Leveraging LLMs in Scholarly Knowledge Graph Question Answering*

Tilahun Abedissa Taffa[1,*], Ricardo Usbeck[2]

[1]Semantic Systems, Universität Hamburg, Vogt-Kölln-Straße 30, 22527 Hamburg, Germany
[2]Leuphana Universität Lüneburg, Universitätsallee 1, C 4.314, 21335 Lüneburg, Germany

Abstract

This paper presents a scholarly Knowledge Graph Question Answering (KGQA) that answers bibliographic natural language questions by leveraging a large language model (LLM) in a few-shot manner. The model initially identifies the top-n similar training questions related to a given test question via a BERT-based sentence encoder and retrieves their corresponding SPARQL. Using the top-n similar question-SPARQL pairs as an example and the test question creates a prompt. Then pass the prompt to the LLM and generate a SPARQL. Finally, runs the SPARQL against the underlying KG - ORKG (Open Research KG) endpoint and returns an answer. Our system achieves an F1 score of 99.0%, on SciQA - one of the Scholarly-QALD-23 challenge benchmarks.

Keywords

Knowledge Graph Question Answering (KGQA), Open Research Knowledge Graph, Large Language Model, Scholarly KGQA, Scholarly-QALD, ORKG, SciQA

1. Introduction

Scholarly **K**nowledge **G**raph **Q**uestion **A**nswering (KGQA) models answer machine or human-generated scholarly natural language questions over KGs that contain bibliographic metadata information [1, 2]. The approaches used in the existing Scholarly KGQA models fall into two categories. The first type is a retriever-reasoner framework, which involves retrieving relevant sub-graphs and then using reasoning to extract entities as answers [3]. The second type is a semantic parsing-based framework, which focuses on transforming questions into executable logical expressions like SQL or SPARQL that can be used to obtain the answer(s) by querying the underlying KG [4]. However, both the retriever-reasoner and semantic parsing approaches need a large amount of training data to create a robust KGQA model. Specifically, the scarcity of scholarly KGQA data sets makes the task more challenging than other general KGQA. Hence, one of the possible solutions is exploring the power of Large Language Models (LLMs) in a zero or few-shot manner.

Scholarly-QALD-23: Scholarly QALD Challenge at The 22nd International Semantic Web Conference (ISWC 2023), November 6 – 10, 2023, Athens, Greece.
*Corresponding author.
tilahun.taffa@uni-hamburg.de (T. A. Taffa); ricardo.usbeck@leuphana.de (R. Usbeck)
https://www.leuphana.de/institute/iis/personen/ricardo-usbeck.html (R. Usbeck)
0000-0002-2476-8335 (T. A. Taffa); 0000-0002-0191-7211 (R. Usbeck)

LLMs are trained on a large amount of textual data for tackling human language understanding and generating tasks [5, 6]. LLMs enormous amount (counted in billions) of parameters and adaptive capability in AI (Artificial Intelligence) applications have contributed to the creation of robust QA models [7, 8, 9, 10]. Besides that, the recent advancement in prompt engineering[1], empowers everyone to get the most out of LLMs [11]. For instance, few-shot LLM prompting, the method of instructing the LLM with a minimal set of examples or context to perform a specific language generation task, generally yields more accurate and contextually relevant results [8]. Thus, by providing a few relevant question-SPARQL pairs, models like GPT-3 [5] and its successors, can generalize and generate correct SPARQL queries to query encyclopedic KGs like Wikidata. For example, as shown in Figure 1 (left), ChatGPT 3.5[2] generates a correct query for the question "What is the capital city of Ethiopia?" in zero-shot mode. Unlike that, for the scholarly question taken from SciQA test set "What are the models that have been benchmarked on the BoolQ dataset?", even though the SPARQL generated in zero-shot has no syntactic errors (see Figure 1 right), it does not yield the right answer when run against the ORKG-dump SPARQL endpoint. This is because ChatGPT does not know the schema of ORKG. One way of addressing this knowledge gap in LLMs is using a few-shot approach.

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?capitalLabel
WHERE
{
  wd:Q115 wdt:P36 ?capital.
  ?capital rdfs:label ?capitalLabel.
  FILTER(LANG(?capitalLabel) = "en")
}
```

ChatGPT generated SPARQL for "What is the capital city of Ethiopia?

```
PREFIX orkg: <https://www.orkg.org/orkg/resource/>
PREFIX dbo: <https://www.orkg.org/orkg/resource/>

SELECT ?model
WHERE {
  ?model orkg:isBenchmarkedBy ?benchmark.
  ?benchmark orkg:isBenchmarkOf ?dataset.
  ?dataset orkg:datasetName "BoolQ".
}
```

ChatGPT generated SPARQL for "What are the models that have been benchmarked on the BoolQ dataset?"

Figure 1: SPARQL queries generated by ChatGPT 3.5 in a zero-shot manner for querying Wikidata (left) and ORKG (right).

Therefore, in this work, we harness the capabilities of LLMs to transform natural language questions into SPARQL queries. We believe that successfully addressing the SciQA challenge at the Scholarly QALD Challenge[3] underway at ISWC 2023[4] allows us to contribute to enhancing access to and utilization of scholarly knowledge.

The contributions of our work are:

- Leveraging LLMs for SPARQL query generation in a few-shot manner;
- Identifying similar questions using a BERT-based model [12];
- Developing a Scholarly KGQA model that ranked second in the SciQA Challenge leaderboard;

[1]The process of designing prompts that help LLMs to perform a task
[2]https://openai.com/blog/chatgpt
[3]https://kgqa.github.io/scholarly-QALD-challenge/2023/
[4]https://iswc2023.semanticweb.org

- Evaluating the impact of single shot and few-shot prompting for SPARQL generation performance.

Our source code can be found at https://github.com/huntila/scholarly-kgqa.

2. Related Works

2.1. Scholarly KGQA

Pipeline-based, semantic parsing-based scholarly KGQA systems first identify the entities and relationships in the given question; and map those entities and relationships to their respective identifier in the KG. Next, formulate a query, e.g. SPARQL, and finally execute the query against the underlying KG and return an answer [13]. JarvisQA [2], create triples from the tables, then convert the triples to text, and extract an answer using a Bidirectional Encoder Representations from Transformers (BERT) [14] based answer retriever. JarvisQA only operates on tabular data. Besides, the performance of the model is highly dependent on the correctness of the transformation of the table entries to triples and triples to text. Instead of using triples to text transformer and retriever, DBLP-QuAD [4], parse questions to a SPARQL by fine-tuning a Text-to-Text Transfer Transformer (T5) [15] model. To use the DBLP-QuAD parser for a new scholarly KG with a different schema, fine-tuning requires a large amount of training data and an entity linker. Unlike JarvisQA, our model translates questions into SPARQL without the need for triple-to-text conversion. Additionally, our approach differs from DBLP-QuAD in that it employs an LLM to generate SPARQL with very few examples, which avoids the LLM pre-training.

2.2. Few-Shot LLM Prompting

In this work, we use Vicuna-13B[5] - an open-source LLM, a descendent of LLaMA [6] fine-tuned using user-shared conversations collected from ShareGPT[6]. LLMs have demonstrated their remarkable ability to understand and generate natural language text in zero-shot and few-shot manner [5, 16, 8]. In few-shot prompting, the LLM is given the task description in natural language such as 'generate SPARQL for the given questions' with few examples, then prompted to accomplish the task without any fine-tuning [8, 11]. So, in our work, we use few-shot prompting for question's SPARQL generation.

Table 1
SciQA dataset size.

Data	train	dev	test
Size	1795	200	200

[5]https://lmsys.org/blog/2023-03-30-vicuna/
[6]https://sharegpt.com

3. The Scholarly QALD Challenge

To foster standard evaluation of KGQA models, there have been a series of QALD (Question Answering over Linked Data) challenges since 2011 [17]. The datasets released in the past QALD challenges are based on generic KGs like Wikidata. Unlike that, the Scholarly QALD Challenge organized at the ISWC 2023 comes up with two new Scholarly QA data sets, namely SciQA (Scientific QA) [18] and DBLP-QUAD [4]; and provides Codalab [19] as a competition platform.

The SciQA benchmark data set - the challenge we participated in - is created following manual and template-based automatic generation methods. That is, first, 100 questions are created manually, afterward, from the manual questions curated eight questions and query templates. The manually created questions and queries underwent rigorous peer review by the authors and domain experts for correctness and relevance. Then, generate an additional three question and query templates from the eight question and query templates using GPT-3 [5]. Finally, 2465 questions are auto-generated by replacing entities and relations in the templates. All questions focus on Computer Science research works [18]. Table 1 shows the train, dev, and test question split size of the SciQA dataset[7].

4. The Scholarly KGQA Model

As shown in Figure 2 for a given question Q, our scholarly KGQA model encodes the training questions and Q. Then, it identifies a set of similar questions to Q from the training set and constructs a prompt. Subsequently, the system generates Q's SPARQL query by prompting the LLM and finally provides an answer A by running the SPARQL against the ORKG SPARQL endpoint[8]. In the following, we explain in detail the three phases: question analysis, SPARQL generation, and answer extraction.

4.1. Question Analysis

This phase aims to identify the top-n questions from the training set similar to test question Q and to fetch their respective SPARQL. As a result, the question and SPARQL pairs are used in the prompt formulation of the query generation. Therefore, the question analyzer first generates the question embedding score of each question in the training set offline using the BERT-based sentence encoder [12]. Besides, encodes the input test question using the same sentence encoder. Then compute the similarity score based on cosine similarity for each test question Q, rank the training questions based on their similarity score with Q, and select the top 5 questions along with their SPARQL.

4.2. Query Generation

This component addresses the challenge of extracting the correct entities and relationships from Q and mapping them to the correct SPARQL query. Therefore, the query generation component

[7]https://github.com/debayan/scholarly-QALD-challenge/tree/main/2023/datasets/sciqa/SciQA-dataset
[8]https://ltdemos.informatik.uni-hamburg.de/orkg/sparql

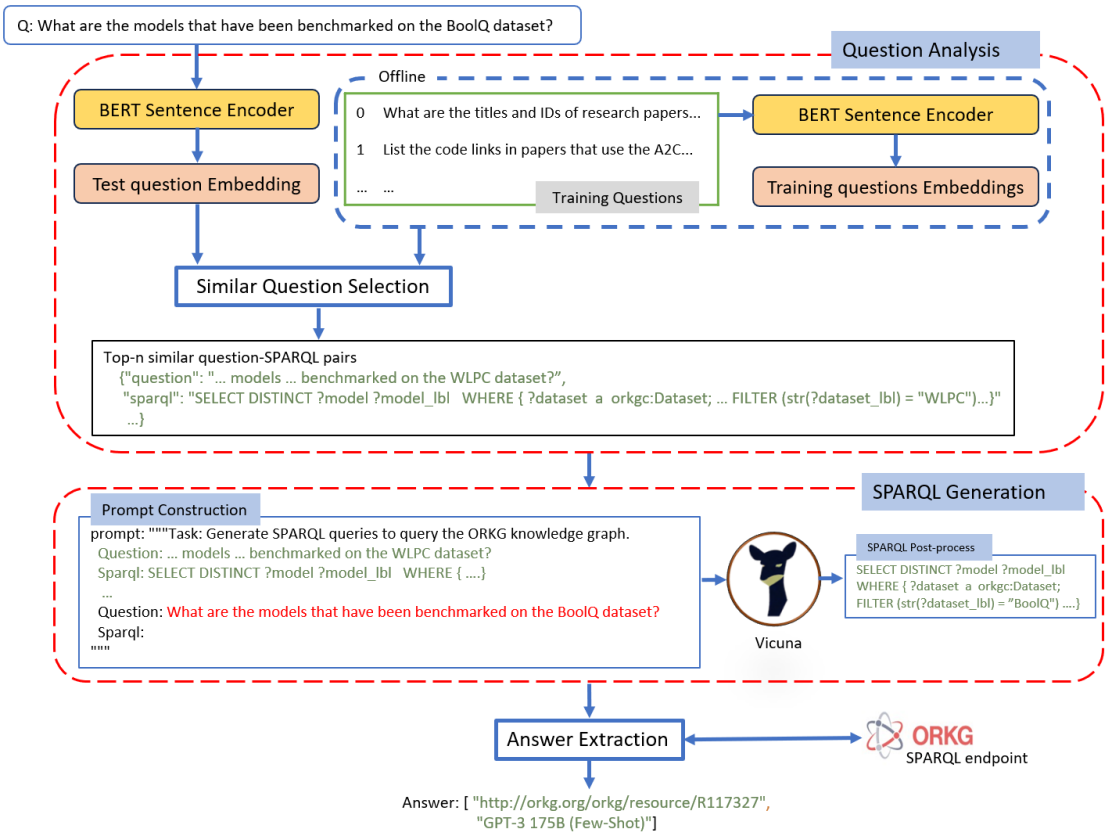

Figure 2: The Scholarly KGQA model.

constructs a prompt using the prompt template shown 4.2. In the prompt, an example is created by concatenating top-n (n=1,3,5) similar questions with their respective SPARQL queries. In the case of one shot, the example variable contains only one top-ranked question SPARQL pair. Whereas, in the three or five shot the example variable contains three or five top-ranked questions SPARQL pairs respectively. Before including the SPARQL queries in the prompt, the query generator removes special characters such as new lines, escaping characters, and extra blank spaces.

```
example = "Question: {sim_question} \n Sparql: {sparql}..."
prompt: """ Task: Generate SPARQL queries to query the ORKG.
Instruction: If you cannot generate a SPARQL query based
             on the provided examples, explain the reason.
             {example}
Question: {test question}
Sparql:
Note:  Output only the SPARQL query. """
```

Then the SPARQL generator sub-component runs the prompt against our own Vicuna instance[9]; and the output of the LLM is returned as the SPARQL of the test question *Q*.

4.3. Answer Extraction

The answer extractor receives the generated SPARQL queries and cleans new lines, escaping characters, and extra blank spaces, again. Finally runs the query against the underlying KG, in our case the ORKG endpoint[10], and returns the result as an answer.

5. Evaluation

5.1. Results and Discussion

The intuitive idea behind our model design is that we can get the best out of the LLM by providing prompts that contain similar question-SPARQL pairs and a test question. This makes the LLM self-learn and generates correct SPARQL based on examples. Hence, as shown in Table 2 and the Codalab SciQA Challenge leaderboard[11] our scholarly KGQA model achieved a near-perfect F1 score of 0.99 using the top-3 similar questions. Queries generated using one-shot recorded the lowest F1 score of 0.96. In the one-shot setting, the LLM receives only the top-most similar question SPARQL pair example. When using the top-5 similar questions, the F1 score reaches 0.989. The top-5 performance of our model is lower than in the top-3 model, because, as the number of question-SPARQL examples increases, it is likely that the probability of including training questions with less similarity to the test question. Thus, the inclusion of dissimilar question-SPARQL pairs confuses the LLM and generates queries that do not give the correct answer.

Table 2
F1 score of our model on SPARQLs generated using one, three, and five-shot methods on dev and test data.

Data	Shot	F1 Score
test	One-shot	0.960
	Three-shot	**0.990**
	Five-shot	0.989
dev	One-shot	0.899
	Three-shot	**0.974**
	Five-shot	0.973

Apart from that, the performance of our model is almost near to 1. The contributing factors are bifold. First, the test questions do not contain additional questions that are generated by those templates used to generate the training set. Thus, the LLM easily memorizes the entities and relations in the data set. Besides, the SPARQL queries in the data set use the literal indicating

[9] https://lmsys.org/blog/2023-03-30-vicuna/
[10] https://ltdemos.informatik.uni-hamburg.de/orkg/sparql
[11] https://codalab.lisn.upsaclay.fr/competitions/public_submissions/14759

Table 3
Number of null gold answers in dev; and null system answers in three and five-shot answer sets on dev and test data.

	data	answer set	total	due to syntax error out of total
Null answers	dev	gold	14	1
		three-shot	23	7
		five-shot	25	7
		both in dev and three-shot	14	1
		both in dev and five-shot	14	1
	test	three-shot	27	3
		five-shot	28	4

prefixes *orkgc*, *orkgp*, and *orkgsh* for classes, predicates, and shapes respectively. Hence, the LLM does not need to resolve and remember the URI (Universal Resource Identifier) of the entities and predicates, rather simply learns from the example questions and generates correct SPARQL queries. Second, the performance is biased due to the number of empty answer sets. For instance, as Table 3 depicts, the number of null gold answers in dev is 14. The number of null system answers on the dev data via the three-shot and five-shot methods is 23 and 25 respectively. All those fourteen questions with null gold answers in dev, also have null system answers in both settings. Among all the fourteen questions that have null values, only one is from the same question in both methods due to syntax error. Furthermore, as the bottom part of Table 3 shows, the number of null system answers on the test data is 27 with three-shot and 28 with the five-shot methods. However, the test data gold answer set is not publicly available as of this writing. Thus, we are unable to run a full analysis. In conclusion, removing the questions that have null answer sets from the dev and test, can reveal the gap between our model and others that participated in the challenge.

5.2. Error Analysis

Since the gold answer of the final phase is not available, our error analysis is based on the dev set experiment. Generally, the errors are 1) syntactic errors: which occur due to missing and improper placement of *closing bracket ('}')*, *period ('.')*, and *semicolon (';')*. As shown in Table 3, on dev data both three-shot and five-shot methods generate 7 SPARQLs that have null results due to syntax error. Besides, on test data, the three-shot method generates 3 SPARQLs that have null answers due to syntactic errors. Likewise, the five-shot method has four SPARQL that are syntactically incorrect. 2) keyword matching: the entities identified by the LLM have extra or missing blank space(s). For example, the filter in "*SELECT ... FILTER (str(?dataset_lbl) = 'Jacquard dataset')...*" has a space at the beginning of the keyword 'Jacquard dataset' in the gold SPARQL, but the LLM-generated SPARQL query does not have space. Thus, our model's SPARQL query results in a null answer. 3) Lack of question understanding: for complex questions like 'Where can all the data sets used in the compared studies be found?', the LLM assumes that the question is about a dataset and generates "*SELECT DISTINCT ?dataset ?dataset_lbl WHERE ...*" which looks for a dataset. However, the question is about the URI where the dataset is stored. Moreover,

on questions like 'What is the top benchmark score and its metric on the Words in Context dataset?', the LLM creates queries that look for the model and its name "*SELECT DISTINCT ?model ?model_lbl WHERE ...*". Here the LLM misses the aim of the question, i.e., the correct answer is the top benchmark score and the respective query should look like "*SELECT DISTINCT ?metric ?metric_lbl (MAX(?value) AS ?score) WHERE...*". Therefore, the misunderstanding of a question leads the LLM to produce SPARQLs that look for incorrect objects and miss operators like *MAX()* in the recent example.

6. Summary

Our Scholarly KGQA system to the Scholarly-QALD-23 challenge at the ISWC 2023, follows a pipeline structure. For a given test question, the question analyzer identifies similar questions using a BERT-sentence encoder. Then, the SPARQL generator creates prompts by composing the top five (three or one) similar question-SPARQL pairs from the training set with the test question and generates a SPARQL by prompting Vicuna. Finally, the answer generator runs the query against the ORKG SPARQL endpoint and returns an answer. Our system achieves an F1-score of 99.0% on the SciQA test set, which is a runner-up of the SciQA leaderboard[12].

Acknowledgments

This work has been partially supported by grants for the DFG project NFDI4DataScience project (DFG project no. 460234259) and by the Federal Ministry for Economics and Climate Action in the project CoyPu (project number 01MK21007G).

References

[1] S. Auer, A. Oelen, M. Haris, M. Stocker, J. D'Souza, K. E. Farfar, L. Vogt, M. Prinz, V. Wiens, M. Y. Jaradeh, Improving access to scientific literature with knowledge graphs, Bibliothek Forschung und Praxis 44 (2020) 516–529. URL: https://www.degruyter.com/document/doi/10.1515/bfp-2020-2042/html.

[2] M. Y. Jaradeh, M. Stocker, S. Auer, Question Answering on Scholarly Knowledge Graphs, in: International Conference on Theory and Practice of Digital Libraries, Springer, 2020, pp. 19–32. URL: https://link.springer.com/chapter/10.1007/978-3-030-54956-5_2.

[3] H. Wang, L. Zhou, W. Zhang, X. Wang, LiteratureQA: A Question Answering Corpus with Graph Knowledge on Academic Literature, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 4623–4632. URL: https://dl.acm.org/doi/10.1145/3459637.3482007.

[4] D. Banerjee, S. Awale, R. Usbeck, C. Biemann, DBLP-QuAD: A Question Answering Dataset over the DBLP Scholarly Knowledge Graph, arXiv preprint arXiv:2303.13351 (2023). URL: http://arxiv.org/abs/2303.13351.

[12] https://kgqa.github.io/scholarly-QALD-challenge/2023/

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[6] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., LLaMA: Open and efficient foundation language models (2023). URL: arXivpreprintarXiv:2302.13971.

[7] D. Banerjee, P. A. Nair, R. Usbeck, C. Biemann, GETT-QA: Graph Embedding Based T2T Transformer for Knowledge Graph Question Answering, in: European Semantic Web Conference, Springer, 2023, pp. 279–297. URL: https://link.springer.com/content/pdf/10.1007/978-3-031-33455-9_17.pdf.

[8] W. Chen, Large Language Models are few(1)-shot Table Reasoners, in: Findings of the Association for Computational Linguistics: EACL 2023, Association for Computational Linguistics, Dubrovnik, Croatia, 2023, pp. 1120–1130. URL: https://aclanthology.org/2023.findings-eacl.83. doi:10.18653/v1/2023.findings-eacl.83.

[9] E. Kamalloo, N. Dziri, C. Clarke, D. Rafiei, Evaluating Open-Domain Question Answering in the Era of Large Language Models, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 5591–5606. URL: https://aclanthology.org/2023.acl-long.307. doi:10.18653/v1/2023.acl-long.307.

[10] N. Ziems, W. Yu, Z. Zhang, M. Jiang, Large Language Models are Built-in Autoregressive Search Engines, in: Findings of the Association for Computational Linguistics: ACL 2023, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 2666–2678. URL: https://aclanthology.org/2023.findings-acl.167. doi:10.18653/v1/2023.findings-acl.167.

[11] T. Sorensen, J. Robinson, C. Rytting, A. Shaw, K. Rogers, A. Delorey, M. Khalil, N. Fulda, D. Wingate, An Information-theoretic Approach to Prompt Engineering Without Ground Truth Labels, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 819–862. URL: https://aclanthology.org/2022.acl-long.60. doi:10.18653/v1/2022.acl-long.60.

[12] V. Kocaman, D. Talby, Spark NLP: Natural language understanding at scale, Software Impacts (2021) 100058. URL: https://www.sciencedirect.com/science/article/pii/S2665963821000063. doi:https://doi.org/10.1016/j.simpa.2021.100058.

[13] L. Zhang, J. Zhang, X. Ke, H. Li, X. Huang, Z. Shao, S. Cao, X. Lv, A survey on complex factual question answering, AI Open 4 (2023) 1–12. URL: https://www.sciencedirect.com/science/article/pii/S2666651022000249. doi:https://doi.org/10.1016/j.aiopen.2022.12.003.

[14] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: J. Burstein, C. Doran, T. Solorio (Eds.),

Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics, 2019, pp. 4171–4186. URL: https://doi.org/10.18653/v1/n19-1423. doi:10.18653/v1/n19-1423.

[15] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, J. Mach. Learn. Res. 21 (2020). URL: https://jmlr.org/papers/volume21/20-074/20-074.pdf.

[16] J. Baek, A. F. Aji, A. Saffari, Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering, in: Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 78–106. URL: https://aclanthology.org/2023.nlrse-1.7. doi:10.18653/v1/2023.nlrse-1.7.

[17] R. Usbeck, X. Yan, A. Perevalov, L. Jiang, J. Schulz, A. Kraft, C. Möller, J. Huang, J. Reineke, A.-C. N. Ngomo, et al., QALD-10—The 10th Challenge on Question Answering over Linked Data, Semantic Web (2023). URL: https://www.semantic-web-journal.net/system/files/swj3471.pdf.

[18] S. Auer, D. A. Barone, C. Bartz, E. G. Cortes, M. Y. Jaradeh, O. Karras, M. Koubarakis, D. Mouromtsev, D. Pliukhin, D. Radyush, et al., The SciQA Scientific Question Answering Benchmark for Scholarly Knowledge, Scientific Reports 13 (2023) 7240. URL: https://www.nature.com/articles/s41598-023-33607-z.

[19] A. Pavao, I. Guyon, A.-C. Letournel, D.-T. Tran, X. Baro, H. J. Escalante, S. Escalera, T. Thomas, Z. Xu, CodaLab Competitions: An Open Source Platform to Organize Scientific Challenges, Journal of Machine Learning Research 24 (2023) 1–6. URL: http://jmlr.org/papers/v24/21-1436.html.