# Impact of Knowledge Graph Representations on Question Answering with Language Models

Daniel Henselmann[1 [0000−0001−6701−0287]], Rene Dorsch[1 [0000−0001−6857−7314]], and Andreas Harth[1,2 [0000−0002−0702−510X]]

[1] Fraunhofer IIS, Nuremberg, Germany
{forename.surname}@iis.fraunhofer.de
[2] Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

**Abstract.** The emergence of Large Language Models (LLMs) brought new approaches to Knowledge Graph Question Answering (KGQA), chasing the vision of querying structured data using natural language. While existing work focuses on improving KGQA approaches, this paper explores the impact of different knowledge graph representations. We consider three dimensions of representation: (i) subsets, (ii) modeling, and (iii) annotations, hypothesizing that different variations impact the $F_1$ scores of KGQA systems. We conduct experiments on a custom knowledge graph featuring integrated data and n-ary relations. Results demonstrate a substantial improvement in the $F_1$ score from 17.6% to 44.5% between the default and best variant, confirming the hypothesis.

**Keywords:** Question answering · Knowledge graphs · Large language models · Knowledge representation .

## 1 Introduction

Knowledge Graph Question Answering (KGQA) persues the vision that users ask questions in natural language that are answered based on factual data. While KGQA is still a hard task to solve, the emergence of Large Language Models (LLMs) has led to new possibilities [3,12]. Research on LLM-augmented KGQA focuses on improving the approaches but assumes a KG with a given, fixed representation.

We consider a different angle and evaluate different representations of KGs with the same KGQA approach. Specifically, we represent the same knowledge in different ways, varying in explicit information. Our hypothesis is the following: *Different representations of KGs impact the $F_1$ score of LLM-augmented KGQA.* To the best of our knowledge, this is the first work that explores this angle.

We identify at least three major dimensions of KG representations:

– **Subsets**. Most KGs are too large to be entirely added to the LLM prompt. Therefore, approaches to KGQA obtain relevant subgraphs to discover and understand the KG.

- **Modeling.** Modeling options for ontologies in KGs are numerous. Different modeling options vary in expressiveness and explicitness of the information used to describe the same knowledge.
- **Annotations.** Annotations do not "contribute to the 'logical' knowledge" [9, Section 8] in a KG but provide additional information as text.

The dimensions have considerable depth and warrant dedicated research. We focus on selected subdimensions, which we quantitatively evaluate through experiments to test our hypothesis.

The experiments are conducted on a custom KG (see Section 5.3) and use Graf von Data (GvD) (see Section 5.1), a KGQA agent similar to the state-of-the-art (SPINACH [13]) that is KG-agnostic. The results show an increase in the $F_1$ score from 17.6% for the default variant to a maximum of 44.5%, an improvement of 26.9pp/153%, confirming the hypotheses.

Our contributions are (i) a KG unknown to LLMs, including integrated data and n-ary relations, and (ii) a quantitative performance evaluation of selected KG representations for KGQA.

The remainder of the paper is structured as follows. Section 2 positions the paper w.r.t. related work. Section 3 introduces an example. Section 4 discusses considered variants. Section 5 explains the experimental setup. Section 6 describes the conducted experiments and discusses the results. Section 7 concludes the paper and outlines future work.

## 2 Related Work

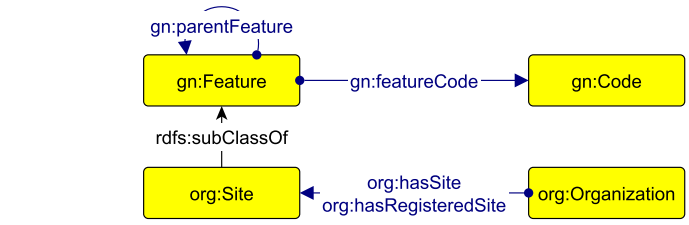
Multiple fundamental approaches to LLM-augmented KGQA have emerged using different interfaces through which the LLM obtains KG data. In finetuning approaches, LLMs are specifically trained on a KG [22]. In workflow approaches, information from the KG in the form of example question-answer-pairs [17,24], ontology terms [10], a KG subgraph [1,11,17], and/or URIs [24] is added to the prompt that instructs the LLM. In agent approaches, an agent iteratively obtains information from the KG through actions [23] to build a SPARQL query that, when executed, retrieves the answer. The core actions are searching entities (Search) [13,21], retrieving entities (Describe) [13,20,21], and executing SPARQL queries (Query) [13,21]. Some agents also include searching paths between two entities [20] and retrieving usage examples of properties [13].

While the approaches may differ, all obtain information from the KG that changes with KG representation. Varying **subsets** impact some workflow and all agent approaches. Varying **modeling** impacts all approaches because modeling is intrinsic to the KG. Varying **annotations** impact all finetuning, some workflow, and all agent approaches.

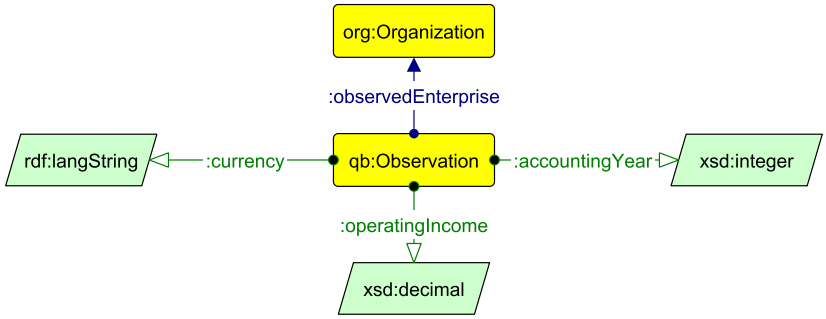
The SPINACH agent [13] achieves state-of-the-art results on Wikidata. Consequently, we focus on agent approaches. However, the SPINACH agent's actions are not knowledge graph-agnostic. Therefore, we conduct our experiments using our own agent (see Section 5.1).

## 3 Example

In the following, we explore how varying KG representations might impact the agent's result for the natural language question: "companies in Germany". Our KG (see Section 5.3) describes enterprises in the semiconductor industry. The KG contains a *Geo* pattern (Figure 1a), in which terms from the Organization Ontology [18] and GeoNames' ontology[3] express in which regions an organization has sites. The hierarchical regions are connected with the transitive `gn:parentFeature` property. For prefixes, we refer to the repository linked in the KG description in Section 5.3.




(a) *Geo* pattern.




(b) *Keyfigure* pattern.


(c) *Supply* pattern.

Fig. 1: Important patterns in our ontology visualized in Graffoo [6] notation.

The required SPARQL query to retrieve companies in Germany uses the Geo pattern and requires the URI of Germany:

```
PREFIX org: <https://www.w3.org/ns/org#>
PREFIX gn: <http://www.geonames.org/ontology#>
```

[3] See http://www.geonames.org/ontology/documentation.html

```
PREFIX : <https://solid.iis.fraunhofer.de/e-40200/2024/10/
          velektronik-graph-clean/region/wdQ183.ttl#>
SELECT ?x WHERE {
    ?x org:hasRegisteredSite _:y .
    _:y gn:parentFeature+ :this . }
```

Discovering the required relation in the KG is easier if (i) the Describe action on Germany returns incoming triples to detect subordinate regions linked from an `org:Organization` and (ii) the Describe action on `gn:Feature` returns the subclass alignment with `org:Site`. Adding the property `gn:locatedIn` could link Germany directly from an organization. To find the class `org:Organization` and understand that a company is an organization, a comment for the class could add the description.

Other important patterns in the KG are *Keyfigure* and *Supply*, both of which represent n-ary relations. In the Keyfigure pattern (Figure 1b), the `qb:Observation` class from the Data Cube Vocabulary [4] expresses key figures (operating income as example) of organizations. In the Supply pattern (Figure 1c), the `sn:SupplyFlow` class from the Abstraction Independent Supply Network Ontology[4] expresses a supply relation between organizations.

## 4 Considered Representation Variants

We identify subsets, modeling, and annotations as dimensions of KG representations. As the foundation, we consider Concise Bounded Descriptions (CBD) [19] for **subsets**, the RDFS/OWL [2,9] features class, object and datatype property, subclass, subproperty, and property characteristics besides 'inverse' for **modeling**, and `rdfs:label` for **annotations**. Subdimensions that introduce alterations include, but are not limited to:

- **Subsets**
  - ***Entity boundaries*** like CBD define the triples returned when describing or dereferencing an entity.
  - **Domain subsets** include triples that belong to a certain domain.
- **Modeling**
  - ***Inverse properties*** link two entities in opposing direction.
  - ***Shortcut properties*** represent a property path to directly link two entities.
  - **Ontology Design Patterns** are distinguishable patterns independent of an ontology that express an established relation.
  - **Ontology reuse and alignment** focus on combining terms from several ontologies in one KG while maintaining semantics.
  - **Upper ontologies** introduce classes on a high abstraction to support applied ontologies with fundamental semantics.
  - **SKOS concepts** offer a hierarchical classification different from classes.

[4] See https://purl.org/supply-network/onto#

  - **Shapes** express ontologies alternatively or supplementary to OWL.
- **Annotations**
  - ***Comments*** are human-readable descriptions of an entity.
  - **Alternative labels** express synonyms or abbreviations.
  - **Examples** show example uses.

This paper focuses on the four subdimensions highlighted in italics, for which we discuss possible variants in the Sections 4.1 to 4.4 and conduct experiments to test our hypothesis.

### 4.1 Entity Boundaries

Entity boundaries refer to a method of extracting a limited set of triples about a specific entity within the KG. Several ways exist: The CBD algorithm [19] includes the outgoing triples of an entity while recursively including the outgoing triples of encountered blank nodes. Alternatively, all neighboring triples of an entity can be retrieved [20], potentially selected by semantic relevance [21]. Wikidata's Linked Data Interface[5] returns outgoing triples of an entity with additional statements according to Wikidata's data model and property information. The returned triples are verbose but may be filtered using the LLM and a dedicated prompt [13].

**Variants.** *Outgoing* triples of a subject are the default variant for our experiments, which aligns with CBD because we do not consider blank nodes.

We retrieve the entire *neighborhood* as another variant by adding incoming triples. The consideration of encountered (potentially nested) blank nodes when retrieving the neighborhood is an open question out of the scope of this paper.

Furthermore, we introduce *specific* class and object property boundaries. Besides outgoing triples, the specific boundaries include selected triples regarding subclasses/subproperties, domain, and range. The motivation is (i) to reduce the number of returned triples without losing valuable information and (ii) to add valuable information (classes in domain and range, types, and labels) outside of the neighborhood. For SPARQL queries for the specific boundaries, see Appendix B and C.

**Examples.** When describing a `gn:Feature` instance (e.g., Germany) in the geo pattern, the returned triples do not contain the backwards traversable links to discover a related `org:Organization` instance. To find the link, the agent must describe the `org:Organization` to follow the outgoing triples.

The classes `qb:Observation` and `sn:SupplyFlow` express n-ary relations. The choice of outgoing triples for an n-ary relation without any single individual standing out as the subject of the relation follows a W3C Working Group Note [16]. Consequently, instances of these classes can never be discovered with default entity boundaries.

Describing the class `gn:Feature` returns no information on the `org:Site` subclass nor on the properties that have `gn:Feature` as domain or range.

[5] See https://www.wikidata.org/wiki/Wikidata:Data_access

### 4.2 Inverse Properties

An inverse property inverts the relation of a property and links two entities in the opposing direction [9]. The information represented by two properties inverse of each other is redundant [15]. Yet, materializing both triples adds explicit information, potentially making it easier for an LLM to grasp a pattern.

The addition of inverse properties also covers some of the effects that extended (i.e., neighborhood or specific) entity boundaries have because either an inverse property or incoming triples add a reverse link to retrieved data.

But unlike entity boundaries, properties are present in all KGQA agent actions: The Search action has more options to match the keyword, the Describe action returns more triples that can be understood, and the Query action has two inverse alternatives for each triple pattern.

**Variants.** For the experiments, we assume no inverse properties as the default. As an alternative variant, we define inverse properties for all object properties in the ontology, which is a baseline approach. We leave sophisticated variants with selected inverse properties to future work. Table 1 shows the inverse properties for the highlighted patterns.

**Examples.** In the highlighted patterns, inverse properties ensure that all information is reachable when traversing the data with Describe actions.

Table 1: Inverse properties for the patterns from Figure 1.

| Inverse property | Default property |
|---|---|
| `org:siteOf` | `org:hasSite` |
| `:registeredSiteOf` | `org:hasRegisteredSite` |
| `:childFeature` | `gn:parentFeature` |
| `:featureCodeOf` | `gn:featureCode` |
| `:enterpriseObservation` | `:observedEnterprise` |
| `:outgoingSupplyFlow` | `esn:supplier` |
| `:incomingSupplyFlow` | `esn:customer` |

### 4.3 Shortcut Properties

A shortcut property is a single property that represents a property path [7,14].

Relations between entities that are not linked directly are less obvious to discover and understand. N-ary relations relate multiple entities through a dedicated entity [16]. As a consequence, two entities that are part of the n-ary relation are not directly linked. An additional shortcut property that links the two directly expresses their binary relation more explicitly.

Similarly, a shortcut property can explicitly express the relation to a specific entity in a chain of transitive properties. For this scenario, OWL supports the definition of a property through a property chain [9]. However, property chains

are a list of properties with fixed length and no reverse properties. Property paths in SPARQL [8] support both and more.

Shortcut properties reduce the number of entities the agent needs to discover and understand because a shortcut property allows to skip some of the modeling if the skipped details are not required to answer the given question. Like inverse properties, shortcut properties impact all KGQA agent actions.

**Variants.** For the experiments, we assume no shortcut properties as the default. As an alternative variant, we add shortcut properties where they shorten reoccurring property paths required in answers. The naive approach of adding all possible shortcut properties is not sensible because it leads to properties expressing extensive relations with dubious meaning. Table 2 shows the shortcut properties for the highlighted patterns.

**Examples.** In the Geo pattern, the shortcut property `gn:parentCountry` skips a path of transitive `gn:parentFeature` properties. Thus, the agent does not have to grasp the transitivity. Additionally, the agent may understand that `gn:parentCountry` points to a country and thus does not have to identify a country by label or by the `gn:featureCode`. The shortcut property `gn:locatedIn` additionally takes the discovery and understanding that the location of an organization comes from its registered site out of the equation.

The shortcut properties `:hasSupplier` and `:hasCustomer` for the Supply pattern abbreviate the path through the n-ary relation by directly linking suppliers and customers. Thus, the agent does not have to understand `sn:SupplyFlow`.

Table 2: All shortcut properties.

| Shortcut property | Default property path (SPARQL syntax) |
|---|---|
| `gn:parentCountry` | `gn:parentFeature+` |
| `gn:locatedIn` | `org:hasRegisteredSite/gn:parentFeature+` |
| `:hasSupplier` | `^esn:customer/esn:supplier` |
| `:hasCustomer` | `^esn:supplier/esn:customer` |

### 4.4 Comments

The `rdfs:comment` property expresses a human-readable description of an entity to clarify the meaning [2]. The usage includes "prose documentation, examples, test cases" [2, Section 3.7], making `rdfs:comment` a versatile property. Consequently, the application of `rdfs:comment` greatly differs between ontologies.

Additional comments on classes and properties help to reduce the ambiguity of the chosen terms. Comments can also increase the likelihood of matches when searching entities. Therefore, the Search action of GvD includes both the `rdfs:label` and `rdfs:comment` into the vector compared to the searched term.

**Variants.** For the experiments, we assume no comments as the default. As an alternative variant, we add a `rdfs:comment` literal to all classes, properties,

and individuals in the ontology. For newly defined terms in the ontology, we write a comment that describes the term in the context of the ontology. If a reused term has a comment in the original ontology, we use that comment regardless of how well the comment fits the use case and selection of ontology terms. All comments (and labels) are in English only.

**Examples.** In the highlighted patterns, comments might reduce ambiguity for the KGQA agent. For example, the meaning of the class `gn:Code` is not clear without a description. The direction of the properties `esn:supplier` and `esn:customer` is ambiguous as they lack a suffix ("hasSupplier" or "supplierOf").

## 5 Experimental Setup

### 5.1 Agent

We use an LLM-based agent called Graf von Data (GvD)[6] that has three actions at its disposal, which return information from the knowledge graph: a list of URIs for a keyword (Search), a subgraph for a URI (Describe), and a SPARQL result set for a SPARQL query (Query). The prompt for GvD is attached in Appendix A. GvD is knowledge graph-agnostic, supports integrated data, and requires no prior knowledge of the KG.

### 5.2 Language Model

We conduct all experiments with Llama 3.3 70B[7] because it is open source and performs well[8] in instruction following, which is required for agent approaches. Our empirical comparison of several LLMs for GvD confirms Llama 3.3 as a strong choice. For our experiments, we set the temperature to 0 and top_p to 1. All experiments were conducted with the Llama 3.3 hosted at Chat AI [5].

### 5.3 Knowledge Graph

We evaluate variants of KG representation with a custom KG[9]. The KG describes enterprises in the semiconductor industry, their sites, and the supply relations between them. The KG integrates data from the Welektonik[10] Wikibase, Wikidata, DBLP, and Library of Congress. We modeled the data with a custom ontology that reuses established design patterns and ontology terms but also defines new terms when required. The KG contains 32,276 to 48,117 triples, depending on the variant.

The KG differs from other KGs for KGQA evaluation (i) because it is unknown to LLMs, (ii) in the integration of multiple data sources and ontologies, and (iii) in the modeling, which includes n-ary relations.

[6] Accessible at https://graf.ti.rw.fau.de/
[7] See https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct
[8] See https://github.com/meta-llama/llama-models/blob/main/models/llama3_3/MODEL_CARD.md and https://t.co/6oQ7b3Yuzc
[9] Available at https://github.com/Quarkse/kg-rep-llm-qa/tree/main/knowledgegraph
[10] See https://welektronik.iis.fraunhofer.de/wiki/Main_Page

### 5.4 Corpus

Our evaluation corpus[11] consists of 58 handcrafted question-query-pairs. Of those, 14 contain the Geo pattern, 15 the Keyfigure pattern, and 11 the Supply pattern. The remaining 18 contain none of the three patterns.

Each question-query-pair consists of a question in natural language and one to four triple patterns of a (gold standard) SPARQL query that expresses the correct answer in the KG. Each question requires one variable, which is always a URI. Focusing on URIs allows the agent (i) to retrieve found results to questions to represent them with added context and (ii) removes the problem of whether attributes (the currency of the operating income) or dimensions (the accounting year of the operating income) shall be included in the answers to the questions.

### 5.5 Evaluation Metrics

As evaluation metrics, we use the Exact Match (EM) and $F_1$ scores. The EM metric checks whether the result of the generated query completely matches the result of the gold standard query in the corpus. The $F_1$ metric considers both how many entities in the results are also in the gold standard result and how many of the entities in the gold standard result are in the results. We adopt the generalization to not penalize additional variables retrieved by the queries [13]. We run all experiments five times and give the mean values.

## 6 Results and Discussion

Table 3 shows the 13 conducted experiments[12] with resulting EM and $F_1$ scores for the entire corpus and the questions covering the three patterns from Figure 1.

Accross the **Entire Corpus**, the neighborhood as entity boundaries considerably improves the scores (experiments 2 vs. 1 and 4 vs. 3). Extending class and property boundaries with specific information improves the scores a little (3 and 4 vs. 1 and 2). The combination of neighborhood triples for individuals and specific information for classes and properties achieves the best scores.

For the following experiments (5 to 13), we consider the default (outgoing/outgoing) and the best (neighborhood/specific) entity boundaries.

Adding inverse properties to the modeling considerably improves the scores for default boundaries (5) but decreases scores for the extended boundaries (6). Shortcut properties considerably improve the scores for both boundary variants (7 and 8). The combination of inverse and shortcut properties further improves scores (9 and 10), especially in combination with extended boundaries (10). The latter achieves the highest EM (38.3%) and $F_1$ (44.5%) scores on the entire corpus. Thus, changing the KG representation for KGQA achieved an improvement from the default's EM score of 13.5% by 24.8pp/184% and the default's $F_1$ score of 17.6% by 26.9pp/153%.

[11] Available at https://github.com/Quarkse/kg-rep-llm-qa/tree/main/corpus
[12] Available at https://github.com/Quarkse/kg-rep-llm-qa/tree/main/experiments

Table 3: Experiment results. The highest scores are marked in bold.

| | Entity Boundaries | | Inverse Properties | Shortcut Properties | Comments | Entire Corpus | | Geo Pattern | | Keyfigure Pattern | | Supply Pattern | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. | Individuals | Classes/ Properties | | | | EM | $F_1$ | EM | $F_1$ | EM | $F_1$ | EM | $F_1$ |
| 1 | outgoing | outgoing | - | - | - | 13.5 | 17.6 | 4.3 | 7.1 | **6.7** | **6.9** | 5.5 | 5.9 |
| 2 | neighborh. | neighborh. | - | - | - | 24.1 | 28.4 | 17.4 | 24.7 | 4.0 | 4.1 | 0.0 | 2.7 |
| 3 | outgoing | specific | - | - | - | 16.9 | 22.3 | 8.5 | 14.9 | 4.0 | 4.0 | 1.8 | 11.4 |
| 4 | neighborh. | specific | - | - | - | 27.3 | 31.7 | 21.4 | 29.5 | 5.4 | 5.4 | 1.8 | 3.6 |
| 5 | outgoing | outgoing | ✓ | - | - | 20.1 | 25.3 | 17.2 | 19.3 | 5.3 | 6.1 | 9.1 | 9.4 |
| 6 | neighborh. | specific | ✓ | - | - | 23.1 | 25.3 | 8.6 | 10.7 | 2.7 | 3.7 | 1.8 | 2.9 |
| 7 | outgoing | outgoing | - | ✓ | - | 24.8 | 31.8 | 30.0 | 39.6 | 4.0 | 4.5 | 27.3 | 31.8 |
| 8 | neighborh. | specific | - | ✓ | - | 32.8 | 40.2 | 45.7 | **59.4** | 4.0 | 4.2 | 16.4 | 25.9 |
| 9 | outgoing | outgoing | ✓ | ✓ | - | 26.0 | 32.3 | 27.2 | 39.1 | 0.0 | 0.2 | 23.7 | 26.0 |
| 10 | neighborh. | specific | ✓ | ✓ | - | **38.3** | **44.5** | **50.0** | 58.3 | 2.7 | 3.6 | **34.6** | **39.2** |
| 11 | outgoing | outgoing | - | - | ✓ | 13.4 | 17.6 | 2.9 | 5.9 | 1.3 | 3.9 | 1.8 | 3.0 |
| 12 | neighborh. | specific | - | - | ✓ | 20.4 | 24.8 | 21.4 | 25.6 | 0.0 | 1.9 | 1.8 | 7.9 |
| 13 | neighborh. | specific | ✓ | ✓ | ✓ | 34.1 | 40.9 | 40.0 | 53.4 | 4.0 | 4.0 | 23.7 | 33.5 |

Adding comments to the default variant has no impact on the scores (11), and to other variants (12 and including the best in 13) lowers the scores.

Potential reasons for decreasing scores despite adding information to the KG representation are (i) a longer prompt, (ii) the inclusion of more irrelevant information, and (iii) the Search action being congested by the comments. All three could lead to the LLM having a harder time identifying the relevant information.

Results for the (transitive) **Geo** pattern follow the trends of the entire corpus, but specifically profit from shortcut properties (7 to 10). The highest $F_1$ score of 59.4% is achieved with shortcut but without inverse properties, improving on the default by 52.3pp/739%, outperforming the entire corpus (8).

Results for the (n-ary) **Keyfigure** pattern show no improvement in any variant and the scores remain on a low level or decrease. Seemingly, the LLM fails to understand the pattern across all considered variants.

Results for the (n-ary) **Supply** pattern show scores mostly improving with shortcut properties (7 to 10). The best variant aligns with the entire corpus with an $F_1$ score of 39.2%, improving on the default by 33.3pp/567% (10).

The results indicate that LLM-augmented KGQA struggles with n-ary relations in comparison to binary relations.

## 7 Conclusion

The paper provides insight into LLM-augmented KGQA, for which we formulated the hypothesis that different KG representations impact the $F_1$ score. The experiments on several variants of KG representation confirm the hypotheses.

Adding inverse and shortcut properties in combination with extended entity boundaries improved the $F_1$ score by 26.9pp/153% to 44.5%.

Future work may consider (i) more variants, (ii) additional subdimensions (iii) questions requiring multiple variables, (iv) other KGQA approaches, and (v) different LLMs.

Additionally, a comparison of runtimes, numbers of tokens, and, consequently, costs of LLM reasoning could uncover drawbacks of different KG representations.

**Acknowledgments.** The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU). The hardware is funded by the German Research Foundation (DFG).

## References

1. Avila, C.V.S., Casanova, M.A., Vidal, V.M.P.: A Framework for Question Answering on Knowledge Graphs Using Large Language Models. In: The Semantic Web: ESWC 2024 Satellite Events, vol. 15344, pp. 168–172. Springer (2025). https://doi.org/10.1007/978-3-031-78952-6_20
2. Brickley, D., Guha, R.: RDF Schema 1.1. https://www.w3.org/TR/rdf-schema/ (Feb 2014), accessed 2021-10-06
3. Chakraborty, N., Lukovnikov, D., Maheshwari, G., Trivedi, P., Lehmann, J., Fischer, A.: Introduction to neural network-based question answering over knowledge graphs. WIREs Data Mining and Knowledge Discovery **11**(3), e1389 (May 2021). https://doi.org/10.1002/widm.1389
4. Cyganiak, R., Reynolds, D.: The RDF Data Cube Vocabulary. https://www.w3.org/TR/vocab-data-cube/ (Jan 2014), accessed 2024-12-11
5. Doosthosseini, A., Decker, J., Nolte, H., Kunkel, J.M.: Chat AI: A Seamless Slurm-Native Solution for HPC-Based Services (Aug 2024). https://doi.org/10.48550/arXiv.2407.00110
6. Falco, R., Gangemi, A., Peroni, S., Shotton, D., Vitali, F.: Modelling OWL Ontologies with Graffoo. In: The Semantic Web: ESWC 2014 Satellite Events. LNCS, vol. 8798, pp. 320–325. Springer (2014). https://doi.org/10.1007/978-3-319-11955-7_42
7. Fichtner, M., Ribaud, V.: Paths and Shortcuts in an Event-Oriented Ontology. In: Metadata and Semantics Research. vol. 343, pp. 214–226. Springer (2012). https://doi.org/10.1007/978-3-642-35233-1_22
8. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. https://www.w3.org/TR/sparql11-query/ (Mar 2013), accessed 2023-09-03
9. Hitzler, P., Krötzsch, M., Parsia, B., Petel-Schneider, P., Rudolph, S.: OWL 2 Web Ontology Language Primer (Second Edition). https://www.w3.org/TR/owl2-primer/ (Dec 2012), accessed 2021-10-06
10. Jiang, L., Yan, X., Usbeck, R.: A Structure and Content Prompt-based Method for Knowledge Graph Question Answering over Scholarly Data. In: Joint Proceedings of Scholarly QALD 2023 and SemREC 2023 Co-Located with ISWC 2023. vol. Vol-3592. CEUR-WS, Athens, Greece (101123)
11. Kovriguina, L., Teucher, R., Radyush, D., Mouromtsev, D.: SPARQLGEN: One-Shot Prompt-based Approach for SPARQL Query Generation. In: Proceedings of the Posters and Demo Track of the 19th International Conference on Semantic Systems (SEMANTiCS 2023). vol. Vol-3526. CEUR, Leipzig, Germany (Sep 2023)

12. Lehmann, J., Gattogi, P., Bhandiwad, D., Ferré, S., Vahdati, S.: Language Models as Controlled Natural Language Semantic Parsers for Knowledge Graph Question Answering. In: Frontiers in Artificial Intelligence and Applications. IOS Press (Sep 2023). https://doi.org/10.3233/FAIA230411
13. Liu, S., Semnani, S.J., Triedman, H., Xu, J., Zhao, I.D., Lam, M.S.: SPINACH: SPARQL-Based Information Navigation for Challenging Real-World Questions. In: Findings of the Association for Computational Linguistics: EMNLP 2024. pp. 15977–16001. Association for Computational Linguistics (Nov 2024). https://doi.org/10.18653/v1/2024.findings-emnlp.938
14. Mungall, C., Ruttenberg, A., Osumi-Sutherland, D.: Taking shortcuts with OWL using safe macros. Nature Precedings (2010). https://doi.org/10.1038/npre.2010.5292.1
15. Noy, N.F., McGuinness, D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Tech. Rep., Stanford University, Stanford (2001)
16. Noy, N., Rector, A.: Defining N-ary Relations on the Semantic Web. https://www.w3.org/TR/swbp-n-aryRelations/ (Apr 2006), accessed 2024-12-19
17. Piao, G., Mountantonakis, M., Papadakos, P., Sonawane, P., OMahony, A.: Toward Exploring Knowledge Graphs with LLMs. In: Joint Proceedings of Posters, Demos, Workshops, and Tutorials of the 20th International Conference on Semantic Systems (SEMANTiCS 2024). vol. Vol-3759. CEUR-WS, Amsterdam, The Netherlands (Sep 2024)
18. Reynolds, D.: The Organization Ontology. https://www.w3.org/TR/vocab-org/ (Jan 2014), accessed 2025-01-27
19. Stickler, P.: CBD - Concise Bounded Description. https://www.w3.org/Submission/CBD/ (Jun 2005), accessed 2021-10-01
20. Sun, L., Tao, Z., Li, Y., Arakawa, H.: ODA: Observation-Driven Agent for integrating LLMs and Knowledge Graphs (Jun 2024). https://doi.org/10.48550/arXiv.2404.07677
21. Xiong, G., Bao, J., Zhao, W.: Interactive-KBQA: Multi-Turn Interactions for Knowledge Base Question Answering with Large Language Models (Jul 2024). https://doi.org/10.48550/arXiv.2402.15131
22. Xu, S., Liu, S., Culhane, T., Pertseva, E., Wu, M.H., Semnani, S., Lam, M.: Fine-tuned LLMs Know More, Hallucinate Less with Few-Shot Sequence-to-Sequence Semantic Parsing over Wikidata. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 5778–5791. Association for Computational Linguistics (2023). https://doi.org/10.18653/v1/2023.emnlp-main.353
23. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y.: ReAct: Synergizing Reasoning and Acting in Language Models. In: The Eleventh International Conference on Learning Representations ICLR 2023. Kigali, Rwanda (Mar 2023). https://doi.org/10.48550/arXiv.2210.03629
24. Zahera, H.M., Ali, M., Sherif, M.A., Moussallem, D., Ngonga Ngomo, A.C.: Generating SPARQL from Natural Language Using Chain-of-Thoughts Prompting. In: Knowledge Graphs in the Age of Language Models and Neuro-Symbolic AI. Studies on the Semantic Web, vol. 60, pp. 353–368. IOS Press, Amsterdam (2024). https://doi.org/10.3233/SSW240028

## A Graf von Data Prompt

Task
You are Graf von Data, an assistant designed to formulate a query based on input from User.

Process
You interact with a knowledge graph in a strict think-act-observe cycle.
1. Think: Analyse the question from User and the descriptions in the representations gathered so far.
2. Act: Choose available actions (listed below) that will best progress towards formulating the query.
3. Observe: HALT. System will provide the result of your chosen actions in the next cycle.
In your response, start with a ‘Think:‘ section followed by an ‘Act:‘ section.

Actions
You can issue one or multiple of the following actions.
* ‘search(keywords: string)‘: Get URIs to resources matching the given ‘keywords‘. Look for URIs that identify entities, i.e., make two calls: ‘search(’foo’)‘ and ‘search(’bar’)‘ instead of ‘search(’foo bar’)‘.
* ‘deref(resource URI: string)‘: Get all triples with ‘resource URI‘ in subject position. Expand (https://www.w3.org/TR/curie)[Compact URIs] to absolute URIs for use with ‘deref()‘.
* ‘query(sparql: string)‘: Evaluate ‘sparql‘ and get solutions. Consider using path expressions, i.e., ‘*‘ or ‘+‘ for paths of transitive properties and ‘|‘ for alternative paths.
You issue run actions concurrently using ‘|‘, e.g., ‘search(’foo’) | search(’bar’)‘.
Format action calls as follows: ‘Act: describe(’http://foo/bar’)‘ or ‘Act: query(’PREFIX : <#> SELECT DISTINCT ?x WHERE ... ’)‘.
Use quotes for arguments.

Chat
Sometimes User does not seek factual information.
* ‘chat(text: string)‘: Issue a ‘chat()‘ action if User does not seek factual information from the knowledge graph.

Conditions
Once you have come up with the final query, check the query via ‘query()‘ and then end.
* ‘success(text: string)‘: Confirm the query as final and optionally give User a concise message explaining the relevant steps you took. Do not mention the query or the query solutions.
If nearing about ten cycles:
* ‘fail(text: string)‘: You have no other choice than to give up. Optionally report to User the relevant steps you tried.

Stages

* Identify key resources in the User question and find URIs via ‘Act: search()‘.
* Next, dereference key resources via ‘Act: deref()‘. Iteratively expand your knowledge about the graph via ‘Act: deref()‘ to gather information required to construct the query. Consider obtaining information about classes and properties via ‘Act: deref()‘, especially domain and range of properties. Consider obtaining information about the graph structure on the level of instances or assertions.
* Once you have collected the information requried to write the query, make sure to include in the query definite descriptions of the key resources. Then, expand the query iteratively with more triple patterns.

Critical Instructions
* Ensure that all URIs used in the query exist in the knowledge graph.
* Only use URIs that have been previously mentioned or discovered via ‘search()‘ or ‘deref()‘.
* Never repeat successfully executed actions, the results will be the same.

## B Specific Class Boundaries SPARQL Query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

CONSTRUCT {
    # subclasses, with type and label
    ?subclass rdfs:subClassOf <{uri}> .
    ?subclass ?p6 ?o6 .

    # parent classes, with type and label
    <{uri}> rdfs:subClassOf ?parent_class .
    ?parent_class ?p4 ?o4 .

    # properties having the specified class as domain or range,
    with type, label, domain, and range
    ?property ?p <{uri}> .
    ?property ?p2 ?o2 .
    # classes in domain or range of such properties, with type and
     label
    ?o2 rdfs:label ?o5 .
    ?o2 rdf:type ?o6 .
}
WHERE {
    {
        # subclasses, with type and label
        ?subclass rdfs:subClassOf <{uri}> .
        OPTIONAL {
            ?subclass ?p6 ?o6 .
```

```
            FILTER (?p6 = rdf:type || ?p6 = rdfs:label)
        }
    } UNION {
        # parent classes, with type and label
        <{uri}> rdfs:subClassOf ?parent_class .
        OPTIONAL {
            ?parent_class ?p4 ?o4 .
            FILTER (?p4 = rdf:type || ?p4 = rdfs:label)
        }
    } UNION {
        # properties having the specified class as domain or range
    , with type, label, domain, and range
        # classes in domain or range of such properties, with type
     and label
        ?property ?p <{uri}> .
        ?property ?p2 ?o2 .
        FILTER (?p = rdfs:domain || ?p = rdfs:range)
        FILTER (?p2 = rdf:type || ?p2 = rdfs:domain || ?p2 = rdfs:
    range || ?p2 = rdfs:label)
        OPTIONAL {
            ?o2 rdfs:label ?o5 .
            ?o2 rdf:type ?o6 .
        }
    }
}
```

## C Specific Object Property Boundaries SPARQL Query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

CONSTRUCT {
    # classes in domain or range of the specified property, with
    type and label
    <{uri}> ?p7 ?o7 .
    ?o7 ?p8 ?o8 .

    # subproperties, with type, label, domain, and range
    ?subproperty rdfs:subPropertyOf <{uri}> .
    ?subproperty ?p5 ?o5 .
    # classes in domain or range of such properties, with type and
     label
    ?o5 ?p6 ?o6 .
```

```
    # parent properties, with type, label, domain, and range
    <{uri}> rdfs:subPropertyOf ?parent_property .
    ?parent_property ?p1 ?o1 .
    # classes in domain or range of such properties, with type and
     label
    ?o1 ?p3 ?o3 .
}
WHERE {
    {
        # classes in domain or range of the specified property,
    with type and label
        <{uri}> ?p7 ?o7 .
        FILTER (?p7 = rdfs:domain || ?p7 = rdfs:range)
        OPTIONAL {
            ?o7 ?p8 ?o8 .
            FILTER (?p8 = rdf:type || ?p8 = rdfs:label)
        }
    } UNION {
        # subproperties, with type, label, domain, and range
        # classes in domain or range of such properties, with type
     and label
        ?subproperty rdfs:subPropertyOf <{uri}> .
        ?subproperty ?p5 ?o5 .
        FILTER (?p5 = rdf:type || ?p5 = rdfs:domain || ?p5 = rdfs:
    range || ?p5 = rdfs:label)
        OPTIONAL {
            ?o5 ?p6 ?o6 .
            FILTER (?p6 = rdf:type || ?p6 = rdfs:label)
        }
    } UNION {
        # parent properties, with type, label, domain, and range
        # classes in domain or range of such properties, with type
     and label
        <{uri}> rdfs:subPropertyOf ?parent_property .
        ?parent_property ?p1 ?o1 .
        FILTER (?p1 = rdf:type || ?p1 = rdfs:domain || ?p1 = rdfs:
    range || ?p1 = rdfs:label)
        OPTIONAL {
            ?o1 ?p3 ?o3 .
            FILTER (?p3 = rdf:type || ?p3 = rdfs:label)
        }
    }
}
```