AIRI team in Text2SPARQL challenge: Text-To-SPARQL Executor for question-answering over knowledge graphs

Daniil Berezin[1], Roman Avdeev[1] and Oleg Somov[1,2]

[1]*Moscow Institute of Physics and Technology, Moscow, Russia*
[2]*AIRI, Moscow, Russia*

Abstract
In this work, we present a pipeline-based solution developed for the ESWC Text-To-SPARQL challenge. Our approach addresses key challenges in knowledge graph question answering (KGQA), including multilingual input processing, entity linking, ontology alignment, and SPARQL query generation. We detail each component of the pipeline and evaluate its contribution through an ablation study. Our system demonstrates strong performance, achieving 3rd place on the DBpedia Knowledge Graph and 5th place on the Corporate Knowledge Graph. The source code for our solution is publicly available at: https://github.com/RomanAvdeev/T2SPARQL.

1. Introduction

Natural language interfaces to databases seek to make structured data more accessible by allowing users to pose questions in natural language and receive precise answers. A common approach involves translating these questions into executable formal queries. This paradigm has been widely explored in two major domains: text-to-SQL for relational databases [1, 2, 3, 4], and text-to-SPARQL for querying semantic knowledge graphs. While the underlying goal is shared, each domain presents distinct challenges [5, 6, 7, 8, 9]. These differences have important implications for system design, evaluation metrics, and the kinds of reasoning required by models. In particular, modern knowledge graphs such as DBpedia [10] and Wikidata [11] encode rich, interconnected information using RDF triples, which introduces additional complexity in terms of ambiguity, sparsity, and ontological reasoning. These KG-specific difficulties compound broader challenges familiar from text-to-SQL research—such as linguistic variability and distributional shift—making the development of accurate and generalizable natural language interfaces especially demanding. To address these challenges, various approaches have been explored, including rule-based pipelines, neural models, and more recently, LLM-based agents [6, 12, 13].

In this paper, we introduce our pipeline-based solution, designed to address four core challenges of Text-To-SPARQL. First, we tackle the issue of multilingual input by integrating GPT-based translation, which significantly improves execution accuracy for non-English queries. Second, our hybrid entity linking module combines DBpedia Spotlight [14] with a GPT-based fallback mechanism to resolve entity ambiguities (e.g., distinguishing "Washington" as a city or state), resulting in improved entity disambiguation and overall performance. Third, we address ontology alignment through two strategies: nearest-neighbor search for DBpedia and Retrieval-Augmented Generation (RAG) [15] applied to turtle files for corporate graphs, achieving improvement in query accuracy. Fourth, our adaptive prompt engineering strategy leverages benchmark datasets such as LcQuad2.0 [16] and QALD-10 [17] to guide SPARQL generation, yielding significant gains in execution accuracy. The query generation module also incorporates SPARQL error-correction loop that automatically detects and repairs malformed queries.

In our work we present two tailored pipelines for completion KGs, its evaluation and ablation study. Our hybrid architecture approach which combines large language models with KG-specific components offers a flexible and competitive alternative to end-to-end and agentic approaches.

First International TEXT2SPARQL Challenge, Co-Located with Text2KG at ESWC25, June 01, 2025, Portorož, Slovenia.
somov@airi.net (O. Somov)

2. Related Work

In the era of LLMs, KGQA systems have emerged as effective tools for enabling natural language interaction with structured data. Query-based KGQA approaches leverage formal query languages such as SPARQL to retrieve precise answers from knowledge graphs, combining the expressiveness of natural language with the rigor of structured querying. The current landscape of KGQA systems can be broadly categorized into three main paradigms: fine-tuning-based, pipeline-based, and agent-based approaches.

Fine-tuning-based methods[18, 19, 6] involve training language models on a specific knowledge graph using aligned question-query pairs. While these models achieve strong performance on in-domain data, they tend to overfit to the training distribution and struggle to generalize under domain shift or when deployed on unseen graphs.

Pipeline-based approaches [20, 21, 12] decompose the SPARQL generation task into modular components—typically including entity recognition and linking, predicate classification, and query construction. This design enables greater flexibility and adaptability to new knowledge graphs, as each module can be independently developed or replaced. However, these systems are susceptible to error propagation due to their sequential nature, where inaccuracies in earlier components can compound in downstream stages.

Agent-based solutions [22, 13, 23] represent a more recent direction in KGQA. These systems use a toolkit-based architecture, where an agent incrementally constructs a SPARQL query by interacting with the knowledge graph through discrete tools or APIs. While promising in terms of interpretability and adaptability, agent-based methods are still maturing and face challenges related to efficiency and tool orchestration.

Our proposed solution adopts a **pipeline-based** architecture enhanced with LLM capabilities via API integration. Each component can be independently optimized and extended—such as incorporating DBpedia Spotlight [14] to improve entity linking. This modular design allows for efficient adaptation to new domains and evolving toolsets. Our system achieved 3rd place on the DBpedia Knowledge Graph [10] and 5th place on the Corporate Graph, demonstrating performance competitive with state-of-the-art approaches[1].

Table 1
The statistics of Text-To-SPARQL challenge datasets.

Metric	DBpedia English	DBpedia Spanish	Corporate
Total questions-query pairs	100	100	50
Average question word count	9.69	9.84	12.36
Average query length	14.24	14.24	31.02
Median hops per query	3.0	3.0	3.0
ORDER BY proportion	0.06	0.06	0.32

3. Challenge objective

The ESWC 2025 Text2SPARQL challenge tasked participants with developing systems capable of translating natural language questions into executable SPARQL queries over RDF knowledge graphs. The competition featured two graphs: the general-purpose multilingual DBpedia, and a proprietary, domain-specific Corporate KG. While the DBpedia evaluation set included questions in both English and Spanish, the presence of multiple languages was not disclosed prior to the final evaluation. The Corporate graph, in contrast, was entirely in English and designed to test generalization to low-resource

[1]In competition we took part under team name MIPT.

domains. As shown in Table 1, a significant portion of the questions in both datasets required multi-hop reasoning, and the Corporate KG featured a notably higher number of ORDER BY clauses, emphasizing the need for systems to handle ranked outputs. Evaluation was performed using Precision, Recall, F1 score, and NDCG, where the latter specifically assessed the quality of ranked results. These metrics jointly measured the correctness, completeness, and relevance of the generated queries.

4. Main method

Our solution is implemented as a dual-branch pipeline, with separate configurations for the two target knowledge graphs: DBpedia and the Corporate Knowledge Graph. Each branch is tailored to the characteristics and constraints of its corresponding dataset, while sharing the same general architecture.

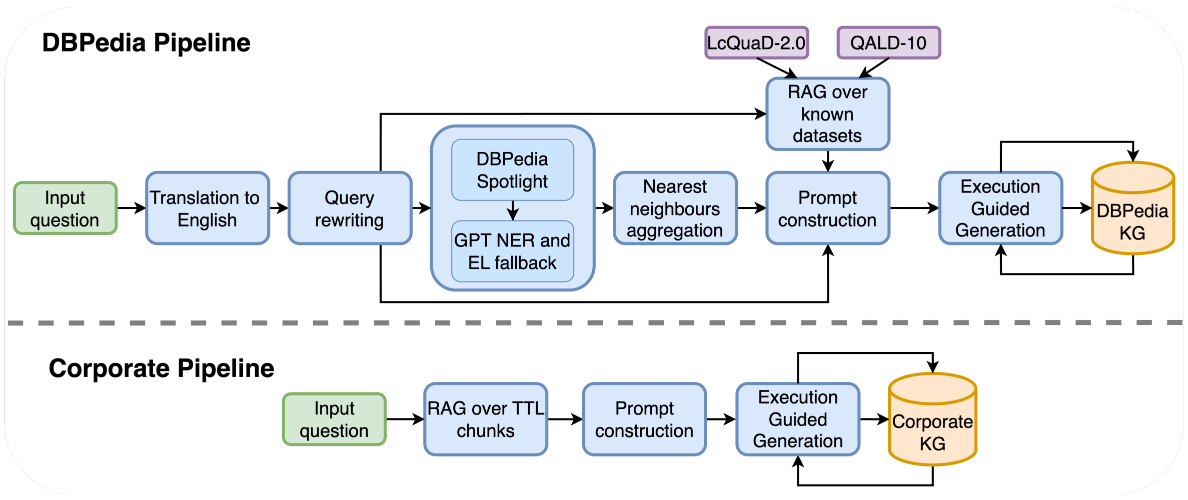

Figure 1: Overview of the DBpedia and Corporate KG Pipelines. The top part illustrates the **DBpedia** pipeline, which includes translation, entity linking (via DBpedia Spotlight and GPT fallback), neighborhood aggregation, and execution-guided SPARQL generation. The bottom part shows the simplified **Corporate** pipeline, which uses retrieval-augmented generation (RAG) over turtle chunks instead of entity linking, followed by prompt construction and query execution.

DBpedia Pipeline. For DBpedia, the pipeline begins with input translation to normalize multilingual queries using a GPT-based translation module. Next, we perform query rewriting, where the question is reformulated to explicitly highlight contextual information and disambiguate entities. Following this, we apply entity recognition and linking to identify relevant entities in the question. For each linked entity, we retrieve its 2-hop neighborhood from the DBpedia graph. This local subgraph structure—comprising nearby entities and predicates—provides contextual grounding for accurate query generation. Then we construct an LLM prompt containing the original question, the set of nearby entities and predicates, and several few-shot examples retrieved from a pre-indexed dataset of known DBpedia question-query pairs. This is achieved by using a Retrieval-Augmented Generation (RAG) approach, which ensures that the LLM is guided by structurally and semantically similar examples. The composed prompt is passed to OpenAI ChatGPT, which generates a candidate SPARQL query. To improve correctness, we employ an execution-guided generation loop, in which the generated query is executed against the DBpedia endpoint and revised if it fails or yields no results.

Corporate Graph Pipeline. For the Corporate Knowledge Graph, which is smaller and more domain-specific, we simplify the pipeline by replacing separate NER, entity linking, and neighborhood retrieval components with a lightweight RAG module over chunked turtle files. This allows us to retrieve relevant entity and predicate context directly from the underlying ontology without pre-linking. The retrieved context is used to construct a tailored prompt, which is then used in the same execution-guided query generation process as in the DBpedia branch.

To evaluate our solution on the DBpedia Knowledge Graph and assess the contribution of individual components, we use subsampled queries from the LcQuad2.0 and benchmark datasets. For the Corporate Knowledge Graph, which lacks publicly available benchmarks, we conduct a manual evaluation to validate system performance. In the following sections, we describe each component of the pipeline in detail and present an ablation study for the DBpedia branch to quantify the impact of each module.

4.1. GPT translation

In the DBpedia track, the input questions were multilingual, requiring a consistent representation in English for downstream processing. To address this, we employed ChatGPT for machine translation. This approach offers greater control compared to traditional translation systems, allowing us to explicitly specify constraints—such as preserving named entities and adapting terminology to the knowledge graph domain. This ensures that critical elements of the question remain intact and contextually appropriate for subsequent entity linking and query generation. The prompt for ChatGPT translation is in Appendix A.

4.2. Query rewriting

After standardizing the input question by translating it into English, the next critical challenge is addressing potential ambiguity in the interpretation of entities. Natural language questions often contain words that can refer to multiple real-world entities, leading to incorrect SPARQL query generation.

To tackle this issue, our system employs a query rewriting step, where an additional request is sent to GPT to clarify the intended meaning of ambiguous terms. For example, given the question "Washington is the capital of what country?", the system identifies that "Washington" could refer to either the U.S. state or the capital city (Washington, D.C.). By prompting GPT to refine question based on the context — e.g., explicitly specifying "Washington, D.C." — we ensure the correct entity is linked to subsequent stages. By resolving ambiguity, the system reduces errors in named entity recognition and entity linking, leading to more precise SPARQL queries. The prompt for the query rewriting is in Appendix B.

4.2.1. DBpedia Spotlight

Following query translation and disambiguation, the next step in our pipeline involves precise entity recognition and linking using DBpedia Spotlight [14], a specialized tool designed to identify and ground named entities directly into the DBpedia knowledge graph. Given the refined natural language question, Spotlight performs joint Named Entity Recognition and Entity Linking, returning not only the detected entities but also their corresponding DBpedia URIs.

To ensure robustness to out-to-distribtuion, the pipeline includes a fallback mechanism: if Spotlight fails (e.g., due to service unavailability or unrecognized entities), the system defaults to a GPT-based query for entity resolution. This hybrid approach balances efficiency with reliability, ensuring uninterrupted processing even in edge cases. The ChatGPT prompt for fallback for entity recognition is in Appendix C.

4.2.2. Nearest neighbors in DBpedia

To further refine the contextual understanding of identified entities, our pipeline retrieves the 2-hop nearest neighbors for each entity from DBpedia's knowledge graph. This step serves two critical purposes. **Ontology Alignment** – by exposing the LLM to the local graph structure around each entity, we compensate for its lack of inherent knowledge about the underlying KG's schema, ensuring generated SPARQL queries adhere to the actual relationships in DBpedia. **Scope Narrowing** – The neighbors act as semantic constraints, additionally reducing ambiguity in query generation. Empirically, this approach boosted execution accuracy, demonstrating its efficacy in bridging the gap between natural language questions and structured KG queries.

4.2.3. Adaptive prompt

To improve the quality of generation, we used adaptive prompt. A RAG system was developed based on pairs (question - query) collected from open datasets QALD-10 and LcQuad2.0. We find the top 5 questions closest in cosine measure to the original one, and add relevant pairs to the prompt. Thus, if the input question implies an ASK query in SPARQL, the adaptive prompt will contain only those question-query pairs in which the query has an ASK structure.

4.2.4. SPARQL generation

At the input stage of the ChatGPT-based SPARQL generation module, we provide the following components: the original natural language question, the entity-annotated version of the question, a set of matched entities with their corresponding DBpedia URIs, the Top-K nearest neighbors (entities and predicates) retrieved from the DBpedia knowledge graph, and a few-shot prompt comprising manually curated question–query pairs, as described in Section 4.2.3.

Using this context, ChatGPT is prompted to generate a corresponding SPARQL query. Given the complexity of our system and the high precision requirements of SPARQL, the generated query is then validated through execution against the DBpedia SPARQL endpoint. If the query execution fails (due to syntax or empty results), we re-invoke the model with all prior inputs, now augmented with the original query and a description of the execution error. The model is then asked to revise the query accordingly. The full query generation prompt is provided in Appendix D.

4.3. Corporate Graph

For the Corporate Knowledge Graph, we adopt a simplified pipeline tailored to the smaller scale and closed-domain nature of the data. Unlike the DBpedia pipeline, which relies on the nearest-neighbor retrieval from a large graph structure, here we employ a RAG approach over pre-processed turtle file chunks representing the corporate ontology.

Given a natural language question, we retrieve the top 10 most relevant turtle chunks based on semantic similarity, same as in Section 4.2.3. This allows us to enrich the prompt with highly relevant, context-specific knowledge, without the need to load the entire graph into LLM context. By selectively incorporating only the most pertinent fragments, the model can generate informed and accurate SPARQL queries while remaining computationally efficient. For query generation, we reuse the execution-guided generation strategy described in Section 4.2.4, enabling the system to iteratively refine the SPARQL output based on runtime feedback from the KG interface.

5. Results

We design a modular pipeline-based solution composed of distinct components, each addressing specific challenges in KGQA. Table 2 presents a reverse ablation study that quantifies the contribution of each component on our validation set. Starting from the complete system with all modules enabled, we observe a consistent drop in execution match accuracy as components are removed. Notably, removing the *nearest neighbors aggregation* module leads to the most significant decline (−12%), highlighting its critical role in grounding the query generation process (see Section 4.2.2). Other components, such as adaptive prompting with retrieval augmentation and the multilingual translation module, also provide meaningful improvements, demonstrating the value of combining LLM capabilities with structured knowledge graph features.

Table 2
Impact of Sequentially Removing Components on the Validation Set. The baseline corresponds to ChatGPT prompted with few-shot exemplars only.

System Variant	Execution Match	
All components included	0.48	
– *Adaptive Prompt with RAG*	0.42	(-0.06)
– *Nearest neighbors aggregation*	0.30	(-0.12)
– *Query Rewriting & Entity Linking module*	0.24	(-0.06)
– *Translation to English*	0.17	(-0.07)

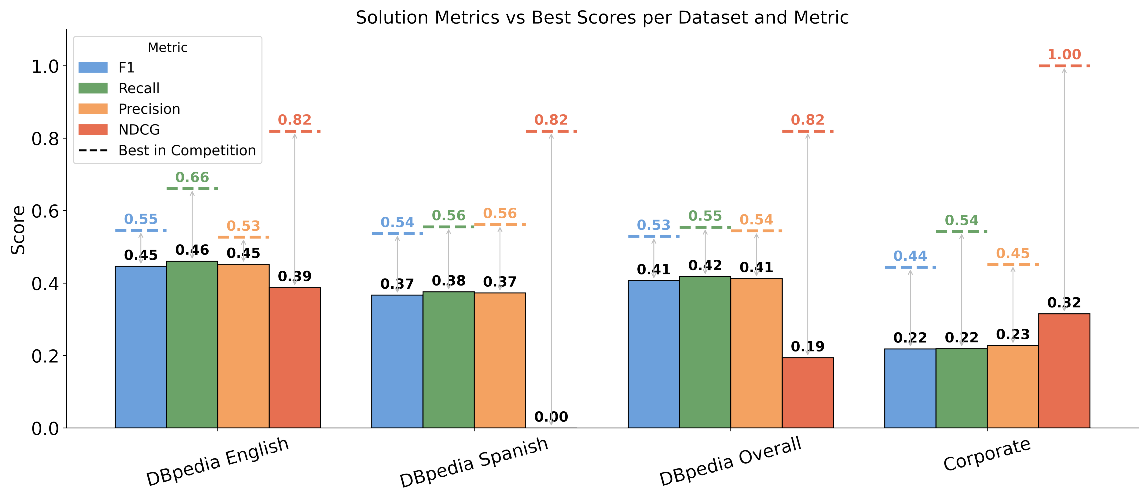

Figure 2: Comparison of the AIRI team's performance to the best scores in the competition across four datasets: DBpedia English, DBpedia Spanish, DBpedia Overall, and Corporate. Each group contains bars for the four evaluation metrics: **F1**, **Recall**, **Precision**, and **NDCG**. Colored dashed lines indicate the best score achieved in the competition for each metric within that dataset. Light grey double-headed arrows visualize the performance gap between the team results and the best competitor.

In Figure 2, we present our final evaluation results based on the official leaderboard published by the organizers[2]. Our system ranked 4th on DBpedia overall, 3rd on DBpedia Spanish, 3rd on DBpedia English, and 5th on the Corporate Knowledge Graph out of 12 participating teams.

The results indicate that our system is highly optimized for the DBpedia domain, but generalizes less effectively to the Corporate KG setting. This performance discrepancy highlights a key limitation of our dual-pipeline approach: while tailored components can boost performance in familiar or well-understood domains, they may struggle to adapt to new or structurally different knowledge graphs. In contrast, top-ranking systems employed agentic approaches that showed more consistent performance across datasets, narrowing the gap between DBpedia and Corporate graphs. This suggests that while our modular components (e.g., ontology alignment, adaptive prompting) are effective, integrating them into more flexible or agent-driven frameworks may improve cross-domain generalization.

Additionally, our current system does not explicitly optimize for result ranking, which negatively impacted our NDCG scores. One potential remedy would be to augment our RAG index with query-response pairs that include ORDER BY clauses, enabling the model to better capture and generate ranked outputs when necessary.

[2]https://github.com/AKSW/text2sparql.aksw.org/tree/develop/docs/results

6. Conclusion

We thoroughly enjoyed participating in the Text2SPARQL challenge Co-Located with Text2KG at ESWC25. One of the key insights gained from this challenge is the clear advantage of agent-based approaches for Knowledge Graph Question Answering (KGQA). Traditional pipeline-based methods—which include tasks such as entity linking and predicate classification—tend to impose rigid constraints on user input and are often brittle in the face of the large scale and structural complexity of real-world knowledge graphs. These hard-coded stages can reduce system robustness, especially when adapting to new domains.

In contrast, agentic systems that dynamically explore the knowledge graph ontology and iteratively refine their hypotheses exhibit behavior more closely aligned with how humans query unfamiliar knowledge sources. Such flexibility is particularly important for domain adaptation and compositional generalization. Another drawback of pipeline-based methods is their sequential structure, where errors in early components propagate and accumulate through the pipeline, ultimately degrading final query quality.

While agent-based systems appear more promising, they are not without fundamental challenges. As shown in Figure 2, even the best-performing solution did not exceed an execution match of 90%, underscoring open problems in the field. These ones include planning effective graph exploration strategies, acquiring a working understanding of new ontologies, managing long-context reasoning and memory, and reliably generating complex queries. Furthermore, entity linking remains a persistent bottleneck: although agents can issue multiple lookups, they still require strong commonsense knowledge and contextual understanding to disambiguate and prioritize entities appropriately.

We thank the organizers of the challenge for an insightful and intellectually stimulating competition that advanced our understanding of scalable and generalizable KGQA systems.

Declaration on Generative AI

Large Language Models (LLMs) were used in this work as an assistive tool for polishing the text, improving clarity, and suggesting alternative phrasings. They were not used for research ideation, experimental design, analysis, or result generation. All scientific contributions, experiments, and conclusions are the responsibility of the authors.

References

[1] O. Somov, The generalization and error detection in llm-based text-to-sql systems, in: Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, WSDM '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 1077–1079. URL: https://doi.org/10.1145/3701551.3707416. doi:10.1145/3701551.3707416.

[2] O. Somov, A. Dontsov, E. Tutubalina, AIRI NLP team at EHRSQL 2024 shared task: T5 and logistic regression to the rescue, in: T. Naumann, A. Ben Abacha, S. Bethard, K. Roberts, D. Bitterman (Eds.), Proceedings of the 6th Clinical Natural Language Processing Workshop, Association for Computational Linguistics, Mexico City, Mexico, 2024, pp. 431–438. URL: https://aclanthology.org/2024.clinicalnlp-1.43/. doi:10.18653/v1/2024.clinicalnlp-1.43.

[3] O. Somov, E. Tutubalina, Confidence estimation for error detection in text-to-sql systems, Proceedings of the AAAI Conference on Artificial Intelligence 39 (2025) 25137–25145. URL: https://arxiv.org/abs/2501.09527. doi:10.1609/aaai.v39i23.34699.

[4] I. Brodskaya, E. Tutubalina, O. Somov, Bridging the gap with RedSQL: A Russian text-to-SQL benchmark for domain-specific applications, in: Proceedings of the 10th Workshop on Slavic Natural Language Processing (Slavic NLP 2025), Association for Computational Linguistics, Vienna, Austria, 2025, pp. 76–83. URL: https://aclanthology.org/2025.bsnlp-1.9/. doi:10.18653/v1/2025.bsnlp-1.9.

[5] 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.

[6] T. Soru, E. Marx, D. Moussallem, G. Publio, A. Valdestilhas, D. Esteves, C. B. Neto, Sparql as a foreign language., in: SEMANTiCS (Posters & Demos), 2017.

[7] A.-K. Hartmann, E. Marx, T. Soru, Generating a large dataset for neural question answering over the dbpedia knowledge base, in: Workshop on Linked Data Management, co-located with the W3C WEBBR, volume 2018, 2018.

[8] D. Bakshandaeva, O. Somov, E. Dmitrieva, V. Davydova, E. Tutubalina, PAUQ: Text-to-SQL in Russian, in: Findings of the Association for Computational Linguistics: EMNLP 2022, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 2355–2376. URL: https://aclanthology.org/2022.findings-emnlp.175/. doi:10.18653/v1/2022.findings-emnlp.175.

[9] O. Somov, E. Tutubalina, Shifted PAUQ: Distribution shift in text-to-SQL, in: D. Hupkes, V. Dankers, K. Batsuren, K. Sinha, A. Kazemnejad, C. Christodoulopoulos, R. Cotterell, E. Bruni (Eds.), Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP, Association for Computational Linguistics, Singapore, 2023, pp. 214–220. URL: https://aclanthology.org/2023.genbench-1.18/. doi:10.18653/v1/2023.genbench-1.18.

[10] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, Dbpedia: A nucleus for a web of open data, in: international semantic web conference, Springer, 2007, pp. 722–735.

[11] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.

[12] D. Yu, S. Zhang, P. Ng, H. Zhu, A. H. Li, J. Wang, Y. Hu, W. Wang, Z. Wang, B. Xiang, Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases, arXiv preprint arXiv:2210.00063 (2022).

[13] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. Ni, H.-Y. Shum, J. Guo, Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph, in: The Twelfth International Conference on Learning Representations, 2024.

[14] P. N. Mendes, M. Jakob, A. García-Silva, C. Bizer, Dbpedia spotlight: shedding light on the web of documents, in: Proceedings of the 7th international conference on semantic systems, 2011, pp. 1–8.

[15] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in neural information processing systems 33 (2020) 9459–9474.

[16] M. Dubey, D. Banerjee, A. Abdelkawi, J. Lehmann, Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia, in: The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II 18, Springer, 2019, pp. 69–78.

[17] R. Usbeck, X. Yan, A. Perevalov, L. Jiang, J. Schulz, A. Kraft, C. Möller, J. Huang, J. Reineke, A.-C. Ngonga Ngomo, et al., Qald-10–the 10th challenge on question answering over linked data: Shifting from dbpedia to wikidata as a kg for kgqa, Semantic Web 15 (2024) 2193–2207.

[18] D. Banerjee, P. A. Nair, J. N. Kaur, R. Usbeck, C. Biemann, Modern baselines for sparql semantic parsing, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 2260–2265.

[19] F. Brei, J. Frey, L.-P. Meyer, Leveraging small language models for text2sparql tasks to improve the resilience of ai assistance (2024).

[20] A. Alekseev, M. Chaichuk, M. Butko, A. Panchenko, E. Tutubalina, O. Somov, The benefits of query-based kgqa systems for complex and temporal questions in llm era, in: International Conference on Applications of Natural Language to Information Systems, Springer, 2025, pp. 426–441.

[21] S. Mohammed, P. Shi, J. Lin, Strong baselines for simple question answering over knowledge graphs with and without neural networks, in: M. Walker, H. Ji, A. Stent (Eds.), Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, Louisiana, 2018, pp. 291–296.

[22] S. Liu, S. Semnani, H. Triedman, J. Xu, I. D. Zhao, M. Lam, SPINACH: SPARQL-based infor-

mation 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. URL: https://aclanthology.org/2024.findings-emnlp.938/. doi:10.18653/v1/2024.findings-emnlp.938.

[23] J. Jiang, K. Zhou, K. Ye, X. Zhao, J.-R. Wen, et al., Structgpt: A general framework for large language model to reason over structured data, in: The 2023 Conference on Empirical Methods in Natural Language Processing, 2023.

A. Translation Prompt

The following prompt was used to translate multilingual inputs into English using ChatGPT:

> You are a precise translation engine. Translate the input to English exactly, preserving:
> 1. All named entities (names, places, titles)
> 2. Technical terms
> 3. Input structure
>
> Output ONLY the translation without commentary.
> Translate this to English exactly:
> [INPUT]

B. Query Rewriting Prompt

The following prompt was used to clarify ambiguous or incomplete user inputs before SPARQL query generation:

> Act as a SPARQL query pre-processor. Your task is to clarify ambiguous / incomplete user input (written below) to ensure they can be accurately translated into a SPARQL query. Focus on disambiguating place names, adding missing context, and correcting assumptions.
> GIVEN INPUT: [INPUT]
> Examples: 1) Input: "Washington is the capital of what country?" Output: "Washington DC is the capital of what country?" (Clarify: Washington state vs. Washington DC)
> 2) Input: "Apple revenue in 2020." Output: "What was Apple Inc.'s revenue in 2020?" (Clarify: Apple the company vs. apple the fruit)
> 3) Input: "USA President during WWII." Output: "Who was the president of the United States during World War II?" (Clarify: country name + World War)

C. LLM Prompts for Entity Recognition and URI Generation

C.1. Entity Recognition Prompt (Simplified)

The following prompt was used to extract named entities and generate intermediate inputs. For the full prompt and logic, refer to:
https://github.com/RomanAvdeev/T2SPARQL/blob/main/text2sparqlAPI/app/t2sparql_dbpedia_prompts.py

> **Entity Recognition Prompt**
>
> Let's think step by step. In the input "Input", we are asked: "...".
> So we need to identify: ....
> The entities are: ....
> So the intermediary input is: "Rewritten input with entity tags"
> Key Requirements: - Maintain EXACT output format including punctuation and spacing
> - Use same entity tags as examples (<relativess>, <restingplace>, etc.)
> - Keep original capitalization in entity names
> - Never add additional explanations outside the template

C.2. URI Generation Prompt (Simplified)

The following prompt was used to map tagged elements from the intermediary input to DBpedia URIs.

URI Generation Prompt

```
Input:
Input: "Original input"
Intermediary: "Intermediary input with entity tags"
Output:
- <entity/class/property> : http://dbpedia.org/[ontology|property|resource]/...
Verification Checklist:
1. For resources: Check exact name matches on DBpedia
2. For classes: Use the most specific type available
3. For properties: Prefer ontology namespace over property
4. Special cases:
- Personal names with initials/middle names
- Geographic names with regional qualifiers
- Technical terms with special characters
```

D. Query Generation and Repair Prompts

D.1. Query Generation Prompt (Simplified)

The following prompt guides the model in generating SPARQL queries from tagged inputs and aligned DBpedia URIs. For full logic and examples, see:
https://github.com/RomanAvdeev/T2SPARQL/blob/main/text2sparqlAPI/app/t2sparql_dbpedia_prompts.py

Query Generation Prompt

```
Input: Original Input: "..."
Input with Entities: "What is the <property> of <entity> ..."
DBpedia URIs:
- <entity> : http://dbpedia.org/resource/...
- <property> : http://dbpedia.org/ontology/...
Thought Process:
Let's think step by step. Identify entity types, properties, filters, and logical structure.
Use variables (?x, ?uri) and connect conditions logically.
SPARQL:
SELECT ... WHERE
... triple patterns ...
FILTER clauses if needed
```

D.2. Query Repair Prompt

This prompt is used when the generated SPARQL query fails due to syntax or execution errors.

Query Repair Prompt

Fix this SPARQL query based on the execution error while preserving intent. Return ONLY the fixed SPARQL query.
Error Context: {ERROR}
Repair Strategies:
1. For empty results:
- Check entity existence
- Replace with broader properties or classes
2. For syntax errors:
- Fix SPARQL formatting issues
- Ensure valid prefix, variables, and nesting
3. URI Adjustments:
- Try alternate URIs or namespaces (dbo: vs. dbp:)
- Use superclasses if specific types fail
4. Always preserve original query intent
Input Query:
{ORIGINAL QUERY}
Context:
- Input: {ORIGINAL INPUT}
- Tagged: {TAGGED INPUT}
- URIs: {URIS}
Output: only the corrected SPARQL query, with no extra text.

D.3. Adaptive Prompt with Few-Shot Examples

To guide the model during query generation, we incorporate few-shot examples that demonstrate how to convert an input with tagged entities into a complete SPARQL query. Each example includes the original input, intermediate tagging, reasoning, and final query structure.

Few-Shot Examples for Adaptive Prompt

Example 1:
Original Input: ”Who wrote the book ’The God Delusion’?”
Tagged: ”Who are the <authors> of <The God Delusion>”
URIs:
- <The God Delusion> : http://dbpedia.org/resource/The_God_Delusion
- <authors> : http://dbpedia.org/ontology/author
Reasoning:
We need the author of a known resource. Direct property lookup suffices.
SPARQL:
SELECT DISTINCT ?author WHERE
<http://dbpedia.org/resource/The_God_Delusion><http://dbpedia.org/ontology/author> ?author
.