SPARQL Query Generation with LLMs: Measuring the Impact of Training Data Memorization and Knowledge Injection

Aleksandr Gashkov[1][0000−0001−6894−2094], Aleksandr Perevalov[1][0000−0002−0877−7063], Maria Eltsova[1,2][0000−0003−3792−8518], and Andreas Both[1][0000−0001−7116−9338]

[1] Web & Software Engineering (WSE) Research Group, Leipzig University of Applied Sciences (HTWK Leipzig), Karl-Liebknecht-Straße 132, 04277 Leipzig, Germany
[2] CBZ München GmbH, Heilbronn, Germany

Abstract. Nowadays, the importance of software with natural-language user interfaces cannot be underestimated. In particular, in Question Answering (QA) systems, generating a SPARQL query for a given natural-language question (often named Query Building) from the information retrieved from the same question is the central task of QA systems working over Knowledge Graphs (KGQA). Due to the rise of Large Language Models (LLMs), they are considered a well-suited method to increase the quality of the question-answering functionality, as there is still a lot of room for improvement aiming for enhanced quality and trustworthiness. However, LLMs are trained on web data, where researchers have no control over whether the benchmark or the knowledge graph was already included in the training data. In this paper, we introduce a novel method that evaluates the quality of LLMs by generating a SPARQL query from a NL question under various conditions: 1) zero-shot SPARQL generation, 2) with knowledge injection, and 3) with "anonymized" knowledge injection. This enables us for the first time to estimate the influence of the training data on the QA quality improved by LLMs. Ultimately, this will help to identify how portable a method is or whether good results might mostly be achieved because a benchmark was already included in the training data (cf. LLM memorization). The developed method is portable, robust, and supports any knowledge graph, therefore, could be easily applied to any KGQA or LLM, s.t., generating consistent insights into the actual LLM capabilities is possible.

Keywords: Question Answering · SPARQL query generation · Large Language Models · Knowledge Graph · LLM memorization.

1 Introduction

Question Answering (QA) is aimed to provide a user with precise answers to questions formulated in a natural language (NL). KGQA systems are filling up

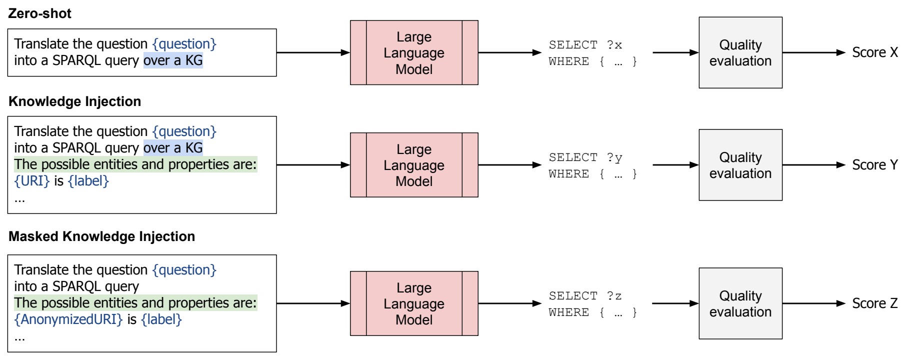
Zero-shot
Translate the question {question} into a SPARQL query over a KG
Large Language Model
SELECT ?x WHERE { … }
Quality evaluation
Score X
Knowledge Injection
Translate the question {question} into a SPARQL query over a KG The possible entities and properties are: {URI} is {label} ...
Large Language Model
SELECT ?y WHERE { … }
Quality evaluation
Score Y
Masked Knowledge Injection
Translate the question {question} into a SPARQL query The possible entities and properties are: {AnonymizedURI} is {label} ...
Large Language Model
SELECT ?z WHERE { … }
Quality evaluation
Score Z

Fig. 1: Overview of the proposed evaluation approach and prompting strategies

the gap between Linked Data and end-users by transforming NL questions into structured queries (e.g., represented as SPARQL[3]) to make the information accessible using NL requests. However, translating NL queries into SPARQL is still a challenge in the field of KGQA [4, 20, 30]. This task is supposed to be solved with the advent of Large Language Models (LLMs) that have significantly furthered the field of natural language processing (NLP) lately [21, 22, 33]. The rapidly growing number of research papers over the past two years confirms the fact that the possibility of LLMs to efficiently generate machine-readable queries from the questions written in NL is being actively discussed. However, according to [24], LLMs often perform well on entities and relationships with high frequencies but face challenges in less popular topics (long-tail knowledge problem). Furthermore, the capabilities of LLMs could be questioned as they show anomalies in terms of *memorization* concerning the actual ability to generate correct SPARQL queries. From this context, we derive the following *research questions*:

RQ1 Assuming a perfect knowledge injection, how well does the SPARQL query generation perform?

RQ2 What is the impact of memorization on the SPARQL query generation capabilities of LLMs?

In summary, these research questions are intended to shed light on the capabilities of LLMs in generating SPARQL queries so that software engineers can better assess the usefulness of LLMs for non-popular or private knowledge graphs.

In this paper, we present a novel method for evaluating the quality of LLMs by generating SPARQL query from a NL question under various conditions:

1. SPARQL generation without additional information provided to the LLM (*zero-shot*),
2. Providing specific and complete knowledge about the information (*knowledge injection*) required to generate the correct SPARQL query (e.g., URI to label mappings),

[3] https://www.w3.org/TR/rdf-sparql-query/

3. Reusing knowledge injection with anonymized URIs (*masked knowledge injection*).

For the general idea of the approach, see Fig. 1. In contrast to prior contributions, our work focuses on estimating the influence of the training data on the QA quality improved by LLMs. Our method is ultimately aimed at recognizing of the fact whether good results are only achieved because an LLM or benchmark was already included in the training data (cf. LLM memorization). In our investigation, we assessed our method using the different KGQA datasets, QALD-9-plus [19] and MCWQ [3]. In addition, we conducted our experiments on various LLMs of different sizes ranging from 7B to 123B parameters.

This paper is structured as follows. In Section2, we summarize related work on the integration of LLMs with KGs for improving semantic parsing and generation of SPARQL queries from a NL questions. Thereafter, in Section 3, we present an overview of our approach, the used datasets and LLMs. Section 4 evaluates and analyzes the experimental results and erroneous queries produced by LLMs during the experiments. The obtained results and limitations are discussed in Section 5. Finally, we conclude our work in Section 6.
Reproducibility statement: The source code and data for experiments are available on GitHub[4].

2 Related Work

The integration of LLMs with various data sources (in particular, the Web of Data, KGs, or knowledge bases) for solving different QA tasks in the last two years has evidenced the high scientific interest in this topic in the research community. Some of the recent approaches try to exploit LLMs as semantic parsers [7,15,17,18] directly or by fine-tuning the models. Mecharnia and d'Aquin [17] presented experiments in fine-tuning LLMs (Llama-3-8B, Llama-2-7B, Llama-3-70B, and Mixtral-8x7B) for the task of NLQ-to-SPARQL transformation on QALD-9-plus [19] and QALD-10 [27]. Their approach demonstrated promising results (Average Macro F1 QALD is near 60% for all LLMs).

Meyer et al. [18] presented a set of automated benchmarking tasks to assess the basic capabilities of LLMs to deal with SPARQL SELECT queries without special architectures or fine-tuning. Completing experiments on different benchmarks like well-known LC-QuAD 2.0 [5], CoyPu-Mini [2], Bestiary [11] and state-of-the-art LLMs families (Claude, Gemini and GPT), the authors conclude that most evaluated LLMs have no significant challenges when perceiving SPARQL `SELECT` query syntax or its semantics, however, generating the `SELECT` queries with correct semantics still seems to be a difficult task for the models.

The last research also tried to integrate LLMs with KG and other solutions [1,11,24,31]. Recent research by Zhang et al. [32] introduced Rule-KBQA, a framework that employs learned rules to guide the generation of logical forms. Initially, rules are extracted from existing data, after that the authors employed

[4] `https://github.com/WSE-research/LLM-generated-SPARQL/`

the Rule-Following Fine-Tuned (RFFT) LLM to generate additional rules, ultimately constructing a comprehensive rule library. Shen et al. [24] suggested a novel framework – Reasoning with Trees – that reformulates KGQA as a discrete decision-making problem, leveraging Monte Carlo Tree Search to iteratively refine reasoning paths.

Zahera et al. [31] developed an approach to generate SPARQL queries that employs Chain-of-thoughts (CoT) prompting [28] and incorporates entities and relations from the input question. Their approach evaluated on the LC-QuAD 2.0, VQuAnDa [10], QALD-9, and QALD-10 datasets demonstrates an improvement in F1 score for both QALD benchmarks.

Another decision is proposed by Kovriguina et al. [11] who developed SPARQLGEN – a one-shot approach for generating SPARQL queries with prompting LLMs. Their approach reached outstanding results on the well-known QALD-9, but did not generalize well on the recently appeared QALD-10 [27] and the BESTIARY benchmark (proposed by the authors) that obviously was not part of the model training. Their results also showed that the model struggled to deal with an unknown KG. The inconsistent performance is explained by the possible memorization of the previously seen datasets by the models.

Some studies (e.g., [6,14,23] etc.) addressed an investigation of benefits from using LLMs for domain-specific or KG-specific tasks in KGQA by leveraging, fine-tuning LLMs, or prompting with intelligent few-shot selection.

Recent papers from Lehmann et al. [12] and Zong et al. [34] leverage novel LLM agent paradigm, where the model calls are streamlined into a predefined workflow and augmented with available tools – external services that supply LLMs with additional information.

From this brief analysis, we can conclude that, on one hand, combining LLMs with KGs might represent a promising technology for constructing KGQA systems; on the other hand, well-known and often exploited benchmarks like QALD, LC-QuAD 2.0 provide the researchers with much better results.

3 Approach and Materials

3.1 Approach

Our primary research objective is to determine if state-of-the-art LLMs are capable of generating valid SPARQL queries over a KG. Our approach (presented in Fig. 1) is tailored to answer the research question while evaluating how well the SPARQL query generation process performs with or without knowledge injection (i.e., incorporation of external knowledge into models to improve their performance). To answer $\boxed{\text{RQ1}}$ and $\boxed{\text{RQ2}}$, we carried out experiments for generating SPARQL queries from NL questions with LLMs by using three prompting techniques. For all LLMs, we used the same prompts.

The first type of prompt used in the experiments was ***zero-shot prompting***. This means that a question is sent to a model "as-is" accompanied only by instructions to prevent extra text generation. Fig. 2a illustrates the structure of this prompt.

The second prompt whose example is presented in Fig. 2b was defined with the presumption that a Named Entity Recognition component was already executed and provided a perfect set of information about the named entities (including its linking to the KG) within the given NL question. Here, the relevant entities were extracted from the gold standard queries and paired with the labels of each entity or property. Pairs of the Entity Name and corresponding URI (in prefixed form) were injected into the prompt. Therefore, we call this a ***knowledge injection prompt***.

The third type of prompt (see Fig. 2c) was tailored to hide sensitive information from the model, trying to prevent the use of the memorized data. Note, we additionally did not refer to the Wikidata KG in the prompts. To mask the Wikidata URIs, we changed them to random (yet unique) numbers with the SPARQL prefix `kg:` pointing to some graph that is unknown to the model. Thus, in comparison to the knowledge injection prompt, all relevant information is recognizable from the prompt, however, there is no direct reference to Wikidata. Hence, we name this approach ***masked knowledge injection prompt.***

We therefore infer that all three types of prompts supplement each other, so they are enough to answer the research question of this paper. Using other types of prompts for investigating their influence of the LLMs performance in generating SPARQL queries from a NL could be a subject of future research.

3.2 Materials

Datasets During the past decade, researchers introduced to the community numerous KGQA benchmarks with logical forms whose quantitative comparison was presented by Liu S. et al. [16] when introducing their own benchmark.

While executing our process, we aim also to find out how much the LLM-generated output is affected by memorized data. Therefore, we suggest that the two benchmarks exploited are quite different in terms of using frequency and, respectively, in training datasets. We suppose that there is a higher tendency of LLMs to memorize popular datasets and this should appear in the resulting metrics. Hence, we require a popular dataset and a dataset that is rarely used where both are defined over data from the same KG.

To evaluate the effectiveness of the proposed method, we carried out our experiments on two public benchmarks executable over the Wikidata knowledge graph: QALD-9-plus [19] and MCWQ [3]. Both datasets are multilingual; however, we used only their English parts for our current research.

QALD-9-plus[5] [19] is an extension of the well-known QALD-9 dataset where extended language support was added, and the translation quality for existing languages was significantly optimized (e.g., via Spanish [25]) though the questions' translations were provided by multiple native speakers. The dataset contains 558 questions incorporating information from the Wikidata knowledge base. In addition to the textual presentation, each question contains the corresponding SPARQL query, the answer entity URI, and the answer type. The

[5] `https://github.com/KGQA/QALD_9_plus`

```
Translate the question ‘‘Who developed Skype?’’ into a SPARQL query
using the Wikidata Knowledge Graph.
Have the query return only resources.
Provide only the generated SPARQL query.
```

(a) Example of zero-shot prompting.

```
Translate the question "Who developed Skype?" into a SPARQL query
using the Wikidata Knowledge Graph.
Have the query return only resources.
Provide only the generated SPARQL query.
The possible entities and properties are:
wd:Q40984 is Skype
wdt:P178 is developer.
```

(b) Example of a prompt with knowledge injection.

```
Translate the question "Who developed Skype?" into a SPARQL query
using a Knowledge Graph.
Have the query return only resources.
Provide only the generated SPARQL query.
The possible entities and properties are:
kg:6211 is Skype
kg:1548 is developer.
```

(c) Example of a prompt with “anonymized” knowledge injection. The information regarding linked URIs and masks was stored to be unmasked after a query generation.

Fig. 2: Examples of prompts to generate the SPARQL query for the question “Who developed Skype?” from the QALD-9-plus dataset by exploiting different LLMs.

datasets of the QALD series are widely used since more than a decade for scientific challenges and publications.

MCWQ[6] [3] is a rarely used dataset transformed from the CFQ (Compositional Freebase Questions) benchmark that was aimed at parsing questions in English into SPARQL queries executable on the Freebase knowledge base. The dataset has 2,444 question patterns (mod entities, verbs, etc.). There are 1,835 unique SPARQL query patterns in MCWQ, resulting in 16.9% instances covering 100% of unique SPARQL query patterns. According to the authors, the dataset “poses a greater challenge for compositional semantic parsing, and exhibits less redundancy in terms of duplicate patterns.” In total, MCWQ contains 124,187 question query pairs, but our experiments were carried out on the gold test set. It should be pointed out that 1) the dataset has a focus on history and cinematography, 2) several questions expressed multiple restrictions to find the correct answer (sometimes much more restrictions than QALD-9-plus has), and 3) multiple questions address the same topic but work with different restrictions.

[6] https://github.com/coastalcph/seq2sparql

Table 1: Overview on the LLMs used within our work

Model name	Developed by	Year	Model size
Qwen 2.5 7B	Alibaba Cloud	2024	7B
Qwen 2.5 14B	Alibaba Cloud	2024	14B
Qwen 2.5 32B	Alibaba Cloud	2024	32B
Qwen 2.5 72B	Alibaba Cloud	2024	72B
DeepSeek-r1 7B	DeepSeek	2025	7B
DeepSeek-r1 14B	DeepSeek	2025	14B
DeepSeek-r1 32B	DeepSeek	2025	32B
DeepSeek-r1 70B	DeepSeek	2025	72B
Mistral-Small	Mistral AI	2025	24B
Mistral-Large	Mistral AI	2025	123B
Llama 3.3 70B	Meta	2024	70B

LLMs We employed 11 LLMs from four developers in our experiments: Qwen 2.5 [29] (7B, 14B, 32B, and 72B), DeepSeek-r1 [8] (7B, 14B, 32B, and 70B), Mistral-Small and Mistral-Large[7], and Llama 3.3 70B[8] presented in Table 1. We chose these latest open-source models because of their significant variability in dimension, which allows us to evaluate different options on the selected benchmarks, as well as the high quality declared by developers.

The models were executed with the Ollama engine[9] on a server with two NVIDIA L40S GPUs (48GB VRAM). All models are quantized in 4-bit with quantization type "Q4_K_M" due to the limited resources.

Metrics For measuring the quality of the experimentally obtained results, we used two metrics: 1) the relative frequency of valid queries (P_{val}) and 2) precision (P) – the relative frequency of correct answers. Only a valid query can produce the correct answer, therefore, $P \leq P_{val}$ applies.

In our setting, a correct answer exists for all questions. This means that precision, recall, and F1 score are always equal in one experiment.

4 Evaluation and Analysis

4.1 Results

Table 2 reports the complete set of experiments on all models, covering all the metrics introduced in Section 3.2. For the sake of space, we will primarily focus on the top results. Smaller models (7B and in some cases 14B) experienced difficulty by solving zero-shot task and masked injection task, especially with MCWQ dataset. The best quality on all tasks demonstrated Qwen 2.5 72B,

[7] `https://mistral.ai/en/news/mistral-small-3`
[8] `https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct`
[9] `https://github.com/ollama/ollama`

Table 2: Experimental results

Model	**Zero-shot**					**Knowledge injection**					**Masked injection**				
	Total records	**Valid queries**	**Correct**	$\mathbf{P_{val}}$	**P = R = F1**	**Total records**	**Valid queries**	**Correct**	$\mathbf{P_{val}}$	**P = R = F1**	**Total records**	**Valid queries**	**Correct**	$\mathbf{P_{val}}$	**P = R = F1**
MCWQ															
Qwen 2.5 7B	155	55	0	0.35	0.00	155	81	11	0.35	0.07	155	90	0	0.35	0.00
Qwen 2.5 14B	155	45	0	0.29	0.00	155	66	6	0.29	0.04	155	78	0	0.29	0.00
Qwen 2.5 32B	155	0	0	0.00	0.00	155	0	0	0.00	0.00	155	1	0	0.00	0.00
Qwen 2.5 72B	155	18	0	0.12	0.00	155	95	24	0.12	**0.15**	155	65	2	0.12	**0.01**
DeepSeek-r1 7B	155	2	0	0.01	0.00	155	11	0	0.01	0.00	155	4	0	0.01	0.00
DeepSeek-r1 14B	155	76	0	0.49	0.00	155	26	2	0.49	0.01	155	45	0	0.49	0.00
DeepSeek-r1 32B	155	59	0	0.38	0.00	155	10	6	0.38	0.04	155	9	0	0.38	0.00
DeepSeek-r1 70B	155	69	0	0.45	0.00	155	12	2	0.45	0.01	155	49	2	0.45	**0.01**
Mistral-Small	155	92	0	0.59	0.00	155	71	11	0.59	0.07	155	38	0	0.59	0.00
Mistral-Large	155	135	1	0.87	**0.01**	155	112	12	0.87	0.08	155	110	0	0.87	0.00
Llama 3.3 70B	155	118	0	0.76	0.00	155	122	16	0.76	0.10	155	112	1	0.76	**0.01**
QALD-9-plus															
Qwen 2.5 7B	471	266	0	0.56	0.00	460	357	109	0.56	0.24	460	347	37	0.56	0.08
qwen2.5 14B	471	268	1	0.57	0.00	460	342	209	0.57	0.45	460	246	126	0.57	0.27
Qwen 2.5 32B	471	10	0	0.02	0.00	460	5	3	0.02	0.01	460	21	8	0.02	0.02
Qwen 2.5 72B	471	247	6	0.52	0.01	460	400	257	0.52	0.56	460	425	229	0.52	**0.50**
DeepSeek-r1 7B	471	31	0	0.07	0.00	460	84	3	0.07	0.01	460	30	0	0.07	0.00
DeepSeek-r1 14B	471	293	0	0.62	0.00	460	198	77	0.62	0.17	460	46	4	0.62	0.01
DeepSeek-r1 32B	471	354	4	0.75	0.01	460	347	212	0.75	0.46	460	335	181	0.75	0.39
DeepSeek-r1 70B	471	370	3	0.79	0.01	460	144	64	0.79	0.14	460	55	14	0.79	0.03
Mistral-Small	471	338	6	0.72	0.01	460	399	241	0.72	0.52	460	347	151	0.72	0.33
Mistral-Large	471	447	30	0.95	**0.06**	460	450	279	0.95	**0.61**	460	426	212	0.95	0.46
Llama 3.3 70B	471	330	14	0.70	0.03	460	371	186	0.70	0.40	460	372	161	0.70	0.35

Table 3: Experimental results for optimized MCWQ

Model	**Zero-shot**					**Knowledge injection**					**Masked injection**				
	Total records	**Valid queries**	**Correct**	$\mathbf{P_{val}}$	**P = R = F1**	**Total records**	**Valid queries**	**Correct**	$\mathbf{P_{val}}$	**P = R = F1**	**Total records**	**Valid queries**	**Correct**	$\mathbf{P_{val}}$	**P = R = F1**
Qwen 2.5 7B	146	51	0	0.35	0.00	140	77	12	0.35	0.09	140	70	0	0.35	0.00
Qwen 2.5 14B	146	24	0	0.16	0.00	140	44	7	0.16	0.05	140	37	0	0.16	0.00
Qwen 2.5 32B	146	0	0	0.00	0.00	140	0	0	0.00	0.00	140	1	0	0.00	0.00
Qwen 2.5 72B	146	22	0	0.15	0.00	140	107	16	0.15	**0.11**	140	99	0	0.15	0.00
DeepSeek-r1 7B	146	4	0	0.03	0.00	140	7	0	0.03	0.00	140	4	0	0.03	0.00
DeepSeek-r1 14B	146	66	0	0.45	0.00	140	51	7	0.45	0.05	140	6	0	0.45	0.00
DeepSeek-r1 32B	146	66	0	0.45	0.00	140	69	16	0.45	**0.11**	140	40	0	0.45	0.00
DeepSeek-r1 70B	146	67	0	0.46	0.00	140	20	2	0.46	0.01	140	27	0	0.46	0.00
Mistral-Small	146	95	0	0.65	0.00	140	56	8	0.65	0.06	140	53	0	0.65	0.00
Mistral-Large	146	123	2	0.84	0.01	140	101	12	0.84	0.09	140	92	0	0.84	0.00
Llama 3.3 70B	146	103	0	0.71	0.00	140	97	20	0.71	0.14	140	89	0	0.71	0.00

Mistral-Large, Llama 3.3 70B, and DeepSeek-r1 32B, the worst one – Qwen 2.5 32B.

Comparing on F1 score in the experiments with knowledge injection prompting, both Mistral models (Mistral-Large= 0.61 vs. Mistral-Small= 0.52), and Qwen 2.5 72B (0.52), demonstrated the best results on QALD-9-plus while only the top result in the same experiment on MCQW only a $F1 = 0.15$ was achieved by Qwen 2.5 72B followed by Llama 3.3 70B (0.1). It is also noteworthy that even much smaller models of Qwen 2.5 demonstrated comparable results in the experiments where knowledge injection prompting was used.

However, the results of experiments with zero-shot prompting on both benchmarks are unsatisfied: only Qwen 2.5 72B, Mistral-Large, Mistral-Small, DeepSeek-r1 32B, DeepSeek-r1 70B, and Llama 3.3 70B were able to generate few correct SPARQL queries from the given NL questions from the QALD-9-plus dataset while only Mistral-Large – from MCWQ.

An experiment with the masked-injection-prompting on MCWQ was expected challenging for all LLMs, only Qwen 2.5 72B, Mistral-Large, and Llama 3.3 70B were able to generate one or two correct SPARQL queries for 155 NL questions. Nevertheless, the same experiment on QALD-9-plus unpredictably did not cause problems for most LLMs besides Qwen 2.5 32B, DeepSeek-r1 7B, DeepSeek-r1 14B, and DeepSeek-r1 70B.

The general observation is that all metrics on the MCWQ benchmark are much worse when compared to QALD-9-plus. Trying to find out the reason for the poor quality of MCWQ results, we selectively analyzed questions from the dataset and ascertained that the fluency, semantic, and grammatical correctness of some questions do not conform to language norms. Therefore, 140 questions were reformulated using OpenAI's well-performing GPT-4o[10] [9], manually validated, and optimized the generated questions, s.t., a smaller, semantically equal, yet (language quality wised) improved dataset was generated[11] to evaluate the impact of the *language quality of the NL questions* in comparison to the memorization effect. The results of the experiment with this improved MCWQ dataset are presented in Table 3. However, they do not provide any noticeable improvement, and, therefore, refute the hypothesis that the MCWQ language quality has heavily impacted the LLM performance. Instead, we can conclude that the (missing) memorization effect was actually the reason for the observed SPARQL query generation quality. It is worth noting that each request was carried out in a new conversation.

4.2 Error Analysis

We performed an analysis of erroneous queries by defining those not corresponding to the expected response. There are research papers offering different error classification (e.g., [1,4,13,14,26]), however, we consider the error analysis as an

[10] `https://platform.openai.com/docs/models#gpt-4o`

[11] The dataset is stored in the online appendix including the generated NL question and the manually optimized queries.

additional result helping us to further evaluate the performance of our method. We identified four error categories, which are presented with some examples in Fig. 3.

1. *Invalid format or query* (cf. Fig. 3a).
   - *Invalid JSON format.* The model's output did not follow the correct JSON format, so the query text cannot be extracted.
   - *Invalid SPARQL query.* The query did not pass the syntax check (done with the help of RDFlib[12]) or caused an error while being executed on Wikidata.
2. *Empty answer* (cf. Fig. 3b): all questions in the considered datasets would lead to a non-empty result from the Wikidata knowledge graph, hence at least one entity as an answer (or true/false value) is expected; therefore, an empty answer is an error.
3. *Incorrect set of entities* (cf. Fig. 3c): we compared the set of expected entities with the set of produced entities. To be a correct answer, the query must produce the same list of entities as the gold standard, without regard to order.
4. *Occurrence of Wikidata URIs* (cf. Fig. 3d): A Wikidata URI occurred in the generated query, while the prompt requires URIs from the sample KG (which is unknown to the LLM).

The error analysis clearly shows that all models are relying on the memorized data while generating SPARQL query. In particular, a model was asked to generate a SPARQL query for an imaginary KG, but it produced the (correct) Wikidata's URI instead (hence, Error Category 4 as presented on Fig. 3d).

Table 4 presents the frequency of erroneous queries for all LLMs. Investigating the data from Table 4, we should point out that Error Category 1 (Invalid format or query) is more usually the case for the MCWQ dataset while Error Category 4 (occurrence of Wikidata URIs) is rather relevant for QALD-9-plus besides the DeepSeek-r1 models which demonstrate (excluding DeepSeek-r1 32B) high rate of Error Category 1 also when experimenting with QALD-9-plus. Error Category 3 (incorrect set of entities) is comparable for both datasets.

When analyzing the error categories by models, the noticeable fact is that Qwen 2.5 32B always produced invalid query or did not follow the correct JSON format, i.e., the query text cannot be extracted while prompts to Mistral-Large hardly result in such errors. In addition, the DeepSeek-r1 models (besides DeepSeek-r1 32B) return the lowest number of empty answers (Error Category 2) on both datasets. However, Mistral-Large often produces the incorrect set of entities, which is typically for all larger models (the size is 24B and over) on the QALD-9-plus dataset and a small one – Qwen 2.5 7B.

Therefore, the error frequency supposes that there is still much to be done to achieve the acceptable results.

[12] `https://github.com/RDFLib/rdflib`

```
SELECT ?resource
WHERE { >// Instance of film
```

(a) Error Category 1 – Invalid format/query

```
SELECT ?newSeriesEpisodes ?oldSeriesEpisodes
WHERE {
    wd:Q162594 wdt:P1113 ?newSeriesEpisodes .
    wd:Q180755 wdt:P1113 ?oldSeriesEpisodes .
ORDER BY DESC(?newSeriesEpisodes)
LIMIT 1
}
```

(b) Error Category 2 – Empty answer

```
SELECT ?mountain
WHERE  ?mountain wdt:P31 wd:Q8502 .
SELECT (MAX(?elevation) AS ?maxElevation)
```

(c) Error Category 3 – Incorrect set of entities

```
SELECT ?moon
WHERE { ?moon kg:is_moon_of_Jupiter wd:Q61702557 .
?moon kg:is_mass wdt:P2067
}
```

(d) Error Category 4 – Occurrence of Wikidata URIs although a masked knowledge injection was used.

Fig. 3: Examples from the identified error categories in the generated SPARQL queries.

5 Limitations and Discussion

This paper focuses on the ability of state-of-the-art LLMs to generate SPARQL queries with or without knowledge injection. However, when appraising the findings, it is essential to take into consideration some limitations and factors w.r.t. the general applicability of the proposed method. First, we did not exploit commercial LLMs like GPT, Claude, or other open-source LLMs. The exploited models are the newest ones, vary significantly in their dimensions, and demonstrate better results declared by developers.

Second, we carried out our experiments only in English as the majority of research in this field is dominated by this language, and we paid most attention to SPARQL queries here and not to the language capabilities. On the other hand, the approach is portable to non-English languages, hence, we leave the exploration of this direction for future work.

A further limitation is that we conducted our experiments on two datasets. Since the approach has been proved, the experiments could be repeated on other datasets, which might make a further contribution to the study of the memoriza-

Table 4: Categories of errors and their frequencies (note, multiple errors per query are possible).

Model	Benchmarks/Error categories							
	QALD-9-plus				MCWQ			
	1	2	3	4	1	2	3	4
Qwen 2.5 7B	0.30	0.19	**0.59**	0.30	0.48	0.03	0.49	0.00
Qwen 2.5 14B	0.38	0.33	0.37	0.31	0.57	0.06	0.42	0.06
Qwen 2.5 32B	**0.97**	0.01	0.02	0.27	**1.00**	0.00	0.00	0.02
Qwen 2.5 72B	0.23	**0.51**	0.42	**0.32**	0.61	0.13	0.34	0.05
DeepSeek-r1 7B	0.90	0.05	0.10	0.16	0.96	0.01	0.04	0.02
DeepSeek-r1 14B	0.61	0.16	0.33	0.08	0.66	0.05	0.34	0.01
DeepSeek-r1 32B	0.26	0.45	0.46	0.29	0.81	0.02	0.18	0.00
DeepSeek-r1 70B	0.59	0.18	0.35	0.20	0.70	0.04	0.29	**0.16**
Mistral-Small	0.22	0.43	0.49	0.29	0.55	0.06	0.43	0.00
Mistral-Large	0.05	0.59	0.58	**0.32**	0.19	**0.14**	**0.78**	0.02
Llama 3.3 70B	0.23	0.42	0.51	**0.32**	0.21	0.08	0.75	0.02

tion phenomenon. However, it is worth noting that finding question-answering datasets that are rarely used is hard.

Several questions are raised for the discussion. First, synergistic integration of LLMs with KG (i.e., “knowledge injection” experiments) is still a promising direction for further research. Even smaller models might provide benefits and, therefore, decent results to the users. However, we have to point out the fact that all LLMs demonstrated very poor quality when zero-shot-prompting. Therefore, the exploited LLMs are hardly able to generate SPARQL queries from given NL questions without knowledge injection. So, one might come to the conclusion that the memorization effect needs to be triggered by a significant knowledge injection.

Second, the results of all experiments on QALD-9-plus outperform those on MCWQ. All LLMs struggled with this task on this infrequently exploited dataset although the data provided via the knowledge injection prompting again provided all needed information to generate a correct SPARQL query. Our final experiment with an improved MCWQ dataset demonstrated that the linguistic quality of NL questions does not impact the LLM’s performance significantly and, therefore, the final results. In this regard, our results correlate with the ones from SPARQLGEN paper [11].

Third, the error analysis revealed that during the “masked injection” experiments, Wikidata’s URIs occurred in the generated query while the prompt does not give any indicator that the generated query should be generated for the Wikidata KG, instead it would have required URIs from the sample (unknown) KG. Hence, this evidences that memorized data significantly influence the output of LLMs. At the moment, there are still additional opportunities for process improvement.

6 Conclusions and Future Work

In this study, we explored the capabilities of Large Language Models (LLMs) in generating SPARQL queries from natural language (NL) questions under three different conditions: zero-shot prompting, knowledge injection, and masked knowledge injection. Our primary goal was to assess the impact of structured knowledge integration on query generation performance, while also examining whether LLMs rely on memorization rather than true reasoning when dealing with knowledge graphs (KGs).

To address $\boxed{\text{RQ1}}$ ("Assuming a perfect knowledge injection, how well does the SPARQL query generation perform?"), our findings demonstrate that knowledge injection significantly enhances LLM performance. Models provided with explicit entity and property mappings consistently outperformed those operating under zero-shot conditions. This suggests that LLMs, despite their vast pre-training, still benefit greatly from additional structured context. Even smaller models showed considerable improvement with knowledge injection, highlighting the importance of integrating external knowledge sources when leveraging LLMs for SPARQL query generation. However, performance varied among different models, with larger architectures generally yielding better results.

Regarding $\boxed{\text{RQ2}}$ ("What is the impact of memorization on the SPARQL query generation capabilities of LLMs"), our experiments revealed strong indications of memorization effects. When using masked knowledge injection, where original Wikidata entity URIs were anonymized to prevent recognition, several models still generated queries containing correct Wikidata URIs.This suggests that LLMs often rely on memorized training data rather than true reasoning capabilities, raising concerns about their generalizability to unseen datasets or novel KGs. Moreover, models performed significantly better on the well-known QALD-9-plus dataset compared to the less frequently used MCWQ dataset, reinforcing the notion that model familiarity with a dataset influences results.

In conclusion, researchers and practitioners need to be cautious when using LLM-based SPARQL generation approaches and always keep in mind that the results of the presented methods are unlikely to be fully reproducible on a private or new dataset because the memorization effect will significantly increase the quality and consequently degrade the results in a dataset without this effect. Further research will be necessary to compensate for this effect.

Future Work. While our study provides valuable insights into SPARQL query generation with LLMs, several limitations should be considered. First, our experiments were conducted using a selected set of open-source models and two benchmark datasets. Further research could expand the scope to involve additional models, including proprietary LLMs such as GPT, Claude, or Gemini, to determine whether similar trends persist. Second, our experiments were performed exclusively on English questions. Given that many knowledge graphs contain multilingual data, assessing LLM performance across different languages remains an important avenue for future exploration. Third, the detailed error analysis and an entire error classification could provide the researchers with the ways to improve the performance of LLM-based SPARQL generation approaches.

Additionally, our findings highlight the need for improved strategies to mitigate memorization effects and the new strategies for creating and publishing benchmarking datasets. Future research should investigate fine-tuning methods or hybrid approaches that combine LLMs with rule-based or symbolic reasoning techniques to enhance generalization. Another promising direction is the integration of external knowledge retrieval mechanisms, such as KG-based embeddings or reinforcement learning strategies, to improve query accuracy while minimizing reliance on pre-trained data.

Finally, evaluating LLM performance across a broader range of KGQA tasks (e.g., handling complex queries, reasoning over multiple hops, and supporting federated queries) could provide further insights into their real-world applicability. By addressing these challenges, we can advance the development of more reliable, interpretable, and generalizable LLM-driven approaches for knowledge graph-based question answering.

References

1. Bhandiwad, D., Gattogi, P., Kangen, A., Basaldella, M., Ferré, S., Vahdati, S., Lehmann, J.: Bridging language models and knowledge graphs with controlled natural languages. Available at SSRN 5009450
2. Brei, F., Frey, J., Meyer, L.P.: Leveraging small language models for Text2SPARQL tasks to improve the resilience of AI assistance. arXiv:2405.17076 (2024)
3. Cui, R., Aralikatte, R., Lent, H., Hershcovich, D.: Compositional generalization in multilingual semantic parsing over Wikidata. Transactions of the ACL **10** (2022)
4. Diallo, P.A.K.K., Reyd, S., Zouaq, A.: A comprehensive evaluation of neural SPARQL query generation from natural language questions. IEEE Access (2024)
5. Dubey, M., Banerjee, D., Abdelkawi, A., Lehmann, J.: LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and DBpedia. In: International semantic web conference. pp. 69–78. Springer (2019)
6. Emonet, V., Bolleman, J., Duvaud, S., de Farias, T.M., Sima, A.C.: LLM-based SPARQL query generation from natural language over federated knowledge graphs. arXiv preprint arXiv:2410.06062 (2024)
7. Faria, B., Perdigão, D., Oliveira, H.G.: Question answering over linked data with GPT-3. Open Access Series in Informatics (OASIcs) **113**, 1–15 (2023)
8. Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al.: DeepSeek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025)
9. Hurst, A., Lerer, A., Goucher, A.P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al.: GPT-4o system card. arXiv preprint arXiv:2410.21276 (2024)
10. Kacupaj, E., Zafar, H., Lehmann, J., Maleshkova, M.: VQuAnDa: Verbalization question answering dataset. In: The Semantic Web. pp. 531–547. Springer International Publishing, Cham (2020)
11. Kovriguina, L., Teucher, R., Radyush, D., Mouromtsev, D., Keshan, N., Neumaier, S., Gentile, A., Vahdati, S.: SPARQLGEN: One-shot prompt-based approach for SPARQL query generation. In: SEMANTiCS (Posters & Demos) (2023)

12. Lehmann, J., Bhandiwad, D., Gattogi, P., Vahdati, S.: Beyond boundaries: A human-like approach for question answering over structured and unstructured information sources. Transactions of the Association for Computational Linguistics **12**, 786–802 (2024)
13. Lehmann, J., Gattogi, P., Bhandiwad, D., Ferré, S., Vahdati, S.: Language models as controlled natural language semantic parsers for knowledge graph question answering. In: ECAI 2023, pp. 1348–1356. IOS Press (2023)
14. Lehmann, J., Meloni, A., Motta, E., Osborne, F., Recupero, D.R., Salatino, A.A., Vahdati, S.: Large language models for scientific question answering: An extensive analysis of the SciQA benchmark. In: European Semantic Web Conference. pp. 199–217. Springer (2024)
15. Liu, J., Cao, S., Shi, J., Zhang, T., Nie, L., Hu, L., Hou, L., Li, J.: How proficient are large language models in formal languages? An in-depth insight for knowledge base question answering. In: Findings of the Association for Computational Linguistics ACL 2024. pp. 792–815 (2024)
16. Liu, S., Semnani, S., Triedman, H., Xu, J., Zhao, I.D., Lam, M.: SPINACH: SPARQL-based information navigation for challenging real-world questions. In: Findings of the Association for Computational Linguistics: EMNLP 2024. pp. 15977–16001. Association for Computational Linguistics (2024)
17. Mecharnia, T., d'Aquin, M.: Performance and limitations of fine-tuned LLMs in SPARQL query generation. In: Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK). pp. 69–77 (2025)
18. Meyer, L.P., Frey, J., Brei, F., Arndt, N.: Assessing SPARQL capabilities of large language models. arXiv preprint arXiv:2409.05925 (2024)
19. Perevalov, A., Diefenbach, D., Usbeck, R., Both, A.: 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). pp. 229–234 (2022)
20. Perevalov, A., Gashkov, A., Eltsova, M., Both, A.: Language models as SPARQL query filtering for improving the quality of multilingual question answering over knowledge graphs. In: International Conference on Web Engineering. pp. 3–18. Springer (2024)
21. Perevalov, A., Gashkov, A., Eltsova, M., Both, A.: Understanding SPARQL queries: Are we already there? Multilingual natural language generation based on SPARQL queries and large language models. In: International Semantic Web Conference. pp. 173–191. Springer (2024)
22. Qin, L., Chen, Q., Feng, X., Wu, Y., Zhang, Y., Li, Y., Li, M., Che, W., Yu, P.S.: Large language models meet NLP: A survey. arXiv:2405.12819 (2024)
23. Rangel, J.C., de Farias, T.M., Sima, A.C., Kobayashi, N.: SPARQL generation: an analysis on fine-tuning OpenLLaMA for question answering over a life science knowledge graph. SWAT4HCLS 2024: The 15th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences (2024)
24. Shen, T., Wang, J., Zhang, X., Cambria, E.: Reasoning with trees: Faithful question answering over knowledge graph. In: Proceedings of the 31st International Conference on Computational Linguistics. pp. 3138–3157 (2025)
25. Soruco, J., Collarana, D., Both, A., Usbeck, R.: QALD-9-ES: A Spanish Dataset for Question Answering Systems, pp. 38–52. Studies on the Semantic Web, IOS Press BV, Netherlands (2023)
26. Taipalus, T., Siponen, M., Vartiainen, T.: Errors and complications in sql query formulation. ACM Transactions on Computing Education (TOCE) **18**(3), 1–29 (2018)

27. Usbeck, R., Yan, X., Perevalov, A., Jiang, L., Schulz, J., Kraft, A., Möller, C., Huang, J., Reineke, J., Ngonga Ngomo, A.C., et al.: QALD-10–the 10th challenge on question answering over linked data: Shifting from DBpedia to Wikidata as a KG for KGQA. Semantic Web **15**(6), 2193–2207 (2024)
28. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems **35**, 24824–24837 (2022)
29. Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al.: Qwen2.5 technical report. arXiv preprint arXiv:2412.15115 (2024)
30. Yani, M., Krisnadhi, A.A.: Challenges, techniques, and trends of simple knowledge graph question answering: a survey. Information **12**(7), 271 (2021)
31. Zahera, H.M., Ali, M., Sherif, M.A., Moussallem, D., Ngomo, A.C.N.: Generating SPARQL from natural language using chain-of-thoughts prompting. SEMANTiCS (2024)
32. Zhang, Z., Wen, L., Zhao, W.: Rule-KBQA: Rule-guided reasoning for complex knowledge base question answering with large language models. In: 31st International Conference on Computational Linguistics. pp. 8399–8417 (2025)
33. Zhuang, Y., Yu, Y., Wang, K., Sun, H., Zhang, C.: ToolQA: A dataset for LLM question answering with external tools. Advances in Neural Information Processing Systems **36** (2024)
34. Zong, C., Yan, Y., Lu, W., Huang, E., Shao, J., Zhuang, Y.: Triad: A framework leveraging a multi-role LLM-based agent to solve knowledge base question answering. Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (2024)