

Improving Subgraph Extraction Algorithms for One-Shot SPARQL Query Generation with Large Language Models

Dmitrii Pliukhin^{1,*}, Daniil Radyush¹, Liubov Kovriguina^{2,*} and Dmitry Mouromtsev³

¹ITMO University, Kronverksky Pr. 49, bldg. A, St. Petersburg, 197101, Russia

²Independent Researcher, Dresden, Germany

³TIB – Leibniz-Informationszentrum Technik und Naturwissenschaften und Universitätsbibliothek, Welfengarten 1B, 30167 Hannover, Germany

Abstract

Question answering over scholarly knowledge graphs involves many challenges: complex graph patterns, long-tail distributed data, revision and evolution of the scholarly ontologies, and knowledge graphs incompleteness due to constant research dynamics. In this work, we present an LLM-based approach for SPARQL query generation over Open Research Knowledge Graph (ORKG) for the ISWC SciQA Challenge. Our approach proposes a couple of improvements to the recently published SPARQLGEN approach, that performs one-shot SPARQL query generation by augmenting Large Language Models (LLMs) with the relevant context within a single prompt. Similar to SPARQLGEN, we include heterogeneous data sources in the SPARQL generation prompt: a question itself, an RDF subgraph required to answer the question, and an example of a correct SPARQL query. In the current work, we focused on designing subgraph extraction algorithms, that are close to real-life scenarios of generative KGQA, and replaced the random choice of example question-query pair with similarity scoring.

Keywords

Scholarly Knowledge Graphs, Knowledge Graphs Question Answering, SPARQL query generation, Augmented Large Language Models, Subgraph Extraction

1. Introduction


Scholarly knowledge graphs has become a recent trend and inspired the adaptation of KGQA systems to new complex domains, that are constantly evolving, bringing new facts and concepts to the knowledge graph. Currently, there is a number of scholarly knowledge graphs, that differ in metadata and coverage, being built on top of various ontologies and data sources [1, 2, 3]. For knowledge graph question answering (KGQA) systems, such landscape creates a lot of challenges due to variative graph patterns, ambiguity, and complex user questions. Previous KGQA benchmarks (i.e. QALD series and LC-QuAD datasets) were build upon DBpedia and Wikidata and allowed cumulative improvements of KGQA systems, especially for template-based approaches. With the diversity of scholarly KGs, template-based and pre-trained approaches may not work as good as before due to adaptation costs. We suggest to employ generative approaches

ISWC 2023: Scholarly QALD Challenge, November 6-10, 2023, Athens, Greece

✉ zeionara@gmail.com (D. Pliukhin); daniil.radyush@gmail.com (D. Radyush); lkovriguina@gmail.com (L. Kovriguina); d.muromtsev@gmail.com (D. Mouromtsev)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

 CEUR Workshop Proceedings (CEUR-WS.org)

to KGQA, namely within the augmented large language models paradigm, when the LLM gets all the context required to generate the SPARQL query, in a prompt. LLMs augmenting approaches try to mitigate the fundamental defect of LLMs: pure statistical language modeling over limited context size, by providing LLMs information from relevant data sources. Several strategies are known to do the augmentation: (i) retrieval-augmented language models [4], knowledge injection [5], reasoning [6], etc. According to the classification, proposed in [7], providing LLMs with extra context via prompting belongs to augmenting with eliciting reasoning.

In the current paper, we propose further improvements to SPARQLGEN, a recent one-shot approach for generating SPARQL queries with prompting LLMs [8]. SPARQLGEN approach lies in augmenting LLMs with a knowledge graph fragment, required to construct the query, and a question-subgraph-query example within a single prompt (see Sec.2). This knowledge graph fragment, further referred as *subgraph*, contains all the triples that are required to build the correct SPARQL query (that is, to answer the question correctly), but no irrelevant triples, and extracting this subgraph is quite challenging. Since the motivation behind SPARQLGEN was to evaluate, whether an LLM can make use of the provided subgraph and infer graph patterns during SPARQL query generation, the algorithm of subgraph extraction is based on target SPARQL queries of QALD-9 and QALD-10 datasets¹. The disadvantage of this approach is that it will not work on inference, since it requires the ground truth data. For the SciQA challenge, we have designed a subgraph extraction algorithm, combining similarity search with Hierarchical Navigable Small World (HNSW) method, that can be used as a replacement of the subgraph extraction algorithm in SPARQLGEN and which matches better the real-life KGQA scenarios (see Sec.5). The random selection of the guiding example for one-shot prompting is replaced with selection based on the Levenstein distance.

Main contributions of the paper are the following:

- improvement of SPARQLGEN, a one-shot method for SPARQL query generation with prompting LLMs, that can be quickly adapted to other datasets and knowledge graphs,
- subgraph extraction algorithm, that can be used in LLM-augmented SPARQL query generation scenarios,
- evaluation of the improved SPARQLGEN method on the ORKG benchmark.

2. Related Work

Fine-tuning LLMs to generate SPARQL queries has already a proven track record with top-leaderboard results on QALD-9 with SGPT [9] and LC-QuAD with GETT-QA [10]. Besides that, there are also no-SPARQL KGQA approaches, allowing to load the knowledge graph into the question answering LLM pipeline in a retrieval fashion, i.e. see², but we excluded them for now despite considering promising.

Rony et al. [9] 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,

¹See algorithm description in the repository: <https://github.com/danrd/sparqlgen>

²<https://github.com/mommi84/rdf-qa>

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 [10], where authors improve on the state of the art KGQA³ by training the T5 model to generate skeleton SPARQL queries 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.

The recent approach, which we are extending upon, is SPARQLGEN [8], that doesn't require any training and instructs LLMs to generate SPARQL queries by providing them the underlying knowledge graph, guiding examples and information about SPARQL grammar within a single prompt. 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. The *description* field depicts the *source* data, i.e. "The RDF knowledge graph", and the *source* contains the data itself, i.e. triples. This altogether allows to quickly and flexibly build custom prompt templates with an arbitrary order and number of elements.

The findings of SPARQLGEN approach show that the model struggles to deal with an unknown knowledge graph. Namespace errors, incomplete triples and ignoring KG structure errors occur more frequently for the unseen dataset and it seems beneficial to introduce the model to the KG in pre-training already (see supplementary material in the repository: <https://github.com/danrd/sparqlgen>).

3. SciQA Challenge and Datasets

This section presents an overview of the SciQA Challenge and the datasets used in this study.

3.1. SciQA Challenge

The primary objective of this research endeavor was to participate in the Scholarly Question-Answering over Linked Data (Scholarly QALD) challenge. This challenge is centered around Knowledge Graph Question Answering utilizing the ORKG Scholarly Knowledge Graph. The challenge was structured into two distinct stages.

The first stage aimed to develop a model and conduct validation using a held-out subset with known true labels, thereby establishing an initial leaderboard. The second stage focused on assessing the model's quality and generating final results for comparison among the various solutions developed. The central challenge task involved transforming a user's natural language question into a SPARQL query and executing this query on the provided knowledge graph to furnish a response. The efficacy of the proposed solutions was evaluated through a set of 200 questions, employing the F1-score metric in both stages.

³<https://github.com/KGQA/leaderboard>

3.2. Datasets

To train and evaluate the developed models, the SciQA dataset was designated as the target data source. This dataset is publicly available in the repository: <https://zenodo.org/record/7744048>. The SciQA dataset encompasses a rich assortment of records. Each record comprises a question expressed in natural language, the corresponding SPARQL query, and a list of knowledge graph triples that represent the answer to that question.

Notably, the SciQA dataset is characterized by a significant diversity of questions, spanning various domains, response types, and sizes. The questions also exhibit structural and computational complexity and a degree of ambiguity. In addition to the SciQA dataset, the repository contains a link to the dump of the knowledge graph designed for question answering. This knowledge graph is primarily structured around scholarly data, including papers, contributions, and connections between them, accompanied by pertinent metadata.

The knowledge graph contains a total of 1,133,217 triples and 21,243 contributions, with an overall dump size of 152 megabytes. This knowledge graph is inherently heterogeneous, encompassing diverse data pertaining to papers from an array of research fields. Its structure is notably intricate, rendering the challenge particularly demanding. Specifically, the graph incorporates both discrete and continuous data entries, encompassing a wide range of primitive types such as numbers in various formats, dates, strings, and binary labels.

4. Architecture Description

Our approach is implemented as a modular architecture with the following components: (1) subgraph retrieval (2) example selection context to populate prompt elements, (3) prompt building, (4) prompt execution, (5) removing hallucinations and validating the query, (6) query execution.

During steps (1) and (2), each data point in the SciQA test set was augmented with the context, represented as (i) the subgraph, required to execute the query (see Sec. 5), and (ii) a question-query pair, similar to the question (see below). Then the prompt builder (3) constructs a prompt from the augmented SciQA 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-3.5 completions endpoint. The resulting query is validated (5) and executed (6). Validation includes removing hallucinated symbols (i.e. generated text prior the query, like *System:*, *Query:*, randomly inserted newlines, etc.) The pipeline architecture is shown in Fig. 1. Subgraph extraction is described in more detail in Sec. 5.

Following the SPARQLGEN approach, to combine heterogeneous data sources in the prompt, we used 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 *Knowledge Graph* 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.

Example Selection For one-shot prompting, the example was sampled from the train subset of the proposed dataset by computing Levenstein distance between input question and every

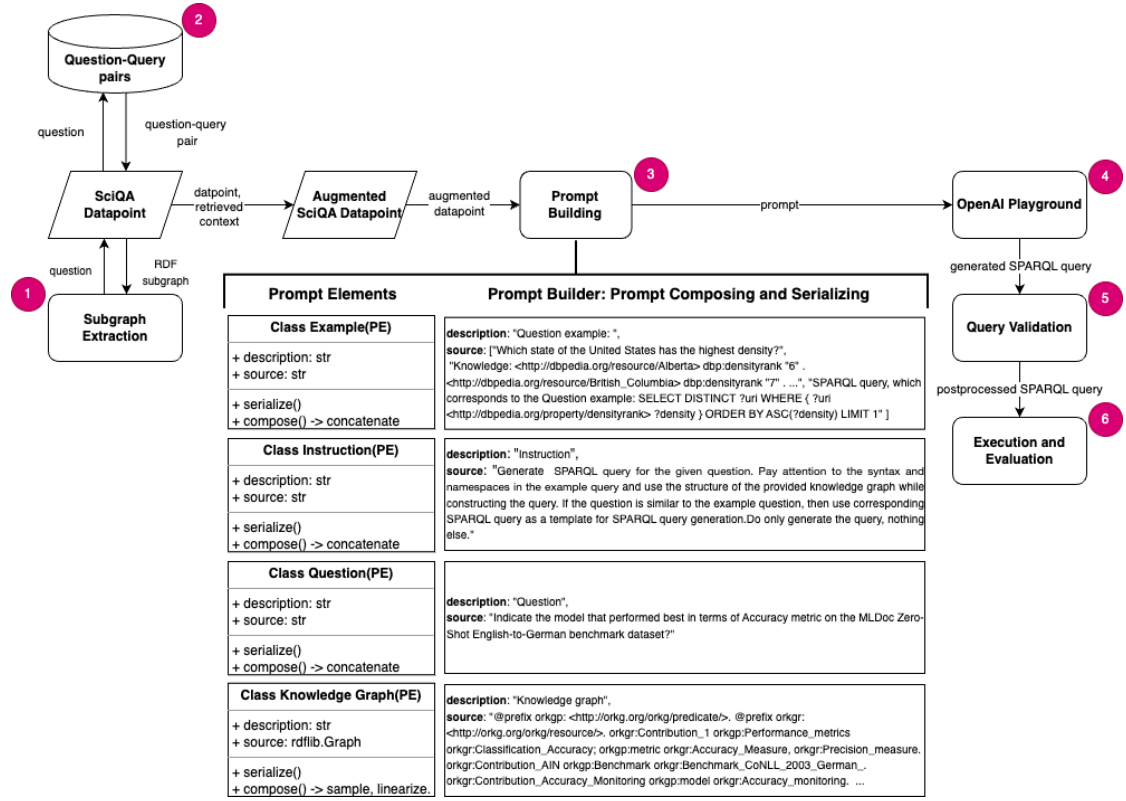


Figure 1: Pipeline Architecture

question from the train subset. Subsequently the list of questions was sorted by Levenstein distance, and record with the lowest distance was selected for prompt generation. The approach based on Levenstein distance instead of semantic similarity has been selected due to the low diversity of the dump dataset, and a lot of instances in the test subset which had at least one training sample with the identical structure except a few property values.

5. Subgraph Extraction Algorithms

Our approach is based on finding for each question semantically related objects in ORKG, namely resources, properties, papers and contributions. For this purpose, 'all-mpnet-base-v2' sentence transformer [11] model was leveraged to measure the similarity between question and object labels embeddings. Furthermore, Hierarchical Navigable Small World (HNSW) method [12] was used for ORKG objects indexing to facilitate vector search that constitutes **preliminary (0) step** for the subgraph extraction approaches.

Essentially, the process of subgraph retrieval comprises the following steps. **(1) question processing:** filtering out parts of speech, except nouns, adjectives and verbs; bigrams generation and encoding. **(2) relevant objects extraction:** defining most relevant to the question ORKG objects based on cosine similarity between its bigrams embeddings and embeddings of ORKG

№	Hyperparameters	Steps	Queries patterns
1	n_prop - list of n properties n_paper - list of n papers n_cont - list of n contributions	1) For each question's bigram retrieve relevant n_prop , n_paper and n_cont by indexes; 2) Form n_prop , n_paper and n_cont for the question keeping most relevant objects; 3) Extract subgraphs for n_paper and n_cont from ORKG; 4) Retain only triples with properties from n_prop .	{<paper> ?x ?y ?y <property> <label>} {<contribution> <property> <label>}
2	n_prop - list of n properties n_res - list of n resources	1) For each question's bigram retrieve relevant n_prop and n_res by indexes; 2) Form n_prop for the question keeping most relevant properties; 3) Discard resources with cosine similarity to the question less than 0.6; 4) Extract subgraphs for n_res from ORKG; 5) Retain only triples with properties from n_prop .	{<resource> ?x ?y ?y <property> <label>} {?x ?y <resource> <resource> <property> <label>}

Figure 2: Subgraphs construction approaches.

object labels leveraging initially constructed with HNSW indexes. **(3) subgraphs retrieval:** deriving 2-hop paths containing given similar objects. **(4) subgraphs merging** and converting into string with postprocessing for prompting.

In our experiments we implemented two approaches for involving derived triples in subgraph construction process as part of step (3). The first approach considers papers and contributions as a main source of structured information for LLMs. Therefore, it relies on retrieving a number of papers and contributions with related titles from ORKG, excluding triples with irrelevant predicates. However, in some cases titles are not specific enough and do not contain needed keywords. To address this issue, the second approach is to directly extract triples containing resources and properties with high similarity to the question, though in some cases it requires particular triple patterns. More detailed description of the implemented approaches is provided with Figure 2, whereas the whole process of subgraph retrieval in general is presented in Fig. 3:

6. Experiments and Results

During the experiments, we evaluated, how the subgraph extraction algorithm influences the SPARQL generation quality. In the baseline version, no subgraph was provided, only an example. The results are summarized in Table 1.

According to the experimental results, even without involving subgraph extraction algorithms the proposed architecture is capable to obtain relatively decent result with 0.922 F1 score. Moreover, the table shows that the second approach to subgraph extraction leads to minor F1 score decreasing. This probably means that structured information, extracted from ORKG, predominantly contributes additional noise to the prompts. Consequently, this approach needs further revision or adaptation to the given KG. On the other hand, the first approach fosters slight F1 metric increase demonstrating its usefulness in general that requires investigation in future works.

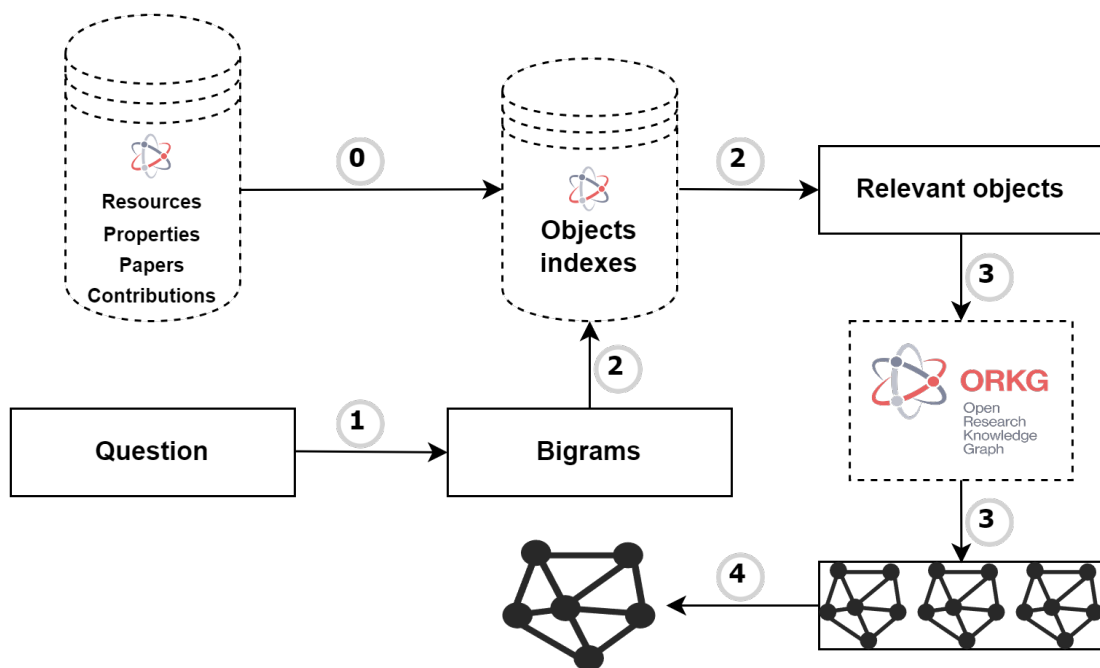


Figure 3: Subgraph extraction procedure.

Table 1
Experimental results

#	Experiment summary	F1 score
1	Instruction + Question + Example	0.922
2	Instruction + Question + Example + Subgraph Approach №1	0.935
3	Instruction + Question + Example + Subgraph Approach №2	0.916

The idea behind hyperparameters selection for the experiments on the one hand was to comply with 4096 tokens limit for GPT-3.5 and at the same time to measure the contribution changing the balance between different parameters. Table 2 indicates that the results mostly robust to variation in hyperparameters that at least partly could be caused by relatively low impact of the subgraph extraction algorithms on the system performance.

7. Error analysis

Using the optimal configuration of the subgraph extraction algorithm, all generated queries (100%) contained one or more missing prefixes, necessitating their manual addition at the beginning of each query to obtain meaningful results from the database. Two generated queries (1%) were entirely incorrect, specifically AQ1806 and AQ1787. These queries consisted solely of

Table 2
Ablation study

Subgraph approach	Hyperparameters	F1 score
№1	n_prop=100, n_paper=25, n_cont=25	0.935
	n_prop=500, n_paper=10, n_cont=10	0.929
№2	n_prop=150, n_res=25	0.916
	n_prop=50, n_res=35	0.913

lists of nodes from the graph database. This could be attributed to the lack of examples in the training set that exhibit semantic similarity to the desired questions, with a similarity score high enough to facilitate the reuse of provided examples for new query generation. Consequently, the model might have attempted to deduce the values that the target query should have returned or replicated elements of the provided subgraph.

However, upon a more detailed investigation, the evidence suggests that the training subset contains relevant examples with Levenshtein similarity scores of 0.944 and 0.8947, indicating very high similarity. The significantly high similarity and structural homogeneity were also confirmed through manual evaluation. Therefore, the model’s inability to utilize these examples and the provided subgraph to generate a correct SPARQL query should be attributed to its stochastic nature.

Furthermore, regarding other configurations of the subgraph retrieval algorithm, the following types of inconsistencies were observed:

1. Incorrect namespace prefixes (such as myontology, ex, foaf) appeared in 2 instances, possibly due to the model’s bias;
2. Instances, in which the model requested additional data using natural language for query generation, occurred once. Notably, in this case, the structure of the provided example closely resembled the format of the input question, and thus, the model should have been capable of handling this case correctly, given that it processed most of the dataset without errors. Such behavior can also be attributed to the inherent stochasticity of the model’s behavior, and an additional request should be sufficient to resolve the observed issue in most cases;
3. Hallucinated node identifiers occurred in one instance, likely due to the complexity of the provided query. In this case, the question was particularly challenging to handle without extensive reference to database content, as the node identifiers could not be straightforwardly derived from text labels. The model essentially had to guess, as the required instances arguably were absent from the provided subgraph.

For more details regarding the classification system used for categorizing errors refer to the supplementary material for SPARQLGEN paper ⁴.

As a result, the majority of the model’s errors were a consequence of its random nature, with only a negligible fraction of cases (0.25% of the total amount of handled questions) in which the model has completely failed to respond correctly. The residual error should be studied with greater attention. Evidently, in these cases, the model has failed to reproduce the query

⁴<https://github.com/danrd/sparqlgen/blob/main/errors.png>

structure, translate the necessary parameter values to corresponding query fields, or efficiently utilize the provided subgraph due to incompleteness of the train corpus, insufficient architecture complexity and other constraints.

8. Conclusion and Future Work

The presented work shares results on one-shot SPARQL query generation with large language models for the fraction of the ORKG, proposed at the SciQA challenge. The results, which were obtained, require further assessment. First of all, the performance of all participating systems, evaluated with F1-score, looks very high for such a complex domain as scholarly knowledge graphs. Our approach was ranked last with 0.935 F1-score, and the winner has the F1-score of 0.99, which is not that far for the one-shot approach without pre-training or fine-tuning. For other benchmarks (i.e. QALD series), the leaderboard is much lower⁵. This might be the result of the lack of structural diversity in the test set. Second, bringing the subgraph to the prompt doesn't result in that considerable performance increase, as it was shown for SPARQLGEN [8]. There, adding the subgraph to the prompt has led to 2.5x increase in accuracy on QALD-9, whereas in the ORKG experiments the best performed subgraph extraction algorithm has achieved only 0.013 F1-score improvement over the baseline without subgraph. The performance of the baseline, implemented as one-shot prompting, is also unexpectedly high, 0.922. For instance, a plausible explanation is the high structural similarity between the questions in the train and test sets of the SciQA challenge.

Given the ongoing research activity on enhancing LLMs with KGs, future work is quite extensive. First of all, we would like to proceed with fine-tuning open source LLMs on multiple knowledge graphs and compare the performance vs. the connecting knowledge graphs to the LLM in a retrieval fashion, as well as evaluate the generalization of the fine-tuned LLM on more datasets.

Developing robust subgraph extraction algorithms, that produce relevant knowledge snippets to augment the LLMs across different tasks, remains another perspective research direction. The requirement to have only relevant triples in the subgraph is also quite challenging to achieve with a single algorithm. Thus, some other subgraph reduction procedures can be added: i.e. instructing the LLM to prune some of the irrelevant triples⁶, using KG embeddings-based classifier to estimate the relevance of the triple, or some actor-critic approach to iteratively refine the subgraph content.

References

- [1] M. Färber, D. Lamprecht, J. Krause, L. Aung, P. Haase, Semopenalex: The scientific landscape in 26 billion rdf triples, arXiv preprint arXiv:2308.03671 (2023).
- [2] D. Dessí, F. Osborne, D. Reforgiato Recupero, D. Buscaldi, E. Motta, Cs-kg: A large-scale knowledge graph of research entities and claims in computer science, in: International Semantic Web Conference, Springer, 2022, pp. 678–696.

⁵<https://github.com/KGQA/leaderboard>

⁶This was suggested by one of the reviewers.

- [3] S. Vahdati, G. Palma, R. J. Nath, C. Lange, S. Auer, M.-E. Vidal, Unveiling scholarly communities over knowledge graphs, in: International Conference on Theory and Practice of Digital Libraries, Springer, 2018, pp. 103–115.
- [4] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, E. Grave, Atlas: Few-shot learning with retrieval augmented language models, 2022. [arXiv:2208.03299](#).
- [5] D. Emelin, D. Bonadiman, S. Alqahtani, Y. Zhang, S. Mansour, Injecting domain knowledge in language models for task-oriented dialogue systems, 2022. [arXiv:2212.08120](#).
- [6] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, 2023. [arXiv:2201.11903](#).
- [7] G. Mialon, R. Dessi, 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).
- [8] L. Kovriguina, R. Teucher, D. Radyush, D. Mouromtsev, Sparqlgen: One-shot prompt-based approach for sparql query generation (2023).
- [9] 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.
- [10] 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.
- [11] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019. URL: <http://arxiv.org/abs/1908.10084>.
- [12] Y. Malkov, D. Yashunin, Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, IEEE Transactions on Pattern Analysis and Machine Intelligence PP (2016). doi:10.1109/TPAMI.2018.2889473.