Large Language Models for Scientific Question Answering: an Extensive Analysis of the SciQA Benchmark

Jens Lehmann[1,2⋆][0000−0001−9108−4278], Antonello Meloni[3][0000−0001−6768−4599], Enrico Motta[4][0000−0003−0015−1952], Francesco Osborne[4,5][0000−0001−6557−3131], Diego Reforgiato Recupero[3][0000−0001−8646−6183], Angelo Antonio Salatino[4][0000−0002−4763−3943], and Sahar Vahdati[1][0000−0002−7171−169X]

[1] ScaDS.AI - TU Dresden, DE
[2] Amazon, DE
[3] Department of Mathematics and Computer Science, University of Cagliari, IT
[4] Knowledge Media Institute, The Open University, UK
[5] Department of Business and Law, University of Milano-Bicocca, IT

Abstract. The SciQA benchmark for scientific question answering aims to represent a challenging task for next-generation question-answering systems on which vanilla large language models fail. In this article, we provide an analysis of the performance of language models on this benchmark including prompting and fine-tuning techniques to adapt them to the SciQA task. We show that both fine-tuning and prompting techniques with intelligent few-shot selection allow us to obtain excellent results on the SciQA benchmark. We discuss the valuable lessons and common error categories, and outline their implications on how to optimise large language models for question answering over knowledge graphs.

Keywords: Knowledge graphs · Question answering · Language models. · Fine-tuning · Few-shot learning

1 Introduction

Knowledge graphs have gained popularity over the last decades as fact storage systems and, more recently, as retrieval backends for large language models (LLMs) [18,30]. Most knowledge graphs are currently relatively simple semantic structures and question-answering (QA) benchmarks have been designed mostly for such structures and encyclopedic knowledge graphs such as DBpedia [25] and Wikidata [39]. The SciQA ORKG benchmark [2] focuses on a new direction by leveraging the Open Research Knowledge Graph[6](ORKG) [36], which models scholarly artifacts and is meant to pose a challenge for next-generation QA systems. Auer et al. [2] underline this by pointing out that the performance

⋆ Corresponding author.
[6] Open Research Knowledge Graph - `https://orkg.org`

of knowledge graph question answering (KGQA) systems as well as ChatGPT (GPT-3.5 backend) on the benchmark is rather low.

The main goal of this paper is to dive deeper into analysing the LLM performance on the SciQA. In particular, we studied the accuracy of different language models on the SciQA ORKG benchmark and whether simple training or prompting approaches are already sufficient to address the challenges or whether more sophisticated techniques are required. The most common approaches to improving the performance of a language model on a given task are appropriate prompting and fine-tuning. We investigated the impact of both strategies on different language models. If a language model can be prompted to solve the task, then this indicates that the model itself has the inherent capability to solve the problem but requires further context to direct its next-token prediction optimisation towards solving the problem [41,14]. We were, therefore, particularly interested in this direction and analysed the impact of zero- and few-shot learning techniques on SciQA performance. For few-shot learning, we investigated different example selection strategies leveraging semantic similarity [28], diversity [26], and entropy criteria [19]. To ensure reproducibility, we released a repository with the codebase and the prompts used in the experiments[7].

The result of this study gives insights into how to optimise language models for SciQA, and more in general, for scientific question answering. Our findings demonstrate that different combinations of LLMs, both fine-tuned and utilizing few-shot learning, achieve remarkable outcomes on the benchmark. Notably, the best-performing model was the fine-tuned version of T5-base, a relatively small language model. The fine-tuned version of GPT-2-large also performed well. This suggests that low-resource models, when meticulously fine-tuned, can match or even surpass the performance of larger, more resource-intensive models in complex tasks like scientific question answering. GPT-3.5 also produced excellent results using a few-shot approach based on the semantic similarity between the question and the samples in the training set.

In summary, the contributions of this paper are the following:

1. Performance analysis of four language models for SciQA using three methodologies (zero-shot learning, few-shot learning, and fine-tuning).
2. Analysis of seven alternative approaches to select samples for few-shot learning for LLM-based KGQA.
3. Provision of several insights which indicate that optimising LLMs with appropriate prompting and fine-tuning techniques is sufficient to obtain excellent results (>94.1% exact query match accuracy and >97.5% F1-score).

The remainder of this paper is structured as follows. Section 2 discusses related works on knowledge graph question answering and LLM few-shot optimizations. The SciQA benchmark and the LLM models are described in Section 3. Section 4 outlines the experiments and the metrics used for evaluation. Section 5 reports the results. Section 6 provides an analysis of the types of errors made by the models while Section 7 discusses the limitations of our study. Finally, Section 8 ends the paper with concluding remarks and future directions.

[7] Codebase and prompts - `https://github.com/NIMI-research/SciQA-LLM`

2 Related Work

Semantic Parsing for Knowledge Graph Question Answering. Semantic parsing [21] is the process of converting an utterance in natural language into a query with a format that machines can understand and by using a specific formal language. This conversion is considered successful when the query accurately represents the original intent of the natural language input. Various strategies have been proposed for semantic parsing, encompassing Combinatory Categorial Grammars (CCG), rule-based systems, and neural-network-based methods [12,34,3,35]. The work introduced in [12] addresses semantic parsing using neural network methods for knowledge graphs. Recent methods of semantic parsing for KGQA often utilize sequence-to-sequence models, such as those described by [34]. In these models, an encoder transforms a natural language utterance into an intermediate representation. Subsequently, a decoder processes this representation to generate a logical form.

A type of neural network architecture that combines the features of both sequence-to-sequence models and pointer networks is called pointer generator networks [3]. These networks enable the replication of specific input segments, such as data values, in the output. Although these methods show high accuracy when trained with large amounts of high-quality data, the difficulties in obtaining such data and the significant effort needed for updates lead to an increased interest in training semantic parsers with fewer datasets. The work in [35] explores an unsupervised approach for semantic parsing which shows promising results. While promising, these methods have scalability issues for real scenarios. Additionally, they present challenges in terms of easy adjustability or tunability for practical applications. LLMs such as GPT-2 [32], GPT-3.5 [9], and Dolly [15], which builds upon the Pythia models [6] show potential in semantic parsing with zero or minimal examples, especially following improvements to mitigate inference costs. However, LLMs do not inherently support KG-specific parsing without access to entities in the underlying knowledge graph. Consequently, they require augmentation with a method for entity retrieval. A work that exploited LLMs for KGQA semantic parsing is described in [24], where authors claimed that training data requirements are substantially reduced when using controlled natural languages as targets for KGQA. Some conversational agents also use language models to map questions to a set of predefined templates that facilitate the generation of queries [29].

The developers of the SciQA benchmark hosted the Scholarly QALD Challenge as a satellite event of the 22nd International Semantic Web Conference [4]. This resulted in the development of some dedicated methods for SciQA [20,37,31], which usually involve additional components for incorporating a representation of the ORKG ontological schema or detecting domain-specific entities. In contrast, our study seeks to evaluate the performance of conventional LLMs when augmented with fine-tuning and prompting techniques.

LLM Optimisation. In this paper, we focus on two standard optimization techniques for LLMs: few-shot learning and fine-tuning. Few-shot optimization in LLMs is a technique where the models are given one or more samples when

performing a certain task. This contrasts with the zero-shot approach, where LLMs function without any training data [22]. There are several ways of optimizing the examples for few-shot learning. Kumagai et al. [23] proposed an approach that employs a feature selector and a decoder: the selector uses concrete random variables to choose relevant features for the decoder to reconstruct the original features of unseen instances. The sampling method in Bansal et al. [5] enables optimization-based meta-learning across diverse tasks with varying numbers of classes. Levy et al. [26] study how to select a diverse set of samples in in-context learning. Similarly, Lin et al. [27] studied the few-shot learning capabilities of LLMs across a broad spectrum of tasks with a diverse selection of examples. The recent Cross Entropy Difference method [19] also demonstrated promising results, especially in semantic parsing. Another optimization method is fine-tuning, a technique where a pre-trained LLM is further trained on a specific task or dataset to improve its performance for that particular domain [40]. Fine-tuning is particularly valuable when the labeled data for the target task are limited, as it maximises the utility of pre-existing knowledge in the model [17].

3 Background

This section describes the SciQA benchmark and introduces the four models employed in our experiments.

3.1 Dataset

We adopted as a benchmark the recent SciQA dataset[8]. SciQA includes 2,565 pairs, each composed of a natural language question and the equivalent SPARQL query. The SPARQL queries are specifically designed to extract information from the Open Research Knowledge Graph (ORKG) [36]. This KG presently consists of 170,000 resources that detail the research contributions of nearly 15,000 scholarly articles spanning 709 research fields.

SciQA is a mix of manually crafted and automatically generated pairs of questions and queries. Specifically, 100 pairs have been manually crafted by a team of researchers from different countries and institutions. First, they selected a number of research fields and relevant studies on ORKG to narrow down the data scope. Next, they formulated natural language questions of various types, such as single comparison, True/False, and aggregation questions. Each question was then associated with a corresponding SPARQL query and relevant metadata (e.g., type, query shape). The entire process underwent multiple peer reviews. The authors sought diverse perspectives and consulted 21 domain experts involved in ORKG curation grants to ensure the relevance and importance of the questions. Next, from these initial 100 pairs, the authors identified eight query templates and used them to automatically produce an additional 2,465 pairs using the GPT-3 model [9]. Specifically, they defined a set of eight templates

[8] SciQA dataset - `https://huggingface.co/datasets/orkg/SciQA`

and 32 questions, with one manual question and three GPT-3-generated variations for each template. The query templates were then filled with all possible entities from ORKG and the resulting queries were randomly assigned to one of the four questions. The results of these queries were collected, and metadata for the final question set was extracted. The addition of automatically generated questions expanded the SciQA dataset to 2,565 questions. We report an example of a question and query pair in Example 1.

The benchmark divides the dataset into three parts: 70% for training (1795 samples), 10% for validation (257 samples), and 20% for testing (513 samples). Table 1 reports relevant statistics for each set. The unbalanced distribution of the template frequency across the datasets is noteworthy. Further details about the SciQA dataset and its construction are available in [2].

Table 1. For the training, validation, and test splits of the SciQA dataset, we show from the left to the right, respectively, the number of human-generated questions, the number of automatically generated questions, the percentage of human-generated questions (H), and the percentage of automatically generated questions for each template T_i.

	#Hum. Gen.	**#Aut. Gen.**	**H**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**
Training	66	1729	3.7%	13.5%	13.3%	12.9%	13.3%	18.1%	2.7%	22.3%	0.3%
Validation	13	244	5.1%	14.0%	15.6%	14.8%	9.7%	16.3%	3.5%	21.0%	0%
Test	21	492	4.1%	12.5%	12.7%	14.2%	15.4%	18.5%	2.9%	19.5%	0.2%

3.2 Approaches

We employed four LLMs in our experiments: i) T5-base [33], ii) GPT-2-large [32], iii) Dolly-v2-3b [15], and iv) GPT-3.5 Turbo [9]. These models are all based on the transformer architecture, which leverages the attention mechanism to learn how words and sentences relate to each other [38]. We chose these models due to their significant variability in dimension (220M to 175B parameters) and structure (both encoder-decoder and decoder-only), enabling us to evaluate various options on the benchmark. In the following, we will briefly illustrate their main characteristics.

The **T5-base** model features an encoder-decoder structure with 220 million parameters and a context capacity of 512 tokens. It was trained on the C4 dataset, a vast collection of text totaling around 750 GB of web pages. The model is trained using a mask-filling task, where 15% of the tokens are randomly masked. The **GPT-2-large** model uses a decoder-only architecture with 774 million parameters. It is capable of handling a maximum of 1024 tokens encompassing both input and output. It was trained on the WebText dataset, which includes approximately 40 GB of text sourced from 45 million web pages.

The **Dolly-v2-3b** is a decoder architecture with 2.8 billion parameters and a token context capacity of 2048. This model was developed by Databricks through the fine-tuning of EleutherAI's Pythia-12b[9]. Specifically, Dolly-v2-3b underwent fine-tuning using a dataset comprising approximately 15,000 instruction-response pairs. Finally, the **GPT-3.5 Turbo** model features a decoder-only architecture with 175 billion parameters and supports a context of up to 4,096 tokens. This model builds upon the architecture and pre-training data of GPT-3, having been trained on roughly 45 terabytes of text from diverse sources, including books, websites, social media, and news articles. GPT-3.5 Turbo underwent further fine-tuning with data and tasks pertinent to chat applications, covering areas like dialogues, question-and-answer sessions, command-response interactions, and code generation and execution.

4 Methodology

The primary objective of the study presented in this paper was to evaluate the performance of various types of LLMs on the Sci-QA datasets when applying common optimization techniques, such as few-shot learning and fine-tuning. To achieve this, we methodically conducted experiments using 7 distinct few-shot learning approaches as well as fine-tuning. We omitted certain experiments that were either unfeasible due to the context window limitations of some language models or too demanding computationally, specifically the fine-tuning of GPT-3.5 and Dolly. We also conducted an error analysis (reported in Section 6) to determine the strengths and weaknesses of the leading models. Contrary to other studies [20,37,31], we did not apply any specialized adjustments to improve the performance of these methods for this specific dataset. This section details the experiments, along with the metrics and setup utilised in our study.

4.1 Experiments

We focused on three approaches for optimizing the LLMs: fine-tuning (FT), zero-shot learning (ZSL), and few-shot learning (FSL). For the FT, we fine-tuned the model on the training set before applying it to the test set. For the ZSL, the test set was processed directly using a basic prompt without any examples. Specifically, we tried for each model different types of prompts, ultimately selecting the most effective prompt for each model based on performance. Finally, for the FSL, the prompt was enhanced by providing a number S of samples extracted from the training set. Given a question of the test set, we evaluated seven methods from the literature to select the most relevant samples for each question:

- **1) Random.** S samples are randomly chosen for each test set element.
- **2) Similarity** [28]. The samples are ranked in descending order based on their semantic similarity to the question associated with the corresponding test set element, and the top S samples are chosen.

[9] Pythia-12b - `https://huggingface.co/EleutherAI/pythia-12b`

– **Diversity of template** [26].
  - **3)** *Test A* (All diverse templates): The samples are arranged in a descending sequence according to their semantic similarity to the question. The top S samples, using all different templates, are selected.
  - **4)** *Test B* (Same template for all): The samples are arranged in a descending sequence according to their semantic similarity to the question. Subsequently, the top S samples that utilize the same template as the first sample are selected.
– **Entropy** [19]. We implemented three different techniques derived from [19]. For each template, we first select the 8 shortest (in terms of string size) samples of the training set and then:
  - **5)** *Same_Templ*: We select the most similar question in the training set and identify its template. Next, we select the sample using the same template from the set of 8 samples;
  - **6)** *Ran_Templ*: We select a random sample from the set of 8 samples;
  - **7)** *Low_Perp*: We generate the SPARQL query for each of the 8 input templates and then compute the perplexity for each of them. We then select the sample which produced the lowest perplexity.

In the following, we will illustrate the experiments conducted on the four models: T5-base, GPT-2, Dolly-v2-3b, and GPT-3.5-Turbo.

T5-base model. The T5-base model underwent only fine-tuning. ZSL and FSL were not applicable due to the limited size of the prompt for this model and its pre-training on specific tasks, such as translating from English to German or summarizing text. Consequently, when prompted for English to SPARQL translation tasks, the model tended to provide a German translation of the question instead. In the FT process, the T5 model was assigned the task of translating a specified English natural language question into a SPARQL query. The prefix used for this purpose was *"Translate English to SPARQL:"*.

The model was fine-tuned with 1,795 labeled samples from the training set over 20 epochs. Each sample consisted of a short prefix and a natural language question, serving as the input request to the model, and was paired with the corresponding SPARQL query, which formed the expected response. Example 4 illustrates the type of data utilized in the fine-tuning process of the T5-base model.

GPT-2-large model. The primary limitation of the GPT-2 model was its restricted context size, which constrained the use of lengthy prompts and multiple samples in few-shot learning (FSL). Despite this limitation, we successfully conducted the full suite of experiments. Specifically, we performed the following experiments:

– ZSL, using the prompt: *"input (English text): [nl question] \n output (Sparql query):"*.

- FSL with 1, 2, and 3 samples, due to the 1024 token context restriction, which did not permit the inclusion of more samples. In the following, we report an example using 1-short learning:
  "input (English text): [the sample question in natural language]"
  "output (Sparql query): [the sample SPARQL query]"
  "input (English text): [the question in natural language of the underlying test set sample]"
  "output (SPARQL query):"
  We tried all the methods described in the previous section (similarity, diversity, and entropy) to select relevant samples.
- FT: The GPT-2-large model was fine-tuned using a text file containing all question and query pairs from the training set. The special token $< |endoftext| >$ was inserted before each pair, serving as a signal for the model to cease output generation. The fine-tuning process was carried out over 20 epochs.

Dolly-v2-3b model. Due to the substantial dimensions of the Dolly model, fine-tuning posed significant challenges. Consequently, we focused on ZSL and FSL. Specifically, we performed the following experiments:

- ZSL, using the prompt: *"Translate to a SPARQL query the following English question:"*.
- FSL: We utilized 1, 3, 5, and 7 examples for few-shot learning. The model's token limit of 2,048 restricted our input to a maximum of seven samples. We employed the same prompt format as used with the GPT-2 model. Similar to the previous approach, we applied the full range of previously discussed metrics to select various samples for testing.

GPT-3.5-Turbo. Similarly to Dolly, fine-tuning was potentially resource-intensive due to the model's large size. Therefore, we concentrated on ZSL and FSL instead. In summary, we performed the following experiments:

- ZSL, using the prompt: *"What is the SPARQL query for:"*.
- FSL: To ensure comparability with the other models, we conducted experiments using 1, 3, 5, and 7 samples. We applied the complete set of methodologies for sample selection, with one exception. We did not test the Entropy - *Low_Perp* methodology because it was not possible to compute the perplexity with OpenAI API. The samples were provided using the following template:
  "input (English text): [the sample question in natural language]"
  "output (SPARQL query): [the sample SPARQL query]"
  "What is the SPARQL query for: [the question in natural language]"

4.2 Experimental Setup

We adopted an extensive set of metrics for evaluating the performance of the models over SciQA: Precision, Recall, F1 Score, Bleu 4, Bleu Cumulative, Rouge

1, and Rouge 2. The precision is calculated as the ratio of common tokens to the total number of tokens in the predicted output. Similarly, recall is computed by dividing the number of common tokens by the total number of tokens in the ground truth. The F1-score is computed as $2 \times \frac{prec \times rec}{prec + rec}$. The BLEU score evaluates the quality of predicted text, referred to as the candidate, by comparing it to a set of references. Representing a precision-based measure, the BLEU score ranges from 0 to 1, with a higher value indicating better prediction quality. A value above 0.3 is generally considered a good score. Cumulative BLEU is determined by adding the n-gram accuracy scores from all reference translations and then taking the geometric mean. Finally, Rouge-n measures the number of matching n-grams between the model-generated text and a human-produced reference. We refer to Chauhan and Daniel [13] for a comprehensive review of these metrics.

The setup we have used for the approaches is the following. T5-base has been first fine-tuned using HuggingFace *Seq2SeqTrainer*[10] with the following parameters: learning_rate=2e-5, weight_decay=0.01, fp16=True. Then, we used the method *generate* of the class *AutoModelForSeq2SeqLM*[11] from the Transformers[12] library with the following setting: max_new_tokens=512, do_sample=True, top_k=30, top_p=0.95. GPT-2-large was fine-tuned using HuggingFace *Trainer*[13] using all the default parameters. GPT-2-large was utilized through the *pipeline*[14] abstraction from the Transformers[12] library and setting the following parameters: task=text-generation, max_new_tokens=400, return_full_text=False. Dolly-v2-3b was also employed using the *pipeline* abstraction from the Transformers[12] library and the following parameters: torch_dtype=torch.bfloat16, trust_remote_code=True. Finally, GPT-3.5 Turbo was used through the OPENAI API[15] via the *create* method of the class *ChatCompletion*[16] from the *openai*[17] Python library. The parameters were set as follows: system role: *"You are a translator from natural language to SPARQL using the provided examples."*, temperature=0.5, top_p=0.95, frequency_penalty=0, presence_penalty=0, stop=None, timeout=30.

[10] Seq2SeqTrainer - `https://huggingface.co/docs/transformers/v4.35.1/en/main_classes/trainer#transformers.Seq2SeqTrainer`

[11] AutoModelForSeq2SeqLM - `https://huggingface.co/docs/transformers/v4.35.1/en/model_doc/auto#transformers.AutoModelForSeq2SeqLM`

[12] Transformers - `https://huggingface.co/docs/transformers/v4.35.1/en/index`

[13] Trainer - `https://huggingface.co/docs/transformers/v4.35.1/en/main_classes/trainer`

[14] Pipeline class - `https://huggingface.co/docs/transformers/v4.35.1/en/main_classes/pipelines`

[15] OPENAI API - `https://openai.com/blog/openai-api`

[16] ChatCompletion - `https://platform.openai.com/docs/guides/text-generation/chat-completions-api`

[17] OpenAI Libraries - `https://platform.openai.com/docs/libraries`

5 Results and Discussion

Tables 2, 3, 4, and 5 report the complete set of experiments on the four models, covering all the metrics introduced in Section 4.2. Table 6 presents a comparative summary of each configuration, focusing specifically on F1-scores and exact matches.

For the sake of space, we will primarily focus on each model's top performance. The highest F1 score was achieved by the fine-tuned T5 model, with a score of 0.9751. This was closely followed by GPT-3.5, which attained a score of 0.9736 using similarity and a 7-sample few-shot approach, and the fine-tuned GPT-2 (0.9669). Dolly's best performance, yielded using a 1-sample few-shot method with the sample selected via similarity, resulted in a score of 0.8792. Although this is the lowest among the compared models, it is still fairly good.

Table 2. T5-base model - Performance when applying the fine-tuning strategy.

Strat.	Prec.	Rec.	F1 Score	Blue 4	Bleu C	Rouge 1	Rouge 2	Exact matches
FT	0.9767	0.9760	0.9751	0.9597	0.9631	0.9790	0.9683	483

Table 3. GPT2-large model - Performance based on the strategy (Strat.) and the criteria for FSL sample selection (C): Random, Similarity, Diversity, Entropy. S is the number of samples. In bold the best results for each metric.

Strat.	C	Test name	S	Prec.	Rec.	F1 Score	Blue 4	Bleu C	Rouge 1	Rouge 2	Exact matches
FT				**0.9693**	**0.9669**	**0.9669**	**0.9462**	**0.9504**	**0.9708**	**0.9580**	**430**
ZSL				0.0464	0.1579	0.0653	0.0004	0.0009	0.0932	0.0087	0
FSL	Rand.		1	0.1119	0.3001	0.1499	0.0191	0.02950	0.1826	0.0692	0
			2	0.1230	0.2968	0.1580	0.0255	0.0372	0.1976	0.0807	0
			3	0.1336	0.3195	0.1730	0.0303	0.0449	0.2287	0.1003	0
	Simil.		1	0.1477	0.4877	0.2076	0.0685	0.0822	0.2535	0.1640	0
			2	0.2054	0.5112	0.2719	0.1205	0.1355	0.3237	0.2454	1
			3	0.2366	0.5692	0.3107	0.1502	0.1670	0.3767	0.3077	2
	Dive.	Test A	3	0.1680	0.4422	0.2215	0.0519	0.0721	0.2879	0.1573	0
		Test B	3	0.2205	0.5636	0.2988	0.1496	0.1635	0.3555	0.2926	1
	Entro.	Same_Templ	1	0.1567	0.3903	0.2029	0.0541	0.0696	0.2383	0.1373	0
		Ran_Templ	1	0.1105	0.2778	0.1421	0.0180	0.0280	0.1798	0.0675	0
		Low_Perp	1	0.2408	0.4360	0.2788	0.0924	0.1263	0.3427	0.2302	0

The scenario is similar when considering exact matches. Once again, the top three performing approaches are the fine-tuned T5 model (scoring 483/513,

Table 4. Dolly-v2-3b model - Performance based on the strategy (Strat.) and the criteria for FSL sample selection (C): Random, Similarity, Diversity, Entropy. S is the number of samples. In bold the best results for each metric.

Strat.	C	Test name	S	Prec.	Rec.	F1 Score	Blue 4	Bleu C	Rouge 1	Rouge 2	Exact matches
ZSL				0.1911	0.0993	0.1087	0.0033	0.0062	0.1734	0.0358	0
FSL	Random		1	0.5976	0.5973	0.5659	0.2799	0.3456	0.6269	0.4559	27
			3	0.5830	0.6676	0.5900	0.3484	0.4038	0.6435	0.5131	31
			5	0.6075	0.7147	0.6242	0.3934	0.4450	0.6847	0.5575	51
			7	0.6358	0.7423	0.6576	0.4460	0.4947	0.7153	0.6001	69
	Similarity		1	**0.8742**	0.9088	**0.8792**	**0.7728**	**0.8015**	**0.8861**	**0.8494**	167
			3	0.7979	0.9163	0.8304	0.7204	0.7432	0.8775	0.8470	182
			5	0.7845	**0.9238**	0.8242	0.7102	0.7322	0.8711	0.8407	180
			7	0.7681	0.9057	0.8052	0.6911	0.7113	0.8515	0.8215	181
	Diversity	Test A	3	0.6621	0.8240	0.7000	0.4587	0.5138	0.7766	0.6565	43
			5	0.6350	0.8151	0.6825	0.4329	0.4912	0.7702	0.6327	39
			7	0.6316	0.7941	0.6729	0.4246	0.4836	0.7529	0.6116	46
		Test B	3	0.7623	0.9068	0.8025	0.6926	0.7139	0.8533	0.8254	171
			5	0.7793	0.9223	0.8181	0.7148	0.7346	0.8647	0.8411	201
			7	0.7961	0.9122	0.8261	0.7279	0.7456	0.8647	0.8398	**212**
	Entr.	Same_Templ	1	0.4866	0.8089	0.5734	0.3662	0.4005	0.5741	0.5474	2
		Ran_Templ	1	0.3789	0.6107	0.4402	0.1604	0.2155	0.4811	0.3288	0
		Low_Perp	1	0.5854	0.8647	0.6757	0.4499	0.5013	0.7262	0.6201	1

Table 5. GPT-3.5-Turbo model - performance based on the strategy (Strat.) and the criteria for FSL sample selection (C): Random, Similarity, Diversity, Entropy. S is the number of samples. In bold the best results for each metric.

Strat.	C	Test name	S	Prec.	Rec.	F1 Score	Blue 4	Bleu C	Rouge 1	Rouge 2	Exact matches
ZSL				0.4963	0.1931	0.2632	0.0039	0.0088	0.4010	0.1019	0
FSL	Random		1	0.8162	0.6962	0.7362	0.3954	0.4828	0.8194	0.6107	45
			3	0.8707	0.8052	0.8259	0.5914	0.6558	0.8817	0.7543	113
			5	0.9024	0.8499	0.8675	0.6813	0.7343	0.9073	0.8141	165
			7	0.9226	0.8726	0.8905	0.7356	0.7799	0.9227	0.8476	189
	Similarity		1	0.9509	0.9305	0.9368	0.8655	0.8833	0.9505	0.9137	356
			3	0.9730	0.9635	0.9667	0.9378	0.9439	0.9750	0.9558	451
			5	0.9759	0.9685	0.9709	0.9481	0.9521	0.9772	0.9609	464
			7	**0.9768**	**0.9727**	**0.9736**	**0.9534**	**0.9571**	**0.9788**	**0.9638**	**475**
	Diversity	Test A	3	0.9693	0.9156	0.9378	0.8613	0.8784	0.9619	0.9232	315
			5	0.9730	0.9209	0.9428	0.8699	0.8875	0.9642	0.9290	328
			7	0.9688	0.9158	0.9375	0.8599	0.8783	0.9610	0.9230	313
		Test B	3	0.9678	0.9492	0.9561	0.9057	0.9181	0.9672	0.9391	412
			5	0.9683	0.9502	0.9566	0.9076	0.9192	0.9681	0.9401	417
			7	0.9673	0.9501	0.9562	0.9061	0.9180	0.9671	0.9386	422
	Ent.	Same_Templ	1	0.9303	0.8781	0.8988	0.7759	0.8119	0.9252	0.8631	205
		Ran_Templ	1	0.8268	0.6345	0.7016	0.3222	0.4070	0.7946	0.5651	26

94.1%), GPT-3.5 employing the similarity-based 7-sample few-shot approach (92.6%), and the fine-tuned GPT-2 (83.8%). However, in contrast to the previous findings, the performance of the Dolly model is consistently subpar across all configurations, with the best outcome achieved using a 7-sample few-shot approach (Diversity - Test B, always using the same template), yielding a score of only 41.3% (212/513). The low performance of Dolly may be due to its exclusive reliance on FSL and ZST, as it has not undergone fine-tuning due to its significant size. The fact that the best version of Dolly uses Diversity - Test B may depend on the fact that using a substantial number of examples following a consistent template enables Dolly to produce comparatively better queries.

When considering all the other metrics introduced in Section 4.2 and reported in Tables 2-5, the trends are very similar. The top three methods in terms of performance for Rouge 1, Rouge 2, Bleu 4, and Bleu C are again the fine-tuned T5, GPT 3.5 with 7-sample few-shot, and the fine-tuned GPT-2.

These results reveal several noteworthy insights. First, despite the challenges presented by the SciQA benchmark and the generally low performance in zero-shot learning observed among all models (no exact matches and <26% F1), the highest-performing solutions that employed fine-tuning and few-shot learning exhibited excellent results. This suggests that as LLMs advance and our understanding of them deepens, more challenging benchmarks may be soon required.

Second, the T5 model, although the smallest model by a large margin (220M parameters), surpassed all the larger models and even robust commercial solutions like GPT-3.5 (175B) when fine-tuned with relevant data. This supports the idea that appropriately tailored datasets can enable the development of efficient, low-resource models that perform comparably to more computationally intensive options. The financial and sustainability implications of this finding warrant further exploration.

Third, our analysis identified semantic similarity as the most effective approach for selecting samples in few-shot learning tasks. This method consistently produced the highest performance across all models. However, it was observed that different models react differently to changes in sample size. For example, GPT 3.5 showed improved performance with larger sample sizes when using similarity-based selection. In contrast, Dolly's performance declined, suggesting that an increase in examples might confuse this model. These phenomena merit additional investigation in future studies.

6 Error Analysis

We performed a comprehensive analysis of erroneous queries, defined as those not matching the expected response. The analysis focused on the errors produced by T5-base (30 errors), which was the top-performing model, and GPT-3.5 (38 errors), which was the second-best performer.

We identified five unique error categories. Table 7 details the definitions of these categories and the respective proportions of erroneous queries for both T5 and GPT-3.5. Each query can be associated with multiple categories due to the

Table 6. Summary of F1-scores and exact matches (in parenthesis) based on the different strategies (Strat.), and the various Criteria (C): Random, Similarity, Diversity, Entropy. S is the number of samples. In bold the best results for each model.

Strat.	C	Test name	S	T5-base	GPT2-large	Dolly-v2-3b	GPT-3.5-turbo
ZSL					0.0653 (0)	0.1087 (0)	0.2632 (0)
FT				**0.9751 (483)**	**0.9669 (430)**		
FSL	Random		1		0.2005 (0)	0.5659 (27)	0.7362 (45)
			3		0.2187 (0)	0.5900 (31)	0.8259 (113)
			5			0.6242 (51)	0.8675 (165)
			7			0.6576 (69)	0.8905 (189)
	Similarity		1		0.2718 (0)	**0.8792** (167)	0.9368 (356)
			3		0.4051 (2)	0.8304 (182)	0.9667 (451)
			5			0.8242 (180)	0.9709 (464)
			7			0.8052 (181)	**0.9736 (475)**
	Diversity	Test A	3		0.2215 (0)	0.7000 (43)	0.9378 (315)
			5			0.6525 (39)	0.9428 (328)
			7			0.6729 (46)	0.9375 (313)
		Test B	3		0.2988 (1)	0.8025 (171)	0.9561 (412)
			5			0.8181 (201)	0.9566 (417)
			7			0.8261 **(212)**	0.9562 (422)
	Ent.	Same_Templ	1		0.2029 (0)	0.5734 (2)	0.8988 (205)
		Ran_Templ	1		0.1421 (0)	0.4402 (0)	0.7016 (26)
		Low_Perp	1		0.2788 (0)	0.6757 (1)	

presence of more than one type of error. Therefore, the sum of all these categories does not necessarily equal to 100%.

The comparison of incorrect results between T5 and GPT-3.5, reveals a total of 21 overlapping queries. They included 10 instances of error type 1 (misspelled entity), 11 of type 2 (wrong entity type), 9 of error type 3 (wrong predicate), and 4 of error type 5 (semantic error). Despite these overlaps, T5 and GPT-3.5 exhibit significant differences across several categories of errors, underscoring their distinct strengths and weaknesses. Indeed, they only exhibit similar behavior with respect to the *misspelled entity* category (60.0% and 60.5%, respectively), which explains the high number of errors of type 1 in the overlapping mistakes. Both models occasionally alter the names of entities, often using synonyms. An example of this is provided in Example 3, where the entity *linear-chain CRFs* was modified to *label-chain CRFs*.

The primary issue with T5 pertains to syntactic errors, accounting for 40.0% of errors compared to only 5.2% in GPT-3.5. This suggests that T5 has a relatively weaker grasp of SPARQL syntax compared to GPT-3.5. Example 4 reports a typical case in which T5 generate a syntactically incorrect query. In all other error categories, T5 demonstrates notably fewer mistakes than GPT-3.5.

GPT-3.5 is characterized by a range of issues, most notably a semantic error rate that is more than double that of T5 (57.8% vs 26.6%). This indicates that while GPT-3.5 typically generates syntactically accurate queries, it tends to

misinterpret their meaning more often than T5. Additionally, GPT-3.5 is more prone to incorrectly generating the wrong entity type or predicate. These problems may partly stem from the inclusion of SPARQL queries from Wikidata and DBpedia in its training data. Consequently, instead of accurately translating queries, GPT-3.5 may hallucinate and erroneously insert SPARQL query fragments that refer to Wikidata and other large knowledge graphs. Example 5 shows a typical case in which GPT-3.5 hallucinates a fictitious predicate (orkgp:P2067), probably influenced by the Wikidata predicate *Property:P2067*. This analysis underscores the challenges LLMs face when attempting to use specialized domain ontologies, which may diverge from standard predicate and type conventions. Consequently, a less complex and more straightforward model, such as T5, may at times produce better results.

Table 7. Categories of errors and their frequencies for GPT-3.5 and T5.

Error Category	Category Definition	GPT-3.5	T5
(1) Misspelled entity	The query misspells an entity name (e.g, "robotic navigation" instead of "robot navigation").	60.5%	60.0%
(2) Wrong entity type	The query uses an entity of a different type (e.g., the query searching an entity of type Energy Source instead of one of type Paper).	52.6%	36.6%
(3) Wrong predicate	The query uses an incorrect predicate in the query (e.g., orkgp:P7046 has been replaced with orkgp:HAS_METHOD).	76.3%	36.6%
(4) SPARQL syntactic error	The query cannot be syntactically parsed by a SPARQL compiler.	5.2%	40.0%
(5) Semantic error	The query does not reflect the meaning of the question.	57.8%	26.6%

7 Limitations

This paper focuses on the SciQA benchmark, since it is meant to be a challenge for next-generation QA systems. However, when analyzing the findings, it is important to consider some limitations and factors regarding the general applicability of the proposed solutions. First, SciQA includes numerous questions that have been automatically generated from initial seed questions. Despite the complexity of these queries, LLMs may be capable of identifying the some inherent patterns. Therefore, the results shown here do not necessarily carry over to other benchmarks, which were entirely handcrafted, e.g., [24] indicates that semantic parsing is still a challenging task for LLMs. Additionally, success on

the SciQA is heavily reliant on the appropriate application of literals, unlike other QA benchmarks where the accuracy of entities plays a more crucial role. While SciQA was just recently released, we would also like to point out that the precise training schedules for OpenAI models are unknown. Therefore, there exists a small yet plausible chance that the GPT-3.5 model may have encountered information from the SciQA benchmark during its training phase.

A further limitation is that we focused on query similarity metrics on the generated query string as opposed to execution metrics for computational reasons. For example, for our exact match metric, there is a risk that an LLM-generated query is rated as incorrect even if retrieves the correct results and the query formulation is aligned with the intent of the question, i.e., we could underestimate model performance. However, the manual examination of the sample of 68 errors produced by T5 and GPT-3.5 reported in Section 6 found that all queries marked as erroneous would not have yielded the correct results upon execution.

8 Conclusions

This paper presented an in-depth examination of the performance of LLMs on the SciQA benchmark. We evaluated four distinct language models, employing three methodologies: zero-shot learning, few-shot learning, and fine-tuning. Additionally, we explored seven different strategies for choosing examples in few-shot learning. Our findings demonstrate that optimizing LLMS with appropriate prompting and fine-tuning techniques produces outstanding results on this benchmark, with the best model yielding a >94.1% exact query match accuracy and >97.5% F1-score. This performance highlights the potential need for more challenging benchmarks in this space.

Interestingly, the most effective results across nearly all metrics were achieved by the fine-tuned T5 model, which is relatively small (220M parameters), followed by GPT-3.5 using a seven-sample few-shot, and the fine-tuned GPT-2. This suggests that low-resource models, if carefully fine-tuned, can rival the performance of larger, more resource-intensive models even on a complex task such as scientific question answering. Finally, the analysis shows that semantic similarity is the most effective method for sample selection in few-shot learning for this specific task.

In our future research, we plan to broaden our examination by conducting comprehensive experiments on the capability of LLMs to perform knowledge graph question answering across various domains. We are also investigating the potential of LLMs in performing related tasks, such as generating scientific hypotheses [8], classifying research articles [11], recommending citations [10], and producing literature reviews [7]. Finally, we plan to apply the valuable insights gained from this study to develop a more challenging benchmark for scientific question answering. Specifically, we plan to produce a new resource based on large-scale knowledge graphs in this space such as the Academia/Industry DynAmics Knowledge Graph (AIDA-KG) [1] and the Computer Science Knowledge Graph (CS-KG) [16].

A Appendix - Examples

Example 1 Question: What are the titles and IDs of research papers that include a benchmark for the DDI extraction 2013 corpus dataset?

```
SELECT DISTINCT ?paper ?paper_lbl
WHERE { ?dataset a orkgc:Dataset; rdfs:label ?dataset_lbl.
   FILTER (str(?dataset_lbl) = ‘‘DDI extraction 2013 corpus’’)
     ?benchmark orkgp:HAS_DATASET ?dataset.
     ?cont orkgp:HAS_BENCHMARK ?benchmark.
     ?paper orkgp:P31 ?cont; rdfs:label ?paper_lbl.
}
```

Example 2 Type of data used in the fine-tuning process of the T5-base model.

Request:

“Translate English to SPARQL: Which model has achieved the highest Accuracy score on the ARC (Challenge) benchmark dataset?”

Expected output:

```
SELECT DISTINCT ?model ?model_lbl
 WHERE { ?metric a orkgc:Metric; rdfs:label ?metric_lbl.
 FILTER (str( ?metric_lbl) = ‘‘Accuracy") { SELECT ?model ?model_lbl
 WHERE { ?dataset a orkgc:Dataset; rdfs:label ?dataset_lbl.
 FILTER (str( ?dataset_lbl) = ‘‘ARC (Challenge)")
 ?benchmark orkgp:HAS_DATASET ?dataset; orkgp:HAS_EVALUATION ?eval.
 ?eval orkgp:HAS_VALUE ?value; orkgp:HAS_METRIC ?metric.
 ?cont orkgp:HAS_BENCHMARK ?benchmark; orkgp:HAS_MODEL ?model.
 ?model rdfs:label ?model_lbl. } ORDER BY DESC( ?value) LIMIT 1 } }
```

Example 3 Example of *misspelled entity* generated by T5. In blue the original entity and in red the misspelled one. Question: *List the code links in papers that use the linear-chain CRFs model in any benchmark?*

Original query:

```
SELECT DISTINCT ?code
WHERE {
    ?model a orkgc:Model;
        rdfs:label ?model_lbl.
    FILTER (str( ?model_lbl) =
    "linear-chain CRFs")
    ?benchmark orkgp:HAS_DATASET ?dataset.
    ?cont orkgp:HAS_BENCHMARK ?benchmark.
    ?cont orkgp:HAS_MODEL ?model;
        orkgp:HAS_SOURCE_CODE ?code.
}
```

Generated output:

```
SELECT DISTINCT ?code
WHERE {
    ?model a orkgc:Model;
        rdfs:label ?model_lbl.
    FILTER (str( ?model_lbl) =
    "label-chain CRFs")
    ?benchmark orkgp:HAS_DATASET ?dataset.
    ?cont orkgp:HAS_BENCHMARK ?benchmark.
    ?cont orkgp:HAS_MODEL ?model;
        orkgp:HAS_SOURCE_CODE ?code.
}
```

Example 4 Example of *SPARQL syntactic error* generated by T5. In red the unmatching part. Question: *What is Raman spectroscopy?*

Original query:

```
SELECT ?definitions
WHERE {
    ?terms orkgp:P24009 ?definitions.
    ?terms rdfs:label ?terms_labels.
    FILTER(REGEX( ?terms_labels,
    "Raman spectroscopy"))
}
```

Generated output:

```
SELECT ?rmdfs:label ?rmdfs:label
?rmdfs:label ?rmdfs:label ?rmdfs:label
?rmdfs:label ?rmdfs:label ?rmdfs:label
?rmdfs:label ?rmdfs:label ?rmdfs:label
?rmdfs:label. }
```

Example 5 Example of *wrong predicate* error generated by GPT-3.5. In red the hallucinated predicate. Question: *What quantity of iron oxide was discovered on Elorza crater?*

Original query:

```
SELECT ?properties_values,
       ?property_description
WHERE {
    ?papers rdf:type orkgc:Paper.
    ?papers rdfs:label ?papers_labels.
    ?papers orkgp:P31 ?contrib.
    ?contrib ?properties ?properties_values.
    ?properties rdfs:label ?properties_labels.
    ?properties orkgp:description ?property_description.
    FILTER(REGEX( ?properties_labels, "FeO"))
    FILTER(REGEX(?papers_labels, "Elorza crater", "i"))
}
```

Generated output:

```
SELECT ?quantity
WHERE {
    ?crater rdf:type orkgc:Crater.
    ?crater rdfs:label "Elorza crater".
    ?crater orkgp:P31 ?iron_oxide_discoveries.
    ?iron_oxide_discoveries orkgp:P2067 ?quantity.
}
```

References

1. Angioni, S., Salatino, A., Osborne, F., Recupero, D.R., Motta, E.: Aida: A knowledge graph about research dynamics in academia and industry. Quantitative Science Studies **2**(4), 1356–1398 (2021)
2. Auer, S., Barone, D.A.C., Bartz, C., Cortes, E.G., Jaradeh, M.Y., Karras, O., Koubarakis, M., Mouromtsev, D., Pliukhin, D., Radyush, D., Shilin, I., Stocker, M., Tsalapati, E.: The sciqa scientific question answering benchmark for scholarly knowledge. Scientific Reports **13**(1), 7240 (May 2023). https://doi.org/10.1038/s41598-023-33607-z, `https://doi.org/10.1038/s41598-023-33607-z`
3. Babu, G.A., Badugu, S.: A survey on automatic text summarisation. In: Proceedings of Third International Conference on Advances in Computer Engineering and Communication Systems: ICACECS 2022. pp. 679–689. Springer (2023)
4. Banerjee, D., Usbeck, R., Mihindukulasooriya, N., Singh, G., Mutharaju, R., Kapanipathi, P. (eds.): Joint Proceedings of Scholarly QALD 2023 and SemREC 2023 co-located with 22nd International Semantic Web Conference ISWC 2023, Athens, Greece, November 6-10, 2023, CEUR Workshop Proceedings, vol. 3592. CEUR-WS.org (2023), `https://ceur-ws.org/Vol-3592`
5. Bansal, T., Jha, R., McCallum, A.: Learning to few-shot learn across diverse natural language classification tasks. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 5108–5123 (2020)
6. Biderman, S., Schoelkopf, H., Anthony, Q.G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M.A., Purohit, S., Prashanth, U.S., Raff, E., et al.: Pythia: A suite

for analyzing large language models across training and scaling. In: International Conference on Machine Learning. pp. 2397–2430. PMLR (2023)
7. Bolanos, F., Salatino, A., Osborne, F., Motta, E.: Artificial intelligence for literature reviews: Opportunities and challenges. arXiv preprint arXiv:2402.08565 (2024)
8. Borrego, A., Dessi, D., Hernández, I., Osborne, F., Recupero, D.R., Ruiz, D., Buscaldi, D., Motta, E.: Completing scientific facts in knowledge graphs of research concepts. IEEE Access **10**, 125867–125880 (2022)
9. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners (2020)
10. Buscaldi, D., Dessí, D., Motta, E., Murgia, M., Osborne, F., Recupero, D.R.: Citation prediction by leveraging transformers and natural language processing heuristics. Information Processing & Management **61**(1), 103583 (2024)
11. Cadeddu, A., Chessa, A., De Leo, V., Fenu, G., Motta, E., Osborne, F., Recupero, D.R., Salatino, A., Secchi, L.: A comparative analysis of knowledge injection strategies for large language models in the scholarly domain. Engineering Applications of Artificial Intelligence **133**, 108166 (2024)
12. Chakraborty, N., Lukovnikov, D., Maheshwari, G., Trivedi, P., Lehmann, J., Fischer, A.: Introduction to neural network-based question answering over knowledge graphs. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **11**(3), e1389 (2021)
13. Chauhan, S., Daniel, P.: A comprehensive survey on various fully automatic machine translation evaluation metrics. Neural Processing Letters (05 2022). https://doi.org/10.1007/s11063-022-10835-4
14. Chen, Y., Kang, H., Zhai, V., Li, L., Singh, R., Raj, B.: Token prediction as implicit classification to identify llm-generated text. arXiv preprint arXiv:2311.08723 (2023)
15. Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., Xin, R.: Free dolly: Introducing the world's first truly open instruction-tuned llm (2023), `https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm`
16. Dessí, D., Osborne, F., Reforgiato Recupero, D., Buscaldi, D., Motta, E.: Cs-kg: A large-scale knowledge graph of research entities and claims in computer science. In: International Semantic Web Conference. pp. 678–696. Springer (2022)
17. Fu, Z., Yang, H., So, A.M.C., Lam, W., Bing, L., Collier, N.: On the effectiveness of parameter-efficient fine-tuning (2022)
18. Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., et al.: Knowledge graphs. ACM Computing Surveys (Csur) **54**(4), 1–37 (2021)
19. Iter, D., Pryzant, R., Xu, R., Wang, S., Liu, Y., Xu, Y., Zhu, C.: In-context demonstration selection with cross entropy difference. arXiv preprint arXiv:2305.14726 (2023)
20. Jiang, L., Yan, X., Usbeck, R.: A structure and content prompt-based method for knowledge graph question answering over scholarly data. CEUR Workshop proceedings **3592** (2023), `https://ceur-ws.org/Vol-3592/paper3.pdf`
21. Kamath, A., Das, R.: A survey on semantic parsing. arXiv preprint arXiv:1812.00978 (2018)

22. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners (2023)
23. Kumagai, A., Iwata, T., Fujiwara, Y.: Few-shot learning for unsupervised feature selection. arXiv preprint arXiv:2107.00816 (2021)
24. Lehmann, J., Gattogi, P., Bhandiwad, D., Ferré, S., Vahdati, S.: Language models as controlled natural language semantic parsers for knowledge graph question answering. In: European Conference on Artificial Intelligence (ECAI). vol. 372, pp. 1348–1356. IOS Press (2023)
25. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic web **6**(2), 167–195 (2015)
26. Levy, I., Bogin, B., Berant, J.: Diverse demonstrations improve in-context compositional generalization. arXiv preprint arXiv:2212.06800 (2022)
27. Lin, X.V., Mihaylov, T., Artetxe, M., Wang, T., Chen, S., Simig, D., Ott, M., Goyal, N., Bhosale, S., Du, J., et al.: Few-shot learning with multilingual generative language models. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. pp. 9019–9052 (2022)
28. Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., Chen, W.: What makes good in-context examples for gpt-3? arXiv preprint arXiv:2101.06804 (2021)
29. Meloni, A., Angioni, S., Salatino, A., Osborne, F., Birukou, A., Reforgiato Recupero, D., Motta, E.: Aida-bot 2.0: Enhancing conversational agents with knowledge graphs for analysing the research landscape. In: International Semantic Web Conference. pp. 400–418. Springer (2023)
30. Peng, C., Xia, F., Naseriparsa, M., Osborne, F.: Knowledge graphs: Opportunities and challenges. Artificial Intelligence Review pp. 1–32 (2023)
31. Pliukhin, D., Radyush, D., Kovriguina, L., Mouromtsev, D.: Improving subgraph extraction algorithms for one-shot sparql query generation with large language models. In: Scholarly-QALD-23: Scholarly QALD Challenge at The 22nd International Semantic Web Conference (ISWC 2023)(Athens, Greece. vol. 3592, pp. 1–10 (2023), https://ceur-ws.org/Vol-3592/paper6.pdf
32. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
33. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(1) (jan 2020)
34. Rongali, S., Soldaini, L., Monti, E., Hamza, W.: Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. In: Proceedings of The Web Conference 2020. pp. 2962–2968 (2020)
35. Rony, M.R.A.H., Chaudhuri, D., Usbeck, R., Lehmann, J.: Tree-kgqa: an unsupervised approach for question answering over knowledge graphs. IEEE Access **10**, 50467–50478 (2022)
36. Stocker, M., Oelen, A., Jaradeh, M.Y., Haris, M., Oghli, O.A., Heidari, G., Hussein, H., Lorenz, A.L., Kabenamualu, S., Farfar, K.E., Prinz, M., Karras, O., D'Souza, J., Vogt, L., Auer, S.: Fair scientific information with the open research knowledge graph **1**, 19–21 (2023). https://doi.org/10.3233/FC-221513
37. Taffa, T.A., Usbeck, R.: Leveraging llms in scholarly knowledge graph question answering. In: Scholarly-QALD-23: Scholarly QALD Challenge at The 22nd International Semantic Web Conference (ISWC 2023)(Athens, Greece. vol. 3592, pp. 1–10 (2023), https://ceur-ws.org/Vol-3592/paper5.pdf

38. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2023)
39. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (2014)
40. Wei, J., Bosma, M., Zhao, V.Y., Guu, K., Yu, A.W., Lester, B., Du, N., Dai, A.M., Le, Q.V.: Finetuned language models are zero-shot learners (2022)
41. Zhao, S., Dang, J., Grover, A.: Group preference optimization: Few-shot alignment of large language models. arXiv preprint arXiv:2310.11523 (2023)