuned LLM to generate a more semantically accurate SPARQL query. In our implementation, we use RAG to retrieve candidate properties from a curated list of distinct ORKG properties, including their URLs and labels. These candidates are then incorporated as contextual information in the prompt template to guide SPARQL generation.

4.3 SPARQL Corrector

Despite improvements through fine-tuning and RAG, generated SPARQL queries may still contain minor structural or syntactic errors that hinder successful execution. These include unnecessary text in the output, missing or extra punctuation, or subtle syntax issues such as missing spaces between variable names. To address this, we introduce a lightweight SPARQL correction layer based on LLMs. This module takes the initially generated query and its natural language question as input and refines it to ensure syntactic validity, which increases the likelihood of generating executable SPARQL queries. The cleaned queries are then passed to the evaluation stage. The prompt we used to clean the input SPARQL query is shown in Listing 2.

Listing 2: Prompt for SPARQL correction.

```
Given a question and its
    corresponding SPARQL query,
    there might be errors in the
    query such as missing spaces
    between variable names,
    unnecessary repetition, etc.
Please clean the SPARQL and return
    only the cleaned SPARQL (no
    explanation):
question: {Natural language question}
query: {Generated SPARQL query}
```

5 EXPERIMENTS

5.1 Datasets

We conduct experiments on the SciQA benchmark dataset, a recently released resource designed to evaluate question answering systems over scholarly knowledge graphs (Auer et al., 2023). SciQA provides a diverse set of natural language questions aligned with the Open Research Knowledge Graph (ORKG). SciQA contains 100 handcrafted natural language questions (HQs) with paraphrases, corresponding SPARQL queries with their results. In addition, a set of 2465 questions (AQs) has been semi-automatically derived from eight question templates. The RDF data dump is also released on Zenodo [3] together with the benchmark.

We choose SciQA because the underlying KG for this dataset, the ORKG, is built for representing and querying the semantic content of research papers, not just their metadata. Unlike DBLP, which mainly provides bibliographic metadata such as citation and authorship information, ORKG captures fine-grained scientific statements in a machine-readable format, enabling fact-based, content-level question answering. ORKG goes beyond citation networks and authorship details to represent fine-grained scientific claims, methodologies, comparisons, and findings. This semantic depth enables fact-centric and content-level QA, making SciQA a more suitable benchmark for evaluating systems that aim to reason about scientific knowledge.

5.2 Baselines and the State of the Art

As a baseline, we adopt a zero-shot setting in which the model generates SPARQL queries without any task-specific fine-tuning or example guidance. This setup evaluates the model's out-of-the-box ability to understand the task and map natural language questions to structured SPARQL queries. For the state-of-the-art comparison, we implement the one-shot approach in which the model is provided with the most semantically similar question from the training set, along with its corresponding SPARQL query, as an in-context demonstration. The most similar example is identified by computing cosine similarity between the input question and all training questions using Sentence-BERT embeddings. To find the most similar question from the training set, Sentence-BERT is better than vanilla BERT because it produces sentence embeddings optimized for semantic similarity.

[3] https://zenodo.org/records/7744048

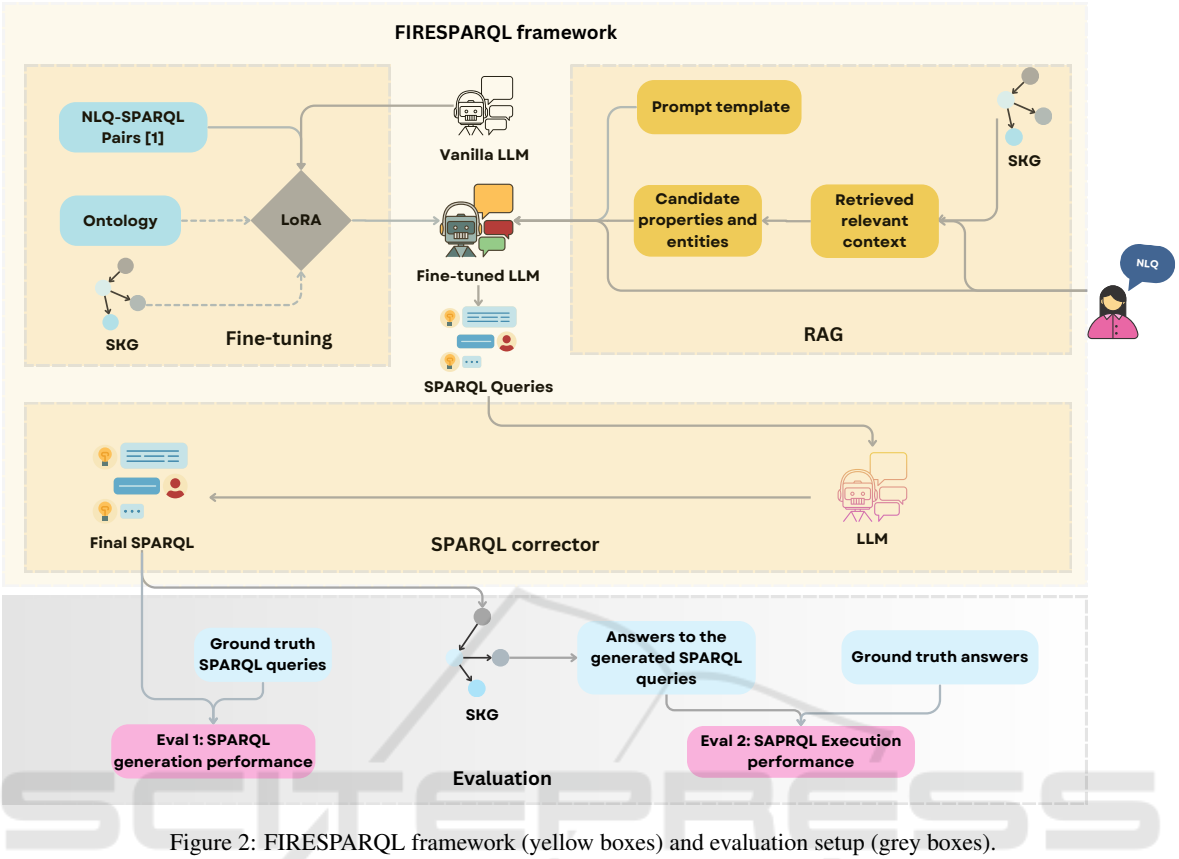

Figure 2: FIRESPARQL framework (yellow boxes) and evaluation setup (grey boxes).

This configuration has shown strong performance in recent studies by helping the model better understand the tasks (Lehmann et al., 2024; Liu et al., 2022).

5.3 Implementation

We fine-tuned two instruction-tuned models, Llama 3.2-3B Instruct and Llama 3-8B Instruct, using 1,795 NLQ-SPARQL pairs from the SciQA training set. The models were trained under various epoch configurations (3, 5, 7, 10, 15, and 20) to analyze performance across training durations. All fine-tuning experiments were conducted on a single NVIDIA H100 GPU. We used DeepSeek-R1-Distill-Llama-70B as the underlying model for the RAG component. All SPARQL queries are executed over the ORKG data dump released together with the benchmark on Zenodo using Qlever (Bast and Buchhold, 2017)

The execution times for fine-tuning the two Llama models across different numbers of epochs are summarized in Table 3 in the Appendix.

5.4 Evaluation Metrics

We employ a combination of string-based and execution-based evaluation metrics to assess the quality of the generated SPARQL queries. Similar to other research, we use BLEU-4 and ROUGE scores, which measure token-level and n-gram overlaps, to evaluate the similarity between generated queries and the ground-truth queries. These metrics provide insights into how closely the structure and content of the generated queries align with the reference queries. Additionally, we assess the execution performance of the generated SPARQL queries using two variants of Relaxed Exact Match (RelaxedEM): success and all. The RelaxedEM(success) metric considers only those queries that were syntactically valid, successfully executed against the ORKG RDF dump, and returned non-empty results. In contrast, the RelaxedEM(all) metric evaluates the correctness of the query results across the entire test set, including queries that may have failed or returned empty results.

Unlike the original Exact Match, which is very strict, our Relaxed Exact Match incorporates several preprocessing steps. First, we remove variable names from the returned results to avoid penalizing differ-

ences that do not affect semantics. Second, we split the results line by line and eliminate duplicate lines to normalize the output structure. Finally, we compare the set of lines in the results using exact matching. The metric RelaxedEM provides a more tolerant and realistic evaluation of query execution performance in scholarly knowledge graph settings. The above-mentioned dual evaluation approach allows us to comprehensively analyze both the syntactic quality and the practical effectiveness of the generated SPARQL queries.

5.5 Results

Table 1 shows the results of different metrics on different strategies, different models, and different epochs. Figure 3 shows the results of different metrics with different epochs for LoRA fine-tuning. All the scores are the average scores of three runs. The standard deviation of BLEU-4, ROUGE scores, RelaxedEM(success), and RelaxedEM(all) across different model variants (e.g., zero-shot, one-shot, fine-tuning, and RAG) and training epochs is consistently low ($std < 0.0265$), indicating stable performance across all three runs. Therefore, we use the average scores as a reliable and representative summary of model effectiveness. In the next section, we discuss the main takeaways from these results.

6 DISCUSSION

In this section, we present the key experimental findings and outline the limitations and directions for future work.

6.1 Discussion on the Results

Our experimental findings provide several key insights into the effectiveness of the FIRESPARQL framework and the performance of different strategies for SPARQL query generation with different epoch settings and different model sizes.

6.1.1 Fine-Tuning Performance

As shown in Table 1, fine-tuning LLMs on NLQ–SPARQL pairs leads to significant improvements over both the zero-shot baseline and the one-shot state-of-the-art methods. The highest performance is achieved by the fine-tuned LLaMA-3-8B-Instruct model trained for 15 epochs, attaining scores of 0.77 (BLEU-4), 0.91 (ROUGE-1), 0.86 (ROUGE-2), 0.90 (ROUGE-L), and 0.85 (RelaxedEM on all test cases). These results indicate that, across both query-level (BLEU, ROUGE) and execution-level (RelaxedEM) evaluations, SPARQL queries generated by finetuned models are not only more accurate but also structurally well-formed and executable. This highlights the effectiveness of supervised adaptation for learning the ontology and structure of the underlying SKG during training.

6.1.2 Model Size Impact

As shown in Fig 3, LLaMA-3-8B-Instruct consistently outperforms LLaMA-3.2-3B-Instruct after fine-tuning across all evaluation metrics. This demonstrates that larger model capacity enhances the ability to internalize domain-specific patterns from training data, including the structure and semantics of the target SKG. Interestingly, the trend reverses in the one-shot setting: LLaMA-3.2-3B-Instruct performs better than the 8B variant on most metrics, except for RelaxedEM(success), as shown in Table 1. This performance gap might be attributed to the fact that LLaMA-3.2-3B-Instruct was released after LLaMA-3-8B-Instruct and incorporates pruning and distillation techniques, which were specifically applied to the 1B and 3B variants (lla,). These techniques help to preserve performance while significantly improving efficiency, making the LLaMA-3.2-3B-Instruct model capable of strong instruction-following performance on resource-constrained devices. As a result, despite its smaller size, LLaMA-3.2-3B-Instruct may benefit from a more refined architecture and training strategies, allowing it to better leverage in-context examples in one-shot settings compared to the larger, but earlier, 8B model.

6.1.3 RAG Performance

As shown in Table 1, the score of RelaxedEM(all) drops from 0.85 to 0.29 when incorporating RAG into the fine-tuned LLaMA-3-8B-Instruct trained for 15 epochs, which does not lead to additional performance gains, and it even degrades the task performance. This decline can be attributed to the noisy or misaligned nature of the retrieved context, such as incorrect or irrelevant property suggestions from the ORKG, which may introduce confusion instead of providing useful guidance since we don't have a context checker to validate whether the context is relevant or not. Prior studies (Jin et al., 2024; Joren et al., 2024) have similarly observed that low-quality RAG context can conflict with the knowledge already encoded in fine-tuned models, ultimately leading to reduced task performance.

Table 1: BLEU, ROUGE and RelaxedEM scores with different strategies, different models and different epochs.

Strategy	Model	Epoch	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	RelaxedEM (success)	RelaxedEM (all)
zero-shot (baseline)	llama-3.2-3b-Instruct	-	0.03	0.36	0.18	0.35	0.00	0.00
	llama-3-8b-Instruct	-	0.03	0.39	0.18	0.38	0.00	0.00
zero-shot_rag	llama-3.2-3b-Instruct	-	0.03	0.36	0.18	0.36	0.00	0.00
	llama-3-8b-Instruct	-	0.03	0.36	0.18	0.38	0.00	0.00
one-shot (SOTA)	llama-3.2-3b-Instruct	-	0.58	0.81	0.73	0.78	0.78	0.40
	llama-3-8b-Instruct	-	0.38	0.61	0.50	0.59	0.89	0.29
ft_rag	llama-3.2-3b-Instruct-lora_ deepseekr1-distill-llama-70b	20	0.32	0.63	0.51	0.60	0.42	0.06
	llama3-8b-Instruct-lora_ deepseekr1-disttill-llama-70b	15	0.58	0.81	0.72	0.77	0.85	0.29
ft	llama-3.2-3b-Instruct-lora	3	0.67	0.86	0.79	0.83	0.88	0.53
		5	0.62	0.83	0.75	0.79	0.71	0.36
		7	0.52	0.77	0.67	0.73	0.79	0.27
		10	0.56	0.79	0.70	0.76	0.70	0.22
		15	0.60	0.81	0.73	0.78	0.88	0.40
		20	**0.70**	**0.86**	**0.79**	**0.84**	**0.80**	**0.54**
	llama-3-8b-Instruct-lora	3	0.75	0.90	0.84	0.88	0.99	0.79
		5	0.74	0.90	0.85	0.87	0.98	0.73
		7	0.70	0.87	0.80	0.84	0.96	0.69
		10	0.71	0.88	0.81	0.85	0.94	0.69
		15	**0.77**	**0.91**	**0.86**	**0.90**	**0.98**	**0.85**
		20	0.74	0.89	0.84	0.88	0.99	0.82

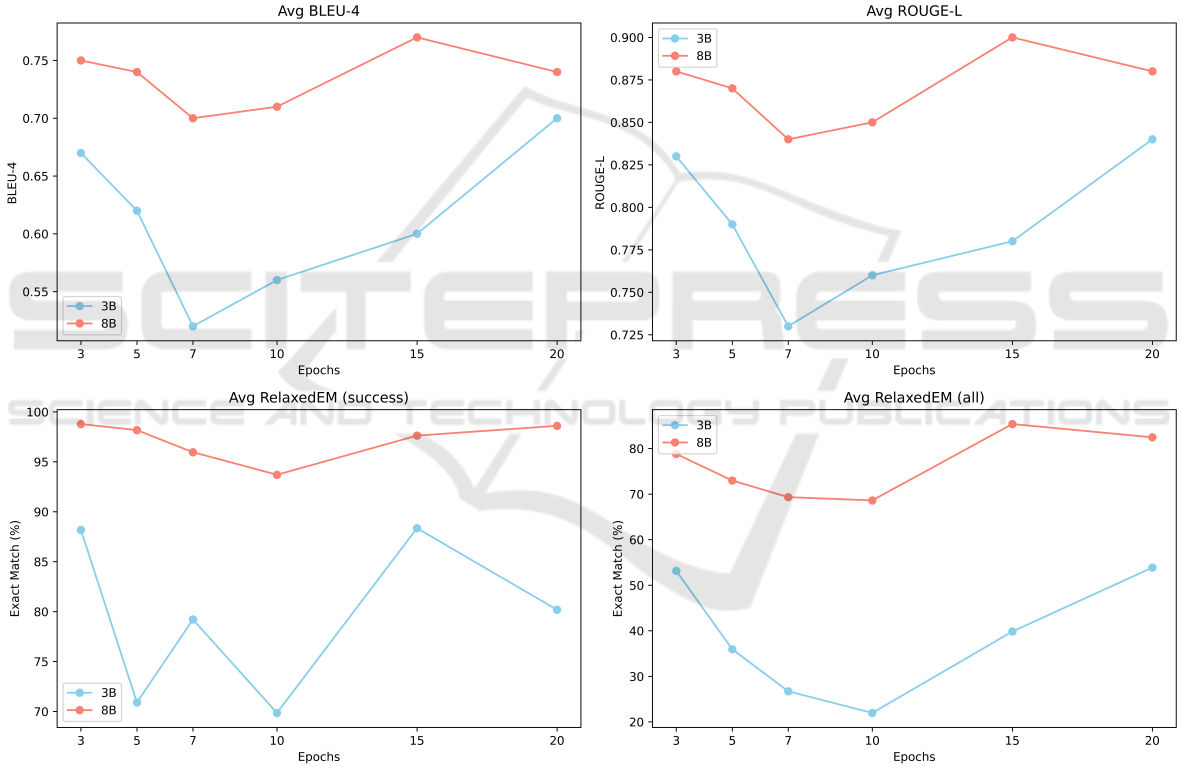

Figure 3: Average BLEU-4, ROUGE-L and RelaxedEM scores with different epochs on different fine-tuned models.

6.1.4 One-Shot Performance

As shown in Table 1, the one-shot setting, using the most similar example from the training set, achieved strong performance, second only to the fine-tuned models. Specifically, the one-shot approach reached scores of 0.58 (BLEU-4), 0.81 (ROUGE-1), 0.73 (ROUGE-2), 0.78 (ROUGE-L), and 0.40 (RelaxedEM(all)) on LLaMA-3.2-3B-Instruct model. Compared to the best-performing fine-tuned model, the one-shot approach achieved comparable performance on query accuracy. However, in terms of execution accuracy, it lagged behind, with a RelaxedEM(all) score of 0.40, substantially lower than the 0.85 achieved by the fine-tuned LLaMA-3-8B-Instruct model. These results suggest that while fine-tuning remains essential for maximizing execution accuracy, one-shot learning provides a simple yet effective alternative in scenarios where high-quality fine-tuning datasets are unavailable.

6.1.5 SPARQL Corrector Performance

The integration of a SPARQL corrector layer significantly improved the syntactic validity and execution accuracy of queries generated by the fine-tuned LLaMA models. As illustrated by the examples provided in the Appendix, the raw outputs often contained structural and formatting errors, such as missing whitespace between keywords (e.g., *SELECT*?*model*?*model_lbl*). This issue leads to syntax errors and failed executions when submitted to the QLever endpoint. The corrector layer effectively resolved these inconsistencies by standardizing spacing, punctuation, and query structure—ensuring compliance with SPARQL syntax. This post-processing step is particularly valuable in handling complex query patterns involving nested clauses and aggregation. Consequently, we observed a notable improvement in the execution success rate and semantic alignment with the reference queries. These results underscore the importance of incorporating lightweight correction mechanisms alongside fine-tuned models for reliable SPARQL query construction.

6.1.6 Training Epoch Sensitivity

As shown in Figure 3, the number of fine-tuning epochs has a significant impact on all metrics for both LLaMA-3.2-3B-Instruct and LLaMA-3-8B-Instruct models. First, both models start with high scores on all metrics at epoch 3 and then slightly decline during the early training phases (epochs 3–7), suggesting that the models may require sufficient training time to properly internalize the SKG-specific structures and semantics. Second, the training dynamics reveal an upward trend in performance from 7 epochs onward, with the best performance at 20 epochs for the 3B model and 15 epochs for the 8B model. This indicates that larger models tend to converge faster and exhibit stronger generalization early on, while smaller models require more epochs to achieve competitive performance.

6.1.7 Performance on HQs and AQs

The execution results in Table 2 reveal a clear performance gap between handcrafted questions (HQs) and auto-generated questions (AQs) on query execution accuracy. For AQs, the best-performing fine-tuned model achieves very strong performance, with 438 queries falling into the `success-execution_non-empty_exact-match_1` category, indicating successful execution, non-empty results, and perfect alignment with the ground-truth answers. In contrast, none of the HQs achieved this level of exact matching. Instead, most generated queries for HQs either produced empty results despite being syntactically valid (14.67 on average across three runs) or only partially overlapped with the ground-truth anwsers (5.67). A small number of the generated queries for HQs (0.67) also failed due to syntax errors. These findings suggest that while the model generalizes effectively to the more uniform, template-based structure of AQs, it faces significant challenges when dealing with the semantic richness of HQs. In particular, the large proportion of empty results for HQs highlights their susceptibility to semantic inaccuracies: even when the generated queries are structurally sound, the model often fails to correctly identify or link the intended entities and properties, resulting empty answers. This reinforces the earlier observation by Sören Auer et al.(Auer et al., 2023) that HQs pose a more realistic and demanding test of model robustness, exposing limitations that remain hidden when evaluating solely on synthetic, template-driven data.

Table 2: Query execution results on HQs and AQs across three runs with the best-performed fine-tuned model.

	#HQ	#AQ	Total
test set	21	492	513
sucess-execution_non-empty_exact-match_1	0	438	438
sucess-execution_non-empty_exact-match_0	5.67	5.00	10.67
sucess-execution_empty	14.67	34.00	48.67
fail-execution-syntax-error	0.67	15.00	15.67

6.2 Error Analysis on Failed SPARQL Queries

We further analyzed the generated SPARQL queries that either failed to execute or returned empty results, by comparing them with the corresponding ground truth queries. Under the best-performing configuration—using the fine-tuned `LLaMA3-8B-Instruct` model trained for 15 epochs—448 out of 513 generated SPARQL queries were executed successfully via QLever without any syntax errors and returned meaningful results. Meanwhile, 14 queries failed due to syntax errors, and 51 queries were executed successfully but returned empty results.

To better understand the causes of failure, we examined the error messages for the 14 syntactically invalid queries and inspected the queries that returned empty results. Our analysis revealed that 11 out of the 14 syntactically invalid queries shared the same error message:

```
Invalid SPARQL query: Variable ?metric
is selected but not aggregated. All
non-aggregated variables must be part
of the GROUP BY clause. Note: The
GROUP BY in this query is implicit
because an aggregate expression was
```

```
used in the SELECT clause.
```

This indicates that these queries included aggregate functions (e.g., `MAX(?value)`) in the `SELECT` clause but did not include the non-aggregated variables (e.g., `?metric`, `?metric_lbl`) in a `GROUP BY` clause. In SPARQL 1.1, such usage is invalid unless all non-aggregated variables are explicitly grouped. This reflects a lack of adherence to SPARQL's syntax rules around aggregation and grouping.

The remaining 3 queries failed with the following error:

```
Invalid SPARQL query: Token 'SELECT':
mismatched input 'SELECT' expecting
'}'.
```

This indicates that the queries contained improperly structured subqueries. Specifically, full `SELECT` statements nested directly inside a `WHERE` clause without being enclosed in curly braces (`{}`). SPARQL requires subqueries to be syntactically isolated within their own scope using curly braces. These errors likely stem from incorrect handling of nested query structures during generation.

These findings highlight the current limitations of fine-tuned LLMs in capturing the formal syntactic constraints of the SPARQL query language, particularly in scenarios involving nested subqueries and aggregation functions. As a potential extension to our approach, prompt engineering techniques that include explicit syntax error examples or constraint reminders could be incorporated during SPARQL generation to encourage the model to produce syntactically valid SPARQL, especially for complex constructs like aggregation and subqueries.

6.3 Limitations and Future Work

While FIRESPARQL demonstrates strong performance in generating SPARQL queries over the ORKG, several limitations remain, which also highlight directions for future research: Our current experiments are limited to a single domain-specific benchmark, SciQA, which is built on top of the ORKG. To assess the generalizability of our approach, it is crucial to evaluate FIRESPARQL on a broader range of benchmarks across different domains and knowledge graphs. This would help determine whether the observed improvements are transferable or highly task-specific. Although our framework includes an optional RAG module, its effectiveness is currently hindered by the quality of the retrieved context. In many cases, irrelevant or incorrect candidate properties are introduced, leading to performance degradation. Future work should focus on developing more accurate and semantically aware retrieval mechanisms that can provide high-quality, contextually relevant information—such as topological subgraphs or query templates—without introducing noise. FIRESPARQL relies on supervised fine-tuning using NLQ–SPARQL pairs, which may not always be available. In future work, we aim to explore alternative data for fine-tuning LLMs such as synthetic training data or leveraging weak supervision from ontology or subgraphs.

7 CONCLUSION

In this paper, we introduced FIRESPARQL, a modular framework for SPARQL query generation over SKGs. By systematically analyzing common error types, structural inconsistencies, and semantic inaccuracies, we designed a three-module architecture comprising fine-tuned LLMs for SPARQL generation, optional RAG for providing relevant context, and a lightweight SPARQL correction layer. Our empirical evaluation on the SciQA benchmark demonstrates that domain-specific fine-tuning, especially using LoRA for efficient parameter updates, significantly improves both the syntactic quality and execution accuracy of generated SPARQL queries. Notably, our best-performing configuration, based on the LLaMA-3-8B-Instruct model fine-tuned for 15 epochs, achieves state-of-the-art results across all evaluation metrics, including BLEU, ROUGE, and relaxed exact match (RelaxedEM). While RAG does not enhance performance in the presence of fine-tuning, this points to the importance of high-quality context retrieval. Importantly, our results also reveal a substantial performance gap between auto-generated and handcrafted questions, underscoring the difficulty of adapting LLMs to the semantic complexity and variability for handcrafted questions. FIRESPARQL offers a reproducible and configurable framework that can be adapted based on resource availability, paving the way for more robust, interpretable, and scalable QA systems over SKGs.

REFERENCES

Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/. Accessed: 2025-05-26.

Auer, S., Barone, D. A. C., Bartz, C., Cortes, E., Jaradeh, M. Y., Karras, O., Koubarakis, M., Mouromtsev, D. I., Pliukhin, D., Radyush, D., Shilin, I., Stocker, M., and Tsalapati, E. (2023). The sciqa scientific question answering benchmark for scholarly knowledge. *Scientific Reports*, 13.

Bast, H. and Buchhold, B. (2017). Qlever: A query engine for efficient sparql+ text search. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 647–656.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Bustamante, D. and Takeda, H. (2024). Sparql generation with entity pre-trained gpt for kg question answering. *arXiv preprint arXiv:2402.00969*.

Cui, H., Peng, T., Han, R., Han, J., and Liu, L. (2023). Path-based multi-hop reasoning over knowledge graph for answering questions via adversarial reinforcement learning. *Knowledge-Based Systems*, 276:110760.

Ghanem, H. and Cruz, C. (2024). Fine-Tuning vs. Prompting: Evaluating the Knowledge Graph Construction with LLMs. In *CEUR Workshop Proceedings*, volume 3747 of *TEXT2KG-DQMLKG-24 (TEXT2KG 2024 and DQMLKG 2024) 3rd International workshop one knowledge graph generation from text. Data Quality meets Machine Learning and Knowledge Graphs 2024*, page 7, Hersonissos, Greece.

Hovcevar, D. and Kenda, K. (2024). *Integrating Knowledge Graphs and Large Language Models for Querying in an Industrial Environment*. PhD thesis, Bachelor's Thesis. University of Ljubljana, Faculty of Computer, Information.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Huang, X., Zhang, J., Li, D., and Li, P. (2019). Knowledge graph embedding based question answering. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 105–113.

Jiang, L., Yan, X., and Usbeck, R. (2023). A structure and content prompt-based method for knowledge graph question answering over scholarly data. In *QALD/SemREC@ ISWC*.

Jin, B., Yoon, J., Han, J., and Arik, S. O. (2024). Long-context llms meet rag: Overcoming challenges for long inputs in rag. *arXiv preprint arXiv:2410.05983*.

Joren, H., Zhang, J., Ferng, C.-S., Juan, D.-C., Taly, A., and Rashtchian, C. (2024). Sufficient context: A new lens on retrieval augmented generation systems. *arXiv preprint arXiv:2411.06037*.

Kakalis, E.-P. D. and Kefalidis, S.-A. (2024). Advancing geosparql query generation on yago2geo: Leveraging large language models and automated uri injection from natural language questions.

Lehmann, J., Meloni, A., Motta, E., Osborne, F., Recupero, D. R., Salatino, A. A., and Vahdati, S. (2024). Large language models for scientific question answering: An extensive analysis of the sciqa benchmark. In *The Semantic Web: 21st International Conference, ESWC 2024, Hersonissos, Crete, Greece, May 26–30, 2024, Proceedings, Part I*, page 199–217. Springer-Verlag.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Liu, J., Shen, D., Zhang, Y., Dolan, W. B., Carin, L., and Chen, W. (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.

Meyer, L.-P., Frey, J., Brei, F., and Arndt, N. (2024). Assessing sparql capabilities of large language models. *arXiv preprint arXiv:2409.05925*.

Muralidharan, B., Beadles, H., Marzban, R., and Mupparaju, K. S. (2024). Knowledge ai: Fine-tuning nlp models for facilitating scientific knowledge extraction and understanding.

Omar, R., Mangukiya, O., Kalnis, P., and Mansour, E. (2023). Chatgpt versus traditional question answering for knowledge graphs: Current status and future directions towards knowledge graph chatbots. *arXiv preprint arXiv:2302.06466*.

Perevalov, A., Diefenbach, D., Usbeck, R., and Both, A. (2022). Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, pages 229–234. IEEE.

Pliukhin, D., Radyush, D., Kovriguina, L., and Mouromtsev, D. (2023). Improving subgraph extraction algorihtms for one-shot sparql query generation with large language models. In *QALD/SemREC@ ISWC*.

Taffa, T. A. and Usbeck, R. (2023). Leveraging llms in scholarly knowledge graph question answering. In *QALD/SemREC@ ISWC*.

Trivedi, P., Maheshwari, G., Dubey, M., and Lehmann, J. (2017). Lc-quad: A corpus for complex question answering over knowledge graphs. In *The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16*, pages 210–218. Springer.

Wang, J., Chang, Y., Li, Z., An, N., Ma, Q., Hei, L., Luo, H., Lu, Y., and Ren, F. (2024a). Techgpt-2.0: A large language model project to solve the task of knowledge graph construction.

Wang, Y., Lipka, N., Rossi, R. A., Siu, A., Zhang, R., and Derr, T. (2024b). Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214.

Yasunaga, M., Ren, H., Bosselut, A., Liang, P., and Leskovec, J. (2021). Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

APPENDIX

Table 3: Epoch-wise fine-tuning runtime of llama models using a single NVIDIA H100 GPU.

model	epochs	running time(hh:mm:ss)
lama-3.2-3b-Instruct	3	00:09:18
	5	00:14:49
	7	00:23:33
	10	00:32:42
	15	00:37:41
	20	00:57:45
lama-3-8b-Instruct	3	00:10:46
	5	00:16:53
	7	00:26:35
	10	00:38:04
	15	00:55:21
	20	01:14:11

Listing 3: The example of generated SPARQL query before the SPARQL corrector.

```
SELECT?model?model_lbl
    WHERE {
     ?dataset        a
                       orkgc:
        Dataset;
                      rdfs:label
                                ?
                       dataset_
                       lbl.
      FILTER (str(?dataset_lbl) = "
       FTD dataset")
     ?benchmark      orkgp:HAS_
        DATASET       ?dataset;
                       orkgp:HAS_
                           EVALUATION
                              ?eval.
     ?paper          orkgp:HAS_
        BENCHMARK      ?benchmark.
      OPTIONAL {?paper       orkgp:
          HAS_MODEL          ?model.
               ?model       rdfs:
                   label
                                ?
                   model_lbl.}
    }<|eot_id|>
```

Listing 4: The example of generated SPARQL query after the SPARQL corrector.

```
SELECT ?model ?model_lbl
WHERE {
  ?dataset a orkgc:Dataset;
           rdfs:label ?dataset_lbl.
  FILTER (str(?dataset_lbl) = "FTD
      dataset")
  ?benchmark orkgp:HAS_DATASET ?
      dataset;
             orkgp:HAS_EVALUATION ?
                eval.
  ?paper orkgp:HAS_BENCHMARK ?
      benchmark.
  OPTIONAL {
    ?paper orkgp:HAS_MODEL ?model.
    ?model rdfs:label ?model_lbl.
  }
}
```