

# Assessing SPARQL Capabilities of Large Language Models

Lars-Peter Meyer<sup>1,2,\*</sup>, Johannes Frey<sup>1,2</sup>, Felix Brei<sup>1</sup> and Natanael Arndt<sup>1</sup>

<sup>1</sup>*Institute for Applied Informatics at Leipzig University, Goerdelerring 9, 04109 Leipzig, Germany*

<sup>2</sup>*Leipzig University, Germany*

## Abstract

The integration of Large Language Models (LLMs) with Knowledge Graphs (KGs) offers significant synergistic potential for knowledge-driven applications. One possible integration is the interpretation and generation of formal languages, such as those used in the Semantic Web, with SPARQL being a core technology for accessing KGs. In this paper, we focus on measuring out-of-the box capabilities of LLMs to work with SPARQL and more specifically with SPARQL SELECT queries applying a quantitative approach.

We implemented various benchmarking tasks in the LLM-KG-Bench framework for automated execution and evaluation with several LLMs. The tasks assess capabilities along the dimensions of syntax, semantic read, semantic create, and the role of knowledge graph prompt inclusion.

With this new benchmarking tasks, we evaluated a selection of GPT, Gemini, and Claude models. Our findings indicate that working with SPARQL SELECT queries is still challenging for LLMs and heavily depends on the specific LLM as well as the complexity of the task. While fixing basic syntax errors seems to pose no problems for the best of the current LLMs evaluated, creating semantically correct SPARQL SELECT queries is difficult in several cases.

## Keywords

LLM benchmarking, SPARQL, LLMs for Knowledge Graphs, RDF, LLM, Knowledge Graph

## 1. Introduction

The combination of Large Language Models (LLMs) and Knowledge Graphs (KGs) is still gaining more traction as the rapidly growing number of research articles<sup>1</sup> and the inclusion of LLMs in several KG related conference calls show. This includes different combinations [1] of LLMs and KGs, especially LLMs supported by KGs and LLMs supporting the work with KGs. An important interface for both combinations is the SPARQL query language as the structured retrieval and update interface for RDF KGs [2].

With the fast evolving list of LLMs available, the need for automated benchmarks increases. LLM performances is evaluated by BigBench [3], on the Open LLM Leaderboard [4], and the Chatbot-Arena [5], but these are very general and do not focus on KGs or SPARQL. In existing

---


*NLP4KGC: 3rd International Workshop on Natural Language Processing for Knowledge Graph Creation, September 17, 2024, Amsterdam, Netherlands*

\*Corresponding author.

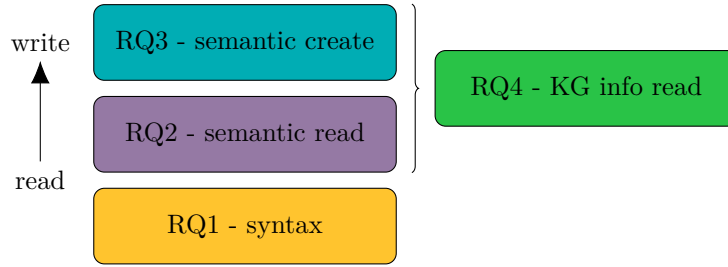
✉ [lpmeyer@infai.org](mailto:lpmeyer@infai.org) (L. Meyer); [frey@informatik.uni-leipzig.de](mailto:frey@informatik.uni-leipzig.de) (J. Frey); [brei@infai.org](mailto:brei@infai.org) (F. Brei)

🆔 0000-0001-5260-5181 (L. Meyer); 0000-0003-3127-0815 (J. Frey); 0009-0008-5245-6655 (F. Brei);

0000-0002-8130-8677 (N. Arndt)

 © 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

<sup>1</sup>list of articles combining KG+LLM: <https://github.com/zjukg/KG-LLM-Papers>



**Figure 1:** An overview on the relation of the research questions.

work, KG related benchmarking is done specifically on the Turtle format [6] or often as discussion and comparison of very specific solutions. LLM benchmarking related to SPARQL was done in a small scale by Meyer et al. [7] and by Kovriguina et al. [8].

In this paper, a set of automated benchmarking tasks is presented, to assess the basic capabilities of LLMs to deal with *SPARQL SELECT* queries without special architectures or fine tuning. We reuse existing work on Knowledge Graph Question Answering (KGQA) datasets, taking lists of question SPARQL pairs for given KGs. The benchmarking tasks are implemented in the LLM-KG-Bench framework<sup>2</sup>[9]. The tasks are executed on selected LLMs and the results are presented and discussed.

### 1.1. Research Questions

In this paper, we consider the following research questions.

**RQ 1.** *Can LLMs follow the syntactic rules of SPARQL SELECT queries?*

**RQ 2.** *Can LLMs parse the semantics of SPARQL SELECT queries and act accordingly?*

**RQ 3.** *Can LLMs write semantically correct SPARQL SELECT queries for a given question and KG, i.e. a query that yields the expected answer?*

**RQ 4.** *Does the KG presentation have an influence on the ability to write SPARQL SELECT queries?*

The research questions depend on each other and build on the previous questions as depicted in fig. 1. RQ1 focusses on the syntax, while RQ2 and RQ3 aim at semantics. RQ2 covers the interpretation and RQ3 the creation of said queries. RQ4 examines the influence of the way knowledge is presented on the results of RQ2 and RQ3.

<sup>2</sup>repository: <https://github.com/AKSW/LLM-KG-Bench>

## 1.2. Contribution

The contribution of the paper is both an evaluation of current LLMs as well as the extended evaluation framework itself. Our evaluation and framework is comprised of tasks that are executed in various configurations. In section 3.2 we introduce the tasks. The main task types are *SPARQL Syntax Fixing (SSF)*, *Text to SPARQL translation (T2S)*, *SPARQL to Answer (S2A)* and *Text to Answer (T2A)*. These tasks evaluate relevant aspects to cover the dimensions along which the capabilities (see section 3.4) of the models are assessed according to our research questions:

- Task type SSF focuses on working with the syntax of *SPARQL SELECT* queries, giving answers for RQ1.
- Task type S2A is asking to work according to semantics of *SPARQL SELECT* queries, helping with RQ2.
- In task type T2S the LLM should create syntactically and semantically correct *SPARQL SELECT* queries, giving us answers to RQ1, RQ3 and RQ4.
- For answering task type T2A the LLM needs to get information from the KG, thus the results are relevant for RQ4.

The purpose of RQ4 is to understand the influence of the knowledge graph’s syntax (Turtle, JSON-LD), the representation of the data (numerical identifiers, human readable identifiers), and the knowledge graph prompt inclusion (schema, size of the sub graph, and selection) in the answers to RQ2 and RQ3. The variants of execution with Turtle and JSON-LD as well as the selection of datasets (KG info type) and their subsets pay in on RQ4.

The remainder of the paper is structured as follows: In section 2 an overview on the related work is given and the paper is located with its contribution to the bigger picture. Section 3 presents the kind of tasks that were executed as part of our benchmark and which datasets were used. In section 4 we do an extensive evaluation of the obtained results and the conclusions that can be drawn. Section 5 summarizes the findings and gives an overview on the directions that our research will head next.

## 2. Related Work

Rangel et al. [10] propose a methodology for fine-tuning OpenLLaMA to generate SPARQL queries for question answering over life science knowledge graphs using data augmentation techniques, such as providing meaningful variable names and inline comments, improving the performance of the model in generating accurate SPARQL queries. Bustamante and Takeda [11] aim at improving the creation of SPARQL queries based on natural language questions. The authors use a GPT model to identify which property of the Text2SPARQL task is the hardest to solve, in order to select appropriate countermeasures. Avila et al. [12] evaluate the ability to answer natural language questions with ChatGPT on KGs. With the Auto-KGQAGPT approach the authors further investigate the ability to translate natural language question to SPARQL queries based on a prompt inclusion of KG fragments.

Li et al. [13] address the challenge of declined quality of results in real-world scenarios where high-quality annotated data is insufficient. To target this challenge the authors present the FlexKBQA framework which employs templated SPARQL queries that are translated into natural language questions using LLMs to generate synthetic training data. This synthetic data allows to fine-tune a lightweight model for SPARQL generation along with further self-guided training on real queries to address a distribution shift between synthetic and real queries.

Diallo et al. [14] give a comprehensive overview and performs a comparison of pre-trained LMs (PLMs), non-pre-trained LMs (NPLMs), and LLMs, testing various fine-tuning methods using LLMs. The error analysis of the models results in the finding, that the primary source of errors are incorrect URIs in SPARQL queries e.g. due to hallucination. Hirigoyen et al. [15] have found though that the hallucination can be prevented by using a copy mechanism.

The SPARQLGEN approach by Kovriguina et al. [8] is a one-shot generative approach to generate SPARQL queries using LLMs. It includes the relevant context in a single prompt, i.e. the question, an RDF sub graph with the relevant information, and an example of a SPARQL query and a question. Pliukhin et al. [16] present an approach for SPARQL query generation on the scholarly knowledge graph ORKG. Its setup is similar to SPARQLGEN but utilizes an advanced sub graph extraction. Zahera et al. [17] went on with leveraging chain of thought prompting and utilizes the GERBIL QA system<sup>3</sup>[18]. Lehmann et al. [19] suggest the usage of Controlled Natural Language as a human interface since it is closer to natural language. The Controlled Natural Language can then be unambiguously translated into a formal language such as SPARQL. They come to the conclusion that this approach substantially reduces the training data requirements.

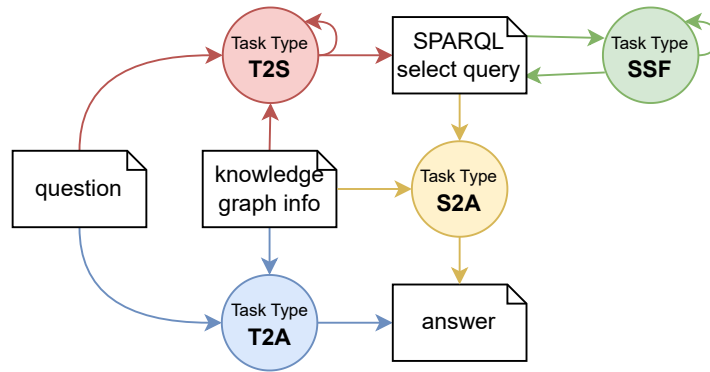
Several datasets of complex questions over knowledge graphs are available. Diefenbach et al. [20] present two datasets for training and benchmarking question answering systems using Wikidata. One is a translation of the SimpleQuestions dataset (cf. Bordes et al. [21]), the other is based on logs and user feedback. LC-QuAD 2.0 by Dubey et al. [22] is an advancement of the Large-Scale Complex Question Answering Dataset (LC-QuAD) [23]. It is compatible with both, the Wikidata and the DBpedia knowledge graphs. It contains “30,000 questions, their paraphrases and their corresponding SPARQL queries”.

Banerjee et al. [24] evaluate different approaches to incorporate LLMs into a common KG workflow. They use language models that can be fine tuned on consumer hardware today, while we focus on very large models that will be used interactively and as is, without any fine-tuning.

Frey et al. [25] show the long term value of automated benchmarking with LLM-KG-Bench. Several version iterations of LLMs were compared with focus on their RDF Turtle language capabilities and the answers were conserved in a time capsule for future (re)evaluation. Hofer et al. [26] showed in an LLM-driven RML mapping (in Turtle format) generation experiment in alignment with [25, 6], that syntactical errors occurred when generating Turtle files, but most LLMs were capable to repair them. Given that SPARQL is based on Turtle, we designed our experiments as multi turn conversations with error feedback loops.

---

<sup>3</sup>projekt page: <https://gerbil-qa.aksw.org/gerbil/>



**Figure 2:** Overview on the task types and their input and output

### 3. Experiment Setup

For automated assessment of the capabilities of LLMs to work with *SPARQL SELECT* statements we extended the LLM-KG-Bench framework. In the following subsections we outline evaluation methods, the different task types, KGs used for testing and specific tasks.

nomenclature that we used:

- A *task type* is a class of tasks that an LLM needs to solve, like SSF or T2S.
- A *task* is one specific instantiation of a task type, like T2A on LC-QuAD using JSON-LD syntax for the knowledge graph.
- *Task entry* refers to one single data point that is used during a task, for example the third question in T2A for LC-QuAD. All tasks presented here have a list of five task entries which they choose from evenly distributed.
- One *execution* of a task entry consists of all steps that are necessary to reach completion of this task entry. For example, the execution of one task entry in SSF entails the complete feedback dialog and ends with the final answer of the LLM and its evaluation.

#### 3.1. Experiment Type Selection

*SPARQL SELECT* queries are connected to a related textual question, a knowledge graph the query is designed for, and an answer. For working with *SPARQL SELECT* queries one usually has to deal with a combination of these concepts. This leads us to the four different task types with inputs and outputs as shown in fig. 2.

#### 3.2. Detailed Explanation of Task Types

##### 3.2.1. Task Type “SPARQL Syntax Fixing” (SSF)

The task SPARQL Syntax Fixing asks to fix a syntax error in a given SPARQL query. To be more precise, the LLM is given a syntactically malformed SPARQL query together with the resulting

rdflib<sup>4</sup> parsing error message. The LLM is asked to answer with just the fixed SPARQL query.

The following prompt is used with variables(`{sparql}` and `{errorMessage}`) filled in according to the specific task:

**Prompt 1:** Please correct a syntax error in the following SPARQL query for Wikidata. Assume common prefixes like `wd` or `wdt` to be defined. To support automated parsing, please answer with just a markdown fenced code block (start and end with `````) containing the sparql query, no other text.

Example for Answer format:

```
```sparql
SELECT ...
```
```

SPARQL:`{sparql}`

Error message: `{errorMessage}`

The answer presented is evaluated according to remaining syntax errors and having the expected result set when executed as described in section 3.3.1. The LLM is given the opportunity to correct the answer in a feedback dialog session with up to three tries.

### 3.2.2. Task Type “Text to SPARQL” (T2S)

The Text-to-SPARQL task type asks to create a *SPARQL SELECT* query for a KG and a textual question. Many KGQA approaches build upon this capability.

The following prompt is used with several variables(`{kgName}`, `{commonPrefixes}`, `{question}` and `{kgInfo}`) filled in according to the specific task:

**Prompt 2:** Please generate a SPARQL query for `{kgName}` and the given question. Assume common prefixes `{commonPrefixes}` to be defined. To support automated parsing, please answer with just a markdown fenced code block (start and end with `````) containing the sparql query, no other text.

Example for Answer format:

```
```sparql
SELECT ...
```
```

Question:`{question}`

---

<sup>4</sup><https://github.com/RDFLib/rdflib>

`#{KgInfo}`

The given SPARQL SELECT query is evaluated on being syntactically correct and having the same result set as expected as described in section 3.3.1. The LLM is given the opportunity to correct the answer in a feedback dialog session with up to three tries.

### 3.2.3. Task Type “SPARQL to Answer” (S2A)

The SPARQL-to-Answer task asks to interpret a given *SPARQL SELECT* query on a given KG and answer with the binding values. The LLM is asked to give one value per line and the F1 measure is used to score the result, as described in more detail in section 3.3.2.

### 3.2.4. Task Type “Text to Answer” (T2A)

This task type was added as a variation of the previous “SPARQL to Answer” task type for comparison. The task is to answer a given natural language textual question on a given KG and answer with the binding values. Expected answer structure and evaluation is the same as described above for task type S2A.

## 3.3. Evaluation Method

In this paper we are applying a quantitative approach. To support automated evaluation within in the LLM-KG-Bench framework we need to define the following evaluation methods for *SPARQL SELECT* queries as generated by task types SSF and T2S, and for answers given in task types S2A and T2A.

### 3.3.1. Evaluation of SPARQL SELECT Queries (SSF and T2S)

*SPARQL SELECT* queries are evaluated regarding having correct syntax and yielding the expected result when executed on the KG. To be more precise, all entries of the result bindings are put into a set (duplicates removed) and are then compared to the expected result set. In case of syntax errors or an empty result set the LLM is given the opportunity to correct the *SPARQL SELECT* query in a feedback dialog session. Therefore a feedback message is generated and sent back to the LLM together with the previous dialog. This feedback dialog session is continued until up to three answers (one initial and up to two additional tries for corrections) are collected or at least one *SPARQL SELECT* query yields a result.

In case of syntax errors the following prompt template was used with values filled in for `#{error}` and `#{sparql}`:

**Prompt 3:** Please try to correct your answer. Your SPARQL query has syntax errors:  
`#{error}`

SPARQL given:

```
‘ ‘ sparql
${sparql}
‘ ‘
```

In case of an empty result the following prompt was used:

**Prompt 4:** Maybe you want to think again about your answer. Your SPARQL query returns an empty result when executed.

The following scores are generated:

**answerParse**  $\in [0, 1]$ : 1 if the *SPARQL SELECT* query given can be parsed with rdflib, 0 otherwise

**{f1measure|precision|recall}**  $\in [0..1]$ : comparison of the result set given and expected.

**sparqlIris{f1measure|precision|recall}**  $\in [0..1]$ : comparison of the IRIs in the *SPARQL SELECT* query given and expected.

**sparqlIriSuffix{f1measure|precision|recall}**  $\in [0..1]$ : comparison of the last segments (the part after last # or \) of the IRIs in the *SPARQL SELECT* query given and expected.

**combined**  $= 0.2 * parse + 0.8 * f1measure \in [0..1]$ : combining syntax and semantic score, resulting e.g. in a score of 0.2 when f1 measure on result set is 0 but the *SPARQL SELECT* query parses.

The values above get prefixed with the specified iteration (**0...**, **1...**, **2...**, **last...**) of the feedback dialog session and aggregates (**mean...**, **max...**) across the whole feedback session are calculated, giving us score names like 0\_answerParse, 1\_answerParse, ... as well as mean\_f1measure or max\_precision to name a few.

### 3.3.2. Evaluation of Answers Given by Tasks S2A and T2A

In the task types S2A and T2A the LLM is asked to give the answer from the KG similar to binding values, one value per line. For comparing the given answer lines with the expected answer lines, precision, recall and f1 measure are calculated. As the LLMs answer does not always follow the expected structure perfectly we implemented a couple of scores for strict and more flexible answer parsing as well:

**exact (f1, precision, recall)**  $\in [0..1]$ : comparison on the exact lines given with expected answer entries

**trimmed (trimF1, trimPrecision, trimRecall)**  $\in [0..1]$ : comparison after leading and trailing whitespace is removed

**fixed format (fixedF1, fixedPrecision, fixedRecall)**  $\in [0..1]$ : comparison after fixing simple format errors like http instead of https, prefix 0: (often given for JSON-LD) instead of : and removal of brackets (< and >) and quotation marks ( ' and " )



**relaxed evaluation (relaxedF1, relaxedPrecision, relaxedRecall)**  $\in [0..1]$ : comparison with format fixed plus ignore case, removing default prefix : (makes instances similar to labels for semantic IRIs). In case a count is expected a list with the right length is also accepted as correct.

**combinedF1** =  $\frac{f1+trimF1+fixedF1+relaxedF1}{4} \in [0..1]$ : mean of the f1 scores above.

### 3.4. Task Aspects Covered by the Task Types

The 4 task types described above in section 3.2 are covering a list of task aspects and are relevant for the research questions. They are mapped on task types in table 1 and described in the following list:

**read SPARQL SELECT query syntax:** To work with a *SPARQL SELECT* query the syntax must be digested.

Part of task types SSF and S2A, related to RQ 1.

**create correct SPARQL SELECT query syntax:** Create or write syntactically correct *SPARQL SELECT* query syntax.

Part of task types SFF and T2S, related to RQ 1.

**read semantics of a SPARQL SELECT query:** Get the semantic meaning of a *SPARQL SELECT* query and act accordingly.

Part of task type S2A, related to RQ 2.

**create a semantically correct SPARQL SELECT query:** create a *SPARQL SELECT* query not only syntactically correct but with an appropriate semantic which yields the expected bindings.

Part of task type T2S, related to RQ 3.

**KG info read :** When dealing with a knowledge graph some kind of information on the KG structure and KG content is needed. The information on the structure could be anything from whole KG over schema to just a list of relevant or all entities and properties.

Part of task types T2S, T2A and S2A, related to RQ 4.

### 3.5. Benchmark Datasets and KGs Used for Task Implementations

There are several benchmark datasets available which contain pairs of *SPARQL SELECT* queries and textual natural language questions for specific knowledge graphs. We selected and implemented benchmark tasks for a couple of current datasets for smaller and bigger knowledge graphs. Only English textual questions were used as we focus on *SPARQL* here and not language capabilities. A total of five tuples, each consisting of a question and corresponding *SPARQL* query, was manually selected from each dataset. This allows to rerun the tasks more often to reduce the random noise in the results.

**Table 1**

Mapping of the task types to the different task aspects covered.

| task type                  | task aspects |               |               |                 |              |
|----------------------------|--------------|---------------|---------------|-----------------|--------------|
|                            | syntax read  | syntax create | semantic read | semantic create | KG info read |
| SPARQL Syntax Fixing (SSF) | x            | x             | -             | -               | -            |
| SPARQL to Answer (S2A)     | x            | -             | x             | -               | x            |
| Text to SPARQL (T2S)       | -            | x             | -             | x               | x            |
| Text to Answer (T2A)       | -            | -             | -             | -               | x            |

**Organizational dataset and KG** The smallest KG used is an organizational KG [7]. We use it here together with a corresponding dataset<sup>5</sup> of question and SPARQL pairs created by Brei et al. [27].

**Organizational Numeric** We created an additional variant of the organizational dataset and KG with numerical IRIs (first 3 digits of hash) and same questions.

**CoyPu-Mini dataset and KG** Brei et al. [27] published as well another dataset based on a small subset of the CoyPu KG<sup>6</sup>. This sub graph is small enough to fit into context size of LLMs evaluated here. We added lists of relevant IRIs and schema information.

**Beastuary dataset and KG** The Beastuary dataset and KG<sup>7</sup> was presented by Kovriguina et al. [8]. It offers for each question a relevant sub graph and a list of IRIs. We derived relevant schema information from these IRI lists.

**LC-QuAD 2.0 and Wikidata** The well known *LC-QuAD 2.0*<sup>8</sup> [22] dataset offers a long list of question and SPARQL pairs for the Wikidata SPARQL endpoint. We focused on the *paraphrased questions* of the test dataset and manually checked them before selecting 5 out of the first 20. Main reason for rejecting question was a missing or ambiguous paraphrased question. We computed lists of relevant IRIs for each question based on the reference SPARQL query. As some questions do not specify whether an IRI or label or both is expected, we configured the evaluation to consider all variations as correct there.

**SPARQL SELECT Query Syntax Errors** We took one question and SPARQL pair from LC-QuAD and derived 5 tests from it by inserting different kinds of syntax errors, one error per test.

The list of benchmark tasks created based on this resources in the *LLM-KG-Bench* framework is given in table 2.

<sup>5</sup>Repository: <https://github.com/AKSW/LMs4Text2SPARQL/tree/main/datasets>

<sup>6</sup>Project page: <https://coypu.org/>

<sup>7</sup>Repository: <https://github.com/danrd/sparqlgen>

<sup>8</sup>Dataset: [https://huggingface.co/datasets/lc\\_quad](https://huggingface.co/datasets/lc_quad)

**Table 2**

Overview on the different tasks implemented within LLM-KG-Bench framework

| task type  | dataset subset         | task                | KG Info type            | KG info format    |
|------------|------------------------|---------------------|-------------------------|-------------------|
| SSF        | Syntax-Errors          | SSF-LC-QuAD         | not needed here         | not needed here   |
| S2A        | Organizational         | S2A-Orga            | full KG                 | Turtle or JSON-LD |
| T2S        | LC-QuAD 2.0            | T2S-LC-QuAD         | IRIs + Labels           | table             |
|            | Organizational         | T2S-Orga            | full KG                 | Turtle            |
|            | Organizational-Numeric | T2S-OrgaNum         | full KG + IRIs + Labels | Turtle + table    |
|            | Beastuary              | T2S-Beast-Graph     | KG subset               | Turtle            |
|            |                        | T2S-Beast-Schema    | schema                  | Turtle            |
|            |                        | T2S-Beast-Subschema | schema subset           | Turtle            |
|            |                        | T2S-Beast-Iris      | IRIs                    | list              |
| CoyPu-Mini | T2S-CoyPu-Graph        | full KG             | Turtle or JSON-LD       |                   |
|            | T2S-Coypu-Schema       | schema              | Turtle or JSON-LD       |                   |
|            | T2S-Coypu-Iris         | IRIs                | list                    |                   |
| T2A        | Organizational         | T2A-Orga            | full KG                 | Turtle or JSON-LD |

**Table 3**

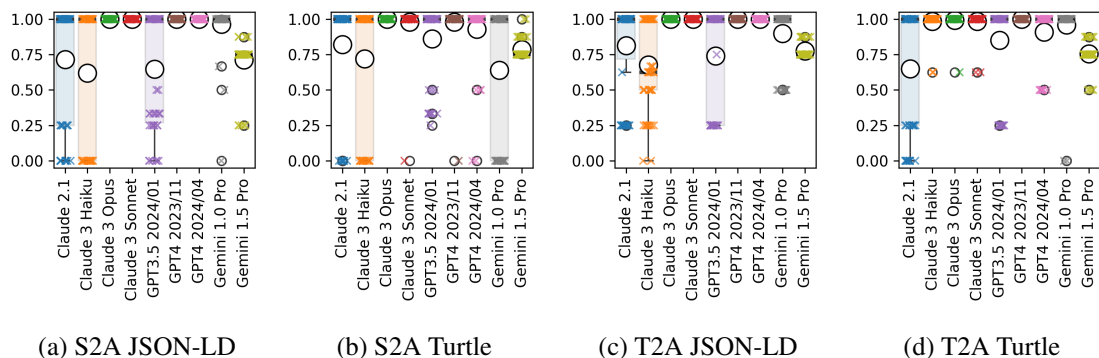
LLMs evaluated

| company   | short name       | long name                | context size |
|-----------|------------------|--------------------------|--------------|
| Anthropic | Claude 2.1       | Claude-2.1               | 200k         |
|           | Claude 3 Haiku   | claude-3-haiku-20240307  | 200k         |
|           | Claude 3 Sonnet  | claude-3-sonnet-20240229 | 200k         |
|           | Claude 3 Opus    | claude-3-opus-20240229   | 200k         |
| Google    | Gemini 1.0       | gemini-1.0-pro           |              |
|           | Gemini 1.5       | gemini-1.5-pro-latest    |              |
| OpenAI    | GPT 3.5t 2024/01 | gpt-3.5-turbo-0125       | 16k          |
|           | GPT 4t 2023/11   | gpt-4-1106-preview       | 128k         |
|           | GPT 4t 2024/04   | gpt-4-turbo-2024-04-09   | 128k         |

## 4. Results

We evaluated the LLMs from OpenAI, Anthropic and Google listed in table 3 with tasks listed in table 2. Each combination of LLM and task shown here was executed 50 times, resulting in 10 executions per task entry and LLM. All data generated is published at GitHub<sup>9</sup>. The evaluation can be reproduced by anyone by using the `--reeval` parameter of the LLM-KG-Bench framework.

<sup>9</sup>link is given at the end of paper



**Figure 3:** *combinedF1* scores for the task types SPARQL to Answer (S2A) and Text to Answer (T2A) with the Organizational KG presented in JSON-LD or Turtle syntax. The score is calculated from  $(f1 + trimF1 + fixedF1 + relaxedF1)/4$

#### 4.1. Results for SPARQL Syntax Fixing (SSF)

Fixing syntax errors in *SPARQL SELECT* queries seems to be easier for LLMs as can be seen in fig. 5a. Only 3 models (Claude 2.1, Claude 3 Haiku and GPT 3.5 2024/01) are not able to fix all errors within 3 feedback iterations, about 80% of the executions got a correct answer on first try. The most common error missed was a variable name containing a hyphen. E.g. *?foo-bar* was not detected as an invalid variable name.

#### 4.2. Results for SPARQL to Answer (S2A)

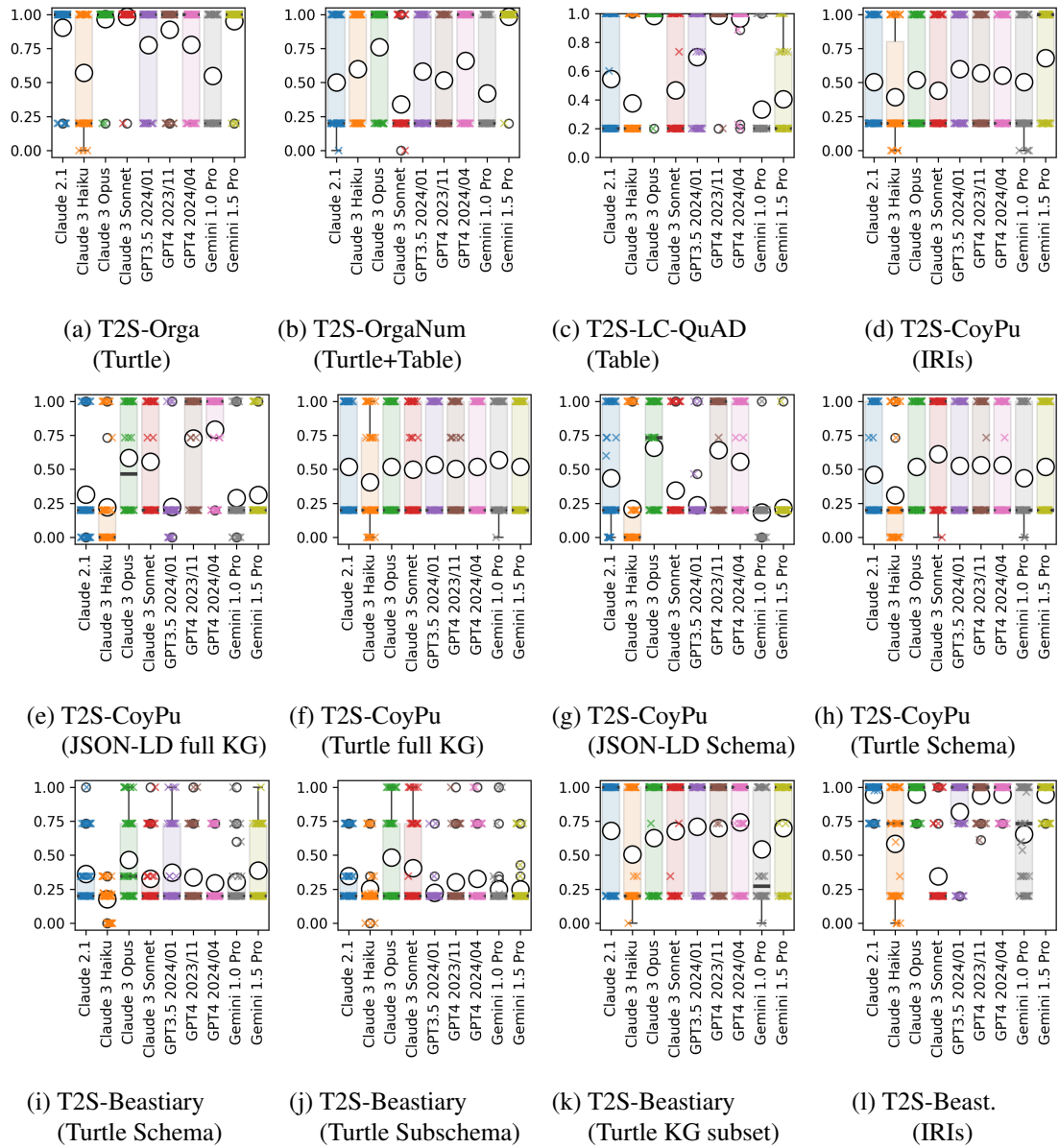
About 75% of the executions had perfect results and 90% had a *relaxedF1* score of 1. The most common formatting problems were caused by insertion of additional spaces. About 10% had just wrong answers, often for task entries that required counting. Plots for the *combinedF1* scores can be seen in figs. 3a and 3b.

#### 4.3. Results for Text to SPARQL (T2S) Organizational Graph

The state-of-the-art LLMs seem to have little problems with generating *SPARQL SELECT* queries for the organizational graph, as can be seen in fig. 4a. But when we look at the results in fig. 4b for the numerical version with a translation table, we can see a different picture, where only Gemini 1.5 scores almost perfect.

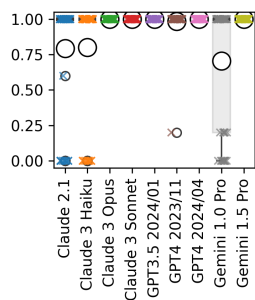
#### 4.4. Results for Text to SPARQL (T2S) LC-QuAD

In fig. 4c we can see that GPT4 in both versions and Claude 3 Opus were quite successful in generating correct *SPARQL SELECT* queries. They managed to generate syntactically correct queries in 99% of the cases on the first try, with all queries having correct syntax after the final feedback iteration. The *f1measure* score for these models increased from 0.699 for their first answer to 0.972 which means they really benefited from the provided feedback and knowledge graph information. Gemini 1.0 on the other hand reached only an *f1* score of 0.274 after the final

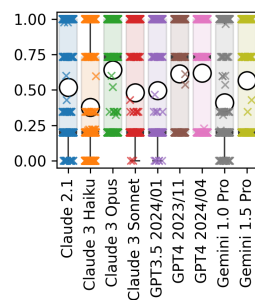


**Figure 4:**  $max\_combined$  scores for each feedback dialog of task type Text to SPARQL (T2S). The combined score equals  $0.2 * parse + 0.8 * f1measure$ , resulting e.g. in a score of 0.2 when f1 measure on result set is 0 but the SPARQL SELECT query parses.

feedback loop, indicating that it still struggles to use the provided properties correctly. However the queries were syntactically correct in almost all cases. For the other models the property structure of Wikidata seems to be just as challenging. We often see generated graph patterns with properties used in a reversed direction (domain and range swapped).



(a) SSF



(b) T2S all combined

**Figure 5:**  $max\_combined$  scores for each feedback dialog of task type SPARQL Syntax Fixing (SSF) and all task type Text to SPARQL (T2S) together. The combined score equals  $0.2 * parse + 0.8 * f1measure$ , resulting e.g. in a score of 0.2 when f1 measure on result set is 0 but the *SPARQL SELECT* query parses.

#### 4.5. Results for Text to SPARQL (T2S) CoyPu

For the CoyPu-Mini graph, the LLMs struggled more to generate syntactically correct SPARQL queries in the **first answer**. When being given the IRIs, Haiku alone is responsible for 21 of the 45 errors, with Gemini generating 8 syntactically wrong queries and GPT4 just one (2023/11 model). The rest is shared between other flavors of Claude. The situation becomes worse when we provide the full KG; only 316 of the responses for JSON-LD and 401 for Turtle contained valid SPARQL (with 450 in total for each). We get the same picture if we give the Schema information, with 341 valid SPARQL queries for JSON-LD and 397 for Turtle. However, if we provide feedback and allow LLMs to correct their mistakes, the numbers increase to 391 for the JSON-LD graph and 434 for Turtle. All model families (GPT, Gemini, Claude) have instances where even the last answer they gave contained no valid SPARQL, so this is a phenomenon that can happen with each LLM. Looking at the  $last\_f1measure$  score (the final answer) of each LLM we found that the global average is about 0.3 for JSON-LD schema information and 0.4 for every other kind of information passed. This is valid across all models, except for one exception: The  $last\_f1measure$  score of the GPT4 models increases to 0.7 if we give it the full KG in JSON-LD format. Surprisingly this did not happen with the Turtle format. The plots for the  $max\_combined$  in figs. 4d to 4h. show that the Gemini models, Claude 2.1 and Haiku perform worse for JSON-LD compared to using Turtle representations of the full KG or schema opposed to Opus and the GPT4 versions that perform better with JSON-LD.

#### 4.6. Results for Text to SPARQL (T2S) Beastiary

Generating syntactically correct SPARQL queries was no problem for the LLMs (besides Haiku) as can be seen in figs. 4i to 4l. No matter what kind of information was given about the KG, there were always about 420 of 450 queries that were parseable. Again, GPT4 performs best here with only answering two times with syntactically wrong queries (at the end of the dialog), and Haiku being responsible for about half of the errors, the rest being evenly distributed among the other flavors of Claude as well as Gemini. Looking at the  $f1$  score we can see that the kind of

**Table 4**

The  $t$  statistic and corresponding  $p$  value for two populations, obtained by grouping the data from all tasks on two aspects ( $A$  and  $B$ ) each and taking the  $f1$  score from *mean\_f1measure*.

| Aspect $A$ | Aspect $B$ | $t$  | $p$     |
|------------|------------|------|---------|
| Turtle     | JSON-LD    | 3.87 | 0.00011 |
| IRIs       | Schema     | 3.00 | 0.0027  |
| IRIs       | full KG    | 1.14 | 0.254   |
| Graph      | Schema     | 2.29 | 0.022   |

information provided has a large influence on the correctness of the query. Providing the schema in Turtle format yields a global *max\_f1measure* of 0.19 across all models, with the sub schema scoring only a 0.16. Providing a relevant portion of the KG as a sub graph however yields a score of 0.61 and providing the IRIs needed for the query nets a 0.76.

#### 4.7. Results for Text to Answer (T2A)

About 75% of the answers given were perfectly correct and about 97% good enough for the relaxed scoring. The most common formatting problem were the addition of spaces in the given answer. The few really wrong answers given were mainly on a question asking for a count, and the LLM answered with the wrong number. Plots for the *combinedF1* score are given in figs. 3c and 3d.

#### 4.8. Statistical Analysis

The above sections deal with the discussion of the results on a split-by-task basis, but we can also analyze the results based on different aspects that were covered, like the input format of a knowledge graph (*JSON-LD* vs *Turtle*) and the kind of information that was provided to the LLM (Full KG vs Schema information vs List of IRIs). We ran pairwise t-Tests and obtained the results shown in table 4. For each comparison we took the  $f1$  scores (*mean\_f1measure*) of two aspects across all tasks and performed a Welch’s t-Test with the two resulting populations, labeled  $A$  and  $B$ . In each case the null hypothesis  $H_0$  was that  $\mu_A = \mu_B$ . Using the common threshold of  $p = 0.05$  we can see that the null hypothesis can be rejected every time except for the difference between providing a list of relevant IRIs to the LLM vs the full KG. In the three remaining cases, we can see that there is a statistically significant difference in the data: Turtle vs JSON-LD, IRIs vs Schema, full KG vs Schema.

## 5. Conclusion and Outlook

Benchmarking LLMs on *SPARQL SELECT* query related tasks is difficult and more experiments are needed. Nonetheless with the work presented in this paper we contributed to the research questions defined and layed the ground for further automated evaluation.

From the results gathered we can give the following answers to the research questions: Most LLMs evaluated had no problems with working with *SPARQL SELECT* query syntax (RQ1) or

reading *SPARQL SELECT* query semantics (RQ2). Creating *SPARQL SELECT* queries with correct semantics (RQ3) seems to be still a difficult task for the LLMs evaluated here, at least when applying the approach to evaluate the answer generated. The results seem to depend on different aspects (RQ4). The varying performance between the 10 different variants of T2S tasks indicate, that we should further extend the tests for serialization formats and content that is provided as KG information to the LLMs (full graph vs. sub graph vs. schema vs. IRIs). Taking the CoyPu-Mini KG as an example, we found that Turtle leads to a higher number of syntactically correct queries, whereas JSON-LD improved the score of GPT4 tremendously and made it stand above every other model in this specific test.

The results of LLMs depend a lot on the concrete task setting as can be seen in fig. 4. Thus we found no clear *winner*. But when focusing on the mean value of f1 scores for T2S tasks Claude 3 Opus and GPT 4 scored best as shown in fig. 5b.

Further research could improve the evaluation of *SPARQL SELECT* queries created by LLMs, especially for queries returning no result when applied on the KG.

In the long term, the fine-tuning or training of LLMs would probably benefit from more SPARQL-related training data.

Unfortunately, we found that existing KGQA benchmarks have ambiguity that hinder a proper automatic evaluation. Given the fact those publicly available benchmarks (like LC-QuAD) can be contained in the training dataset and memorized by the LLMs, emphasizes the need for new and diverse test datasets. Fortunately, with the extended framework presented in this work, it is easy to include new evaluation datasets.

## Acknowledgments

This work was partially supported by grants from the German Federal Ministry of Education and Research (BMBF) to the project StahlDigital (13XP5116B) and ScaleTrust (16DTM312D) as well as from the German Federal Ministry for Economic Affairs and Climate Action (BMWK) to the KISS project (01MK22001A) and CoyPu project (01MK21007A).

## Conflicts of Interest

The authors have no competing interests to declare that are relevant to the content of this article. The authors used a free evaluation license for Claude and Gemini models, however due to the setup and technical nature of the evaluation this has no effect on the results.

## Online Resources

**code:** <https://github.com/AKSW/LLM-KG-Bench>, DOI:10.5281/zenodo.13622575

**results:** <https://github.com/AKSW/LLM-KG-Bench-Results/tree/main/2024-NLP4KGC-SPARQL>, DOI:10.5281/zenodo.13621581



## References

- [1] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: A roadmap, *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2024). doi:10.1109/TKDE.2024.3352100.
- [2] A. Hogan, E. Blomqvist, M. Cochez, C. d’Amato, G. de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.-C. N. Ngomo, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, A. Zimmermann, Knowledge graphs, *ACM Computing Surveys (CSUR)* 54 (2020) 1 – 37. doi:10.1145/3447772.
- [3] A. Srivastava, et al., Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, *Transactions on Machine Learning Research* (2023). arXiv:2206.04615.
- [4] D. Park, Open-llm-leaderboard-report, 2023. URL: <https://github.com/dsdanielpark/Open-LLM-Leaderboard-Report>.
- [5] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez, I. Stoica, Chatbot arena: An open platform for evaluating llms by human preference, 2024. arXiv:2403.04132.
- [6] J. Frey, L. Meyer, N. Arndt, F. Brei, K. Bulert, Benchmarking the abilities of large language models for RDF knowledge graph creation and comprehension: How well do llms speak turtle?, in: M. Alam, M. Cochez (Eds.), *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG 2023) co-located with the 21th International Semantic Web Conference (ISWC 2023)*, Athens, November 6-10, 2023, volume 3559 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: <https://ceur-ws.org/Vol-3559/paper-3.pdf>.
- [7] L.-P. Meyer, C. Stadler, J. Frey, N. Radtke, K. Junghanns, R. Meissner, G. Dziwis, K. Bulert, M. Martin, Llm-assisted knowledge graph engineering: Experiments with chatgpt, in: C. Zinke-Wehlmann, J. Friedrich (Eds.), *First Working Conference on Artificial Intelligence Development for a Resilient and Sustainable Tomorrow (AITomorrow) 2023*, *Informatik aktuell*, 2023, pp. 101–112. doi:10.1007/978-3-658-43705-3\_8.
- [8] L. Kovriguina, R. Teucher, D. Radyush, D. Mouromtsev, Sparqlgen: One-shot prompt-based approach for sparql query generation, in: *International Conference on Semantic Systems*, volume 3526 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: <https://ceur-ws.org/Vol-3526/paper-08.pdf>.
- [9] L.-P. Meyer, J. Frey, K. Junghanns, F. Brei, K. Bulert, S. Gründer-Fahrer, M. Martin, Developing a scalable benchmark for assessing large language models in knowledge graph engineering, in: N. Keshan, S. Neumaier, A. L. Gentile, S. Vahdati (Eds.), *Proceedings of the Posters and Demo Track of the 19th International Conference on Semantic Systems (SEMANTICS 2023)*, 2023. URL: <https://ceur-ws.org/Vol-3526/paper-04.pdf>.
- [10] J. C. Rangel, T. M. de Farias, A. C. Sima, N. Kobayashi, Sparql generation: an analysis on fine-tuning openllama for question answering over a life science knowledge graph, 2024. arXiv:2402.04627, to appear in *Proceedings of SWAT4HCLS 2024: Semantic Web Tools and Applications for Healthcare and Life Sciences*.
- [11] D. Bustamante, H. Takeda, Sparql generation with entity pre-trained gpt for kg question answering, 2024. arXiv:2402.00969.
- [12] C. V. S. Avila, V. M. P. Vidal, W. Franco, M. A. Casanova, Experiments with text-to-

- sparql based on chatgpt, in: 18th IEEE International Conference on Semantic Computing, ICSC 2024, Laguna Hills, CA, USA, February 5-7, 2024, IEEE, 2024, pp. 277–284. doi:10.1109/ICSC59802.2024.00050.
- [13] Z. Li, S. Fan, Y. Gu, X. Li, Z. Duan, B. Dong, N. Liu, J. Wang, Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering, 2024. doi:10.1609/aaai.v38i17.29823. arXiv:2308.12060.
- [14] P. A. K. K. Diallo, S. Reyd, A. Zouaq, A comprehensive evaluation of neural sparql query generation from natural language questions, 2024. arXiv:2304.07772.
- [15] R. Hirigoyen, A. Zouaq, S. Reyd, A copy mechanism for handling knowledge base elements in SPARQL neural machine translation, in: Y. He, H. Ji, S. Li, Y. Liu, C.-H. Chang (Eds.), Findings of the Association for Computational Linguistics: ACL-IJCNLP 2022, Association for Computational Linguistics, Online only, 2022, pp. 226–236. URL: <https://aclanthology.org/2022.findings-aacl.22>.
- [16] D. Pliukhin, D. Radyush, L. Kovriguina, D. Mouromtsev, Improving subgraph extraction algorithms for one-shot sparql query generation with large language models, in: 1st Scholarly QALD Challenge 2023 and 4th SeMantic Answer Type, Relation and Entity Prediction Tasks Challenge 2023, volume 3592 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: <https://ceur-ws.org/Vol-3592/paper6.pdf>.
- [17] H. M. Zahera, M. Ali, M. A. Sherif, D. Moussallem, A.-C. Ngonga Ngomo, Generating sparql from natural language using chain-of-thoughts prompting, in: SEMANTiCS 2024: Knowledge Graphs in the Age of Language Models and Neuro-Symbolic AI, Amsterdam, Netherlands, 2024. doi:10.3233/ssw240028.
- [18] R. Usbeck, M. Röder, M. Hoffmann, F. Conrads, J. Huthmann, A.-C. Ngonga-Ngomo, C. Demmler, C. Unger, Benchmarking question answering systems, *Semantic Web 10* (2019) 293–304. doi:10.3233/sw-180312.
- [19] J. Lehmann, S. Ferré, S. Vahdati, Language models as controlled natural language semantic parsers for knowledge graph question answering, in: K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, R. Radulescu (Eds.), ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023), volume 372 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2023, pp. 1348–1356. doi:10.3233/FAIA230411.
- [20] D. Diefenbach, T. P. Tanon, K. D. Singh, P. Maret, Question answering benchmarks for wikidata, in: N. Nikitina, D. Song, A. Fokoue, P. Haase (Eds.), Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017, volume 1963 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017. URL: <https://ceur-ws.org/Vol-1963/paper555.pdf>.
- [21] A. Bordes, N. Usunier, S. Chopra, J. Weston, Large-scale simple question answering with memory networks, 2015. arXiv:1506.02075.
- [22] M. Dubey, D. Banerjee, A. Abdelkawi, J. Lehmann, Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia, in: Proceedings of the 18th International Semantic Web Conference (ISWC), Springer, 2019. doi:10.1007/978-3-030-30796-7\_5.

- [23] P. Trivedi, G. Maheshwari, M. Dubey, J. Lehmann, Lc-quad: A corpus for complex question answering over knowledge graphs, in: C. d’Amato, M. Fernández, V. A. M. Tamma, F. Lécué, P. Cudré-Mauroux, J. F. Sequeda, C. Lange, J. Heflin (Eds.), *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference*, Vienna, Austria, October 21-25, 2017, Proceedings, Part II, volume 10588 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 210–218. doi:10.1007/978-3-319-68204-4\\_22.
- [24] D. Banerjee, P. A. Nair, J. N. Kaur, R. Usbeck, C. Biemann, Modern baselines for sparql semantic parsing, in: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’22*, ACM, 2022. doi:10.1145/3477495.3531841.
- [25] J. Frey, L.-P. Meyer, F. Brei, S. Gruender, M. Martin, Assessing the evolution of llm capabilities for knowledge graph engineering in 2023, in: *Proceedings of Special Track Large Language Models for Knowledge Engineering at Extended Semantic Web Conference 2024 (ESWC24)*, 2024. URL: <https://2024.eswc-conferences.org/wp-content/uploads/2024/05/77770050.pdf>.
- [26] M. Hofer, J. Frey, E. Rahm, Towards self-configuring knowledge graph construction pipelines using llms - a case study with rml, in: *Fifth International Workshop on Knowledge Graph Construction @ ESWC2024*, volume 3718 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024. URL: <https://ceur-ws.org/Vol-3718/paper6.pdf>.
- [27] F. Brei, J. Frey, L.-P. Meyer, Leveraging small language models for text2sparqltasks to improve the resilience of ai assistance, in: J. Holze, S. Tramp, M. Martin, S. Auer, R. Usbeck, N. Krdzavac (Eds.), *Proceedings of the Third International Workshop on Linked Data-driven Resilience Research 2024 (D2R2’24)*, colocated with ESWC 2024, volume 3707 of *CEUR-WS*, 2024. URL: [https://ceur-ws.org/Vol-3707/D2R224\\_paper\\_5.pdf](https://ceur-ws.org/Vol-3707/D2R224_paper_5.pdf).