SPARQLGEN: One-Shot Prompt-based Approach for SPARQL Query Generation

Liubov Kovriguina[1,2,*], Roman Teucher[2], Daniil Radyush[3] and Dmitry Mouromtsev[4]

[1]*metaphacts GmbH, Daimlerstraße 36, 69190, Walldorf, Germany*

[2]*Fraunhofer IAIS Dresden, Schloss Birlinghoven 1, 53757, Sankt Augustin, Germany*

[3]*ITMO University, Kronverksky Pr. 49, bldg. A, St. Petersburg, 197101, Russia*

[4]*TIB – Leibniz-Informationszentrum Technik und Naturwissenschaften und Universitätsbibliothek, Welfengarten 1B, 30167 Hannover, Germany*

Abstract

In this work, we present a one-shot generative approach (further referred to as SPARQLGEN) for generating SPARQL queries by augmenting Large Language Models (LLMs) with the relevant context within a single prompt. The prompt includes heterogeneous data sources: a question itself, an RDF subgraph required to answer the question, and an example of a correct SPARQL query for a different question. In the experiments, GPT-3, a popular pre-trained language model from OpenAI, was leveraged, but it is possible to extend the approach to any other generative LLM. We evaluate, how different types of context in the prompt influence the query generation performance on QALD-9, QALD-10 and Bestiary dataset (BESTIARY), which was created to test LLM performance on unseen data, and provide a detailed error analysis. One of the findings is that providing the model with the underlying KG and a random correct query improve the generation results. The approach shows strong results on QALD-9 dataset, but doesn't generalize on QALD-10 and BESTIARY, which can be caused by memorization problem.

Keywords

Knowledge Graphs Question Answering, SPARQL query generation, Augmented Large Language Models, Prompt Template Design

1. Introduction

In the current paper, we propose a one-shot approach for generating SPARQL queries with prompting LLMs, further referred as SPARQLGEN. Our approach lies in augmenting LLMs [1] with a knowledge graph fragment, required to construct the query, and a question-subgraph-query example, randomly sampled from the training set. Assembling all the context, required to generate a query, in a single prompt, is performed via loosely coupled heterogeneous structured information snippets, further referred as *prompt elements*. The prompt element is represented as a structure, having *description* and *source* and a set of pre-processing methods (i.e. for sampling, serializing to string, ranking, linearizing), that are specific to the prompt element and allow to combine heterogeneous data sources within a single prompt (Fig. 1). This allows to quickly and flexibly build custom prompt templates with an arbitrary order and number of elements.

SEMANTICS 2023 EU: 19th International Conference on Semantic Systems, September 20-22, 2023, Leipzig, Germany

*Corresponding author.

lk@metaphacts.com (L. Kovriguina); roman.teucher@iais.fraunhofer.de (R. Teucher); daniil.radyush@gmail.com (D. Radyush); d.muromtsev@gmail.com (D. Mouromtsev)

The experiments, implemented by now, pursue two main goals: (i) leverage LLM for SPARQL query generation in one-shot / zero-shot setup without fine-tuning, (ii) estimate the impact of different prompt elements on the query generation task and how the model attends to them: i.e., does the model focus on reasoning over the provided contextual information, rather than retrieving what it already knows, at the inference step. Since memorization is a known problem of LLMs [2], we created an extra dataset on a fantasy bestiary topic, following the QALD format, to evaluate the method on the data, which could not be seen during LLM training[1].

Main contributions of the paper are the following:

- SPARQLGEN, a one-shot method for SPARQL query generation with prompting LLMs, that can be quickly adapted to other datasets and knowledge graphs,
- unseen BESTIARY dataset, that can be particularly useful for evaluating reasoning capabilities of LLMs,
- detailed error analysis of LLM generated queries across QALD-9, QALD-10 and BESTIARY datasets.

2. Related Work

There are several recent approaches for generating SPARQL queries from natural language queries based on neural architectures. Soru et al. [3] designed a sequence-to-sequence system that utilizes bi-directional LSTM for generating SPARQL templates. Using a rule-based approach, the SPARQL query is then created from the generated templates. However, this translational approach cannot handle out-of-vocabulary tokens. Rony et al. [4] propose SGPT, an approach using a stack of Transformer encoders to embed linguistic features from natural language questions, as well as entity and relation information, to the GPT-2 model. While entities and relations representations are fed to the model in SGPT, providing their connections in the underlying KG is missing. Thus, generating correct triple sequences in the final SPARQL queries is error prone due to unknown graph structures. Another strong approach was introduced in [5], where authors improve on the state of the art KGQA[2] by train the T5 model to generate skeleton SPARQL gueries and truncated KG embeddings, that are used to fetch candidate entities for the skeleton query. However, all these approaches assume training or fine-tuning an existing model, whereas SPARQLGEN doesn't require any training.

3. Datasets

QALD-9 [6] is a small yet challenging multilingual question answering dataset based on DBpedia. The dataset contains 150 questions in 3 to 8 different languages, for our experiments the test data in English were used. **QALD-10** is a multilingual question answering dataset based on Wikidata. It contains 394 samples with questions in English, Chinese, Russian and German. The dataset is more complex than QALD-9, for instance, by incorporating property paths instead

[1]The augmented data used for prompting, the BESTIARY dataset, as well as supplementary material, are uploaded to the repository:https://github.com/danrd/sparqlgen

[2]https://github.com/KGQA/leaderboard

of only using single relations. **BESTIARY** dataset consists of 100 manually created queries related to a custom Bestiary knowledge graph. This graph contains diverse information about creatures from the Dungeons & Dragons fantasy role-playing game. The graph and dataset description are presented in the supplementary material.

4. Architecture Description

SPARQLGEN is implemented as a modular architecture with the following business logic: (1) retrieving context to populate prompt elements, (2) composing and executing the prompt, (3) removing hallucinations and validating the query. At the preprocessing step, each datapoint in the QALD-formatted dataset was augmented with the knowledge, represented as the minimal subgraph, required to execute the query. To combine heterogeneous data sources in the prompt, we designed an abstract structure, called prompt element, that is instantiated during the experiment. Each prompt element has fields *description* and *source*, as well as methods for pre-processing the *source* data (see prompt elements *Example*, *Instruction*, *Question* and *KnowledgeGraph* in Fig. 1). A prompt in SPARQLGEN is a serialized sequence of prompt elements. The implemented structure allows to configure experiments with minimal changes in the code structure and quickly design custom prompt templates.

For one-shot prompting we created a set of guiding examples, which includes 20 question-subgraph-query samples, selected from QALD-9 training set and representing different query types (ASK or SELECT), patterns (number of hops), combinations of modifiers (FILTER, ORDER BY, etc.). Guiding examples repeat the structure of prompt, but already provide the correct answers (one-shot prompting).

The architecture of the SPARQLGEN pipeline is shown in Fig. 1. Firstly, for each datapoint in QALD format, already augmented with a subgraph, a guiding example is randomly selected (prompt element *Example*). Then the prompt builder constructs a prompt from the provided datapoint and a guiding example with the order of the prompt elements, defined in the experiment config, as shown in Fig. 1), and the serialized prompt is sent to the GPT-3 Completions endpoint. The resulting query is validated and evaluated. Validation includes removing hallucinated symbols (i.e. generated text prior the query, like *System:*, *Query:*, randomly inserted newlines, etc.)

To investigate whether adding subgraph information to the model improves the performance, we enriched each sample in the test sets of QALD-9, QALD-10 and BESTIARY with a subgraph of the source RDF graph of that dataset. These subgraphs contain all the triples that are required to answer the question correctly, but no irrelevant triples. First, we strip the ground truth SPARQL queries from any modifiers leaving a simple *SELECT ** query. By doing that, we get all possible bindings for the variables in the ground truth query. Second, we extract the individual triples from the query. They contain entities, relations but also still the variables. Third, we take the result bindings from the *SELECT ** query and replace the variables in the extracted triples. The result is a set of triples, which is based on the original ground truth query, representing the subgraph that is sufficient to answer the given query. Further, this subgraph is considerably smaller than taking all triples from all the given entities and relations of the ground truth query (see process diagram in supplementary material).

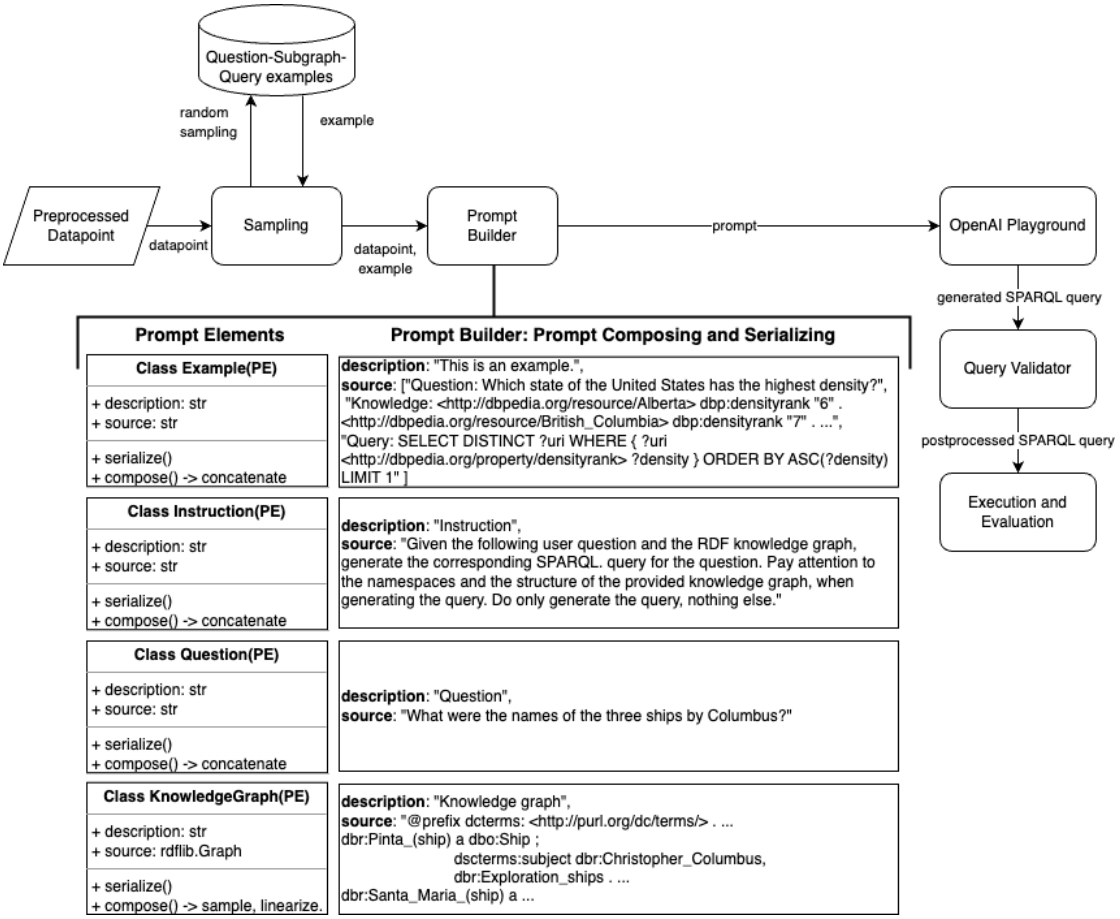

Figure 1: SPARQLGEN architecture and prompt composing in the prompt builder.

5. Experiments

During experiments, we addressed 2 research questions. **RQ1**: How do different types of context (guiding example, knowledge graph, constraints, etc.), provided in the prompt affect LLM inference for the SPARQL query generation? To answer this question, we ran experiments on QALD-9 with different prompts templates: with and without the knowledge graph, with or without guiding example, etc. The results are summarized in Table 1. **RQ2**: How well does the method perform in one-shot setups across different datasets? To tackle this question, we took the best configuration and ran it for 3 datasets with further error analysis. Evaluation results are shown in Table 2 and error analysis is provided in supplementary materials. Results in the

[3]Accuracy values in the table 1 are calculated on a smaller number of samples, than the available version of the QALD-9 contains. This happened for the following reasons: 1) 37 of 150 samples, provided in QALD-9, are no longer executable under the current DBPedia version in the endpoint https://dbpedia.org/sparql and we couldn't augment such samples with a subgraph; 2) when running the query generation for the first time, we didn't take into consideration, that adding the extracted subgraph to the prompt might make the prompt exceed the maximum number of tokens (4097 with the completion) for GPT-3 (number of omitted samples due to too long input is given

Table 1
SPARQL query generation accuracy with different prompt configurations for on QALD-9[3]

Prompt Configuration	Total augmented samples	Executed correctly	Samples not processed due to limited input	Accuracy
Instruction + Question	113	14	0	0.1239
Instruction + Question + Subgraph	113	28	26	0.3218
Instruction + Question + Subgraph + Guiding example	113	38	30	**0.4578**

Table 1 shows, that providing the subgraph to the model increases performance. This suggests that the model can make use of the information and structure of the graph at inference step. One-shot prompting increases the performance as well.

6. Evaluation and Results

We used F1-macro to evaluate the whole query generation as suggested in GERBIL benchmarking system [7]. Results are reported in Table 2. On QALD-9 SPARQLGEN reaches 67.07, that matches the performance of pre-trained SGPT system (67.82), which is currently at the top of the leaderboard [8]. However, the approach doesn't generalize well on the recent QALD-10 and unseen BESTIARY.

Table 2
SPARQLGEN performance across datasets

Dataset		Description	
	F1-macro	Knowledge graph	Prompt configuration
QALD-9	**67.07**	DBPedia	Example + Instruction + Question + Subgraph
QALD-10	28.75	Wikidata	Example + Instruction + Question + Subgraph
BESTIARY	15.01	Bestiary graph	Example + Instruction + Question + Subgraph

Objective evaluation with QALD F1-macro [7] has shown quite diverse results across the 3 datasets. Therefore, we tried to categorize the errors in a small set of categories, that cover all types of errors witnessed during LLM inference. Error classification and distribution is presented in the supplementary materials.

The experimental results show that the model struggles to deal with an unknown knowledge graph. It is safe to assume that GPT-3 has encountered DBpedia as well as Wikidata information

in column 4); 3) for experiments in table 1 we didn't fix hallucinations. When running the experiments on all 3 datasets (Table 2), we addressed these errors: 1) for QALD-9, from 37 samples with non-executable queries we managed to fix 20 queries; 2) also, we added sampling of triples from the subgraph, so that the resulting prompt could never exceed the token limit. Following the evaluation guidelines, suggested in GERBIL [7], if both reference and generated query return empty set, QALD-F1 is set to 1.

in pre-training. This is visible in the overall better performance on the corresponding datasets, as well as in the higher number of knowledge related errors in BESTIARY. Namespace errors, incomplete triples and ignoring KG structure errors occur more frequently for BESTIARY. In general, it seems beneficial to introduce the model to the KG in pre-training already.

7. Conclusion and Future Work

The performance of the one-shot SPARQLGEN approach on QALD-9 is only slightly below the SGPT approach, that requires additional training of the stack of Transformer-encoders to leverage the pre-trained GPT-2 model, and significantly outperforms the rest of the QALD-9 leaderboard. Assuming that SPARQLGEN can be adapted to a new dataset and KG faster than approaches, requiring fine-tuning, continuing experimenting with prompt-based approaches in KGQA can be definitely a winning strategy. However, this approach doesn't generalize perfectly, given the evidence from QALD-10 and BESTIARY.

Memorization can be one explanation: QALD-10 is a recent dataset, which OpenAI models might not seen, and BESTIARY dataset have never been used for training any model. BESTIARY queries are based on the knowledge graph, that was designed specially for structuring the Dungeons & Dragons domain, and this graph has been never made available to the GPT-3. The availability of the unstructured data about Dungeons & Dragons on the Web doesn't imply, that the LLM can synthesize the knowledge graph, following by generating the adequate query. Given that, we assume that a niche domain (for knowledge engineering) combined with an unseen knowledge graph makes the model rely only on its reasoning skills, when querying BESTIARY. The existence of these skills is questionable at its own: parallel research lines prove that LLMs can and can not reason at full scale. However, during query generation against the BESTIARY KG, GPT-3 copies entities and relations from the provided BESTIARY subgraph, without large hallucinations from DBPedia and Wikidata, indicating that providing a subgraph of domain-specific KGs can improve SPARQL query generation.

Future work is manifold. First of all, we would like to proceed with fine-tuning open source LLMs on available train sets of QALD / LC-QuAD series and evaluate the generalization of the fine-tuned model on more benchmarks. Another direction can be embedding the current SPARQLGEN approach in a reinforcement learning pipeline.

References

[1] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, et al., Augmented language models: a survey, arXiv preprint arXiv:2302.07842 (2023).

[2] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, C. Zhang, Quantifying memorization across neural language models, 2023. arXiv:2202.07646.

[3] T. Soru, E. Marx, A. Valdestilhas, D. Esteves, D. Moussallem, G. Publio, Neural machine translation for query construction and composition, 2018. arXiv:1806.10478.

[4] M. R. A. H. Rony, U. Kumar, R. Teucher, L. Kovriguina, J. Lehmann, Sgpt: A generative

approach for sparql query generation from natural language questions, IEEE Access 10 (2022) 70712–70723. doi:10.1109/ACCESS.2022.3188714.

[5] D. Banerjee, P. A. Nair, J. N. Kaur, R. Usbeck, C. Biemann, Modern baselines for sparql semantic parsing, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 2260–2265.

[6] N. Ngomo, 9th challenge on question answering over linked data (qald-9), language 7 (2018) 58–64.

[7] R. Usbeck, M. Röder, M. Hoffmann, F. Conrads, J. Huthmann, A.-C. Ngonga-Ngomo, C. Demmler, C. Unger, Benchmarking question answering systems, Semantic Web 10 (2019) 293–304.

[8] 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: Proceedings of the Thirteenth Language Resources and Evaluation Conference, 2022, pp. 2998–3007.