

A Prompt-Based Approach to WikiKGQA 2026: SPARQL Generation with Large Language Models

Melisa Marian^{1,*}, Alexandra Nanu¹, Alexia Dan¹ and Gheorghe Cosmin Silaghi¹

¹Babeş-Bolyai University, Business Informatics Research Center, Cluj-Napoca, Romania

Abstract

This paper presents WikiProSP (Wikidata Prompt-based SPARQL Generation), our system submitted to the WikiKGQA 2026 challenge for automated SPARQL query generation over Wikidata. We compare two retrieval-based methods (TF-IDF, BM25) with three LLM-based methods (zero-shot, few-shot, few-shot with automated repair), evaluating them on natural language questions, with and without additional mentions of relevant Wikidata entities and properties. The evaluation is performed using Macro QALD-F1, together with Exact Answer Accuracy and Execution Rate metrics. The results show that LLM-based methods outperform the retrieval-based sanity checks, while BM25-selected few-shot demonstrations and the integration of entity and property mentions substantially improve performance.

Keywords

Knowledge Graph Question Answering, SPARQL Generation, Large Language Models, Prompt Engineering, Wikidata

1. Introduction

Over the last few years, Knowledge Graph Question Answering (KGQA) has become an important direction for research in Natural Language Processing and Semantic Web, having as an objective the transformation of natural language questions into queries that are executable on knowledge bases [1], [2]. The emergence of Large Language Models (LLMs) has reshaped research on text-to-SPARQL, with recent studies showing that prompting and in-context learning alone can generate SPARQL queries without additional fine-tuning [3]. However, their performance depends on the prompting strategy and on whether the prompt includes knowledge-base information such as relevant ontology fragments [4], [5].

This paper reports our empirical submission to WikiKGQA 2026¹, evaluating prompt-based SPARQL generation across the English Simple and With Mentions tracks. We compare zero-shot, few-shot, and query-repair configurations, alongside retrieval-based sanity checks, and analyze the effects of demonstration selection and entity/property annotations.

Therefore, we investigate the effectiveness of prompting strategies for SPARQL generation regarding Wikidata knowledge graph question answering, by providing several contributions. First, we perform a comparative evaluation of the retrieval and LLM prompting-based methods for SPARQL query generation, over the benchmark provided by WikiKGQA 2026. Then, we investigate the impact of few-shot prompting using query examples retrieved with BM25, as well as an additional automated repair step, aiming to improve the quality of the generated queries. Finally, we analyze the effect of introducing entity and property mentions, comparing the performance of the system in the two configurations. The obtained results outline the strengths and weaknesses of our system for prompt-based SPARQL query generation, offering a perspective over how such a model could contribute to the development of efficient KGQA systems. To facilitate reproducibility, the complete source code and experimental setup are publicly available at https://github.com/meli1230/wikikgqa2026_submission.git.

Proceedings of the Wiki-Based Knowledge Graph Question Answering Challenge (WikiKGQA), co-located with the 25th International Semantic Web Conference (ISWC 2026), Bari, Italy, October 25-29, 2026

✉ melisa_marian@yahoo.fr (M. Marian); alexandrananu5@gmail.com (A. Nanu); alexia1402@icloud.com (A. Dan); gheorghe.silaghi@ubbcluj.ro (G. C. Silaghi)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

¹<https://www.wikikgqa.org>

2. Related Work

KGQA aims to respond to natural language questions by querying knowledge graphs using standardized query languages, such as SPARQL, or other reasoning mechanisms. DBpedia [6], Freebase [7] and Wikidata [8] are among the most widely used knowledge graphs in KGQA research and evaluation [9]. Although many KGQA evaluation datasets have historically relied on Freebase and DBpedia, Wikidata is becoming an increasingly important knowledge base, due to its scale and multilingual coverage [10], [11].

Benchmarks, such as Mintaka [12], have introduced complex multilingual natural language questions, including factual, comparative and multi-hop questions over Wikidata, making them challenging evaluation frameworks for KGQA systems. A notable challenge in KGQA is the transformation of natural language questions into executable SPARQL queries. Early text-to-SPARQL approaches were generally built as modular pipelines, combining entity and relation linking. Such approaches were sensitive to error propagation between steps and required hand-engineered components [1], [13], [14], [15].

In contrast, research focusing on semantic-parsing approaches employ neural sequence-to-sequence models that learn mappings between natural language questions and SPARQL queries, substantially reducing hand-crafted rules and templates in the parsing component, therefore achieving improved generalization on standard benchmarks [1], [16].

Recent developments in LLMs steer toward prompting-based SPARQL generation. SPARQLGEN [4] demonstrates that LLMs can generate executable SPARQL queries even in one-shot configurations, without providing many examples specific to the task. More recent findings investigate the use of retrievals for the selection of the examples, integrating information about entities and properties, as well as automated refinement mechanisms, aiming to improve their quality [3], [17], [5].

Our work follows the same trend, comparing classic retrieval with LLM prompting-based methods, analyzing the impact of few-shot prompting and of the query repair mechanism on the final answer, as well as the improvement in performance achieved by integrating explicit mentions of entities and relations in the official benchmark of WikiKGQA 2026.

3. Task and Dataset

3.1. WikiKGQA 2026

The experiments presented in this paper were conducted within WikiKGQA 2026; for each question, the dataset provides the expected answer in the training and development configuration, allowing for the evaluation of the system’s performance. In the test split, the gold answer is withheld for the official Codabench evaluation. The provided benchmark is based on QAWiki v2026.2 [18].

3.2. Subtasks

This challenge defines four official subtasks, split into two categories. In its **Simple** setting, the models receive the natural language questions and they must provide an equivalent valid SPARQL query. In the version **With Mentions**, each question has additional Wikidata entity and property annotations for each of the questions, offering an explicit semantic grounding mechanism and reducing the ambiguity of the SPARQL generation process [18]. Both of these configurations provide questions in English and Spanish; our work focuses on the English subset, and the methods have been evaluated on both subtasks in order to investigate the impact of the annotations.

3.3. Dataset

For each of the two subtasks, the organizers provide *train*, *development* and *test* splits, as seen in Table 1. The training split was used for retrieval-based methods and for the selection of the examples provided to the LLM in its few-shot configuration. We used the development split for the comparison of the methods, while the test split was used exclusively for the generation of the final predictions, uploaded

Table 1
Dataset Split

Split	Samples
train	317
development	100
test	75



Figure 1: Architecture Pipeline

on Codabench. The experimental pipeline was implemented and executed in Google Colab using the CPU runtime configuration.

This benchmark is suitable for the evaluation of SPARQL generation methods, as it contains natural language questions with generated SPARQL queries, executed over Wikidata. Furthermore, the two configurations (with and without mentions) allow for the assessment of the impact that entity and property annotations have on the generated query.

4. Methodology

4.1. Evaluated Methods

We evaluate five configurations for SPARQL prediction: two retrieval-based sanity checks (TF-IDF and BM25) and three LLM-based strategies (zero-shot, few-shot, and few-shot with repair). Figure 1 displays the general architecture of our system.

All methods start from the same input question, but differ in how the SPARQL query is generated or selected. The retrieval-based methods are used as nearest-neighbor query-copy sanity checks: they retrieve the most similar question from the training split and copy its associated SPARQL query unchanged as the final prediction, without adapting its entities or relations to the target question. The LLM-based methods, in contrast, generate a new SPARQL query for each input.

Based on the experimental configuration, the prompt may contain additional examples (few-shot configuration), and the generated answer may undergo an additional step of automated repair (few-shot with repair configuration).

4.1.1. Retrieval-Based Sanity Checks

We implemented TF-IDF and BM25 as nearest-neighbor query-copy sanity checks. Both approaches operate exclusively on the examples from the training dataset split and do not employ any LLMs. For each question from the dataset, the method identifies the most similar question from the training split and directly retrieves the gold SPARQL query associated to it, passing it as the final prediction. No entity, property or relation adaptation is performed on the retrieved query.

For TF-IDF, questions are represented as TF-IDF weighted vectors and ranked using cosine similarity. BM25 however ranks the questions from the dataset based on term frequency, inverse document frequency and document-length normalization, using a probabilistic relevance function. In the Simple track, retrieval is based on the question text alone. In the With Mentions track, the retrieval representation additionally incorporates the provided mention strings and their entity and property identifiers. Thus, the annotations can affect which training query is retrieved even though they are not used to modify the retrieved SPARQL query itself.

Because the selected SPARQL query is copied without adapting its entities or relations to the target question, these approaches are not intended as general text-to-SPARQL systems. They can produce a correct answer primarily when the retrieved training example is sufficiently compatible with the target for direct query reuse, such as in duplicate, near-duplicate, or semantically equivalent cases. We therefore use them as sanity checks to measure how far lexical similarity and direct reuse of training queries can perform without query generation or adaptation.

4.1.2. LLM Prompting Configurations

In the zero-shot configuration, the prompt contains no question-SPARQL demonstrations. For the Simple track, it contains the instructions and input question, requiring the model to infer the relevant Wikidata identifiers. For the With Mentions track, the provided entity and property annotations are additionally included in the prompt. No dedicated entity or relation linking component is used.

In the few-shot configuration, the prompt is extended with three question-SPARQL examples dynamically retrieved from the training split using BM25. The top three questions with the highest lexical relevance scores are selected for each input question and their corresponding SPARQL queries are included as demonstrations.

In both configurations, the model is instructed to return only the SPARQL query without additional explanations, allowing the generated output to be executed directly on the WikiKGQA endpoint.

4.1.3. Query Repair

The generated queries may contain syntactic or semantic errors that affect their correctness or execution. To evaluate whether a second LLM pass can mitigate these errors, we introduce an automatic query repair stage. The initially generated SPARQL query is submitted to the same LLM together with the original question and, when available, the entity and property mentions. The model is instructed to correct the query only when necessary and return only the resulting SPARQL query. The repaired query is then used as the final prediction for evaluation.

The idea of refining an initial model output through an additional revision stage has also been explored in previous work, such as RARR (Researching and Revising what Language Models Say) [19], which uses LLMs to iteratively improve generated responses based on external evidence.

To ensure the reproducibility of WikiProSP, Figure 2 presents the prompts used during the interaction with the LLM. The first prompt was employed for SPARQL query generation in both zero-shot and few-shot configurations, with the few-shot variant additionally including retrieved question-query examples, while the second prompt was used during the automatic repair stage to identify and correct syntactic or semantic errors in the initially generated SPARQL query before its execution.

5. Experimental Setup

All the experiments were conducted within the WikiKGQA 2026 challenge, using the official training, development and testing datasets provided by the organizers [18]. The experimental configurations were developed and compared exclusively on the validation dataset split, while the test set was only used to generate the final predictions, uploaded on Codabench.

For the TF-IDF method, the questions were represented with the use of the TF-IDF weights and cosine similarity. We used uni-grams and bi-grams to cover as many individual words as possible, as well as short but relevant expressions used in the questions. The resulting vocabularies, constructed from the training split, contains 4422 unique terms.

For BM25 we used the default parameters of the BM25Okapi implementation ($k_1 = 1.5$, $b = 0.75$, $\epsilon = 0.25$) [20], [21], without additional changes. We retain these default parameters to provide a reproducible retrieval configuration without tuning BM25 on the development set.

For the LLM-based methods we opted to use OpenAI’s GPT-4.1, with the temperature set to 0, in order to avoid creativity-induced hallucinations. We also considered Gemini’s 2.5 Flash, but the API

SPARQL Query Generation Prompt	SPARQL Query Repair Prompt
Generate a valid Wikidata SPARQL query that answers the question. Use Wikidata prefixes: - wd: for entities - wdt: for direct properties - p:, ps:, pq:, psv: only when quantities or qualifiers are needed - Do not use SERVICE wikibase:label or any external SERVICE clause - Return Wikidata entity and property identifiers, not labels. Return only the SPARQL query. Do not explain. Question: {question} {mention_text}	The following SPARQL query was generated for the question, but it may contain mistakes. Repair it so that it correctly answers the question, if needed. Question: {question} Generated SPARQL: {original_sparql} {mention_text} Rules: - Keep the query valid for Wikidata - Use wd: for entities and wdt: for direct properties - Use p:, ps:, pq:, psv: only when qualifiers or quantities are needed - Return only the repaired SPARQL query - Do not explain anything - Do not use SERVICE wikibase:label or any external SERVICE clause - Return Wikidata entity and property identifiers, not labels.

Figure 2: Prompts used by the LLM. Left: SPARQL query generation prompt used for the zero-shot and few-shot configurations. Right: prompt used during the automatic query repair stage.

Table 2
Experimental Configuration Used Throughout the Evaluation

Parameter	Value
Large Language Model	GPT-4.1
Temperature	0.0
Few-shot Examples	3
Few-shot Retriever	BM25
TF-IDF n-grams	(1,2)
TF-IDF Similarity Metric	Cosine Similarity
BM25 Implementation	BM25Okapi (default parameters)
Retrieval Corpus	Training set
Evaluation Dataset	Development set
SPARQL Endpoint	Official WikiKGQA QLever endpoint
Evaluation Metrics	Macro QALD-F1, Exact Accuracy, Execution Rate

calls were demonstrating unstable behavior, as well as GPT-4.0-mini and GPT-4.5-mini, which produced significantly weaker results. In an effort to balance cost and performance, for the few-shot variants of the LLM approach we used top-three most relevant examples retrieved with BM25 from the training dataset. In order to facilitate the reproducibility of the experiments, Table 2 synthesizes the main experimental configurations used for the evaluation step.

The generated SPARQL queries were executed on the provided SPARQL endpoint, which hosts the Wikidata copy used to evaluate the participating systems. Development results were evaluated using a local implementation of Macro QALD-F1 aligned with the competition metric, which measures the overlap between the generated and the expected answer. Exact Answer Accuracy measures the proportion of questions for which the predicted answer set exactly matches the gold answer set (equivalent to a QALD-F1 score of 1.0). Execution Rate measures the proportion of generated SPARQL queries that were successfully executed on the Wikidata endpoint, regardless of whether the returned answers were correct. Together, these metrics assess answer quality, exact correctness, and query

executability.

5.1. Additional Experimental Variants

Beyond the experiments presented in the previous section, we investigated additional experimental directions, implemented independently by *Alexia Dan* and submitted separately to the official WikiKGQA 2026 challenge under the Codabench team *alexia*. Since both implementations shared the same objective of prompt-based SPARQL query generation for WikiKGQA, the findings were incorporated into the overall analysis presented in this paper.

5.1.1. Fine-Tuned T5

As an additional supervised baseline, we fine-tuned **t5-small**, to directly map English natural language questions to SPARQL queries. The model was trained on the 317 questions With Mentions training split for 10 epochs, using a batch size of 8 and a learning rate of 3×10^{-4} . Input questions were prefixed with "translate to SPARQL", with a maximum input and target sequence length of 64, respectively 256 tokens. The model obtained a macro QALD-F1 score of 0.000 on the 100 questions development set. Inspection of its predictions showed that it frequently generated identifiers observed during training, rather than the entity and property identifiers required by the target question, showing the difficulty of learning the large Wikidata identifier space from such few supervised examples.

5.1.2. Iterative Schema-Grounded Repair

We have additionally evaluated an iterative generation strategy, employing *Llama-3.3-70B*. Five structurally similar training examples are retrieved dynamically using Sentence-BERT (*all-MiniLM-L6-v2*) and included in the prompt. Before generation, properties associated with the Wikidata entities provided in the question annotations are retrieved from the SPARQL endpoint and supplied as additional schema context. After having generated the candidate query, it is executed and evaluated by an execution-based gate. A candidate is accepted if it executed successfully and returned at least one result. Therefore, repair is triggered both when query execution fails and when execution succeeds, but produces an empty result set. The model then re-prompts the original context, the query and its execution error or empty-result signal. The process is bounded to a maximum of three iterations.

The iterative strategy obtains a macro QALD-F1 of 0.402, compared to 0.439 for the corresponding static few-shot approach. This does not imply that repairing an execution-failing query directly transforms a positively-scored prediction into a zero-scoring one. First, the iterative configuration differs from the static few-shot from its initial generation, as the schema information retrieved from Wikidata is already included in the first prompt. The two configurations therefore do not necessarily produce the same initial candidate query. Moreover, the execution gate only distinguishes execution failures or empty results from non-empty results and it does not determine whether the returned answers are correct. An incorrect query that executes successfully and returns unrelated results is accepted without any further repair. Among queries for which repair was triggered, regeneration could also replace a partially useful candidate with another incorrect query.

6. Results and Discussion

6.1. Overall Results

The proposed system was evaluated both on the development dataset, used for comparing the experimental configurations, as well as on the official test set. The final results that were obtained on Codabench confirm the trends observed during the development stage.

On the official dataset, our system obtained a score of 0.63 on the subtask **With Mentions**, respectively 0.11 for the **Simple** subtask. The significant difference in performance between the two configurations confirms the importance of additional information about Wikidata entities and properties in the

Table 3
Experimental Results Across all Evaluated Methods on the Development Split

Dataset	Method	QALD-F1	Exact Accuracy	Execution Rate
Simple	TF-IDF	0.0339	0.02	1.00
	BM25	0.0339	0.02	1.00
	Zero-shot	0.1781	0.11	0.96
	Few-shot	0.3163	0.29	1.00
	Few-shot + Repair	0.2813	0.25	0.96
With Mentions	TF-IDF	0.0439	0.03	1.00
	BM25	0.0439	0.03	1.00
	Zero-shot	0.4670	0.38	0.98
	Few-shot	0.5597	0.51	1.00
	Few-shot + Repair	0.5125	0.45	0.99

generation process of the queries. In order to further analyze the performance of the different methods, we present the results obtained on the development dataset split.

6.2. Retrieval vs LLM Prompting

As seen in Table 3, the LLM-based methods substantially outperform the TF-IDF and BM25 query-copy sanity checks in both QALD-F1 and Exact Answer Accuracy. The retrieval approaches do not generate or adapt SPARQL queries: they retrieve the nearest training question and reuse its gold query unchanged. Consequently, both methods achieve an Execution Rate of 1.00, since the retrieved queries are valid gold queries from the training set. However, their low QALD-F1 scores show that executability does not imply semantic correctness: lexical similarity alone is generally insufficient for transferring complete SPARQL queries between questions. Correct predictions are limited to cases in which the retrieved training query is also applicable to the target question. In contrast, the LLM-based methods generate a new query conditioned on the target input, allowing entities, properties, and query structure to vary between examples.

6.3. Few-Shot and Repair

As seen in Table 3, introducing examples in the few-shot configuration contributes to consistent improvements compared to the zero-shot variant. The automated retrieval of similar examples from the training split offers the model the expected structure of the SPARQL queries, facilitating the adaptation to new questions, therefore resulting in an improvement of QALD-F1 and Exact Match Accuracy.

However, the query repair mechanism does not show the same trend. An analysis of the original and repaired predictions shows that, on the Simple split, repair improved the QALD-F1 of 5 queries, but worsened 7, while on the With Mentions split, repair improved 6 queries and worsened 11. The observed degradation is primarily semantic rather than syntactic, the results indicating that the second LLM pass can over-correct already valid queries. Overall, the experiments show that the use of relevant examples in the prompt is the main factor that contributes to a better performance, while query repair would benefit from an additional validation or selective application step to avoid degrading already correct predictions.

6.4. Few-Shot Demonstration Selection Ablation

In order to assess whether the improvement of the few-shot configuration comes from the presence of demonstrations alone or from their BM25-based selection, we compare three 3-shot strategies on the development split: a fixed set of three training examples used for every question, three randomly selected examples and three examples dynamically retrieved with BM25 for each target question. For the random strategy, we report the mean and standard deviation across five runs using seeds from

Table 4
Few-Shot Demonstration Selection Ablation on the Development Split

Dataset	Strategy	QALD-F1	Exact Accuracy	Execution Rate
Simple	Zero-shot	0.1781	0.110	0.960
	Fixed 3-shot	0.1983	0.180	0.990
	Random 3-shot	0.1966 (0.0101)	0.168	0.976
	BM25 3-shot	0.3163	0.290	1.000
With Mentions	Zero-shot	0.4670	0.380	0.980
	Fixed 3-shot	0.4305	0.370	0.990
	Random 3-shot	0.4811 (0.0289)	0.416	0.980
	BM25 3-shot	0.5597	0.510	1.000

42 to 46. All other experimental conditions were kept unchanged. The results of this ablation can be consulted in Table 4.

The BM25-based demonstration selection consistently achieves the highest QALD-F1 among the evaluated strategies. In the Simple configuration, BM25-selected examples achieve QALD-F1 of 0.3163, outperforming both the fixed strategy (0.1983) and the random strategy (0.1966). In the With Mentions version, BM25 achieves again the highest QALD-F1 with 0.5597, compared to 0.4305 for fixed demonstrations and 0.4811 for random demonstrations. These results indicate that the gains of the few-shot method are not explained by the existence of examples alone. Dynamically selecting demonstrations that are lexically relevant to the target question provides an additional performance benefit, particularly in the Simple variant.

6.5. Effect of Mentions

In order to evaluate the impact of mentions, we compare the performance of each of the methods in the two experimental configurations: **Simple** and **With Mentions**. The results obtained on the development dataset can be consulted in Table 3, while the differences between the two configurations are displayed in Figure 3. The results show that introducing mentions generally leads to an improvement in performance, the biggest improvement being observed in the case of LLM-based methods, suggesting that additional information is more efficiently exploited for query generation than in query retrieval.

This observation shows that mentions not only offer additional context, but directly contribute to the generation process. By reducing ambiguity and offering explicit hints related to entities and properties, the model can generate better queries. The same trend can be observed in the official evaluation, where the **With Mentions** configuration obtained a QALD-F1 score of 0.63, compared to 0.11 in the **Simple** configuration. These results support the conclusion that the explicit integration of information about entities and properties represents one of the most important factors in performance improvement for prompting-based systems in KGQA tasks.

6.6. Limitations

Although the experimental results demonstrate the efficacy of the prompt-based approaches for SPARQL query generation, our proposed system has a few limitations that can influence performance. A first such limitation is the exclusive use of prompt engineering techniques, without adaptation or fine-tuning.

A further limitation is the absence of a dedicated entity and relation linking component. In the *Simple* configuration, the LLM must directly deduce from the natural language question the relevant Wikidata identifiers, which creates a dependence on the LLM’s capacity of identifying and generating the QIDs and PIDs. Therefore, an incorrect identifier can produce the wrong SPARQL query, even if the general structure is correct. In the *With Mentions* configuration, this limitation is partially mitigated by entity and property annotations. However, these annotations are a form of external grounding, provided by the benchmark. Therefore, the results obtained in the *With Mentions* variant should not be interpreted

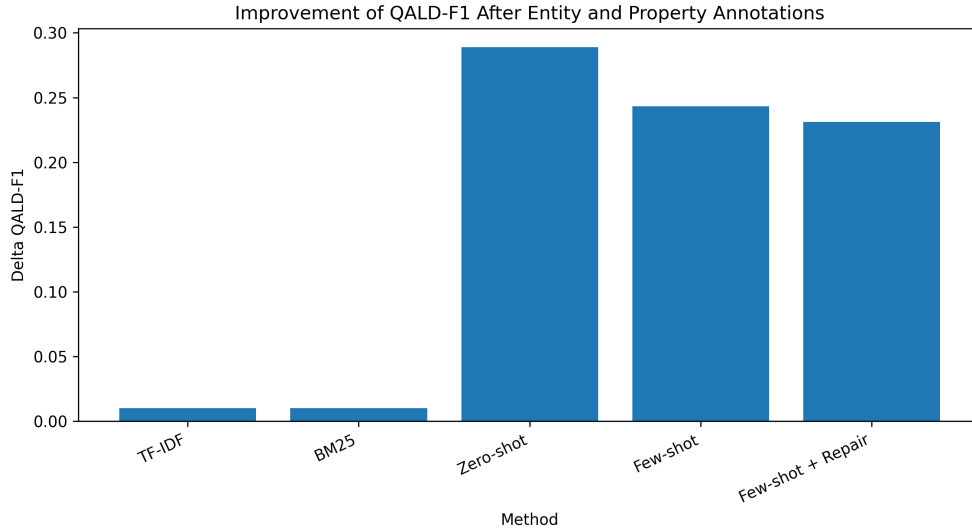


Figure 3: Impact of Annotations on QALD-F1

as measuring end-to-end entity-linking performance.

Another limitation is the selection mechanism of the examples provided to the LLM in the two few-shot variants. The examples are retrieved using BM25, a method based on lexical similarity. Therefore, the quality of the retrieved examples depends directly on lexical overlap between the questions in the training set and the input question, without using embeddings.

Moreover, the experiments were evaluated on the development dataset given by the organizers, while the test set was used exclusively for generating the final predictions. The size of the benchmark is quite small (317 training questions, 100 development and 75 for testing), which can limit the generalization of the conclusions and can influence the performance of example retrieval.

7. Conclusion

In this study we investigated the efficiency of prompt engineering based strategies for automated generation of SPARQL queries on the WikiKGQA 2026 benchmark. We compare two retrieval-based methods (TF-IDF and BM25) and three LLM-based ones (zero-shot, few-shot and few-shot with repair), evaluating them on two datasets: a natural language questions only dataset, as well as one with additional annotations about relevant entities and properties. The experimental results showed that LLM prompting-based methods consistently outperform the retrieval-based methods; the use of few-shot configurations brings significant performance improvements, the best results having been obtained on the dataset with mentions. However, the automated repair step produces competitive results without overtaking its simple few-shot variant.

Overall, the results confirm that modern prompting strategies represent an efficient approach for SPARQL query generation for Wikidata, without the need of additional training of the models. Moreover, this paper highlights the importance of using structural information about entities and properties and offers an experimental analysis that can serve as a reference point for the development and evaluation of future KGQA systems based on LLMs.

Declaration on Generative AI

The authors used OpenAI GPT-5.5 and Perplexity AI to assist with stylistic language editing to improve the clarity of this article. The tool was not used to generate scientific content or draw conclusions; the authors reviewed and edited the content as needed and take full responsibility for the content.

References

- [1] D. Banerjee, P. A. Nair, J. N. Kaur, R. Usbeck, C. Biemann, Modern Baselines for SPARQL Semantic Parsing, in: Proc. of the 45th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval, ACM, 2022, pp. 2260–2265. doi:10.1145/3477495.3531841.
- [2] L. Jiang, R. Usbeck, Knowledge Graph Question Answering Datasets and Their Generalizability: Are They Enough for Future Research?, in: Proc. of the 45th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval, ACM, 2022, pp. 3209–3218. doi:10.1145/3477495.3531751.
- [3] J. D’Abramo, A. Zugarini, P. Torroni, Investigating Large Language Models for Text-to-SPARQL Generation, in: Proc. of the 4th Intl. Workshop on Knowledge-Augmented Methods for Natural Language Processing, ACL, 2025, pp. 66–80. doi:10.18653/v1/2025.knowledgenlp-1.5.
- [4] L. Kovriguina, R. Teucher, D. Radyush, D. Mourmstsev, SPARQLGEN: One-Shot Prompt-based Approach for SPARQL Query Generation, in: Proc. of the 19th Intl. Conf. on Semantic Systems (SEMANTiCS 2023), volume 3526 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: <https://ceur-ws.org/Vol-3526/paper-08.pdf>.
- [5] A. Gashkov, A. Perevalov, M. Eltsova, A. Both, SPARQL Query Generation with LLMs: Measuring the Impact of Training Data Memorization and Knowledge Injection, in: Web Engineering, Springer Nature Switzerland, 2026, pp. 177–192. doi:10.1007/978-3-031-97207-2_14.
- [6] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: A Nucleus for a Web of Open Data, in: The Semantic Web, Springer, 2007, pp. 722–735. doi:10.1007/978-3-540-76298-0_52.
- [7] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proc. of the 2008 ACM SIGMOD Intl. Conf. on Management of Data, ACM, 2008, pp. 1247–1250. doi:10.1145/1376616.1376746.
- [8] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85. doi:10.1145/2629489.
- [9] D. Diefenbach, T. P. Tanon, K. D. Singh, P. Maret, Question Answering Benchmarks for Wikidata, in: Proc. of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th Intl. Semantic Web Conference (ISWC 2017), 2017, volume 1963 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017. URL: <https://ceur-ws.org/Vol-1963/paper555.pdf>.
- [10] A. Perevalov, X. Yan, L. Kovriguina, L. Jiang, A. Both, R. Usbeck, Knowledge Graph Question Answering Leaderboard: A Community Resource to Prevent a Replication Crisis, in: Proc. of the Thirteenth Language Resources and Evaluation Conference, European Language Resources Association, 2022, pp. 2998–3007. URL: <https://aclanthology.org/2022.lrec-1.321/>.
- [11] A. Waagmeester, G. Stupp, S. Burgstaller-Muehlbacher, B. M. Good, M. Griffith, O. L. Griffith, K. Hanspers, H. Hermjakob, T. S. Hudson, K. Hybiske, S. M. Keating, M. Manske, M. Mayers, D. Mietchen, E. Mitraka, A. R. Pico, T. Putman, A. Riutta, N. Queralt-Rosinach, L. M. Schriml, T. Shafee, D. Slenter, R. Stephan, K. Thornton, G. Tsueng, R. Tu, S. Ul-Hasan, E. Willighagen, C. Wu, A. I. Su, Wikidata as a knowledge graph for the life sciences, eLife 9 (2020) e52614. doi:10.7554/eLife.52614.
- [12] P. Sen, A. F. Aji, A. Saffari, Mintaka: A Complex, Natural, and Multilingual Dataset for End-to-End Question Answering, in: Proc. of the 29th Intl. Conf. on Computational Linguistics, International Committee on Computational Linguistics, 2022, pp. 1604–1619. URL: <https://aclanthology.org/2022.coling-1.138/>.
- [13] M. Dubey, D. Banerjee, D. Chaudhuri, J. Lehmann, EARL: Joint Entity and Relation Linking for Question Answering over Knowledge Graphs, in: The Semantic Web – ISWC 2018: 17th Intl. Semantic Web Conference, 2018, Proc. Part I, Springer-Verlag, 2018, p. 108–126. doi:10.1007/978-3-030-00671-6_7.
- [14] M. Li, S. Ji, Semantic Structure Based Query Graph Prediction for Question Answering over Knowledge Graph, in: Proc. of the 29th Intl. Conf. on Computational Linguistics, International Committee on Computational Linguistics, Gyeongju, Republic of Korea, 2022, pp. 1569–1579. URL:

<https://aclanthology.org/2022.coling-1.135/>.

- [15] S. Srivastava, M. Patidar, S. Chowdhury, P. Agarwal, I. Bhattacharya, G. Shroff, Complex Question Answering on knowledge graphs using machine translation and multi-task learning, in: Proc. of the 16th Conf. of the European Chapter of the Association for Computational Linguistics: Main Volume, ACL, Online, 2021, pp. 3428–3439. doi:10.18653/v1/2021.eacl-main.300.
- [16] F. F. Luz, M. Finger, Semantic Parsing Natural Language into SPARQL: Improving Target Language Representation with Neural Attention, 2018. doi:10.48550/arXiv.1803.04329, arXiv:1803.04329.
- [17] V. Emonet, J. T. Bolleman, S. Duvaud, T. M. de Farias, A. C. Sima, LLM-based SPARQL Query Generation from Natural Language over Federated Knowledge Graphs, in: Proc. of the Special Session on Harmonising Generative AI and Semantic Web Technologies (HGAIS 2024) co-located with the 23rd Intl. Semantic Web Conference (ISWC 2024), 2024, volume 3953 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: <https://ceur-ws.org/Vol-3953/355.pdf>.
- [18] S. Arzoo, A. Hogan, A. Moya Loustaunau, D. M. Schmidt, QAWiki v2026.2: Knowledge Graph Question Answering (KGQA) / SPARQL Query Generation Dataset for Wikidata, 2026. doi:10.5281/ZENODO.19474009.
- [19] L. Gao, Z. Dai, P. Pasupat, A. Chen, A. T. Chaganty, Y. Fan, V. Zhao, N. Lao, H. Lee, D.-C. Juan, K. Guu, RARR: Researching and revising what language models say, using language models, in: A. Rogers, J. Boyd-Graber, N. Okazaki (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 16477–16508. URL: <https://aclanthology.org/2023.acl-long.910/>. doi:10.18653/v1/2023.acl-long.910.
- [20] D. Brown, Rank-BM25: A Collection of BM25 Algorithms in Python, 2020. URL: <https://doi.org/10.5281/zenodo.4520057>. doi:10.5281/zenodo.4520057.
- [21] C. Spinoso-Di Piano, S. Rahimi, J. Cheung, Qualitative Code Suggestion: A Human-Centric Approach to Qualitative Coding, in: H. Bouamor, J. Pino, K. Bali (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore, 2023, pp. 14887–14909. URL: <https://aclanthology.org/2023.findings-emnlp.993/>. doi:10.18653/v1/2023.findings-emnlp.993.