Leveraging Data Shapes in Large Language Model Contexts for Question Answering on Public and Private Knowledge Graphs

Jan G. Wardenga[1,*], Tobias Käfer[1]

[1]Karlsruhe Institute of Technology (KIT), Kaiserstraße 12, 76131 Karlsruhe, Germany

Abstract

Knowledge Graph Question Answering aims to make structured semantic data accessible through natural language interfaces. While recent Large Language Models can generate SPARQL queries from natural language questions, their effectiveness is limited by their reliance on prior exposure to vocabularies and datasets during pretraining. This work explores how Knowledge Graph specific schema information, so called shape constraints, can be used to provide Large Language Models with context about the dataset to be queried, enabling more accurate and generalizable query generation. We show that ShEx-augmented prompting achieves an F1-score of 0.28 on unseen knowledge graphs, compared to a baseline score of 0.00, demonstrating its ability to generalize beyond the training distribution. A pipeline is developed that integrates knowledge graph data shape extraction, prompt construction, and automatic SPARQL validation. Experiments were conducted on benchmarks with public and proprietary data sets using different state-of-the-art Large Language Models and demonstrate that shape-informed prompting improves the execution accuracy of generated queries. These results suggest that augmentation with data shapes can reduce the dependency on Large Language Model pre-training or fine-tuning on a specific dataset and offer a practical pathway toward domain-agnostic, robust Knowledge Graph Question Answering systems.

Keywords

Knowledge Graphs, Question Answering, Large Language Models, SPARQL Generation, Data Shapes, Prompt Engineering, Semantic Web, Domain Adaptation, Retrieval-Augmented Generation

1. Introduction

Question Answering (QA) systems have a long history, initially focusing on retrieving and extracting answers from unstructured textual sources such as documents or the Web [1]. The goal of QA systems, both then and now, has been to enable end users to leverage data sources while abstracting away the complexity of the underlying retrieval mechanisms. With the advent of large-scale structured Knowledge Bases (KBs) like Freebase [2], Wikidata [3], and DBpedia [4] a new paradigm emerged: answering questions based on structured, machine-readable data [5]. These KBs are commonly represented as Knowledge Graphs (KGs) and structured based on the Web Ontology Language (OWL) and the Resource Description Framework (RDF). They store factual knowledge as triples and can be queried using formal languages such as SPARQL Protocol and RDF Query Language (SPARQL) [6].

The goal of the Knowledge Graph Question Answering (KGQA) domain is to abstract the complexity of the underlying retrieval mechanisms, such as SPARQL, RDF, or OWL, in order to enable non-expert users to access structured information intuitively. KGQA systems aim to make semantic data more accessible to non-expert users, using natural language: a commonly adopted approach is to translate Natural Language Questions (NLQs) into formal query representations, such as SPARQL, thereby bridging the gap between human language and machine-interpretable knowledge [7, 8]. KGQA can also be considered as a special form of Retrieval Augmented Generation (RAG) [9], which traditionally is done using vector data bases, but other RAG approaches based on query generation can be shown to provide fresher and more holistic results [10].

First International TEXT2SPARQL Challenge, Co-Located with Text2KG at ESWC25, June 01, 2025, Portorož, Slovenia.
*Corresponding author.
jan.wardenga@gmail.com (J. G. Wardenga); tobias.kaefer@kit.edu (T. Käfer)
https://github.com/Branchenprimus/ (J. G. Wardenga)
0009-0007-1374-5588 (J. G. Wardenga); 0000-0003-0576-7457 (T. Käfer)

This problem is of considerable relevance due to the widespread adoption of KGs as core infrastructure in many data-driven enterprises. Companies such as Google, Facebook, eBay, and IBM rely on knowledge graphs to improve search capabilities, personalize user interactions, support product discovery, and enhance knowledge extraction and decision-making processes [11].

Despite the richness of large-scale KGs, accessing the knowledge they contain remains challenging. While recent advances in Large Language Models (LLMs) have significantly improved the automatic generation of syntactically correct SPARQL queries from natural language questions, a key limitation remains: the semantic correctness of such queries often depends on ”knowledge” of the underlying graph, including entity identifiers, relations, and constraints specific to the target KG. Furthermore, a critical limitation of current KGQA systems based on LLMs is their reliance on prior exposure to the target KG during pre-training. Many recent approaches implicitly assume that the model has seen the entities, properties, and structural patterns of widely used public graphs such as Wikidata or DBpedia during training, which introduces a bias and hinders generalization [12, 13, 14]. However, this assumption does not hold in industrial or proprietary settings, where custom KGs are often domain-specific, access-restricted, or entirely absent from public training corpora. For such cases, a system is needed that can robustly handle KGQA tasks without requiring the LLM to have been pre-trained on the specific target graph.

To address this issue, this paper builds on recent work proposing the use of KG schema information, so-called data shapes, as a ”semantic footprint” that captures the underlying structure and semantic relationships of the KG, and integrates them into the prompt context of LLMs [8, 12, 7]. Data shapes, expressed for instance via ShEx [15] or SHACL [16], offer structured metadata about the types and properties of entities within a KG [17, 18].

This paper is based on a Master’s thesis that investigated the impact of shape-informed prompting on KGQA performance. As part of this work, a modular LLM-driven pipeline was developed that integrates entity extraction, data shape generation, and query synthesis. The system was subsequently evaluated in the First International TEXT2SPARQL Challenge, where it was evaluated against the CK25 and DB25 benchmarks (see section 3.2). While the foundations of the artifact remain consistent with the challenge submission, this paper presents extended experimental findings and deeper insights derived from the thesis work, with a focus on generalization behavior and methodological implications.

2. Research Objectives

This work is situated in the broader problem space of Knowledge-based Question Answering (KBQA) with a specific focus on the subdomain of KGQA, which addresses the challenge of answering natural language questions over structured KGs.

The main contribution of this work is a comprehensive investigation into the integration of structured schema information, in the form of KG data shapes, into the prompt context of LLMs for improved KGQA. Specifically, this study comprises three core contributions.

First, it provides a comparative analysis of the two most widely used constraint languages, SHACL and ShEx with respect to their impact on the quality of the generated SPARQL queries.

Second, it examines to what extent incorporating tailored KGs data shapes enhances the quality of SPARQL query generation from natural language input, and quantifies the performance improvements over a non shape-informed baseline.

Third, it evaluates the generalizability of the proposed pipeline to previously unseen or proprietary KGs, highlighting its potential to enable robust and comparable KGQA results across proprietary KGs and their corresponding datasets.

Through the development of a modular and extensible experimental pipeline, this work enables systematic evaluation across multiple LLMs, datasets, and data shape configurations, offering novel insights into the interplay between structural schema information and language model prompting in the context of KGQA.

3. Methodology and Variables

The developed artifact is a modular data pipeline that aims to investigate whether incorporating KG data shapes into the prompt context of a LLM improves the quality of SPARQL query generation from natural language questions. The independent variables are the LLMs, the datasets and the data shape types. The pipeline is capable of executing any number of test cases, allowing statistically meaningful sample sizes to be generated for empirical evaluation. It includes an automated evaluation mechanism that calculates the F1-Score, which serves as the dependent variable. The development process followed an objective-centered approach as described in the Design Science Research (DSR) methodology, with a strong focus on iterative prototyping and optimization of the artifact [19]. While the research problem is conceptual in nature, the technical efforts focused on realizing a stable, extensible, and testable system.

3.1. Large Language Models

Three state-of-the-art LLMs from different providers were selected. GPT-4o-mini (following: GPT) was selected for its cost-efficient inference capabilities. DeepSeek-V3 (following: Deepseek) offers a similarly strong performance profile at a lower cost and was included as a promising alternative. As an open-source counterpart, LLaMA-3.3-70B-Versatile (following: Llama) was selected to investigate performance in a possibly fully self-hosted environment. All models were accessed via their respective Application Programming Interfaces (APIs).

3.2. Datasets

The artifact is tailored to the structure of QA datasets in the QALD format [20], which makes it partially domain-specific and not fully generalized. Nevertheless, it supports a wide range of KGs, including Wikidata [3], DBpedia [4], and custom local RDF graphs.

To empirically evaluate the effectiveness of the proposed approach, four datasets were selected, covering both public benchmarks and proprietary knowledge graphs:

- **QALD-9-plus (DBpedia subset)**: A subset of the QALD-9-plus benchmark [20], containing natural language questions and corresponding golden SPARQL queries targeting the DBpedia KG. It supports multilingual evaluation and is widely used in KGQA research.
- **QALD-9-plus (Wikidata subset)**: The complementary subset of QALD-9-plus, containing the same questions as the QALD-9-plus (DBpedia subset) but mapped to Wikidata instead of DBpedia [20]. This facilitates cross-KG comparison using aligned questions.
- **DB25**: A benchmark dataset from the Text2SPARQL Challenge [21], comprising natural language questions and SPARQL queries over the DBpedia KG. It is designed to evaluate LLM-based KGQA systems under realistic linguistic variation and schema complexity.
- **CK25**: A proprietary dataset introduced in the same challenge series [22], based on an enterprise knowledge graph not publicly available. It provides an evaluation scenario with unseen entities and schema, simulating industrial application settings.

To ensure consistency in evaluation, QALD-9-plus (DBpedia subset) and DB25 were consolidated under the unified label `DBpedia`.

It is noteworthy that we could not validate approximately 11% of the Wikidata 6% of the DBpedia and 2% of CK25 answers. In part due to the natural evolution of KGs [20]. For example, the golden query related to the question *"Butch Otter is the governor of which U.S. state?"*:

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
SELECT ?res WHERE { ?res wdt:P6 wd:Q39593 . }
```

currently yields no result, because Butch Otter is no longer serving as a governor. Such queries were classified as invalid and excluded from the F1-Score calculation to ensure that the evaluation is not biased by outdated ground-truth data.

The SPARQL queries in our experiments were executed against live versions of Wikidata and DBpedia during the period from May to June 2025. Due to the nature of these experiments and the reliance on public SPARQL endpoints, we did not persist complete dumps of the knowledge graphs at the time. However, for the sake of reproducibility, we reference the closest available static snapshots: the May 20, 2025 Wikidata dump[1] and the most recent DBpedia Live snapshot available as of August 2025[2]. These versions closely reflect the data state used in our experiments, although slight variations may exist due to the dynamic nature of the endpoints during query execution.

3.3. Data Shapes

Validation or data shape constraint languages are used to ensure data quality in KGs by defining integrity constraints in the form of various KG data shape types. Data shape types are high-level structural constraints that represent entity relations within a KG [18, 23, 24].

Given, the importance of validation schemas for KGs, a couple of different technologies have emerged: SHACL, ShEx, ReSh [18], DSP [25] or SPIN [26, 27]. However, the two most prominent constraint languages, ShEx and SHACL will be discussed in greater detail [28, 17].

SHACL, the Shapes Constraint Language, was developed by the World Wide Web Consortium (W3C) Data Shapes Working Group as a draft community group report for RDF validation [16]. Its design was influenced by earlier technologies such as SPIN and OSLC Resource Shapes. SPIN, in particular, was used for expressing constraints in RDF, primarily within TopQuadrant's TopBraid Composer, and served both as a foundation and precursor to SHACL's formalization [17].

Shape Expressions or short: **ShEx** were proposed as a user-friendly and high-level language for RDF validation. Initially proposed as a human-readable syntax for OSLC Resource Shapes, ShEx grew to embrace more complex user requirements coming from clinical and library use cases [18]. ShEx now has a rigorous semantics and interchangeable representations: JSON-LD and RDF [17].

Respectively adapted, the shape type is configurable in the artifact settings of this works software and supports both SHACL and ShEx formats, enabling the impact analysis of different types of data shapes on the quality of the generated SPARQL queries.

An example of a ShEx data shape can be found in appendix 3.

3.4. ShapeMaps

The ShapeMap standard, developed by the Shape Expressions Community Group, defines a formal mechanism to associate RDF nodes with ShEx shape expressions. A ShapeMap consists of a finite set of shape associations, where each association includes at minimum a node identifier and a data shape label, and may optionally include status metadata (conformant or nonconformant), explanatory reasons, or application-specific information. The primary role of ShapeMaps is to specify validation intentions or outcomes. As input, a ShapeMap determines which node-shape pairs should be checked for conformance against a ShEx schema. As output, it can capture the results of such validation, including whether each node conforms to the associated data shape. [29]

This work exclusively relies on fixed ShapeMaps for the entity based data shape generation. This structure ensures compatibility with ShEx data shape validation engines and supports transparent and reproducible high-level descriptions of the knowledge graphs examined throughout the experiments. Within this work, the fixed ShapeMap serves as the initiating link between the entity extraction and the data shape generation step, which further leads to The SPARQL query generation by augmenting the LLM input prompt.

[1] https://dumps.wikimedia.org/wikidatawiki/20250520/
[2] https://downloads.dbpedia.org/live.tar.gz

The SheXer library is used to automatically infer data shape expressions based on the neighborhood structure of selected entities [30]. The resulting data shapes are derived based on the source entities transformed into a fixed ShapeMap.

It is notable that the SheXer library adopts a slightly modified approach to handling fixed ShapeMaps for the purpose of data shape generation [30]. While ShapeMaps were originally introduced as part of the ShEx ecosystem, primarily to associate RDF nodes with shape expressions for validation purposes SheXer uses them differently. In the official ShapeMap draft, ShapeMaps are intended to specify candidate node-shape pairs as input to the validation process, or to report the conformance results of a ShEx validation engine. In contrast, SheXer utilizes fixed ShapeMaps as an input mechanism for shape induction: the specified RDF nodes serve as anchors for the automatic generation of data shapes. These shapes can be expressed in either ShEx or SHACL syntax, enabling flexible downstream use in validation or shape-informed applications. An example of such a shape map is shown below:

```
<[.]/entity/Q1248784>@<Shape: [.]/entity/Q1248784 = airp.>
<[.]/entity/Q99>@<Shape: [.]/entity/Q99 = Calif.>
<[.]/entity/Q229623>@<Shape: [.]/entity/Q229623 = USA>
```

In this example, the placeholder `[.]` is replaced by `http://www.wikidata.org`. The identifiers: `entity/Q1248784 = airport`, `entity/Q99 = California`, `entity/Q229623 = USA` in this case were derived from the entity extraction process. See listing 3 for an insightful example of what is produced by SheXer. It can be observed, that the part after the "@" in the ShapeMap is simply used as the data shape identifier, while the part before the "@" is parsed and used for the data shape generation.

This configuration is completed by a namespace dictionary that defines common prefixes, along with a specification of namespaces to be ignored in order to avoid overly bloated data shapes. Together, these elements enable the `Shaper` object to generate concise and effective data shapes.

Furthermore in case of the Wikidata KGs, the SheXer library offers a `wikidata_annotation` flag, that resolves entity identifiers to readable names. This option was enabled to generate more semantic clarity in regards to the LLM in In-Context Learning (ICL) capabilities. While shexer provides numerous additional parameters for fine-tuning the output, the chosen configuration was sufficient to meet the requirements of the experiments conducted in this work.

3.5. Evaluation Metric

The F1-Score provides a standard yet informative measure of answer quality by quantifying the harmonic mean of precision and recall. In the context of this work, it is used to evaluate how well the system-generated SPARQL queries approximate the correct result sets defined in the benchmark datasets. A high F1-Score indicates that the LLM was able to generate SPARQL queries that return the correct answers while avoiding incorrect ones, thereby balancing the trade-off between false positives and false negatives.

Following the recommendations by Usbeck et al., a QALD-specific macro-averaged F1-Score is employed to reflect the system's answer quality across all test questions [31]:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

The classification of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) is based on a comparison of the result sets derived from the execution of LLM-generated and golden SPARQL queries provided in the datasets:

- **True Positive (TP)**: Counted if the result set produced by the LLM-generated SPARQL query is identical to the annotated golden answer. The order of results is disregarded.

- **False Positive (FP)**: Counted if the result set of the LLM-generated SPARQL query differs from the golden answer set.
- **False Negative (FN)**: Counted if the LLM-generated SPARQL query yields an empty result set or returns values such as `0`, `null`, or similar indicators of failure.
- **True Negative (TN)**: Would be counted if both the LLM-generated and golden SPARQL queries return empty result sets. However, no such cases were examined in any of the datasets used.

In the context of KGQA, the F1-Score serves as a strict measure of success, rewarding only fully correct responses while penalizing both missing and incorrect results. The F1-Score is a bounded metric ranging from 0 to 1, where a score of 1 indicates optimal performance, meaning the model consistently generates answers that are both precise and complete.

4. Pipeline Architecture

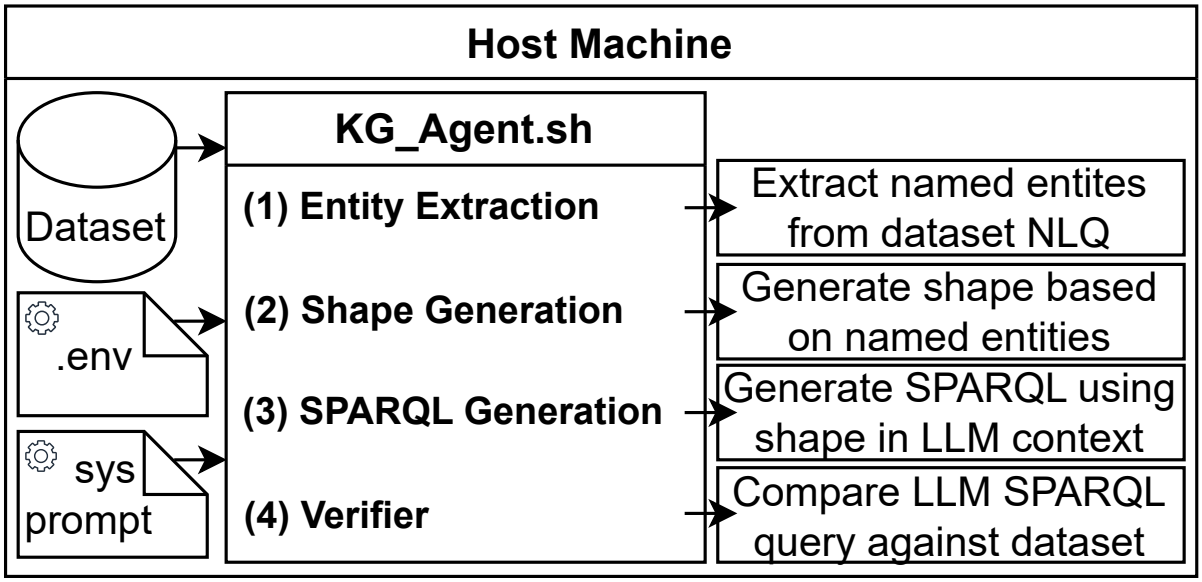

Figure 1: Modular pipeline architecture for shape-informed SPARQL generation. Each step corresponds to a module in the KGQA workflow.

Figure 1 illustrates the experimental pipeline and shows the orchestration shell script that receives information from the dataset, the .env file, and a series of system prompts. The four steps are labeled with numbers (1-4). Each step is responsible for a distinct processing step in the KGQA workflow to support testing in various KGQA scenarios. The artifact is mainly implemented in Python and orchestrated via a Bash script that sets up the structured test environment and logging. This configuration ensures that each test run can be uniquely and reproducibly identified.

- **Step1: Entity Extraction**: Identifies and extracts candidate entities from the translated question using a LLM. The output is used to construct targeted data shape queries in the subsequent step.
- **Step 2: Shape Generation**: Interfaces with the SheXer [32] library to generate a data shape based on a fixed ShapeMap derived from the extracted entities. The resulting data shape captures schema-level constraints relevant to the query context.
- **Step 3: SPARQL Generation**: Constructs a system prompt based on selected dataset, KG and data shape type. Then invokes the selected LLM to generate a candidate SPARQL query.
- **Step 4: Verifier**: Verifies the LLM generated SPARQL query against the dataset golden answer and calculates the F1-Score for each subsequent experiment.

The pipeline is designed to automate the experimental workflow and support testing in various KGQA scenarios. The artifact is mainly implemented in Python using a number of open source libraries, including but not limited to `rdflib` for RDF graph manipulation, `requests` and `SPARQLWrapper` for querying endpoints and openai or for accessing LLM APIs.

4.1. Infrastructure

The pipeline can be run on most host machines with sufficient compute and storage capabilities. The experiments were conducted on a login node in the JUWELS Booster, which consists of 2× AMD EPYC Rome 7402 CPU, 2× 24 cores, 2.8 GHz, Simultaneous Multithreading, 512GB DDR4, 3200 MHz [33].

4.2. Step 1: Entity Extraction

The purpose of the entity extraction is to identify relevant entities from the NLQs provided in the benchmark dataset to be evaluated. The process begins by loading the dataset NLQs. For each entry, the natural language question is combined with a predefined system prompt (see listing 1) and submitted to the configured LLM endpoint for entity extraction.

When testing the pipeline, we found that the proprietary graph linked to the CK25 dataset was small enough to generate a complete data shape, making the entity extraction process obsolete. In such cases, the class skips the extraction step and proceeds directly to the next processing stage. However, since most of the tested datasets were too large to generate complete data shapes, we focused on including entity extraction as the default mode.

When entity extraction is performed, the LLM returns a list of relevant entity names. This list is parsed and forwarded to a SPARQL endpoint for disambiguation and identifier resolution. The result is a dictionary that maps entity names to their corresponding unique identifiers. The dereferencing process is dataset-specific and implemented separately for Wikidata and DBpedia to account for differences in endpoint structure, query formulation, and identifier formats.

Our evaluation does not include a dedicated assessment of the entity extraction step. Since this step precedes shape construction and query generation, errors at this stage can propagate and affect overall performance.

4.3. Step 2: Shape Generation

The shape generation step processes the entities extracted in the previous step and generates corresponding SHACL or ShEx data shapes for each entity. The resulting data shapes are then written to a designated output directory for subsequent use.

The KG data shapes are derived from the extracted entities from step 1 if the target KG did not allow to generate a complete data shape. This enables targeted extraction of structural constraints by guiding the generation process based on pre-selected entities. A more detailed description of KG data shape types and ShapeMaps can be found in section 3.3 and 3.4.

Particular emphasis is placed on the use of the shexer library, a tool capable of generating SHACL or ShEx data shapes based on a variety of configurable parameters [32]. After extensive testing, the optimal configuration for this use case was found to involve initializing the `Shaper` object with a combined fixed ShapeMap [29] is constructed from the previously extracted entities.

4.4. Step 3: SPARQL Generation

The SPARQL generation step queries is responsible for creating a valid SPARQL query that retrieves the information requested by the specified NLQ from the underlying KG.

This construction process involves reading a predefined system prompt template (see listing 2) and replacing its placeholders with context-specific parameters, including the natural language question, the associated data shape, and optional retry parameters. Once the prompt is complete, it is sent to the LLM API, whose endpoint is specified via environment variables.

To ensure the accuracy of the generated queries, a retry mechanism is implemented. A query is considered incorrect if the processed result contains an explicit error message, for example due to an endpoint communication error or SPARQL syntax problems. If the above condition is met, the query is marked as faulty and a retry is triggered. Each retry includes the previously generated faulty query in the new prompt payload, allowing the LLM to revise its output based on the observed error pattern.

Furthermore, the temperature LLM is increased by 0.1 for each consecutive retry, starting from 0 on the first attempt. This implementation was chosen to give the LLM more flexibility in finding a creative solution to the given problem. If none of the error conditions apply, the query is considered valid and the generation step is completed.

The system prompt (Listing 2) includes both a grounding constraint to use only the properties defined in the shape and a fallback instruction to guess missing properties when needed. This was intended to improve answer recall in cases where the extracted shape was sparse or incomplete.

4.5. Step 4: Verifier

The main purpose of this step is to calculate the F1-Score, which serves as the primary evaluation metric for measuring the semantic correctness of LLM-generated SPARQL queries. To achieve this, the program first executes the golden reference queries from the dataset against the corresponding public or local KG endpoint to retrieve the expected set of answer entities. This result set forms the ground truth for comparison.

Subsequently, the generated SPARQL query produced by the pipeline is executed against the same endpoint under identical conditions. The returned entity set is compared to the gold set using set-based operations, disregarding the order of results. The outcome of this comparison is categorized into True Positives, False Positives, and False Negatives on a per-question basis.

These values are used to compute precision, recall, and ultimately the F1-Score, offering a balanced measure that captures both correctness and completeness of query results. In addition, the program logs detailed mismatch information, retry counts, tokens used and other secondary metrics, which facilitates error analysis and robustness evaluation of the query generation pipeline. This comparison method aligns with established practices in KGQA benchmarking, ensuring that results are reproducible and interpretable across datasets and models.

5. Experiments and Results

A total of 36 different experiments were conducted, 24 of which used shape-informed configurations and 12 of which were baseline experiments without data shape specifications. Each of the 36 experiments encompasses 50 different questions from the corresponding dataset minus the disregarded invalid questions (see section 3.2). Each experiment was defined by a unique combination of variables, including the choice of LLM, the dataset and the data shape type.

Table 1
Distribution of variable values across 36 experimental configurations

Feature	Unique Values (Count)
Shape-informed	True (24), False (12)
Model Entity	Deepseek (12), Llama (12), GPT (12)
Dataset	DBpedia (18), Wikidata (9), CK25 (9)
Data Shape Type	ShEx (12), SHACL (12), None (12)

The experiments were executed sequentially, and each run was monitored manually to detect errors, anomalies, or unexpected behavior. All outputs were logged in detail, including the number of LLM retries, token usage, and result metrics.

5.1. Data Shape Type Comparison

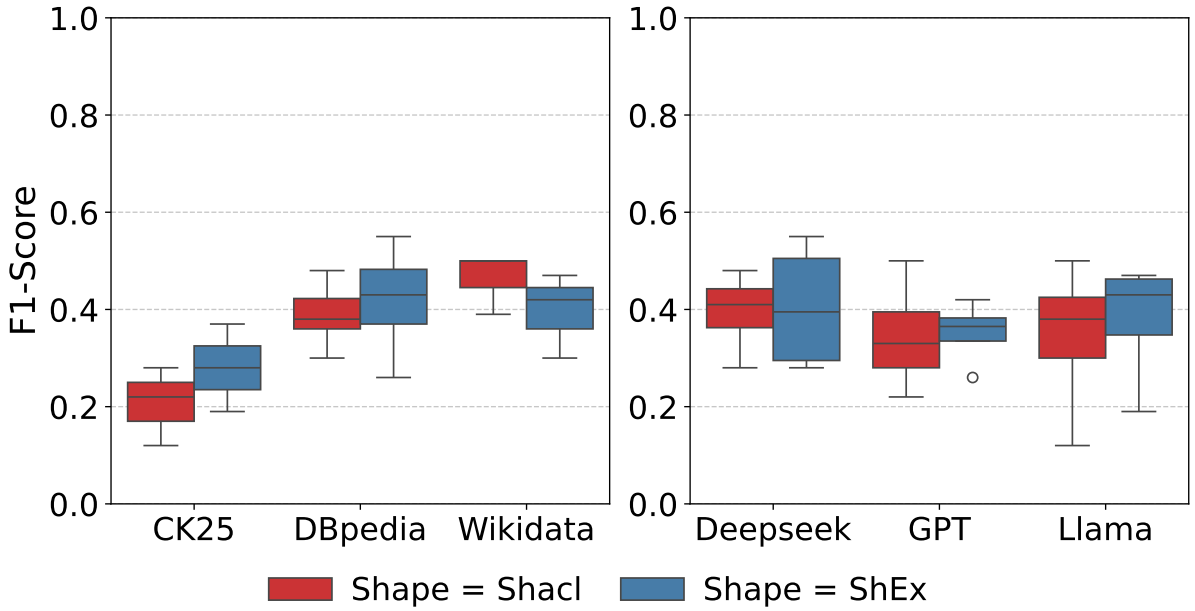

Figure 2: F1-Score by data shape type and dataset (left); F1-Score by data shape type and LLM (right)

Figure 2 compares the F1-Score distributions across data shape types: ShEx and SHACL, grouped by dataset (left) and LLM (right).

On the left, the comparison across the K25 and DBpedia dataset shows that ShEx-based configurations yield slightly higher median F1-Scores than their SHACL counterparts. For Wikidata, a slight advantage of SHACL was found. In case of DBpedia, the spread for ShEx is notably higher than for SHACL. In the case of Wikidata, however, the difference is minor, with SHACL configurations showing less tightly grouped performance with slightly higher medians and higher third quartile performance. For the Corporate Graph, the interquartile range (IQR) and median look similar but shifted up relatively around 27,27%.

On the right, the performance of all language models is presented. For Deepseek, ShEx-based configurations show a noticeably higher upper IQR and maximum F1-Score. However the median is slightly lower. In contrast, GPT essentially performs better on ShEx configurations with the IQR considerably smaller. Llama shows higher median F1 Scores for ShEx as well.

Overall, while both formalisms support comparable central performance levels, ShEx tends to yield slightly higher median F1-scores, particularly on datasets such as DBpedia or when used with models like Deepseek. However, this advantage is accompanied by increased variability in some cases. Conversely, SHACL results appear more variable across benchmarks, suggesting less stable but occasionally more performant behavior.

5.2. Impact of ShEx Shape-Informed Prompting Versus Baseline

The left diagram in figure 3 shows the F1-Score for a data shape configuration set to ShEx compared to non shape-informed configuration. It is notable that shape-informed experiments outperform non shape-informed experiments with a relative increase of 83.33% in median, with the IQR of 0.17 compared to 0.33 being exceptionally low in these experiments.

The analysis of shape-informed SPARQL query generation using ShEx constraints highlights clear performance differences across benchmarks and models (right).

For the Corporate Graph (CK25), GPT achieved the highest median F1-score (0.37), followed by Deepseek (0.28) and Llama (0.19). In the DBpedia benchmark, Deepseek performed best with a median

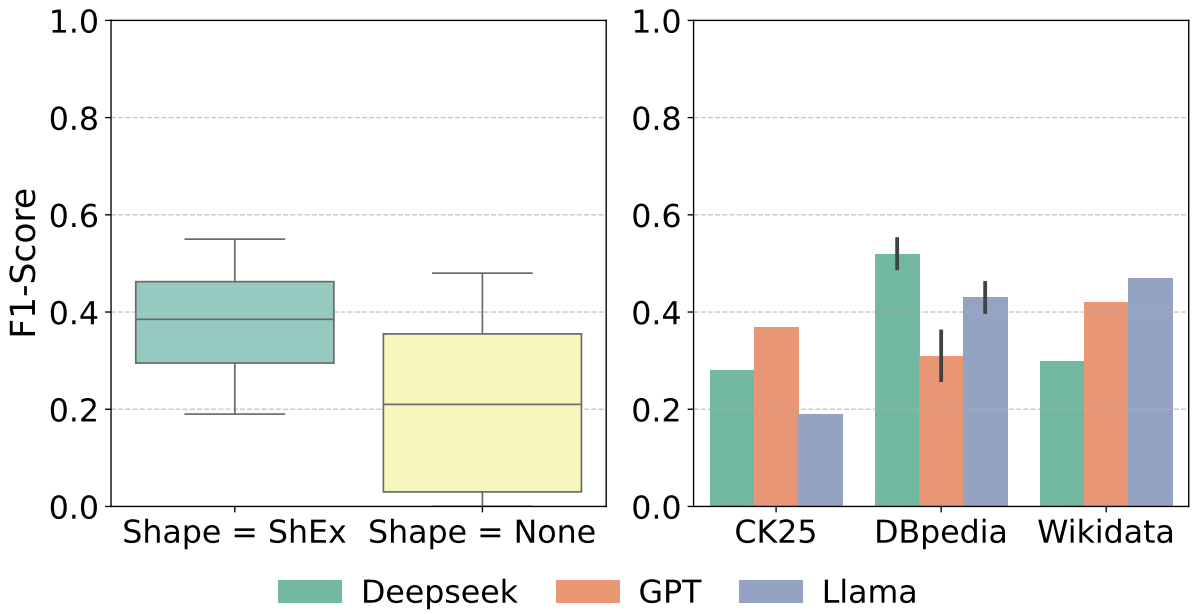

Figure 3: F1-Score by data shape configuration set to ShEx vs non shape-informed (left); F1-Score by LLM and dataset in only ShEx shape-informed configurations (right)

F1-score of 0.52, outperforming Llama (0.43) and GPT (0.31). This benchmark exposes meaningful differences in model behavior and serves as the most discriminative setting in the evaluation. On Wikidata, Llama reached the highest median (0.47), followed by GPT-4o-mini (0.42) and Deepseek (0.30).

In summary, Llama exhibited the strongest performance on Wikidata, while GPT showed leading results in the CK25 benchmark, but with greater variation on DBpedia. Deepseek performs best on DBpedia but comparatively lower on the other datasets. Among the three benchmarks, DBpedia proved most effective in differentiating model quality under ShEx constraints.

5.3. Per-Model and Per-Dataset Effects of Shape Integration

Looking at the F1-Score in figure 4, broken down by the different categories LLMs and datasets, each grouped by shape-informed configuration (ShEx) versus non shape-informed configurations. It can be observed that shape-informed runs consistently achieve better results than their baseline variants.

At the LLM level, it can be observed that Llama benefits particularly strongly from the integration of data shape information: The median F1-Score increases from 0.13 in non shape-informed configurations to 0.43, using data shape information. Similar effects can also be observed with GPT (from 0.25 to 0.37) and Deepseek (from 0.32 to 0.4). At the same time, it is noticeable that the IQR is lower in all ShEx configurations, which indicates more stable generation results.

Visible improvements are also evident at the benchmark level. Particularly noteworthy is Wikidata, whose median rises from 0.16 to 0.42. The effect is also positive for DBpedia, where the median increases from 0.38 to 0.43, with a lower IQR variance at the same time. The proprietary CK25 benchmark shows no output in non shape-informed configuration (F1-Score median = 0), while the integration of shapes allows a median F1-Score of 0.28 to be achieved. This means that the LLMs were not able to generate any positive results, when no data shape information was provided.

Particularly noteworthy is the ability of LLMs to generate consistent and valid results using ShEx data shapes, achieving performance levels that are comparable to those on public datasets. The deviation in median F1-score amounts to only 0.15 points relative to DBpedia and 0.14 points relative to Wikidata, indicating strong generalization despite the proprietary nature of the evaluated dataset.

Overall, the results show that the use of data shapes via ShEx leads to more robust and significantly better response generation in all models and datasets examined, especially in the proprietary dataset

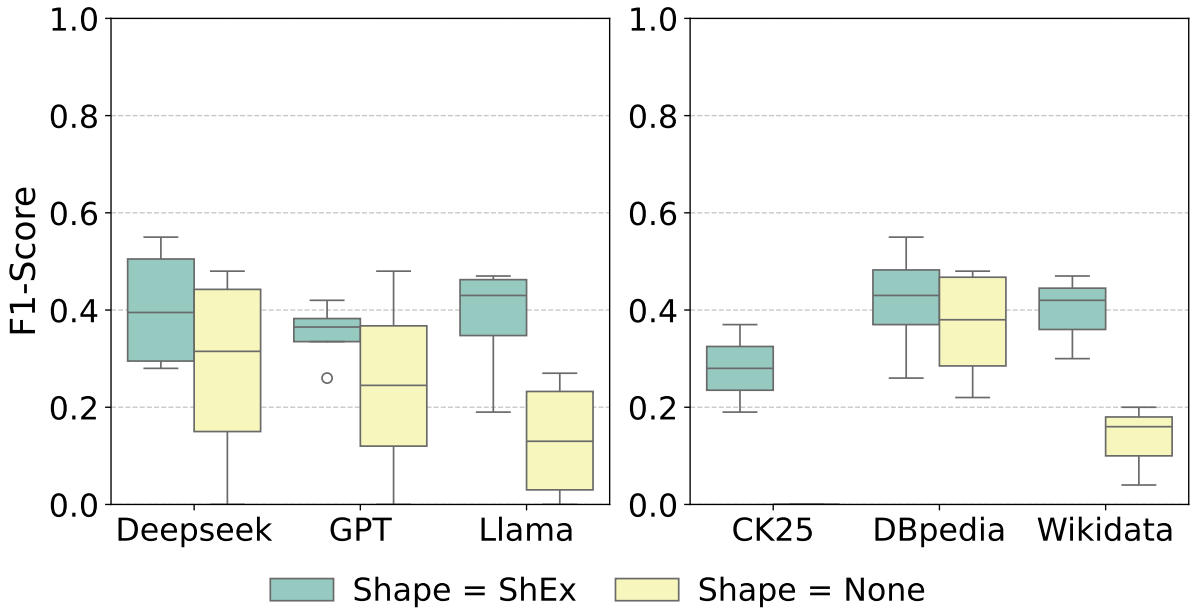

Figure 4: F1-Score by data shape configuration set to ShEx vs non shape-informed (left); F1-Score by LLM and dataset in only ShEx shape-informed configurations (right)

and Wikidata.

6. Discussion and Limitations

Comparing the two data shape languages (Figure 2), ShEx outperforms SHACL across most configurations. The advantage is especially pronounced in the DBpedia and CK25 datasets, and when used in combination with GPT and LLaMA models. However, the relative strength of ShEx must be interpreted with caution, as its superiority may depend on dataset characteristics, annotation richness, or model alignment with the data shape format and possible toolchain bias toward ShEx. Our shape extraction and validation pipeline is based on the SheXer library and relies on ShapeMaps, both of which are rooted in the ShEx ecosystem. This may introduce a systematic bias favoring ShEx over SHACL, particularly if tool-specific optimizations or assumptions influence extraction quality.

The experiments demonstrate that shape-informed prompting generally improves the performance and stability of SPARQL query generation across all tested LLMs and datasets. As shown in Figure 3 (left), configurations enriched with declarative ShEx constraints exhibit a marked improvement over non shape-informed baseline experiments, with an observed median F1-Score increase of 83.33% and a corresponding reduction in IQR. This confirms the hypothesis that explicitly integrating structural knowledge into prompt contexts can improve semantic grounding and guidance during query generation. A potential extension of the baseline could involve incorporating simple schema information, such as a list of commonly used predicates per entity. This would give the LLM some structural guidance while remaining less complex than full-fledged ShEx or SHACL shapes.

Figure 4 further highlights that shape-informed configurations not only achieve higher F1-Scores but also lead to more consistent generation behavior. In all tested models, the IQR decreased under shape-informed conditions, indicating reduced variability and improved reliability. Notably, the proprietary CK25 dataset yielded no valid SPARQL queries in baseline runs, but produced median F1-Scores up to 0.37 when augmented with ShEx constraints. This underlines the role of data shape prompting in enabling generalization to previously unseen or proprietary KGs.

The experiments were limited to three datasets: two public (DBpedia and Wikidata) and one proprietary (CK25). While this selection offers a useful degree of diversity in terms of comparability

and domain specificity, the generalizability of the results to other domains and knowledge graphs remains untested. In particular, the performance gains observed on CK25 may not necessarily extend to other proprietary knowledge graphs. Despite efforts to evaluate the pipeline on additional proprietary datasets, this remains a challenging task due to limited accessibility. By their very nature, proprietary datasets are significantly harder to obtain and share than public benchmarks, making this a persistent limitation, both for this work and for future research.

Public datasets such as DBpedia and Wikidata may have been seen during LLM pretraining, potentially inflating model performance through latent memorization. This confound is partially mitigated by the inclusion of the CK25 benchmark, though further experiments with proprietary or synthetic graphs would strengthen causal interpretation.

Prompt construction was manually engineered and optimized iteratively. While effective in this context, transferability to other tasks, LLMs, or graph structures is not guaranteed. The system prompt induced fallback strategy, mentioned in 4.4 introduces a tension between precision and flexibility. While it helps generate non-empty queries, it also allows the model to hallucinate properties not defined in the shape. A stricter prompting strategy could offer better control, which should be examined in future studies.

Although variance decreased in shape-informed runs, full reproducibility cannot be ensured due to the inherent non-determinism of autoregressive LLMs (e.g., temperature settings, sampling strategies).

Data shapes were generated using SheXer. Although this tool supports large-scale data shape extraction through a streamlined implementation, it may yield more potential by fully utilizing its configuration parameters. Such limitations can affect the completeness and quality of the data shape context provided to the LLM.

7. Conclusion

This work contributes to the advancement of KGQA by demonstrating that integrating declarative data shapes into prompt-based SPARQL query generation significantly improves both accuracy and stability across multiple LLMs and knowledge graphs. Through a unified and reproducible experimental framework, we evaluated the effect of ShEx and SHACL constraints on query generation performance using F1-Score as the primary evaluation metric. Results indicate that shape-informed prompting enables more reliable query generation, particularly for previously unseen or proprietary graphs such as CK25.

We can therefore conclude that the developed technology for generating data shapes enables the generation of results on previously unseen datasets that are comparable in quality to those achieved on publicly available benchmarks, despite the latter being likely included in the pretraining corpus of the underlying language models.

The artifact developed in this study supports modular data shape extraction, prompt enrichment, and automated evaluation, making it suitable for systematic benchmarking across heterogeneous KGs and model configurations. By enabling comparisons across data shape types, models, and datasets, it provides a foundation for further exploration into structure-aware language modeling.

Taken together, these directions point toward a more generalizable, robust, and semantically grounded approach to KGQA using large language models.

Declaration on Generative AI

During the preparation of this paper, generative AI systems and other digital tools were used as a supporting capacity. Their use was limited to improving productivity, translating from and to English, sentence polishing and rephrasing, and clarifying domain-specific concepts. The authors critically reviewed and revised all AI-generated content before its integration. The authors take full responsibility for the paper's content. Specifically, the authors used the following AI systems and tools: ChatGPT (GPT-4, ChatGPT Plus), GitHub Copilot, DeepL Translator.

Acknowledgements

This work has been supported in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 459291153.

References

[1] E. M. Voorhees, D. Harman, Overview of the ninth text retrieval conference (trec-9) question answering track, in: Proceedings of the Ninth Text Retrieval Conference (TREC-9), National Institute of Standards and Technology (NIST), 2000, pp. 1–14. URL: https://trec.nist.gov/pubs/trec9/papers/qa_overview.pdf, editors: Ellen M. Voorhees and Donna Harman.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery (ACM), 2008, pp. 1247–1250. URL: https://doi.org/10.1145/1376616.1376746. doi:10.1145/1376616.1376746, presented at SIGMOD/PODS '08.

[3] D. Vrandečić, M. Krötzsch, Wikidata: A free collaborative Knowledgebase, Communications of the ACM 57 (2014) 78–85. URL: https://doi.org/10.1145/2629489. doi:10.1145/2629489.

[4] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, C. Bizer, DBpedia – a large-scale, multilingual knowledge base extracted from Wikipedia, Semantic Web 6 (2015) 167–195. URL: https://doi.org/10.3233/SW-140134. doi:10.3233/SW-140134.

[5] A. Bordes, N. Usunier, S. Chopra, J. Weston, Large-scale simple question answering with memory networks, arXiv preprint (2015). URL: https://arxiv.org/abs/1506.02075. doi:10.48550/arXiv.1506.02075. arXiv:1506.02075.

[6] E. Prud'hommeaux, SPARQL Query Language for RDF, 2008. URL: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[7] B. Zhang, Y. He, L. Pintscher, A. M. Peñuela, E. Simperl, Schema generation for large knowledge graphs using large language models, 2025. URL: http://arxiv.org/abs/2506.04512. arXiv:2506.04512.

[8] L. Kovriguina, R. Teucher, D. Radyush, D. Mouromtsev, SPARQLGEN: One-shot prompt-based approach for sparql query generation, in: Proceedings of SEMANTICS 2023 EU: 19th International Conference on Semantic Systems, September 20–22, 2023, Leipzig, Germany, 2023. URL: https://github.com/danrd/sparqlgen?tab=readme-ov-file.

[9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive NLP tasks, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Proceedings of the 34th Annual Conference on Neural Information Processing Systems (NeurIPS), 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[10] Y. Li, P. Kerbusch, R. Pruim, T. Käfer, Evaluating the performance of RAG methods for conversational AI in the airport domain, in: W. Chen, Y. Yang, M. Kachuee, X.-Y. Fu (Eds.), Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track), Association for Computational Linguistics, Albuquerque, New Mexico, 2025, pp. 794–808. URL: https://aclanthology.org/2025.naacl-industry.61/. doi:10.18653/v1/2025.naacl-industry.61.

[11] K. S. Aggour, A. Detor, A. Gabaldon, V. Mulwad, A. Moitra, P. Cuddihy, V. S. Kumar, Compound Knowledge Graph-Enabled AI Assistant for Accelerated Materials Discovery 11 (2022) 467–478. doi:10.1007/s40192-022-00286-z.

[12] V. Emonet, J. Bolleman, S. Duvaud, T. M. de Farias, A. C. Sima, LLM-based SPARQL Query Generation from Natural Language over Federated Knowledge Graphs, 2024. URL: http://arxiv.org/abs/2410.06062. arXiv:2410.06062.

[13] X. Huang, J. Zhang, D. Li, P. Li, Knowledge graph embedding based question answering, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, ACM, 2019, pp. 105–113. doi:10.1145/3289600.3290956.
[14] J. Baek, A. Aji, A. Saffari, Knowledge-augmented language model prompting for zero-shot knowledge graph question answering, in: Proceedings of the First Workshop on Matching From Unstructured and Structured Data (MATCHING 2023), Association for Computational Linguistics, 2023, pp. 70–98. doi:10.18653/v1/2023.matching-1.7.
[15] P. Eric, B. Iovka, Labra Gayo, Jose Emilio, Kellogg, Gregg, Shape Expressions Language 2.1, 2019. URL: http://shex.io/shex-semantics-20191008/.
[16] Knublauch, Holger, Kontokostas, Dimitris, Shapes Constraint Language (SHACL), 2017-07-20. URL: https://www.w3.org/TR/2017/REC-shacl-20170720/.
[17] J. E. L. Gayo, E. Prud'hommeaux, I. Boneva, D. Kontokostas, Validating RDF Data, Synthesis Lectures on Data, Semantics, and Knowledge, Springer International Publishing, 2018. doi:10.1007/978-3-031-79478-0.
[18] A. G. Ryman, S. Speicher, A. Le Hors, OSLC Resource Shape: A language for defining constraints on Linked Data 996 (2013) 96. URL: http://ceur-ws.org/Vol-996/.
[19] K. Peffers, T. Tuunanen, M. A. Rothenberger, S. Chatterjee, A Design Science Research Methodology for Information Systems Research 24 (2007) 45–77. doi:10.2753/MIS0742-1222240302.
[20] A. Perevalov, D. Diefenbach, R. Usbeck, A. Both, QALD-9-plus: A Multilingual Dataset for Question Answering over DBpedia and Wikidata Translated by Native Speakers (2022) 229–234. doi:10.48550/arXiv.2202.00120. arXiv:2202.00120.
[21] AKSW, questions_db25.yaml – benchmark questions for Text2SPARQL, 2025. URL: https://github.com/AKSW/text2sparql.aksw.org/blob/develop/docs/benchmark/questions_db25.yaml.
[22] e. GmbH, CK25 corporate knowledge reference dataset for benchmarking text 2 SPARQL question answering approaches, 2025. URL: https://github.com/eccenca/ck25-dataset.
[23] K. Rabbani, M. Lissandrini, K. Hose, SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption, in: Companion Proceedings of the Web Conference 2022, ACM, 2022-04-25, pp. 260–263. doi:10.1145/3487553.3524253.
[24] K. Rabbani, M. Lissandrini, K. Hose, Extraction of Validating Shapes from Very Large Knowledge Graphs 16 (2023) 1023–1032. doi:10.14778/3579075.3579078.
[25] T. Bosch, K. Eckert, Towards description set profiles for RDF using SPARQL as intermediate language, in: Proceedings of the 2014 international conference on dublin core and metadata applications, DCMI'14, Dublin Core Metadata Initiative, 2014, pp. 129–137.
[26] H. Knublauch, J. A. Hendler, K. Idehen, SPIN - overview and motivation, 2011. URL: http://www.w3.org/submissions/2011/SUBM-spin-overview-20110222/.
[27] H. Knublauch, J. A. Hendler, K. Idehen, Sparql inferencing notation (spin) - overview and motivation, 2011. URL: https://www.w3.org/submissions/spin-overview/, w3C Member Submission.
[28] D. Tomaszuk, RDF Validation: A Brief Survey, in: Beyond Databases, Architectures and Structures. Towards Efficient Solutions for Data Analysis and Knowledge Representation, volume 716, Springer International Publishing, 2017, pp. 344–355. doi:10.1007/978-3-319-58274-0_28.
[29] Eric Prud'hommeaux, Thomas Baker, ShapeMap Structure and Language, 2017-07-13. URL: http://shex.io/shape-map-20170713.
[30] D. Fernández-Álvarez, Shexer: Shape extractor for RDF graphs, 2025. URL: https://github.com/weso/shexer.
[31] R. Usbeck, M. Röder, M. Hoffmann, F. Conrads, J. Huthmann, A.-C. Ngonga-Ngomo, C. Demmler, C. Unger, Benchmarking question answering systems 10 (2019) 293–304. doi:10.3233/SW-180312.
[32] D. Fernandez-Álvarez, J. E. Labra-Gayo, D. Gayo-Avello, Automatic extraction of shapes using sheXer 238 (2022) 107975. doi:10.1016/j.knosys.2021.107975.
[33] Stefan Kesselheim, A. Herten, K. Krajsek, J. Ebert, J. Jitsev, M. Cherti, M. Langguth, B. Gong, S. Stadtler, A. Mozaffari, G. Cavallaro, R. Sedona, A. Schug, A. Strube, R. Kamath, M. G. Schultz, M. Riedel, T. Lippert, JUWELS Booster – A Supercomputer for Large-Scale AI Research, 2021. URL: http://arxiv.org/abs/2108.11976. arXiv:2108.11976.

A. Extended Results

Table 2
F1-Score Statistics by Benchmark and data shape type

benchmark	shape_type_label	count	median	mean	std	min	max	q1	q3	iqr
CK25	Shape = None	3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CK25	Shape = ShEx	3	0.280	0.280	0.090	0.190	0.370	0.235	0.325	0.090
CK25	Shape = Shacl	3	0.220	0.207	0.081	0.120	0.280	0.170	0.250	0.080
DBpedia	Shape = None	6	0.380	0.368	0.111	0.220	0.480	0.285	0.467	0.182
DBpedia	Shape = ShEx	6	0.430	0.420	0.103	0.260	0.550	0.370	0.482	0.112
DBpedia	Shape = Shacl	6	0.380	0.388	0.063	0.300	0.480	0.360	0.422	0.062
Wikidata	Shape = None	3	0.160	0.133	0.083	0.040	0.200	0.100	0.180	0.080
Wikidata	Shape = ShEx	3	0.420	0.397	0.087	0.300	0.470	0.360	0.445	0.085
Wikidata	Shape = Shacl	3	0.500	0.463	0.064	0.390	0.500	0.445	0.500	0.055

Table 3
F1-Score Statistics by Model and data shape type

model_entity	shape_type_label	count	median	mean	std	min	max	q1	q3	iqr
Deepseek	Shape = None	4	0.315	0.277	0.222	0.000	0.480	0.150	0.443	0.292
Deepseek	Shape = ShEx	4	0.395	0.405	0.135	0.280	0.550	0.295	0.505	0.210
Deepseek	Shape = Shacl	4	0.410	0.395	0.085	0.280	0.480	0.363	0.443	0.080
GPT	Shape = None	4	0.245	0.242	0.208	0.000	0.480	0.120	0.367	0.247
GPT	Shape = ShEx	4	0.365	0.352	0.067	0.260	0.420	0.335	0.383	0.048
GPT	Shape = Shacl	4	0.330	0.345	0.118	0.220	0.500	0.280	0.395	0.115
Llama	Shape = None	4	0.130	0.133	0.133	0.000	0.270	0.030	0.233	0.203
Llama	Shape = ShEx	4	0.430	0.380	0.130	0.190	0.470	0.348	0.463	0.115
Llama	Shape = Shacl	4	0.380	0.345	0.161	0.120	0.500	0.300	0.425	0.125

Table 4
F1-Score Statistics by Model Entity grouped by LLM and data shape type = ShEx

shape_type_label	count	median	mean	std	min	max	q1	q3	iqr
Shape = None	12	0.210	0.217	0.185	0.000	0.480	0.030	0.355	0.325
Shape = ShEx	12	0.385	0.379	0.107	0.190	0.550	0.295	0.463	0.168

Table 5
F1-Score Statistics by Baseline VS. data shape type = ShEx

benchmark	model_entity	count	median	mean	std	min	max	q1	q3	iqr
CK25	Deepseek	1	0.280	0.280	NaN	0.280	0.280	0.280	0.280	0.000
CK25	GPT	1	0.370	0.370	NaN	0.370	0.370	0.370	0.370	0.000
CK25	Llama	1	0.190	0.190	NaN	0.190	0.190	0.190	0.190	0.000
DBpedia	Deepseek	2	0.520	0.520	0.042	0.490	0.550	0.505	0.535	0.030
DBpedia	GPT	2	0.310	0.310	0.071	0.260	0.360	0.285	0.335	0.050
DBpedia	Llama	2	0.430	0.430	0.042	0.400	0.460	0.415	0.445	0.030
Wikidata	Deepseek	1	0.300	0.300	NaN	0.300	0.300	0.300	0.300	0.000
Wikidata	GPT	1	0.420	0.420	NaN	0.420	0.420	0.420	0.420	0.000
Wikidata	Llama	1	0.470	0.470	NaN	0.470	0.470	0.470	0.470	0.000

Table 6
F1-Score Statistics by Dataset, grouped by data shape type = ShEx or None

benchmark	shape_type_label	count	median	mean	std	min	max	q1	q3	iqr
CK25	Shape = None	3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CK25	Shape = ShEx	3	0.280	0.280	0.090	0.190	0.370	0.235	0.325	0.090
DBpedia	Shape = None	6	0.380	0.368	0.111	0.220	0.480	0.285	0.467	0.182
DBpedia	Shape = ShEx	6	0.430	0.420	0.103	0.260	0.550	0.370	0.482	0.112
Wikidata	Shape = None	3	0.160	0.133	0.083	0.040	0.200	0.100	0.180	0.080
Wikidata	Shape = ShEx	3	0.420	0.397	0.087	0.300	0.470	0.360	0.445	0.085

Table 7
F1-Score Statistics by LLM, grouped by data shape type = ShEx or None

model_entity	shape_type_label	count	median	mean	std	min	max	q1	q3	iqr
Deepseek	Shape = None	4	0.315	0.277	0.222	0.000	0.480	0.150	0.443	0.292
Deepseek	Shape = ShEx	4	0.395	0.405	0.135	0.280	0.550	0.295	0.505	0.210
GPT	Shape = None	4	0.245	0.242	0.208	0.000	0.480	0.120	0.367	0.247
GPT	Shape = ShEx	4	0.365	0.352	0.067	0.260	0.420	0.335	0.383	0.048
Llama	Shape = None	4	0.130	0.133	0.133	0.000	0.270	0.030	0.233	0.203
Llama	Shape = ShEx	4	0.430	0.380	0.130	0.190	0.470	0.348	0.463	0.115

B. System Prompts

```
You are an expert in named entity recognition for knowledge graphs, specializing in {ont}. The goal is, to use these extracted entities as a step in order to generate {ont} shema information based on the extracted entities.

### Task
Extract the most relevant named {ont} entities directly mentioned in the user's question.

### Rules
- Focus **only** on entities that are **literally named** in the question.
- Return **only** a **comma-separated list** of entity names, **no explanations**, **no additional text**.
- Entities must be in **singular form**, even if they appear in plural in the question.
- **Think carefully** about the context, but **respond only** with the exact entity names mentioned.

### Important
- Do not infer related concepts that are not explicitly named.
- Do not add any {ont} specific prefixes.
- Do not include question words or general terms (e.g., "city", "person", "event" unless explicitly named as entities).
- Do not add brackets, quotes, or list markers - **only raw names**, separated by commas.

### Input Format
Question: "{nlq}"
```

Listing 1: System prompt for the entity extraction. {ont} is replaced by the ontology type, {nlq} by the natural language question.

```
You are an expert in structured query languages, specializing in SPARQL. Your only purpose is to generate valid SPARQL queries to query a {ont} graph.

### Task
Generate a valid SPARQL query that answers the user's question, strictly on the provided {shp_typ} shape constraints.

### Rules
- Write your response **in a single line** without any line breaks or additional formatting.
- **Use** the properties and classes defined in the provided {shp_typ} shape.
- If the properties or entities are not found in the {shp_typ} shape, try to best guess them.
- Return **only raw SPARQL code**. No explanations, no comments, no natural language.
- Ensure the query is **syntactically correct** according to SPARQL standards.
- If a previous attempt failed, **reformulate creatively** to find a working alternative.
- **Assume** external knowledge beyond what is available through the shape if necessary.

### Important
- this {shp_typ} shape is derived from a {ont} graph, the generated query must adhere to the {ont} specifics
- Focus purely on constructing the query.
- Do not add headings, bullet points, or any other text output except the query itself.
- Maintain strict compliance with SPARQL syntax.

### Input Format
Question: "{nlq}"

{shp_typ} Shape:
{shp_dat}
```

````
### SPARQL Query:
```sparql
````

Listing 2: System prompt for shape-informed SPARQL query generation. {ont} is replaced by the ontology type, {shp_typ} by the shape language, and {shp_dat} by the concrete shape.

C. Data Shape Example

```
PREFIX ex: <http://example.org/>
PREFIX xml: <http://www.w3.org/XML/1998/namespace/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX shapes: <http://shapes.wikidata.org/>
PREFIX : <http://weso.es/shapes/>

shape: http://www.wikidata.org/entity/Q133270230 = Berlin
{
   rdfs:label  xsd:string
}

shape: http://www.wikidata.org/entity/Q30185 = mayor
{
   rdfs:label  xsd:string  {181};
   wdt:P18 IRI  -->  image
   wdt:P31 IRI  -->  instance of
   wdt:P227 xsd:string {2}  -->  GND ID
   wdt:P244 xsd:string {2}  -->  Library of Congress authority ID
   wdt:P268 xsd:integer  -->  Bibliotheque nationale de France ID
   wdt:P279 IRI {6}  -->  subclass of
   wdt:P373 xsd:string  -->  Commons category
   wdt:P460 IRI {3}  -->  said to be the same as
   wdt:P508 xsd:integer  -->  BNCF Thesaurus ID
   wdt:P511 IRI  -->  honorific prefix
   wdt:P646 xsd:string  -->  Freebase ID
   wdt:P691 xsd:string {2}  -->  NL CR AUT ID
   wdt:P902 xsd:integer  -->  HDS ID
   wdt:P910 IRI  -->  topic's main category
   wdt:P1001 IRI  -->  applies to jurisdiction
   wdt:P1014 xsd:integer  -->  Art & Architecture Thesaurus ID
   wdt:P1036 xsd:float {2}  -->  Dewey Decimal Classification
   wdt:P1225 xsd:integer  -->  U.S. National Archives Identifier
   wdt:P1245 xsd:integer  -->  OmegaWiki Defined Meaning
   wdt:P1296 xsd:integer  -->  Gran Enciclopedia Catalana ID (former scheme)
   wdt:P1343 IRI {6}  -->  described by source
   wdt:P1417 xsd:string  -->  Encyclopedia Britannica Online ID
   wdt:P1709 IRI  -->  equivalent class
   wdt:P1807 xsd:integer  -->  Great Aragonese Encyclopedia ID
   wdt:P1889 IRI {5}  -->  different from
   ... (omitted)
}
```
```

Listing 3: Example ShEx shape with annotation. Note: Some information of the shape was omitted for brevity. Each predicate is annotated with either a cardinality constraint (e.g., `{2}`) or a label (e.g., `--> GND ID`), guiding Large Language Models in understanding the expected structure and semantics of the data.