Question Answering over Linked Data with GPT-3

Bruno Faria
Department of Informatics Engineering, University of Coimbra, Portugal
Centre for Informatics and Systems of the University of Coimbra, Portugal

Dylan Perdigão
Department of Informatics Engineering, University of Coimbra, Portugal
Centre for Informatics and Systems of the University of Coimbra, Portugal

Hugo Gonçalo Oliveira
Department of Informatics Engineering, University of Coimbra, Portugal
Centre for Informatics and Systems of the University of Coimbra, Portugal

Abstract

This paper explores *GPT-3* for answering natural language questions over Linked Data. Different engines of the model and different approaches are adopted for answering questions in the *QALD-9* dataset, namely: zero and few-shot *SPARQL* generation, as well as fine-tuning in the training portion of the dataset. Answers retrieved by the generated queries and answers generated directly by the model are also compared. Overall results are generally poor, but several insights are provided on using *GPT-3* for the proposed task.

2012 ACM Subject Classification Computing methodologies → Natural language processing

Keywords and phrases SPARQL Generation, Prompt Engineering, Few-Shot Learning, Question Answering, GPT-3

Digital Object Identifier 10.4230/OASIcs.SLATE.2023.1

Supplementary Material
Software (Repository): `https://github.com/brunofaria1322/GPT3-over-QALD9`, archived at `swh:1:dir:5c52a1c2df0a799ceee6ac97ea1fe3ff6e056694`

Funding This work was partially supported by the Portuguese Recovery and Resilience Plan (PRR) through project C645008882-00000055, Center for Responsible AI; and by FCT – Foundation for Science and Technology, I.P., within the scope of the project CISUC – UID/CEC/00326/2020 and by the European Social Fund, through the Regional Operational Program Centro 2020. It is also based upon work in COST Action CA18209 Nexus Linguarum, supported by COST (European Cooperation in Science and Technology). `http://www.cost.eu/`.

1 Introduction

The Generative Pre-trained Transformer 3 (*GPT-3*) [7] Language Model (*LM*), developed by *OpenAI*, is known to perform a broad range of Natural Language Processing (*NLP*) and generation tasks, like summarisation, classification, or translation, in a zero or few-shot scenario. However, there is not much work concerning its use for generating Simple Protocol And RDF Query Language (*SPARQL*). This gap, to which access limitations contribute, is the primary motivation for exploring *GPT-3* in this task. We explore this model in the generation of *SPARQL* queries for generic questions in Natural Language (*NL*). Such queries should be able to retrieve answers from Linked Data (*LD*). The advantage of using a Large Language Model (*LLM*) like *GPT-3* is that we are not limited to a Knowledge Base (*KB*) with static finite information. Not that the *LLM*s has infinite information, but it is much more flexible: it can learn, even from only a few examples (i.e., in few-shot learning), and, independently of the quality, will generate outputs for any prompt.

On the other hand, *KB* and *LD* are aligned with the FAIR data principles (Findable, Accessible, Interpretable, Reusable) [23], in opposition to black-box *LLM*. Therefore, instead of using *GPT-3* directly for answering questions, a middle-ground would be using this model for generating human-interpretable *SPARQL*, which may then be used for querying *LD*, represented in Resource Description Framework (*RDF*).

For exploring *GPT-3* in this task, we rely on the Question Answering over Linked Data 9 (*QALD-9*) [22] dataset, which has: *NL* questions; *SPARQL* for retrieving their answers from *DBpedia* [2]; and the actual answers retrieved by these queries. Question Answering over Linked Data (*QALD*) is a series of challenges that started in 2011, and are currently in the 10^{th} edition[1]. Questions are available in several languages, but most translations lack the necessary high quality, so we focus on English.

Using *QALD-9*, experiments are conducted for generating *SPARQL* queries for *DBpedia* with *GPT-3*, using different engines (i.e., *text-davinci-002* and *text-davinci-003*) and approaches (i.e., zero-shot, few-shot, fine-tuning). Generated queries are evaluated with BLEU [15] scores. Evaluation is complemented with the F1-score, computed on the results of running the generated queries, and on the answers directly generated by *GPT-3* when the *NL* questions are asked.

Amongst our findings, we highlight that the zero-shot approach generates many invalid *SPARQL* queries and that the queries by the fine-tuned model are the closest to the reference, followed by the few-shot approach. On the other hand, answers retrieved from *DBpedia* with queries by the few-shot approach are comparable to those of the fine-tuned model, which learned from many more examples. Still, the best answers are obtained by asking the *NL* question directly to *GPT-3*, for which the query is not necessary. Despite the insights provided by this exploration of *GPT-3*, overall, all results end up being poor according to the adopted metrics.

The remainder of this paper is organised as follows. Section 2 overviews existing *LLM*s and their use cases in the scope of Question Answering (*QA*). Section 3 highlights essential tools and frameworks for our experimentation. Section 4 describes the adopted methodologies. Section 5 presents the obtained results, further discussed in Section 6. Finally, Section 7 concludes the paper and points to possible future directions.

2 Related Work

Bidirectional Encoder Representations from Transformers (*BERT*) [9] and Generative Pre-trained Transformer (*GPT*) are two of the most popular *LM* based on the Transformer architecture. Among many other tasks, they have both been used for *QA*.

BERT, developed by *Google*, uses only encoder blocks, and can be used for providing contextual word embeddings or fine-tuned for many *NLP* tasks, including Extractive *QA*, as long as data is available. *GPT*, an auto-regressive *LM* developed by *OpenAI*, has only decoder blocks and is mostly used for text generation. However, this is enough for current versions of this model, namely *GPT-3* [7] and the recent *GPT-4* [14], performing a broad range of *NLP* tasks based on text prompts, not requiring fine-tuning (zero and few-shot), which can still be performed for specific applications.

There is much work on automatic *QA*, mainly from unstructured text, often referred to as Information Retrieval (*IR*)-based *QA*. Recent approaches rely on fine-tuning transformers for extractive *QA* [9] or *QA* on the domain of the training data [16].

[1] `https://www.nliwod.org/challenge` (accessed on 20/03/23)

Alternatively, knowledge-based *QA* gets answers from a structured *KB*. For this, *NL* questions must be converted to logical constraints or structured queries, e.g., through semantic parsing [6], or, more recently, deep neural networks [8].

When it comes to generating *SPARQL* queries, for *KB* in *RDF*, there are datasets of *NL* questions and their translation to *SPARQL*. These include *LC-QuAD* [20] and the *QALD* [22]. The latter results from a series of challenges, currently in their tenth edition[2].

SparseQA [3] is a framework used for answering complex questions tested in several datasets, including those previously mentioned. It adopts a word-reordering approach for creating and refining a graph based on each question. This encompasses:

(i) the classification of the question type;
(ii) the identification of entities and variables;
(iii) the construction of a graph from the sequential analysis of the question words.

The search space is then reduced by creating a knowledge sub-graph, and an approximate match is performed with the relation pattern-based graph similarity. *SParseQA* was shown to perform better than other systems that generate *SPARQL* with a broad range of approaches, such as: graph traversal [21] and other graph-based [11, 12]; traditional supervised machine learning [4]; parsing [24, 5] and rules on the underlying *KB* semantics [10]; query template learning [22] and pattern recognition [28].

The performance of *SPARQL* generation with *BERT* and *GPT-3* was compared in a *KB* of aviation accident reports [1]. Four models, namely *BM25-BERT* (baseline), *KGQA*, *BERT-QA*, *GPT-3-QA*, and two combinations, *KGQA+BERT-QA* and *KGQA+GPT-3-QA*, were tested. Results were assessed with Exact Match (*EM*), Exact Recall (*ER*), accuracy, and recall. *KGQA+GPT-3-QA* was the best approach in most metrics, which shows the benefits of combining models. Even though *GPT-3-QA* was based on *GPT-3*, it used older engines (*ada* and *curie*) and is focused on aviation reports. There are very recent reports [18] on using *GPT-3* and related models for *QA*, in *QALD* and other datasets. When noting that some of the models have difficulties for generating SPARQL, they focus only on the answer, and report a F1 of 46% (text-davinci-003). In a related task, knowledge-based visual *QA*, the steps of knowledge retrieval and reasoning were unified by prompting *GPT-3*, used implicitly as a *KB* [25].

SPBERT [19] was the first transformer-based *LM* pre-trained on a large quantity of *SPARQL* queries. After fine-tuning, it was tested in four datasets: *QALD-9*, *LC-QuAD*, *Mon* [17] and Verbalization QUestion ANswering DAtaset (*VQuAnDa*) [13] datasets, where it outperformed other approaches that model *SPARQL* generation from *NL* as Neural Machine Translation (*NMT*) [26], with Recurrent Neural Networks (*RNN*s), Convolutional Neural Networks (*CNN*s), or an encoder-decoder Transformer model. Evaluation relied on BiLingual Evaluation Understudy (*BLEU*) [15] and *EM*.

Our work complements existing research with the use of *GPT-3* for *SPARQL* generation. As in other works, *SPARQL* is evaluated with *BLEU* and the retrieved answers with F1-score.

3 Experimentation Setup

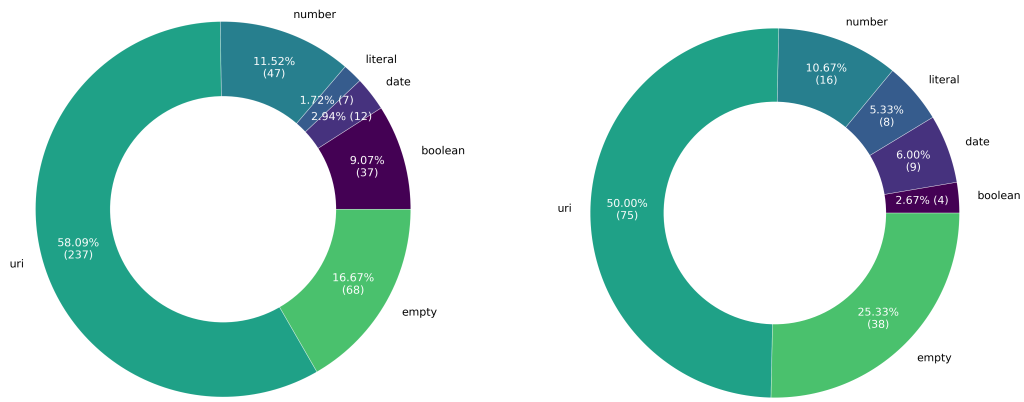
The main tools used in our experiments were:

(i) *QALD-9*, a dataset of *NL* questions and their respective *SPARQL* queries;
(ii) *OpenAI*'s Application Programming Interface (*API*), for text completion with different engines of *GPT-3*;
(iii) *SPARQLWrapper*, for executing *SPARQL* queries and getting their respective results.

[2] `https://www.nliwod.org/challenge` (accessed on 20/03/23)

```
-- Boolean
Q: Was Marc Chagall a jew?
A: False

-- Date
Q: When was Olof Palme shot?
A: 1986-02-28

-- Literal
Q: What is the birth name of Angela Merkel?
A: Angela Dorothea Kasner

-- Number
Q: How much is the elevation of Düsseldorf Airport?
A: 44.8

-- URI
Q: What are the specialities of the UNC Health Care?
A: http://dbpedia.org/resource/Cancer; http://dbpedia.org/resource/Trauma_center
```

Figure 1 Five categories of questions with their respective answers.



(a) Train (408 questions) **(b)** Test (150 questions)

Figure 2 Composition of *QALD-9* dataset.

Each entry of the *QALD-9* [22] dataset has:

(i) a *NL* question, in a number of languages;

(ii) the gold *SPARQL* query for getting the answer of the question from *DBpedia* (*2016-10* dump)[3];

(iii) the gold answers to the previous queries.

Answers may belong to one of the following five categories: boolean, date, literal, number, Uniform Resource Identifiers (*URI*s). Figure 1 shows an example question/answer pair for each category. We have only considered their English version. *QALD-9* is split into training and testing portions, each with 408 and 150 questions, respectively. However, we noted that some queries return an empty result due to wrong formatting or to changes in the current version of *DBpedia*. Since these queries did not work, they were discarded for our experimentation. Afterwards, we are left with 340 training and 112 testing questions. Figures 2a and 2b have the distribution of the *QALD-9* dataset, regarding the type of answers.

[3] Instead of *DBpedia*, version 9-plus of the dataset includes queries to *Wikidata*

OpenAI offers an *API*[4] for generating any kind of text (e.g., *NL* or code), i.e., the user prompts the model with some text and the model will generate following text. For instance, if the prompt is a question, the model is expected to generate an answer. In our case, the prompt is an instruction for generating a *SPARQL* query, and this is what we expect to be generated. A spectrum of models and engines is available for performing different tasks, with more or fewer capabilities and different prices. These models include: *davinci*, *curie*, *babbage*, or *ada*. We tested two variants of *davinci*, the most powerful for text completion: *text-davinci-002* and *text-davinci-003*. *OpenAI* also allows fine-tuning one of the available engines, which we did with *QALD*'s training set.

For executing *SPARQL* queries on the *DBpedia* endpoint[5], we use *SPARQLWrapper*[6] a Python wrapper for executing *SPARQL* queries, part of *RDFLib*. *SPARQLWrapper* also validates *GPT-3* generated queries. If the query is well-formatted, results are retrieved in a suitable format for further analysis.

4 Methods

This section describes the approaches adopted for testing *GPT-3* in the *QALD* dataset, namely: zero-shot *SPARQL* generation, few-shot *SPARQL* generation, generation with a fine-tuned model, and direct answer generation. All of them are tested in *QALD-9*'s testing data. Evaluation approaches and adopted metrics are also described.

4.1 Zero and Few-Shot

Zero and few-shot were tested in both pre-trained *GPT-3* engines, *davinci-002* and *davinci-003*. These were used with the ten prompts in Table 1, where the ⟨*question*⟩ placeholder is replaced by the questions from the *QALD-9* dataset. The result can be, for example:

- *Turn this into a DBpedia SPARQL query: "What is the time zone of Salt Lake City?"*

Since the *SPARQL* queries in *QALD-9* are meant for *DBpedia*, five prompts refer it specifically and the others do not, for later analysis of the impact of this inclusion. The response of *GPT-3* to these prompts should be a *SPARQL* query. For example, an expected query for the previous question is shown in Figure 6.

Table 1 Prompts tested for getting *SPARQL* queries.

ID	Prompt	ID	Prompt
Q1	`The SPARQL query for the question "`⟨*question*⟩`" is`	Q6	`The DBpedia SPARQL query for the question "`⟨*question*⟩`" is`
Q2	`What is the SPARQL query for the question "`⟨*question*⟩`"?`	Q7	`What is the DBpedia SPARQL query for the question "`⟨*question*⟩`"?`
Q3	`SPARQL for "`⟨*question*⟩`" is`	Q8	`The DBpedia SPARQL for "`⟨*question*⟩`" is`
Q4	`Write the complete SPARQL query to answer the question:` ⟨*question*⟩	Q9	`Write the complete DBpedia SPARQL query to answer the question: "`⟨*question*⟩`"`
Q5	`Turn this into a SPARQL query: "`⟨*question*⟩`"`	Q10	`Turn this into a DBpedia SPARQL query: "`⟨*question*⟩`"`

4 `https://openai.com/api` (accessed on 20/03/23)
5 `https://dbpedia.org/sparql` (accessed on 20/03/23)
6 `https://github.com/RDFLib/sparqlwrapper` (accessed on 20/03/23)

The main difference between zero and few-shot relies in the prompts. In zero-shot, they consist of a single *NL* instruction, followed by the *NL* question from *QALD*. The expectation is that *GPT-3* generates the *SPARQL* for the question. In few-shot, the prompt includes a number of instruction-question-*SPARQL* blocks, followed by an instruction-question pair. We only tested five-shot learning, with a prompt illustrated in Figure 3 for the previous example. The five questions for few-shot are selected from the training dataset and include one example from each question category (Figure 1).

```
Turn this into a DBpedia SPARQL query: "What are the specialities of the UNC Health Care?"
SELECT DISTINCT ?uri WHERE { <http://dbpedia.org/resource/UNC_Health_Care>
<http://dbpedia.org/property/speciality> ?uri }

Turn this into a DBpedia SPARQL query: "When was Olof Palme shot?"
SELECT DISTINCT ?date WHERE { <http://dbpedia.org/resource/Olof_Palme>
<http://dbpedia.org/ontology/deathDate> ?date }

Turn this into a DBpedia SPARQL query: "How much is the elevation of Düsseldorf Airport ?"
SELECT ?ele WHERE { <http://dbpedia.org/resource/Düsseldorf_Airport>
<http://dbpedia.org/ontology/elevation> ?ele } LIMIT 1

Turn this into a DBpedia SPARQL query: "Was Marc Chagall a jew?"
ASK WHERE { <http://dbpedia.org/resource/Marc_Chagall>
<http://dbpedia.org/property/ethnicity> \"Jewish\"@en }

Turn this into a DBpedia SPARQL query: "What is the birth name of Angela Merkel?"
SELECT DISTINCT ?string WHERE { <http://dbpedia.org/resource/Angela_Merkel
<http://dbpedia.org/property/birthName> ?string }

Turn this into a DBpedia SPARQL query: "What is the time zone of Salt Lake City?"
```

Figure 3 Prompt for few-shot learning.

4.2 Fine-tuning

Fine-tuning is performed in the custom *davinci* engine with the 340 questions of the *QALD-9* training data. For this purpose, a *JSONL* file is produced (see Figure 4), with each question ending in a "`->`" followed by its *SPARQL* query. To avoid lengthy answers, an end-token (i.e., `\n<EOQ>\n`)) was added after each query.

```
{
    "prompt":
        "List all boardgames by GMT. ->",
    "completion":
        " PREFIX dbo: <http:\/\/dbpedia.org\/ontology\/>
          PREFIX res: <http:\/\/dbpedia.org\/resource\/>
          PREFIX rdfs: <http:\/\/www.w3.org\/2000\/01\/rdf-schema#>
          SELECT ?uri WHERE {
              ?uri dbo:publisher res:GMT_Games
          }\n<EOQ>\n"
}
```

Figure 4 First row of the *JSONL* file containing the pre-processed dataset.

4.3 Hyperparameters

The following hyperparameters were set for all experiments: `temperature`, `max_tokens`, `top_p`, `frequency_penalty`, `presence_penalty`. The `temperature` controls the randomness of the string completion and is set to 0 to avoid randomness. The maximum number of `tokens` is `max_tokens` and is set to twice the length of the expected answer L_{EA} from *QALD*.

The `top_p` controls diversity via nucleus sampling (e.g., 0.5 means that half of all likelihood-weighted options are considered). Finally, `frequency_penalty` and `presence_penalty` are both set to 0. The former penalises new tokens based on their existing frequency in the text so far, and the latter penalises new tokens based on whether they have appeared in the text so far.

4.4 Direct Answer

The final approach does not involve *SPARQL* generation. It consists of making the *NL* question directly to the model, with the gear of finally comparing the generated answer with the query answers in *QALD*. Due to cost limitations, only *davinci-002* was used for this.

To evaluate the model's performance, and since the answers in the dataset are frequently *URI*s in *DBpedia*, the first step was to convert *URI* to text. For this, *DBpedia* is queried for a textual representation of the resource through the value of its `rdfs:label` or, if not available, of its `foaf:name`. If none is available, the *URI* is parsed, and its final part (i.e., past the last `/`) is extracted, with `_` replaced by white spaces. When the answer is a list of *URI*s, the previous steps are applied to each *URI*, and the results are joined in a single string, separated by white spaces.

The last step of this process is to normalise answers from the dataset and by *GPT-3*. This involves converting numbers and dates to a textual format, removing punctuation and stopwords[7], converting special characters (e.g., accents, cedillas) to *ASCII*, and lowercasing everything. The result of this process is illustrated in Figure 5.

```
-- Original Answer
08/01/2020 was a good day to visit Monção, Portugal, with my 2 dogs.

-- Normalised Answer
01 august 2020 good day visit moncao portugal two dogs
```

Figure 5 Answer Normalisation.

```
PREFIX res: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>
SELECT DISTINCT ?uri WHERE {
    res:Salt_Lake_City <http://dbpedia.org/ontology/timeZone> ?uri
}
```

Figure 6 Expected *SPARQL* query for the question "*What is the time zone of Salt Lake City?*".

For evaluation against the answers in *QALD*, the same normalisation was performed on the results retrieved by the generated *SPARQL*.

4.5 Metrics

Two approaches were adopted for evaluating generated *SPARQL* queries:

(i) comparison with the gold *SPARQL* queries in the dataset;

(ii) comparison of their answers, i.e., results retrieved from *DBpedia* by the generated query with the actual answer in the dataset.

[7] We considered the list of English stopwords from NLTK, `https://www.nltk.org/` (accessed on 20/03/23)

Answers generated by *GPT-3*, when asked the question directly, were also compared with the answers in *QALD-9*.

Since *EM* would be too strict, as in related work, we rely on *BLEU*[8] for comparing how close two queries are. This has in mind that some queries might be invalid due to simple syntactic errors that a human could quickly fix. *BLEU* is typically used in Machine Translation, in our case, of English to *SPARQL*. It compares the gold answer with the generated one and measures the weighted geometric average of all modified n-grams precision (p_n). Different values of n originate different variants of *BLEU*, such as *BLEU*-1 for unigrams and *BLEU*-2 for bigrams. We report on *BLEU*-1, *BLEU*-2, and a combined measure, Sentence-*BLEU*, which averages *BLEU*-1, 2, 3 and 4.

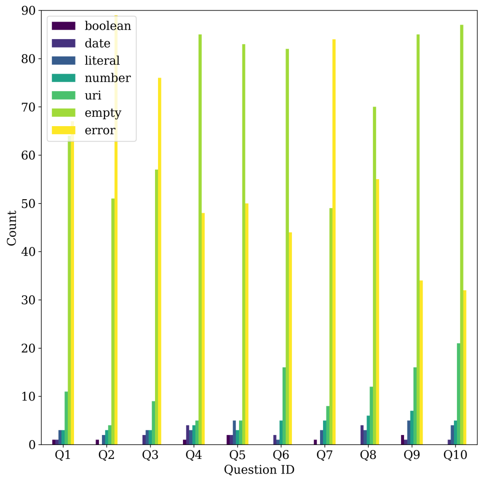
As in the *QALD* challenge, typical *IR* measures, i.e., precision, recall and F1-score, are computed for comparing generated and retrieved answers with the gold answers. When used for assessing the direct answers, their normalisation is performed (see Section 4.4).

5 Evaluation

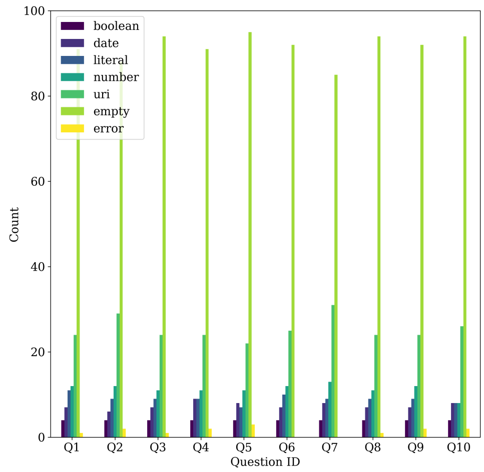
After analysing the type of the generated queries, this section reports on the evaluation of *SPARQL* queries generated with the three methods, against the gold queries, followed by the evaluation of their results, and of the direct questions, against the gold results.

5.1 Analysis of Generated Query Types

The proportion of valid queries is an initial insight into how *GPT-3* can be used for *SPARQL* generation. Figure 7a shows a distribution of answer types, including invalid queries and empty answers, for queries generated when *QALD* test questions are concatenated to the prompts in Table 1. Results are similar for each engine, so we present them only for *davinci-002*. There are many invalid queries (yellow bar) with zero-shot, but most errors



(a) zero-shot.

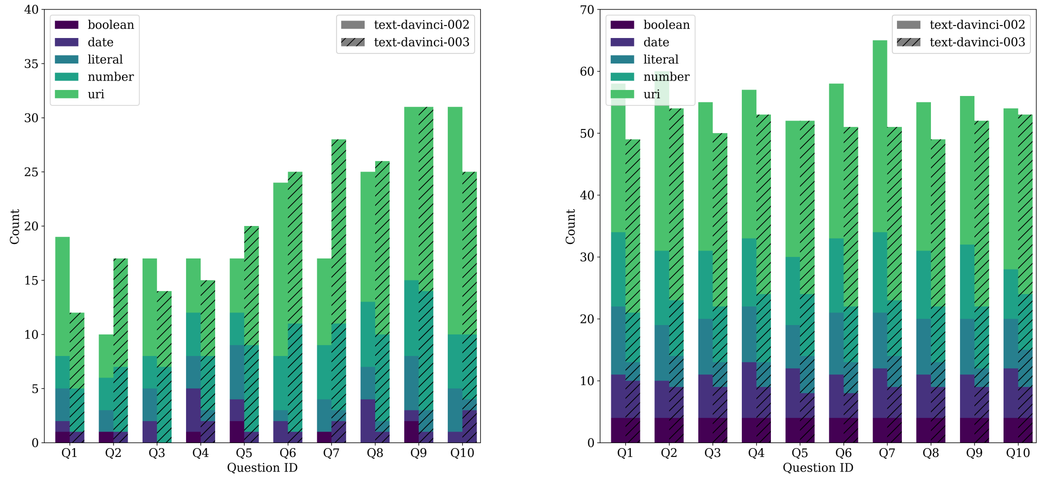


(b) few-shot.

Figure 7 All *SPARQL* queries generated by *text-davinci-002*.

are fixed in the few-shot scenario. However, the increase in valid answer types comes at the cost of an increase in empty answers.

[8] We have used the *BLEU* implementation of NLTK, `https://www.nltk.org/_modules/nltk/translate/bleu_score` (accessed on 10/05/23)



(a) zero-shot. **(b)** few-shot.

Figure 8 Valid *SPARQL* queries generated by each engine.

Figure 8 does the same analysis after removing empty and error queries. For each prompt, two columns are presented, one for each engine. For zero and few-shot, more valid queries can be generated with *davinci-002* than with *davinci-003*.

5.2 Evaluation of Generated SPARQL

BLEU-1, 2, 3 and 4, as well as Sentence-*BLEU* were computed for the *SPARQL* generated for the *QALD* test questions with each prompt, approach and engine, as well as with the fine-tuned model. Table 2 reports on the average *BLEU*-1, *BLEU*-2, and Sentence-*BLEU*. Besides considering the full gold query, we also report the scores when the declaration of prefixes is ignored not only in the gold query but also in the generated one. This has in mind that these declarations are not always necessary. For instance, standard prefixes like `rdf`, or `dbp` and `dbr` for *DBpedia*, are often preloaded by *SPARQL* endpoints.

When considering prefixes, differences between *davinci-002* and *davinci-003* and between different prompts are minimal. Referring *DBpedia* specifically on the prompt also seems to make no difference. When prefix declarations are ignored, performance improves. In this case, the few-shot approach performs better than the zero-shot. Still, low *BLEU*-2 and Sentence-*BLEU* scores suggest that generated queries lack consistency and that *GPT-3* is not suitable for *SPARQL* generation, neither in a zero nor in a few-shot approach.

Despite being far from perfect, the best performance for every metric is achieved by the fine-tuned model. To some extent, this was expected, because this approach was trained in more data (340 examples), and confirms the benefits of fine-tuning.

5.3 Evaluation of SPARQL Results

A different perspective is given by running the generated queries in *DBpedia* and comparing the obtained results with the gold results in *QALD-9*. This is not immune to changes in *DBpedia* because, due to hardware limitations, we queried its most recent version through its public *SPARQL* endpoint, and not the source dump of the dataset, and we know that some answers are only valid in a specific time frame (e.g., *Who is the mayor of Berlin?*).

Table 2 BLEU scores for different prompts and engines.

Engine	Shots	Prompt	With Prefix			Without Prefix		
			BLEU-1	*BLEU-2*	*Sent-BLEU*	*BLEU-1*	*BLEU-2*	*Sent-BLEU*
davinci-002	0	Q1	0.255	0.089	0.024	0.284	0.086	0.010
		Q2	0.238	0.080	0.020	0.256	0.075	0.007
		Q3	0.245	0.084	0.023	0.269	0.080	0.008
		Q4	0.254	0.087	0.024	0.276	0.082	0.007
		Q5	0.259	0.088	0.024	0.283	0.084	0.007
		Q6	0.259	0.087	0.022	0.288	0.085	0.007
		Q7	0.254	0.085	0.021	0.279	0.082	0.007
		Q8	0.254	0.084	0.020	0.283	0.083	0.007
		Q9	0.257	0.086	0.021	0.286	0.084	0.007
		Q10	0.260	0.087	0.022	0.288	0.085	0.007
	5	Q1	0.340	0.179	0.067	0.442	0.227	0.078
		Q2	0.340	0.178	0.064	0.441	0.224	0.074
		Q3	0.341	0.180	0.066	0.445	0.229	0.078
		Q4	0.340	0.178	0.064	0.443	0.226	0.075
		Q5	0.342	0.178	0.064	0.441	0.224	0.073
		Q6	0.341	0.179	0.065	0.444	0.226	0.075
		Q7	0.341	0.179	0.065	0.445	0.227	0.076
		Q8	0.341	0.178	0.065	0.445	0.227	0.076
		Q9	0.340	0.177	0.065	0.444	0.226	0.076
		Q10	0.341	0.177	0.064	0.444	0.226	0.075
davinci-003	0	Q1	0.254	0.076	0.007	0.305	0.089	0.007
		Q2	0.257	0.076	0.007	0.304	0.086	0.005
		Q3	0.252	0.074	0.005	0.303	0.088	0.005
		Q4	0.253	0.075	0.006	0.303	0.087	0.006
		Q5	0.258	0.076	0.006	0.306	0.087	0.006
		Q6	0.256	0.075	0.005	0.306	0.088	0.006
		Q7	0.257	0.074	0.005	0.307	0.087	0.006
		Q8	0.256	0.074	0.005	0.308	0.088	0.006
		Q9	0.257	0.075	0.005	0.310	0.088	0.006
		Q10	0.259	0.074	0.005	0.313	0.089	0.006
	5	Q1	0.361	0.198	0.094	0.481	0.262	0.121
		Q2	0.351	0.183	0.069	0.467	0.242	0.090
		Q3	0.349	0.181	0.067	0.465	0.240	0.088
		Q4	0.348	0.178	0.062	0.462	0.234	0.080
		Q5	0.347	0.176	0.059	0.460	0.231	0.076
		Q6	0.348	0.177	0.060	0.462	0.234	0.078
		Q7	0.347	0.176	0.058	0.461	0.232	0.075
		Q8	0.348	0.177	0.059	0.462	0.233	0.077
		Q9	0.348	0.176	0.059	0.461	0.231	0.075
		Q10	0.348	0.177	0.060	0.462	0.232	0.077
davinci-ft	-	-	**0.473**	**0.313**	**0.245**	**0.519**	**0.345**	**0.261**

Table 3 reports the evaluation of the results of the queries generated by each engine, approach, and prompt. Here, recall and precision are both low, thus leading to low F1-scores. Of course, the high number of invalid queries, considered empty, has a negative impact on the results. Towards an alternative comparison with the answers directly generated (Section 5.4), which might include unexpected results, *BLEU* metrics, this time between natural language answers, were also computed, but do not bring much more to the table.

Performance is again better for the few-shot approach than for the zero-shot. Yet, surprisingly, the few-shot compares well to the fine-tuned model. In fact, even if by an insignificant margin, the best F1-score is achieved by the few-shot approach, in *davinci-003*, using prompt Q2.

Table 3 Scores of answers retrieved by generated queries or generated directly by the model.

Engine	Shots	Prompt	Precision	Recall	F1-Score	BLEU-Score	
						BLEU-1	*BLEU-2*
davinci-002	0	Q1	0.028	0.043	0.034	0.022	0.000
		Q2	0.008	0.010	0.009	0.008	0.000
		Q3	0.033	0.035	0.034	0.026	0.000
		Q4	0.027	0.024	0.025	0.020	0.000
		Q5	0.011	0.018	0.014	0.011	0.000
		Q6	0.038	0.039	0.038	0.031	0.007
		Q7	0.023	0.028	0.025	0.016	0.000
		Q8	0.064	0.072	0.068	0.057	0.007
		Q9	0.055	0.062	0.058	0.049	0.000
		Q10	0.063	0.068	0.065	0.050	0.000
	5	Q1	0.113	**0.142**	0.126	0.110	0.000
		Q2	0.116	0.130	0.122	0.113	0.001
		Q3	0.100	0.125	0.111	0.100	0.000
		Q4	0.105	0.123	0.113	0.102	0.001
		Q5	0.094	0.119	0.105	0.093	0.001
		Q6	0.118	0.141	0.128	0.112	0.000
		Q7	0.113	0.134	0.123	0.108	0.001
		Q8	0.104	0.133	0.117	0.104	0.000
		Q9	0.098	0.124	0.109	0.095	0.001
		Q10	0.096	0.109	0.102	0.093	0.001
davinci-003	0	Q1	0.026	0.028	0.027	0.019	0.000
		Q2	0.032	0.040	0.035	0.025	0.000
		Q3	0.026	0.029	0.028	0.019	0.000
		Q4	0.027	0.029	0.028	0.021	0.000
		Q5	0.043	0.053	0.048	0.036	0.000
		Q6	0.027	0.024	0.026	0.015	0.000
		Q7	0.054	0.053	0.054	0.043	0.000
		Q8	0.026	0.024	0.025	0.015	0.000
		Q9	0.077	0.075	0.076	0.061	0.000
		Q10	0.039	0.043	0.041	0.033	0.007
	5	Q1	0.104	0.121	0.112	0.096	0.007
		Q2	0.124	0.139	**0.131**	0.112	0.007
		Q3	0.115	0.131	0.122	0.107	0.007
		Q4	0.120	0.134	0.121	0.103	0.007
		Q5	0.103	0.119	0.110	0.098	0.007
		Q6	0.112	0.119	0.115	0.104	0.007
		Q7	0.114	0.130	0.122	0.104	0.007
		Q8	0.104	0.121	0.112	0.096	0.007
		Q9	0.092	0.107	0.099	0.085	0.007
		Q10	0.114	0.130	0.122	0.104	0.007
davinci-ft	-	-	**0.126**	0.132	0.129	**0.115**	0.008
davinci-002-dir	-	-	**0.317**	**0.419**	**0.361**	**0.240**	**0.124**

Out of curiosity, considering only valid and non-empty queries, the best F1-score is 0.40, specifically with the zero-shot approach in *davinci-002*, using prompt Q8. This is, however, not comparable among approaches, because such queries and their number vary.

5.4 Evaluation of Direct Answers

In addition to *SPARQL* generation, *GPT-3* was used for answering the *QALD*-9 test questions directly, in *NL*. The generated answers were then compared with the gold answers, and performance is included the last line of Table 3. Though not especially high, all the scores are greater than for any other approach. This suggests that, if the query is not important, it is preferable to avoid the extra step of query generation.

6 Discussion

Objectively, *GPT-3* performed poorly for both *SPARQL* generation and *QA*. Yet, if we look at the official results in the *QALD*-9 challenge [22], the 0.131 F1 would rank the best few-shot approach third, which also shows that this is a challenging task. On top of that, asking the questions directly to *GPT-3* would rank it first (0.361 vs 0.298 F1).

However, we note that, since a portion of the entries were discarded from the dataset (see Section 3), these scores are not directly compared to the official. Moreover, official scores date from 2018 and, since then, there has been much progress in text to text generation. In fact, the very recent work [18] that uses *GPT-3* reports on a F1 of 46%. Besides using the full dataset, they consider its 13 languages, not just English, and we do not about some details of the experiment, e.g., whether they applied any kind of pre and post-processing, or how they handled answers that have changed.

In any case, our results suggest that it is preferable to use *GPT-3* directly. And asking a question in *NL* is indeed straightforward, while queries must comply with a formal language, to be made to a *KB* as *DBpedia*. If they are invalid, they will simply not be accepted. Moreover, when it comes to comparing queries, it is usual that the generated query will be different than the one in the dataset, even if slightly (e.g., name of a variable) because there are many different ways to query *DBpedia* and obtain the same results. On the other hand, queries are fixable and human-readable, and they are made to a transparent source of knowledge, represented in *RDF*, in opposition to the black-box reasoning of *GPT-3*. So, when interpretability is a requirement, using *GPT-3* directly is not a solution.

Despite slight improvements in the few-shot scenario with *davinci-003*, differences between *davinci-002* and *davinci-003* engines are minimal. However, we note that *davinci-002* insists on generating *Wikidata* queries, instead of *DBpedia*, which ends up producing erroneous queries. This was also why *DBpedia* was specified in half of the prompts but, apparently, it made no noticeable difference on the quality of the generated *SPARQL*.

Fine-tuning the *davinci* engine led to improvements in the generated *SPARQL*. This was somewhat expected because it was trained in 340 question-query pairs, whereas the few-shot approach only saw five and the zero-shot none. Performance could be possibly improved if more training examples were used, but this would have to resort to a different dataset.

Differences in *SPARQL* generation are, however, not reflected when comparing the results of the generated queries, where the performance of the few-shot approach is comparable to the fine-tuned model. This may be due to different queries that lead to similar results. However, we recall that the best-scored answers were obtained by querying *GPT-3* directly, without *SPARQL* and *DBpedia*.

7 Conclusion and Future Work

GPT-3 has been used for many tasks, and *SPARQL* generation has been attempted with different approaches. Yet, until recently, *GPT-3* had not been explored for the automatic generation of *SPARQL* queries.

Ideally, this would combine the best of text generation with *KB*-*QA*. Current text generation models are known for their capacity of adapting to many tasks, taking advantage of zero and few-shot learning. However, their inference is not transparent for humans, which hinders their application to critical domains. On the other hand, both *SPARQL* queries and *LD* can be easily scrutinised.

We tested different *GPT-3* engines in zero and few-shot learning with ten different prompts. We also fine-tuned a model for *SPARQL* generation. Results were analysed from the perspective of valid queries produced and their resemblance with correct ones. The evaluation was complemented by scoring the results of running the generated queries and comparing them to those obtained when the original question is asked directly to the model, which also generates an answer in *NL*.

Briefly, in the zero-shot scenario, *GPT-3* generates many invalid queries. Performance increases with the five-shot approach, and even more with fine-tuning, but *BLEU* scores show that generated queries are still far from the gold ones. On the other hand, the results of queries by the few-shot approach are comparable to those of queries by the fine-tuned model. Nevertheless, answers generated directly by the model are the best, even if still far from the gold answers.

Overall, the results were poor and show that we were far from the aforementioned ideal combination. Yet, we learned about the performance of *GPT-3* for this specific task and reported on insights that will hopefully open the door to further exploration of zero and few-shot learning for *SPARQL* generation, using recent *LLM*s. This work was developed as a course mini-project at the University of Coimbra, and some experiments were left to do due to lack of time and resources. For instance, the reported performance could possibly be improved with simple changes, such as: considering the type of question when selecting the training examples for the few-shot approach; as others have done [17, 19], pre-processing SPARQL queries for making them closer to *NL* (e.g., replace $?x$ variables or brackets, respectively by tokens var_x or bra_left); and, most of all, using the correct DBPedia version. The fine-tuned model could be further improved if more training data is used, but this would have to resort to larger datasets (e.g., *LC-QuAD* [20]). Moreover, there are many *LLM*s left to explore, e.g., *OPT-175B* [27], or the recent *GPT-4* [14].

References

1 Ankush Agarwal, Raj Gite, Shreya Laddha, Pushpak Bhattacharyya, Satyanarayan Kar, Asif Ekbal, Prabhjit Thind, Rajesh Zele, and Ravi Shankar. Knowledge Graph–Deep Learning: A Case Study in Question Answering in Aviation Safety Domain. *arXiv preprint arXiv:2205.15952*, 2022. doi:10.48550/arXiv.2205.15952.

2 Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, LNCS, pages 722–735. Springer, 2007. doi:10.1007/978-3-540-76298-0_52.

3 Mahdi Bakhshi, Mohammadali Nematbakhsh, Mehran Mohsenzadeh, and Amir Masoud Rahmani. SParseQA: Sequential word reordering and parsing for answering complex natural language questions over knowledge graphs. *Knowledge-Based Systems*, 235:107626, 2022. doi:10.1016/j.knosys.2021.107626.

4 Petr Baudiš. YodaQA: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*, pages 1156–1165, 2015.

5 Romain Beaumont, Brigitte Grau, and Anne-Laure Ligozat. SemGraphQA@ QALD5: LIMSI participation at QALD5@ CLEF. In *Working Notes of CLEF 2015 – Conference and Labs of the Evaluation Forum*, 2015.

6 Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from Question-Answer pairs. In *Proceedings of 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, 2013.

7 Tom et al. Brown. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

8 Yongrui Chen, Huiying Li, and Zejian Xu. Convolutional Neural Network-Based Question Answering Over Knowledge Base with Type Constraint. In *China Conference on Knowledge Graph and Semantic Computing*, pages 28–39. Springer, 2018. doi:10.1007/978-981-13-3146-6_3.

9 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota, 2019. ACL.

10 Dennis Diefenbach, Andreas Both, Kamal Singh, and Pierre Maret. Towards a question answering system over the Semantic Web. *Semantic Web*, 11(3):421–439, 2020. doi:10.3233/SW-190343.

11 Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):824–837, 2018. Conference Name: IEEE Transactions on Knowledge and Data Engineering. doi:10.1109/TKDE.2017.2766634.

12 Hai Jin, Yi Luo, Chenjing Gao, Xunzhu Tang, and Pingpeng Yuan. ComQA: Question Answering Over Knowledge Base via Semantic Matching. *IEEE Access*, 7:75235–75246, 2019. Conference Name: IEEE Access. doi:10.1109/ACCESS.2019.2918675.

13 Endri Kacupaj, Hamid Zafar, Jens Lehmann, and Maria Maleshkova. VQuAnDa: Verbalization QUestion ANswering DAtaset. In Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez, editors, *The Semantic Web*, pages 531–547, Cham, 2020. Springer International Publishing.

14 OpenAI. GPT-4 Technical Report, March 2023. arXiv:2303.08774 [cs]. doi:10.48550/arXiv.2303.08774.

15 Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, page 311, Philadelphia, Pennsylvania, 2002. ACL. doi:10.3115/1073083.1073135.

16 Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online, 2020. ACL. doi:10.18653/v1/2020.emnlp-main.437.

17 Tommaso Soru, Edgard Marx, Diego Moussallem, Gustavo Publio, Andre Valdestilhas, Diego Esteves, and Ciro Baron Neto. Sparql as a foreign language. In *Proceedings of the Posters and Demos Track of the 13th International Conference on Semantic Systems - SEMANTiCS2017*, 2017.

18 Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. Evaluation of chatgpt as a question answering system for answering complex questions. *arXiv preprint arXiv:2303.07992*, 2023.

19 Hieu Tran, Long Phan, James Anibal, Binh T Nguyen, and Truong-Son Nguyen. SPBERT: an Efficient Pre-training BERT on SPARQL Queries for Question Answering over Knowledge Graphs. In *Neural Information Processing: 28th International Conference, ICONIP 2021, Proceedings*, pages 512–523. Springer, 2021.

20 Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs. In Claudia d'Amato, Miriam Fernandez, Valentina Tamma, Freddy Lecue, Philippe Cudré-Mauroux, Juan Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web – ISWC 2017*, LNCS, pages 210–218, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-68204-4_22.

21 Christina Unger, Corina Forascu, Vanessa Lopez, Axel-Cyrille Ngonga, Elena Cabrio, Philipp Cimiano, and Sebastian Walter. Question Answering over Linked Data (QALD-5). In *Working Notes of CLEF 2015 – Conference and Labs of the Evaluation Forum*, volume 1391 of *CEUR Workshop Proceedings*, page 10. CEUR-WS.org, 2015.

22 Ricardo Usbeck, Ria Hari Gusmita, Axel-Cyrille Ngonga Ngomo, and Muhammad Saleem. 9th Challenge on Question Answering over Linked Data (QALD-9). *Language*, 7(1):58–64, 2018.

23 Mark D. Wilkinson et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018, 2016. Number: 1 Publisher: Nature Publishing Group. doi:10.1038/sdata.2016.18.

24 Kun Xu, Sheng Zhang, Yansong Feng, and Dongyan Zhao. Answering Natural Language Questions via Phrasal Semantic Parsing. In Chengqing Zong, Jian-Yun Nie, Dongyan Zhao, and Yansong Feng, editors, *Natural Language Processing and Chinese Computing*, Communications in Computer and Information Science, pages 333–344. Springer, 2014. doi:10.1007/978-3-662-45924-9_30.

25 Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Yumao Lu, Zicheng Liu, and Lijuan Wang. An Empirical Study of GPT-3 for Few-Shot Knowledge-Based VQA. *Proceedings of AAAI Conference on Artificial Intelligence*, 36(3):3081–3089, 2022. doi:10.1609/aaai.v36i3.20215.

26 Xiaoyu Yin, Dagmar Gromann, and Sebastian Rudolph. Neural machine translating from natural language to SPARQL. *Future Generation Computer Systems*, 117:510–519, 2021. doi:10.1016/j.future.2020.12.013.

27 Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open Pre-trained Transformer Language Models, 2022. arXiv:2205.01068 [cs]. doi:10.48550/arXiv.2205.01068.

28 Weiguo Zheng and Mei Zhang. Question Answering over Knowledge Graphs via Structural Query Patterns, 2019. arXiv:1910.09760 [cs]. doi:10.48550/arXiv.1910.09760.