CEUR-WS.org/Vol-4094/paper4.pdf

ARUQULA - An LLM based Text2SPARQL Approach using ReAct and Knowledge Graph Exploration Utilities

Felix Brei[1,*,†], Lorenz Bühmann[1,*,†], Johannes Frey[1,2,*,†], Daniel Gerber[1,†], Lars-Peter Meyer[1,3,*,†], Claus Stadler[1,†] and Kirill Bulert[1]

[1]*Institute for Applied Informatics at Leipzig University, Goerdelerring 9, 04109 Leipzig, Germany*
[2]*Leipzig University, Germany*
[3]*Chemnitz Technical University, Germany*

Abstract
Interacting with knowledge graphs can be a daunting task for people without a background in computer science since the query language that is used (SPARQL) has a high barrier of entry. Large language models (LLMs) can lower that barrier by providing support in the form of Text2SPARQL translation. In this paper we introduce a generalized method based on SPINACH, an LLM backed agent that translates natural language questions to SPARQL queries not in a single shot, but as an iterative process of exploration and execution. We describe the overall architecture and reasoning behind our design decisions, and also conduct a thorough analysis of the agent behavior to gain insights into future areas for targeted improvements. This work was motivated by the Text2SPARQL challenge, a challenge that was held to facilitate improvements in the Text2SPARQL domain.

Keywords
Text2SPARQL, ReAct, SPARQL, RDF

1. Introduction

Knowledge graphs are a modern approach at storing and linking information, that have made their way into several large projects like Wikidata, DBpedia or the Open Research knowledge graph. Their inherent structure enables the storage of not only the information entities themselves, but also the meaningful relationships between them. Information from such a graph can be retrieved in various ways, including the use of structured query languages like SPARQL, Cypher, or GQL, as well as alternative methods such as faceted browsing. While SPARQL is a powerful tool widely used in the semantic web community, it can be challenging for those without training in these technologies. The last couple of years have shown that large language models (LLMs) can be helpful in translating the intent of a user into a matching SPARQL query, but their precision is still very limited. To facilitate further research in this area, a contest was created by eccenca GmbH where any individual or group could submit a URL that would return a SPARQL query for a given question (and dataset that the question relates to). In this paper, we describe the details of our submission to this contest.

1.1. The TEXT2SPARQL Challenge

The first TEXT2SPARQL Challenge[1] was designed to evaluate systems capable of translating natural language questions into SPARQL queries across multiple datasets and languages. Participants were required to deploy their solutions as publicly accessible RESTful web services. Each registered system had to expose a uniform API interface accepting two GET parameters: `question` (the input in natural

First International TEXT2SPARQL Challenge, Co-Located with Text2KG at ESWC25, June 01, 2025, Portorož, Slovenia.
*Corresponding author.
†These authors contributed equally. Authors in alphabetical order
brei@infai.org (F. Brei); buehmann@informatik.uni-leipzig.de (L. Bühmann); frey@informatik.uni-leipzig.de (J. Frey); LPMeyer@infai.org (L. Meyer)
0009-0008-5245-6655 (F. Brei); 0000-0003-3127-0815 (J. Frey); 0000-0001-5260-5181 (L. Meyer); 0000-0001-9948-6458 (C. Stadler); 0000-0002-1459-3754 (K. Bulert)
© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
[1]https://text2sparql.aksw.org/challenge/

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2025-11-13

language) and `dataset` (a URL identifying the target knowledge graph). The service was expected to return a valid SPARQL query string in response, within a timeout limit of ten minutes.

The evaluation infrastructure orchestrated 250 queries per endpoint during a 5-day testing phase. Two datasets[2] were employed to test different aspects of system performance:

- **DBpedia (DB25)** An English and Spanish subset of DBpedia 2015-10 - a large-scale, multilingual knowledge graph derived from Wikipedia - encompassing 200 selected questions equally split between English and Spanish. The challenge organizers state that they manually curated and validated question-query pairs, with refinements such as `GROUP BY` and `ORDER BY` clauses to ensure structural diversity.
- **Corporate Knowledge Graph (CK25):** A domain-specific KG built from scratch by the challenge organizers. It contained 50 English questions reported to be created manually in consultation with corporate stakeholders to simulate realistic enterprise queries.

The challenge thereby tested scalability and multilingual robustness on open-domain data (DBpedia) while assessing domain adaptation and precision on specialized data (Corporate).

1.2. Contribution

A recent approach to tackle Text2SPARQL is the use of large language models as agents that can traverse the knowledge graph, retrieve information from it, and create SPARQL queries. One such agent is SPINACH, released by Stanford [1]. While SPINACH is an impressive proof of concept implementation targeted at Wikidata, it cannot be easily configured to support other knowledge graphs.

The major contribution of our work consists of the generalization of SPINACH to RDF graphs and adapting and deploying it for the multilingual and multi-KG setting of the TEXT2SPARQL Challenge. To this end, we extended SPINACHs codebase and prompting such that it would use our own knowledge graph exploration utilities instead of Wikidata API endpoints. These utilities were realized w.r.t. RDF and OWL standard vocabularies, such that they can be adapted to work with other RDF knowledge graphs and languages as well. In addition, we conducted a qualitative and quantitative analysis of our approach in the TEXT2SPARQL challenge, combining the evaluation logs of the challenge organizers with ARUQULAs action, observation, and reasoning logs.

2. Related Work

Although the TEXT2SPARQL Challenge represents the first edition under this specific name, there exists earlier work addressing the problem of translating natural language questions into SPARQL queries. This task is related to the field Knowledge Graph or Knowledge Base Question Answering (KGQA & KBQA) [2, 3, 4] and corresponding benchmarks & leaderboards. In the scope of this paper, we focus on LLM-powered SPARQL-based KGQA systems & Text2SPARQL approaches, as well as relevant benchmarks & leaderboards.

Since the rise of LLMs several colleagues evaluated the capabilities around KGQA and SPARQL. Lehmann et al. [5] propose the usage of a controlled natural language as an intermediate step in the Text2SPARQL translation for KGQA. Meyer et al. [6] manually evaluated several KG related capabilities of ChatGPT 3.5 and ChatGPT 4.0, including Text2SPARQL. The LLMs were generating syntactically correct SPARQL queries with semantic problems on bigger KGs (Mondial KG). Kovriguina et al. [7] applied SPARQLGEN to evaluate a Text2SPARQL approach on bigger KGs (Bestiary, Wikidata and DBpedia) with low F1 score. Taffa and Usbeck [8] present a KGQA system that finds similar questions in a dataset and uses the corresponding SPARQL queries for a few shot prompt Text2SPARQL translation on ORKG. This results in a high F1 score, but was tested only for the SciQA benchmark. The potential of small language models is evaluated by Brei et al. [9].

[2]https://web.archive.org/web/20250626113401/https://text2sparql.aksw.org/assets/talks/1-sebastian-tramp-introduction.pdf

Text2SPARQL capabilities are evaluated as well in the LLM-KG-Bench [10, 11, 12]. Here we were able to add several SPARQL SELECT query related tasks and assess the capabilities of more than 40 LLMs [13, 14, 12, 15, 16]. Even for the best state-of-the-art models the F1 scores vary on different datasets.

For Wikidata Liu et al. [1] present SPINACH. They apply a ReAct [17, 18] approach on the Wikidata KG for a new benchmark dataset. They reached a quite good performance and the approach looks promising for us.

There exist several efforts for benchmarking KGQA systems. The KGQA Leaderboard [19][3] collects results for common datasets like LC-QuAD [20, 21], QALD [22]. Other notable datasets are SciQA [23], Beastiary [7] or SPINACH [1]. New datasets for a KG can get generated as well [24]. The GERBIL [25] benchmarking platform helps evaluating KGQA systems on datasets.

3. ARUQULA Approach, Architecture, and Implementation

As the results of SPINACH [1] and the ReAct approach looked quite promising to us, we decided to generalize the existing code to the RDF graphs given in the Text2SPARQL challenge. The system is build around a ReAct [17, 18] configuration implemented with LangGraph[4], a SPARQL endpoint setup realized with RPT[5] & Qlever [26][6], a hybrid search realized with the vector database Qdrant[7], a text search with Lucene, and an API endpoint for the Text2SPARQL challenge.

3.1. ReAct with KG Exploration Utilities

The ReAct (reason and act) approach [17] is built around a graph of actions the LLM can navigate through. ReAct proposes a template consisting of groups of thoughts, actions and observations which make up the prompt. The idea here is to not have the LLM try to solve a given task in a single attempt, but instead allow it to split the task into smaller sub-tasks as it sees fit and give it tools to interact with the task as well as a history of all preceding actions and resulting observations. The available actions and how they interact with the controller (aka. the action graph) are shown in Figure 1.

In the initialization, the language of the query is detected to switch between the English or Spanish DBpedia. At the "controller" action, the LLM can choose to stop and report the final answer or invoke one out of six knowledge graph exploration utilities:

search: search relevant entities. The `search_entity` action offers a lookup for instance data, `search_property` searches for properties and `search_class` searches for classes. The `search_entity` action is implemented as full-text search while the `search_property` and `search_class` actions are implemented with a hybrid vector Search.

inspect: get more details on knowledge graph entities. Either with an excerpt on a given entry with `get_knowledgegraph_entry` or some usage examples with `get_property_examples`. The action `get_knowledgegraph_entry` is implemented with a SPARQL query searching for outgoing edges. The action `get_property_examples` is implemented with a SPARQL query for 5 usage examples for the given property.

execute: use `execute_sparql` to test a SPARQL query on the KG. The action executes the given SPARQL query on the KG and returns the result.

This actions are described as well in the controller prompt we adopted from SPINACH [27] as can be seen in Listing 1. The 'controller' step follows after each search/inspect/execute step to select the next action. This process can be repeated up to 15 iterations which can be changed in the code.

[3]KGQA Leaderboard: https://github.com/kgqa/leaderboard
[4]web page: https://www.langchain.com/langgraph
[5]Repository: https://github.com/Scaseco/RdfProcessingToolkit/tree/master
[6]Repository: https://github.com/ad-freiburg/qlever
[7]web page: https://qdrant.tech/qdrant-vector-database/

Listing 1: Action description from the controller prompt

```
- get_knowledgegraph_entry(entity URI): Retrieves all outgoing edges (linked
  entities, properties) of a specified knowledge graph entity using its full URI. Example:
  ‘http://dbpedia.org/resource/Sufism‘.

- search_entity_by_label(string): Searches the {dataset} knowledge graph for
  individual real-world entities like companies, people, locations, or things (e.g. “Apple”, “Sufism”,
  “Barack Obama”).

- search_property_by_label(string): Searches the {dataset} knowledge graph for
  *properties* (also called predicates or relationships) like “price”, “hasLocation”, or “producedBy”.
  Use this when you’re trying to find the right property to complete a triple.

- search_class_by_label(string): Searches for *classes* (types/categories) in the
  knowledge graph like “Company”, “Service”, “Book”, or “Organization”.

- get_property_examples(property URI): Retrieves a few usage examples of the
  specified property, given as a full URI.

- execute_sparql(SPARQL): Executes a SPARQL query on the {dataset} knowledge graph.
  Use this when you’re confident in your query structure and ready to test a hypothesis.

- stop(): Marks the most recent SPARQL query as your final answer and ends the process.
```

The search/inspect/execute actions all take a single argument. This could be a string to search for, or an entity to look up or a SPARQL query to execute on the SPARQL endpoint. In the Python code this actions get translated into function calls with additional parameters like the name of the KG to use or the language detected in the initialization.

The LLM used needs to have tool support to interact with LangGraph and the tools. After some internal evaluations of various LLMs including Llama, DeepSeek and various GPT variants we decided to use GPT 4.1 mini as the LLM with the best cost-result-ratio for our case.

3.2. A Dual-Strategy Approach for Semantic Grounding

A critical challenge for any natural language to SPARQL system is semantic grounding: the process of accurately mapping ambiguous or varied natural language phrases from a user’s question to the precise, canonical IRIs of classes and properties in the knowledge graph’s schema. This process involves two distinct sub-tasks: resolving conceptual terms (e.g., *“population”*, *“who made this”*) to schema elements (instances of type `owl:Class`, `owl:ObjectProperty`, `owl:DatatypeProperty`), and resolving proper nouns (e.g., *“Berlin”*, *“Google”*) to specific entity instances. Our agent employs a tailored, dual-strategy approach, recognizing that these two tasks have different requirements for precision and semantic nuance.

Strategy 1: Hybrid Vector Search for Schema Entities For grounding conceptual terms against the KG schema, where semantic ambiguity is high, we use the sophisticated hybrid search method, natively supported by the Qdrant vector store[8]. This approach combines dense and lexical search to provide a deep understanding of the user’s intent.

1. Schema Indexing: We first create a searchable index in Qdrant of all schema entities (instances

[8]Qdrant hybrid search query docs: https://qdrant.tech/documentation/concepts/hybrid-queries/

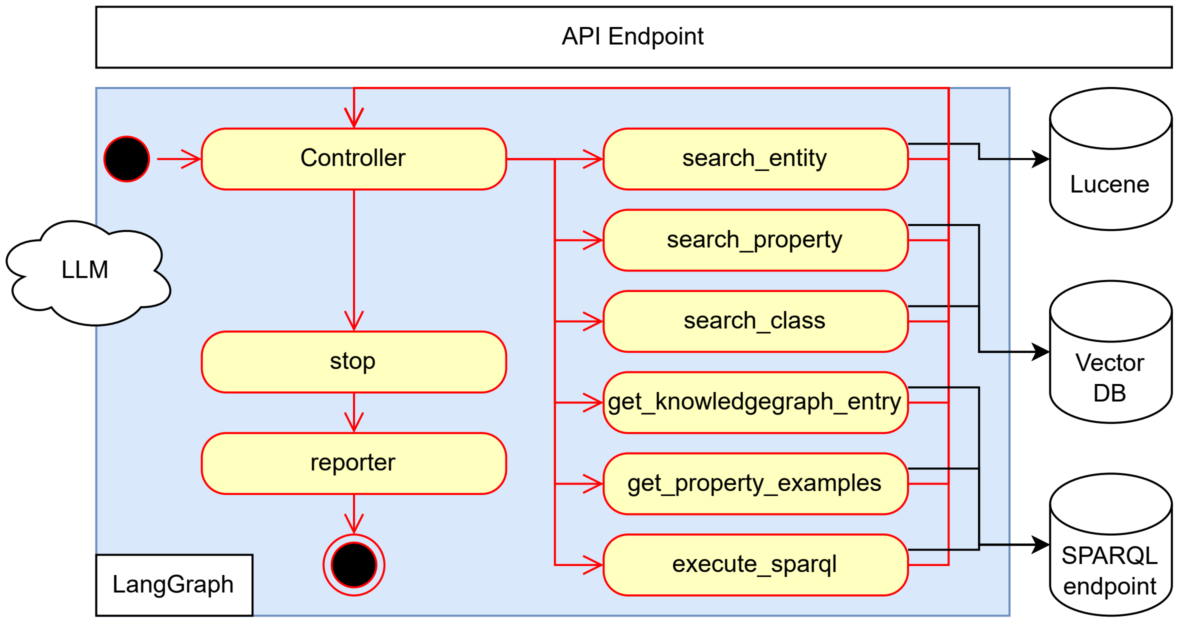

Figure 1: Architecture and Action Graph (in the blue area). At the controller step the LLM can decide whether to search, inspect, execute or stop.

of type `owl:Class`, `owl:ObjectProperty`, `owl:DatatypeProperty`). For each entity, we concatenate its `rdfs:label` and `rdfs:comment` into a single text document. This document is then encoded into two distinct vector representations:

- Dense Vector: A transformer model (BGE Large English[9]) generates a dense embedding that captures the semantic meaning of the entity. This allows for matching based on conceptual similarity.
- Sparse Vector: A BM25 based model[10] generates a sparse, high-dimensional vector that excels at keyword-centric matching, ensuring lexical precision for technical or domain-specific terms.

Both vectors are stored in a Qdrant collection, indexed by the entity's IRI. In addition, we also store the domain and range of properties (if available) in metadata fields of the collection.

2. Agentic Workflow for Schema Grounding: When the agent needs to resolve a term like *"population"*, it generates a dense vector and performs a hybrid query using Reciprocal Rank Fusion (RRF). This robustly identifies the correct schema element (e.g., `dbo:populationTotal`) by balancing semantic relevance with keyword accuracy.

Strategy 2: Full-Text Search for Named Entity Resolution For grounding named entities, the challenge is less about conceptual ambiguity and more about efficiently matching strings against a massive set of instances. For this task, a pragmatic and highly performant full-text search is more appropriate.

1. Instance Indexing: We use a standard Lucene index, a mature and powerful full-text search library. All entity instances from the knowledge graph are indexed. The indexed document for each entity includes its name (`rdfs:label`) and description (`rdfs:comment`) (if available).
2. Agentic Workflow for Named Entity Resolution: When the agent's LLM called in the controller node extracts a proper noun like "Berlin," it does not use the vector store. Instead, it queries the Lucene index. This provides a fast, scalable, and lexically precise method for resolving *"Berlin"* to its canonical IRI, `dbr:Berlin`.

[9]Qdrant BGE large-en model: https://huggingface.co/Qdrant/bge-large-en-v1.5-onnx
[10]Qdrant BM25 sparse embedding model: https://huggingface.co/Qdrant/bm25

By employing this dual strategy, our agent effectively uses the right tool for the right job. It leverages the semantic depth of hybrid vector search for the nuanced task of schema mapping, while relying on the speed and lexical precision of Lucene for the high-volume task of named entity resolution. This division is demonstrated when parsing *"What is the population of Berlin?"*: the agent uses hybrid search to ground *"population"* to `dbo:populationTotal` and Lucene to ground *"Berlin"* to `dbr:Berlin`, obtaining both components needed to construct the final query.

3.3. KG Setup & Querying

In order to realize the inspection and execution functions, we loaded the CK25 and DB25 KGs into a dedicated SPARQL endpoint each (without making use of named graphs). In order to make ARUQULA's RDF data management not only conform to the FAIR principles[11] [28] but also *reproducible* using a conventional build system, we employed our Maven-based data publishing workflow described in [29]. The key aspects that make the Maven ecosystem interact well with Semantic Web technology are summarized as follows:

- Maven artifacts are addressed using *Maven coordinates* which can be represented as URNs of patterns `<urn:mvn:{artifact}>`. A Maven coordinate is resolved using a well-defined conversion that derives a relative URL and prepends it with a repository's base URL. Multiple repositories can be configured for artifact lookups. Changing a repository URL does not necessitate any change in the coordinates of the artifacts.
- Maven builds are extensible using plugins and there already exist many for common tasks, such as for signing artifacts and validating checksums. We created plugins that build and package RDF databases from a set of RDF data dependencies.
- Maven has native support for deployment to local folders[12]. In combination with a generic file server, this can be leveraged as a lightweight approach to (self-)publishing artifacts that does not require maintenance of a dedicated repository system.

For Text2SPARQL, the original dataset downloads were available from the website[13]. We re-published the individual files as Maven artifacts[14], which makes it possible to use them as dependencies in a Maven-aware[15] build process. Initially, we used the `tdb2-maven-plugin`[16] to load the data into an instance of Apache Jena's TDB2 database. While TDB2's performance is sufficient for small datasets, it becomes a bottleneck for larger ones. For this reason, we created the `qlever-maven-plugin`[17], which features building QLever[18] databases (via Docker). QLever is presently among the fastest RDF engines and supports the processing of billions of triples on conventional hardware. Our Maven plugin abstraction makes it easy to build a database for either system with a "push of a button": Regardless whether one uses TDB2 or QLever, an invocation of `mvn package` creates a pre-built database archive, whereas `mvn deploy` deploys the archive as well as the `pom.xml` file to the configured repository. For example, the QLever database is available as yet another Maven artifact[19]. Note, that the deployed `pom.xml` file serves as a historic snapshot that can be downloaded and used to rebuild the database locally at any point in time (provided that the Maven and Docker ecosystems still exist). By default, our plugins place all triples of an RDF dependency into the graph with the artifact's URN. Consequently, the provenance of data in an RDF store can be tracked using the graph name. However, in practice, it is often desirable to merge multiple datasets into a single graph. For this purpose, our plugins also support the specification of the target graph. The pre-created databases are used in two ways: The

[11]FAIR = findable, accessible, interoperable and reusable
[12]`mvn deploy -D'altDeploymentRepository=snapshot-repo::default::file:./repo-folder'`
[13]https://text2sparql.aksw.org/challenge/#corporate-knowledge-small-knowledge-graph
[14]https://maven.aksw.org/archiva/#artifact-details-download-content~internal/org.aksw.data.text2sparql.2025/dbpedia/1.0.0
[15]There are several build tools that can interact with Maven repositories, such as Gradle, SBT, or bld.
[16]https://github.com/Scaseco/tdb2-maven-plugin
[17]https://github.com/Scaseco/qlever-maven-plugin
[18]https://github.com/ad-freiburg/qlever
[19]https://maven.aksw.org/archiva/#artifact-details-download-content~internal/qlever.org.aksw.data.text2sparql.2025/dbpedia/1.0.0

self-contained ARUQULA docker image is built by downloading the QLever database archives directly from our Maven repository. Endpoints are also hosted under our public Apache Jena Fuseki setup[20]. The integration of QLever into Fuseki is part of our JenaX project[21]. Our Maven plugins have been published to Maven Central and are thus globally available for use in builds. Other relevant approaches that combine Maven and Semantic Web are *OntoMaven* [30], which provides plugins for ontology development and management, and the *DataBus* project [31], which features a large data catalogue that reuses Maven concepts.

3.4. Challenge API

As stated in subsection 1.1, all challengers had to provide a uniform API to expose their service to the judges of the challenge. This API was specified via an OpenAPI conforming JSON document found at https://text2sparql.aksw.org/openapi.json and defines a single route `/text2sparql` which accepts two parameters via `HTTP-GET`: The name of the dataset that should be queried (in this case limited to `https://text2sparql.aksw.org/2025/DBpedia/` and `https://text2sparql.aksw.org/2025/corporate/`) along with one URL-encoded question. Assuming a base URL of `http://example.com`, a valid request might look like this:

```
http://example.com/text2sparql \
    ?dataset=https%3A%2F%2Ftext2sparql.aksw.org%2F2025%2FDBpedia%2F \
    &question=Who%20designed%20the%20Python%20programming%20language%3F
```

This API was implemented in Python with the Flask framework[22].

4. Evaluation

For each request that was sent to our agent, we logged every prompt, response, action taken, results and total runtime. Since the challenge consisted of sending 250 such requests, we ended up with a wealth of information.

The first statistic we calculated from data extracted from the agent logs was the average runtime needed to generate a SPARQL query along with the average number of steps that the agent needed to reach this goal. The results can be seen in table 4. Given that the timeout defined by the challenge holders was ten minutes, we can see that our agent clearly stayed well below this limit the whole time.

From the data we extracted, we can run some statistical analysis. We limit ourselves here to questions from `DBpedia-EN` and `Corporate`.

Running a t-test on the number of agent steps between these two datasets reveals indeed a significant difference. The calculation results in a t-value of -2.734 and a $p = 0.00744$ which supports the hypothesis that the agent performs notably different between these two datasets.

However, repeating this calculation for the runtimes gives us a $t = -1.770$ and $p = 0.079$ which hints in the direction that `DBpedia-en` was quicker to be processed than `corporate`, but the evidence is not strong enough to reach that conclusion.

Given that bandwidth and transfer speed between the agent and the SPARQL endpoint is a contributing factor, as well as the processing power of that endpoint to process a SPARQL query, we must assume that there is a lot of noise in the duration data. Going with the findings for the number of agent steps though, it seems promising to explore this direction further.

The second avenue of analysis we want to explore is the behavior of the agent itself. Data scientists follow a certain methodology when searching for new findings in large amounts of data, which consists of drawing samples from the data, analysing their properties, trying to automate this process and finally rolling out this process to a larger portion or even all the data at hand. The functions that were provided

[20]https://copper.coypu.org/#/dataset/text2sparql-2025-dbpedia-qlever/info
[21]https://github.com/Scaseco/jenax
[22]web page: https://flask.palletsprojects.com/en/stable/

Table 1
Average time to generate a SPARQL query from a given question and average number of steps the agent took during generation. The data is grouped by dataset and language. The standard deviation is also shown to illustrate the large spread across the different executions.

	Time to answer (seconds)		Number of Steps by agent	
	mean	std	mean	std
Name of dataset				
corporate	59.620000	25.210858	9.920000	3.515795
DBpedia-en	51.440000	29.395018	8.260000	3.486250
DBpedia-es	66.326531	34.449956	10.520408	3.985100

to the LLM were created with that process in mind, giving the LLM the opportunity to approach the process of answer extraction the same way a human would. But so far, there is no data that conclusively shows that an LLM would indeed follow this process.

Observing the LLM agent trying to answer roughly 250 questions and noting at each step which action was taken how often, we arrive at figure 4. As first step, the agent mostly considered searching around entities from the graph (`search_entity` and `search_class`). Step two is mainly concerned with exploring the surroundings of entities by utilizing `search_property` and `get_knowledgegraph_entry`. In step three and onward, we observed an increasing rate of `execute_sparql`, showing that the agent is ready to send SPARQL queries to the endpoint and expect actionable results.

Figure 4 shows the cumulative amount of actions taken up until a certain step index. Again, we can see that during the first steps there is a steep increase in the total amount of subject related exploration which starts to stagnate a bit after step seven, while the total number of executed SPARQL queries grows at an almost steady pace once the agent is at step three. The most notable takeaway here is that the `stop` action was called about 200 times by the agent on purpose, meaning that for roughly 50 questions, the agent failed to generate a conclusive SPARQL query and had to scrape one from the conversation trace as a last hail mary.

Lastly, we want to investigate further which action precedes which. To this end, we start by looking at each `stop` action and count the actions that came before. We find that all 203 `stop` actions were preceded by `execute_sparql` which shows that the agent always verifies its final SPARQL query before issuing a `stop`. Looking in the other direction, the pie chart in section 4 shows the ratio of each action following an `execute_sparql` step. In one out of four cases, the agent deems the query results satisfying enough to continue with a `stop` action. However, 60% of all SPARQL executions are followed by another execution immediately. The reasons for this encompass a swath of different things, mainly empty result sets, syntactic errors and even timeouts from the SPARQL endpoint. A deeper analysis of why two or more `execute_sparql` actions are chosen consecutively is one goal of our future research. And lastly, we find that in one out of seven cases the agent refrains from executing SPARQL again and instead returns to searching and inspecting entities and properties of the knowledge graph directly. The ratios between the agent doing this on its own accord versus the controller forcing the agent to backtrack because it executed the same step twice, is yet another interesting analysis that shall be done in the future.

5. Discussion

When comparing our system's query results to the expected results for the DB25 dataset (using the evaluation artifact produced by the challenge organizers), we discovered several pitfalls that need to be considered when interpreting the challenge results and the reported systems‘ performances[23].

In case of an automatic evaluation of Text2SPARQL performance via SPARQL result set analysis, the

[23]https://web.archive.org/web/20250627131218/https://text2sparql.aksw.org/assets/talks/8-edgard-marx-result-presentation.pdf

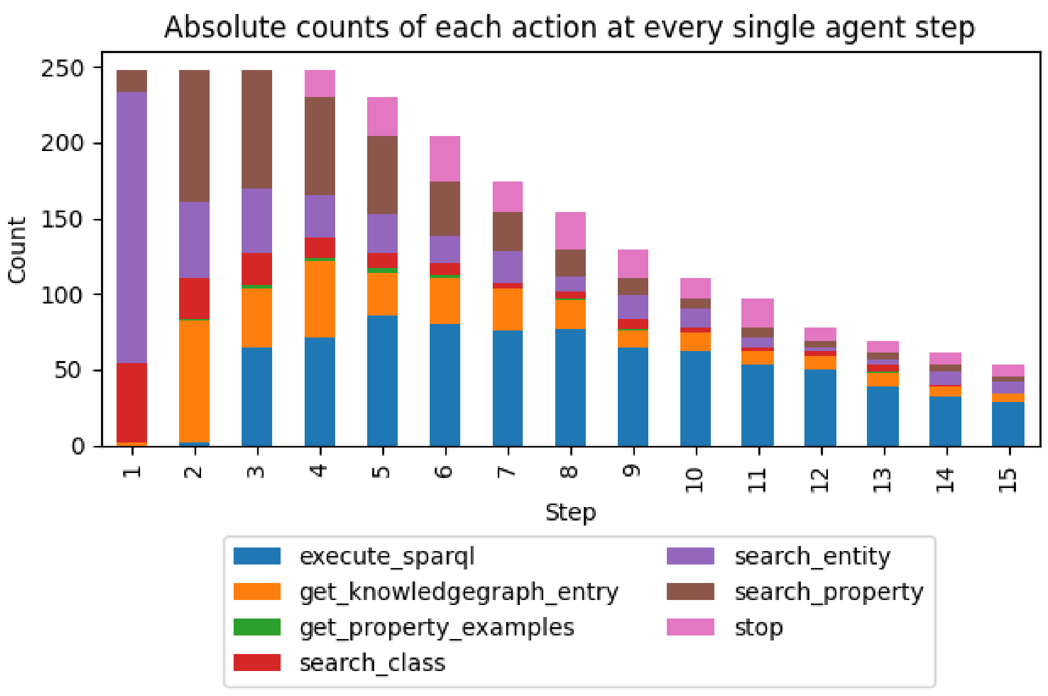

Figure 2: At each individual step of the agent we noted which action was chosen how often from the about 250 queries of the benchmark. The step index is shown on the x axis and the height of the bar corresponds to the number of times the agent has reached that step. The colors indicate the action selected at a step.

query and its result can be viable but do not match with the results of the gold query. As a result, a low score will be assigned, albeit a good and meaningful query (w.r.t. the question) has been proposed. On the other hand, a query can achieve a good score w.r.t. its results but in fact it has been constructed in a way that emulates the answer (e.g. using VALUES clauses and BIND statements in combination with internal knowledge of the LLM) without making proper use of the KG. We could observe instances of misjudged responses of our system for both categories.

When it comes to the first category, the specification of the setup and expected SPARQL query outcome is of significant importance. Natural language questions typically lack precise instructions w.r.t. the nature of the SPARQL result projection. E.g. in case of the question *"What are the 10 most populated countries?"* it is not clear whether the results should return either the entity labels or entity IRIs, or both of them and whether the entities should be reported with the population number, or a further column that specifies the rank. While we consider rank and population numbers as not required, given the question, we would also not see providing it as invalid - given the setup instructions of the challenge. Nevertheless, queries returning labels or population numbers will have a significantly lower score in the challenge. In terms of the nature of DB25, that consists of DBpedia-EN and DBpedia-ES, it is unclear what kind of entities should be queried and returned. Our system was configured in a language-aware way, such that Spanish questions would issue a search for DBpedia-ES entities, but we also saw that the LLM gave preference to Spanish entities during querying by applying Spanish language filters on the entity labels accordingly (thus effectively selecting/querying for DBpedia-ES entities). However, neither the gold queries nor the challenge setup and specification seem to address this appropriately, leading to scores of 0 for viable queries that return DBpedia-ES entities instead of DBpedia-EN.

Another problem occurs in case the question can be semantically grounded in several ways for the KG. Unfortunately, the DB25 KG provides in several instances multiple options to be queried to

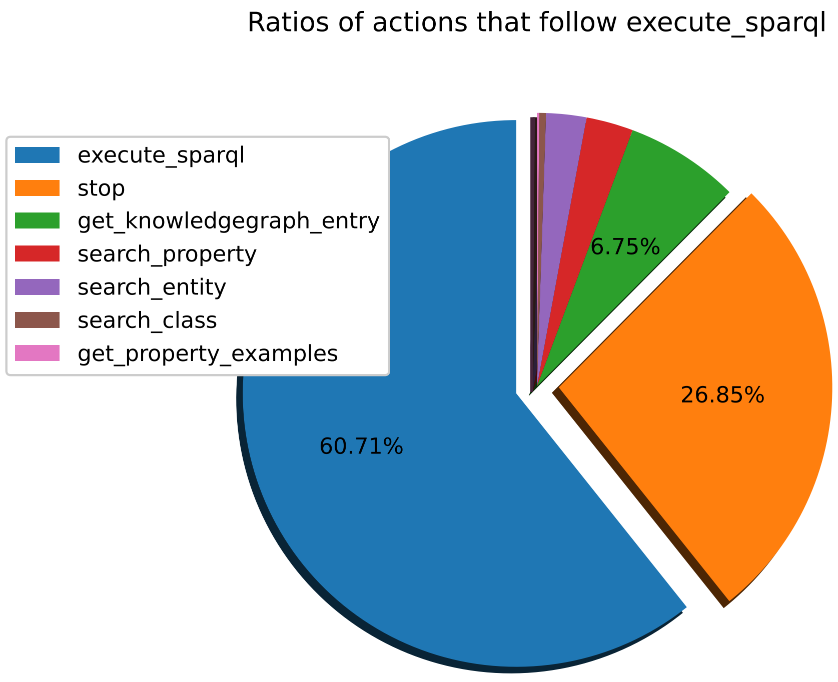

Figure 3: A large portion of actions that the agent takes starting at step three is execute_sparql. This pie chart shows the ratios of actions that come directly after such an action, i.e. in three out of five times it is followed by yet another execute_sparql action and in one out of four times the agent decides to stop afterwards, whereas in one out of seven cases the agent resorts to retrieving information from the knowledge graph directly via search or get functions to refine future queries

answer a particular question. In case of the population question mentioned above, there are at least 4 different property candidates: `dbo:population`, `dbo:populationTotal`, `dbp:population`, and `dbp-es:población`. Unfortunately, all of them return different results. While our approach never used `dbo:population` (which is only used for 7 entities), the Spanish property actually returns factually more accurate results than using the DBpedia Ontology property (which seems to be used in the gold query). When using the ontology property in combination with the `dbo:Country` class, the majority of the top-10 results represent associations of countries like the Commonwealth, due to an issue in DBpedia-EN. Our approach often accounted for such data quality or schema fuzziness issues (that are inherent to the DBpedia extraction and mapping process) and refined the query in that instance such it would e.g. filter out `dbo:Organisations` in the query. This poses a significant capability that even outperforms the correctness of the gold query, yet leading to a lower scoring. We detected several instances where ARUQULA accounted for this with systematic and meaningful constraints on the graph, however, we also found instances where it tried to enforce the "truthfulness" of queries by filtering out result rows with patterns like regex filtering on instance labels or IRIs or faking the outcome with `BIND` statements and `VALUE` clauses. While we consider the enforcement of specific query outcomes an undesirable form of overcompensation in the context of Text2SPARQL, the frequency of this behaviour suggests that both the DB25 dataset and the evaluation setup require refinement to enable precise and reliable automatic evaluation. Unfortunately, the question reported as running example is only one instance from several problematic question-resultset pairs (some of them even contained false empty result sets). Nevertheless, we saw a manual analysis of the behaviour and responses of our system w.r.t. DB25 still as an interesting and insightful study.

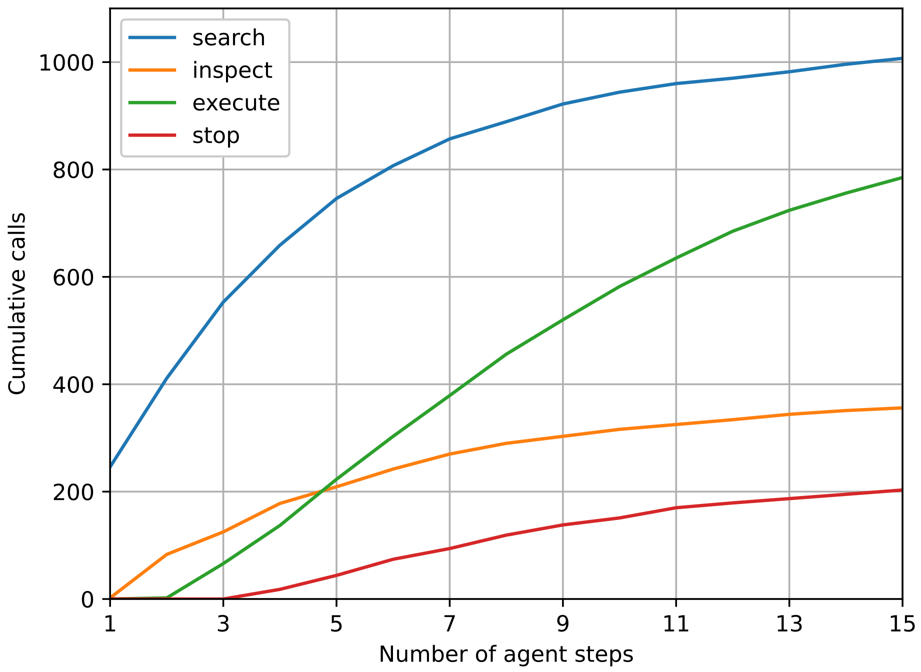

Figure 4: For this plot we grouped the actions into categories to increase readability. It shows the total number of times an action from each category was taken up until a certain step index (i.e. cumulative). For example, we can see that at step seven a total of almost 400 SPARQL queries was executed across all 250 questions. Most notably, only about 200 times total was the stop action called, meaning that circa 50 times the resulting SPARQL query was simply scraped from the conversation trace up until this point without confidence in its correctness.

6. Future Work

In this paper, we introduced ARUQULA, a Text2SPARQL agent built upon SPINACH and enhanced with KG exploration utilities to support RDF-based knowledge graphs beyond Wikidata. Our successful participation in the TEXT2SPARQL challenge demonstrated the potential of the approach, and our evaluation provided insights into agent behaviour, performance characteristics, and limitations. Looking ahead, there are several avenues to extend and refine our work. We consider improvements to latency or increased responsiveness of the agent as a requirement to be used within interactive settings, e.g. a chatbot system. An in-depth comparison of different LLM models in conjunction with a sophisticated selection of test data could help to better understand trade-offs between LLM performance and cost and how the utilities can be enhanced to further improve their helpfulness in order to reduce the number of steps/actions performed. Furthermore, it would be interesting to evaluate whether the approach can be transferred to knowledge graphs from other domains and integrate it into evaluation frameworks like LLM-KG-Bench [12] or GERBIL [32]. Thus, it would be beneficial to improve the automation of setup and deployment so that it can be easily configured and deployed for various knowledge graphs. An evaluation of the ontology grounding by comparing different embedding models, embedding strategies, and search approaches, also in comparison to POTS [33] can be beneficial especially for large knowledge graphs that are not available in the training data of LLMs.

Acknowledgments

This work was partially supported by grants from the German Federal Ministry of Education and Research (BMBF) to the projects ScaleTrust (16DTM312D) and KupferDigital2 (13XP5230L), as well as from the German Federal Ministry for Economic Affairs and Climate Action (BMWK) to the KISS project

(01MK22001A), as well as from the German Federal Ministry of Transport (BMV) to the MobyDex project (19F2266A).

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT to: Grammar and spelling check, paraphrase, and reword to improve the writing style. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Liu, S. Semnani, H. Triedman, J. Xu, I. D. Zhao, M. Lam, SPINACH: SPARQL-based information navigation for challenging real-world questions, in: Y. Al-Onaizan, M. Bansal, Y.-N. Chen (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2024, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 15977–16001. doi:10.18653/v1/2024.findings-emnlp.938.

[2] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann, A.-C. Ngonga Ngomo, Survey on challenges of question answering in the semantic web, Semantic Web 8 (2017) 895–920. doi:10.3233/sw-160247.

[3] B. Fu, Y. Qiu, C. Tang, Y. Li, H. Yu, J. Sun, A survey on complex question answering over knowledge base: Recent advances and challenges (2020). doi:10.48550/ARXIV.2007.13069.

[4] A. Perevalov, A. Both, A.-C. Ngonga Ngomo, Multilingual question answering systems for knowledge graphs – a survey, Semantic Web 15 (2024) 2089–2124. doi:10.3233/sw-243633.

[5] J. Lehmann, S. Ferré, S. Vahdati, Language models as controlled natural language semantic parsers for knowledge graph question answering, in: K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, R. Radulescu (Eds.), ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023), volume 372 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2023, pp. 1348–1356. doi:10.3233/FAIA230411.

[6] L.-P. Meyer, C. Stadler, J. Frey, N. Radtke, K. Junghanns, R. Meissner, G. Dziwis, K. Bulert, M. Martin, LLM-assisted knowledge graph engineering: Experiments with ChatGPT, in: C. Zinke-Wehlmann, J. Friedrich (Eds.), First Working Conference on Artificial Intelligence Development for a Resilient and Sustainable Tomorrow (AITomorrow) 2023, Informatik aktuell, 2024, pp. 103–115. doi:10.1007/978-3-658-43705-3_8.

[7] L. Kovriguina, R. Teucher, D. Radyush, D. Mouromtsev, Sparqlgen: One-shot prompt-based approach for sparql query generation, in: International Conference on Semantic Systems, volume 3526 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3526/paper-08.pdf.

[8] T. A. Taffa, R. Usbeck, Leveraging llms in scholarly knowledge graph question answering, in: QALD/SemREC@ ISWC, 2023.

[9] F. Brei, J. Frey, L.-P. Meyer, Leveraging small language models for Text2SPARQL tasks to improve the resilience of AI assistance, in: J. Holze, S. Tramp, M. Martin, S. Auer, R. Usbeck, N. Krdzavac (Eds.), Proceedings of the Third International Workshop on Linked Data-driven Resilience Research 2024 (D2R2'24), colocated with ESWC 2024, volume 3707 of *CEUR-WS*, 2024. URL: https://ceur-ws.org/Vol-3707/D2R224_paper_5.pdf.

[10] L.-P. Meyer, J. Frey, K. Junghanns, F. Brei, K. Bulert, S. Gründer-Fahrer, M. Martin, Developing a scalable benchmark for assessing large language models in knowledge graph engineering, in: N. Keshan, S. Neumaier, A. L. Gentile, S. Vahdati (Eds.), Proceedings of the Posters and Demo Track of the 19th International Conference on Semantic Systems (SEMANTICS 2023), volume 3526 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3526/paper-04.pdf.

[11] J. Frey, L.-P. Meyer, N. Arndt, F. Brei, K. Bulert, Benchmarking the abilities of large language models for RDF knowledge graph creation and comprehension: How well do llms speak turtle?,

in: M. Alam, M. Cochez (Eds.), Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG 2023) co-located with the 21th International Semantic Web Conference (ISWC 2023), Athens, November 6-10, 2023, volume 3559 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3559/paper-3.pdf.

[12] L.-P. Meyer, J. Frey, D. Heim, F. Brei, C. Stadler, K. Junghanns, M. Martin, LLM-KG-Bench 3.0: A compass for semantic technology capabilities in the ocean of LLMs, 2025.

[13] J. Frey, L.-P. Meyer, F. Brei, S. Gruender, M. Martin, Assessing the evolution of llm capabilities for knowledge graph engineering in 2023, in: The Semantic Web: ESWC 2024 Satellite Events, 2025. doi:10.1007/978-3-031-78952-6_5.

[14] L.-P. Meyer, J. Frey, F. Brei, N. Arndt, Assessing SPARQL capabilities of large language models, in: E. Vakaj, S. Iranmanesh, R. Stamartina, N. Mihindukulasooriya, S. Tiwari, F. Ortiz-Rodríguez, R. Mcgranaghan (Eds.), Proceedings of the 3rd International Workshop on Natural Language Processing for Knowledge Graph Creation co-located with 20th International Conference on Semantic Systems (SEMANTiCS 2024), volume 3874 of *CEUR Workshop Proceedings*, 2024. URL: https://ceur-ws.org/Vol-3874/paper3.pdf.

[15] D. Heim, L.-P. Meyer, M. Schröder, J. Frey, A. Dengel, How do scaling laws apply to knowledge graph engineering tasks? the impact of model size on large language model performance, in: ESWC 2025 Workshops and Tutorials Joint Proceedings, volume 3977 of *CEUR Workshop Proceedings*, 2025. URL: https://ceur-ws.org/Vol-3977/elmke-3.pdf.

[16] L.-P. Meyer, J. Frey, F. Brei, D. Heim, S. Gründer-Fahrer, S. Todorovikj2, C. S. Stadler, M. Schröder, N. Arndt, M. Martin, Evaluating large language models for RDF knowledge graph related tasks - the LLM-KG-Bench-Framework 3, Semantic Web (2025). doi:10.5281/zenodo.16779481, submitted for review 05/2025.

[17] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, React: Synergizing reasoning and acting in language models, in: International Conference on Learning Representations (ICLR), 2023.

[18] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. M. Ni, H.-Y. Shum, J. Guo, Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph (2023). doi:10.48550/ARXIV.2307.07697. arXiv:2307.07697.

[19] A. Perevalov, X. Yan, L. Kovriguina, L. Jiang, A. Both, R. Usbeck, Knowledge graph question answering leaderboard: A community resource to prevent a replication crisis, in: N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk, S. Piperidis (Eds.), Proceedings of the Thirteenth Language Resources and Evaluation Conference, LREC 2022, Marseille, France, 20-25 June 2022, European Language Resources Association, 2022, pp. 2998–3007. URL: https://aclanthology.org/2022.lrec-1.321.

[20] P. Trivedi, G. Maheshwari, M. Dubey, J. Lehmann, Lc-quad: A corpus for complex question answering over knowledge graphs, in: C. d'Amato, M. Fernández, V. A. M. Tamma, F. Lécué, P. Cudré-Mauroux, J. F. Sequeda, C. Lange, J. Heflin (Eds.), The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II, volume 10588 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 210–218. doi:10.1007/978-3-319-68204-4_22.

[21] M. Dubey, D. Banerjee, A. Abdelkawi, J. Lehmann, Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia, in: Proceedings of the 18th International Semantic Web Conference (ISWC), Springer, 2019. doi:10.1007/978-3-030-30796-7_5.

[22] R. Usbeck, X. Yan, A. Perevalov, L. Jiang, J. Schulz, A. Kraft, C. Möller, J. Huang, J. Reineke, A.-C. Ngonga Ngomo, M. Saleem, A. Both, Qald-10 – the 10th challenge on question answering over linked data: Shifting from dbpedia to wikidata as a kg for kgqa, Semantic Web (2023) 1–15. doi:10.3233/sw-233471.

[23] S. Auer, D. A. C. Barone, C. Bartz, E. G. Cortes, M. Y. Jaradeh, O. Karras, M. Koubarakis, D. Mouromtsev, D. Pliukhin, D. Radyush, I. Shilin, M. Stocker, E. Tsalapati, The sciqa scientific question answering benchmark for scholarly knowledge, Scientific Reports 13 (2023). doi:10.1038/s41598-023-33607-z.

[24] F. Brei, L.-P. Meyer, M. Martin, Queryfy: from knowledge graphs to questions using open large language models: Enabling finetuning by question generation on given knowledge, it - Information Technology (2025). doi:10.1515/itit-2024-0079.

[25] R. Usbeck, M. Röder, M. Hoffmann, F. Conrads, J. Huthmann, A.-C. Ngonga-Ngomo, C. Demmler, C. Unger, Benchmarking question answering systems, Semantic Web 10 (2019) 293–304. doi:10.3233/sw-180312.

[26] H. Bast, B. Buchhold, Qlever: A query engine for efficient sparql+text search, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, ACM, 2017, pp. 647–656. doi:10.1145/3132847.3132921.

[27] N. Hu, Y. Wu, G. Qi, D. Min, J. Chen, J. Z. Pan, Z. Ali, An empirical study of pre-trained language models in simple knowledge graph question answering, World Wide Web 26 (2023) 2855–2886.

[28] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The fair guiding principles for scientific data management and stewardship, Scientific data 3 (2016) 1–9.

[29] C. Stadler, L. Bühmann, S. Bin, FAIR data publishing with apache maven, in: L. J. Castro, D. Rebholz-Schuhmann, D. Dessì, S. Schimmler (Eds.), Proceedings of the Fourth Workshop on Metadata and Research (objects) Management for Linked Open Science - DaMaLOS 2024 co-located with Extended Semantic Web Conference (ESWC), PUBLISSO, Hersonissos, Greece, 2024. doi:10.4126/FRL01-006474023.

[30] A. Paschke, R. Schäfermeier, Ontomaven-maven-based ontology development and management of distributed ontology repositories, Synergies Between Knowledge Engineering and Software Engineering (2018) 251–273.

[31] J. Frey, F. Götz, M. Hofer, S. Hellmann, Managing and compiling data dependencies for semantic applications using databus client, in: E. Garoufallou, M. A. Ovalle-Perandones, A. Vlachidis (Eds.), Metadata and Semantic Research - 15th International Conference, MTSR 2021, Virtual Event, November 29 - December 3, 2021, Revised Selected Papers, volume 1537 of *Communications in Computer and Information Science*, Springer, 2021, pp. 114–125.

[32] M. Röder, R. Usbeck, A. N. Ngomo, GERBIL - benchmarking named entity recognition and linking consistently, Semantic Web 9 (2018) 605–625. doi:10.3233/SW-170286.

[33] J. Frey, L. Ferraz, M. Hofer, Pots - a polyparadigmatic ontology term search with fine-grained context steering using hyper-level vector spaces, in: Companion Proceedings of the ACM on Web Conference 2025, WWW '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 2831–2834. URL: https://doi.org/10.1145/3701716.3715194. doi:10.1145/3701716.3715194.

A. Online Resources

Code Repository: https://github.com/AKSW/ARUQULA/tree/aruqula