Boosting Knowledge Graph Question Answering with Open Source Lightweight Large Language Models and RAG techniques*

Mario Caruso[1,*,†], Giorgia Lodi[2,*,†], Carlo Macis[1,†], Simone Persiani[1,†] and Valentina Presutti[2,3,†]

[1]BUP srl, Rome, Italy
[2]CNR- Institute of Cognitive Sciences and Technologies, Rome, Italy
[3]University of Bologna, Bologna, Italy

Abstract

The Linked Open Data (LOD) ecosystem provides a vast potential for structuring and sharing knowledge. However, accessing and querying this data remains challenging, particularly in contexts like the public sector, where technical capabilities are often limited. To bridge this gap, we introduce a Knowledge Graph Question Answering (KGQA) system that leverages open-source and lightweight Large Language Models (LLMs), along with Retrieval-Augmented Generation (RAG) techniques, to automatically produce SPARQL queries. Our experiments demonstrate the effectiveness of RAG-enhanced few-shot learning and Low-Rank Adaptation (LoRA) fine-tuning for generating precise, ontology-specific SPARQL queries. The presented results are obtained by employing our system within the cultural heritage domain, using ArCo, the Italian Ministry of Culture's open knowledge graph of cultural properties.

Keywords

Knowledge Graph, Question Answering, Retrieval Augmented Generation, Linked Open Data

1. Introduction

Despite its vast potential, the Linked Open Data (LOD) paradigm for data representation is still perceived as a niche technical domain, partly due to the technical difficulties of accessing and querying data via languages like SPARQL. Yet, its open nature suggests that a broader audience including non experts should be able to benefit from it. This creates barriers to wider adoption and utilisation of valuable open datasets [1]. To bridge this gap, there is a pressing need for intuitive tools for SPARQL query generation. Generative AI offers a promising solution in this regard: users can pose questions in natural language, which are then automatically transformed into precise SPARQL queries tailored to specific domain ontologies [2].

To be truly effective while minimizing hallucination risks, such systems based on Large Language Models (LLMs) may require specific fine-tuning of generative models. However, their operational demands present a significant challenge: potential users, including Small and Medium-Enterprises (SMEs) and medium-sized public institutions, may lack the resources and expertise to operate them.

In this scenario, we present a Knowledge Graph Question Answering (KGQA) system that uses open-source, lightweight LLMs and Retrieval-Augmented Generation (RAG) techniques to automatically generate SPARQL queries. Our approach enables effective fine-tuning on specific domains and ontologies, offering accurate natural language-to-SPARQL transformation without prohibitive resource demands. We describe the methodology behind our KGQA system and present preliminary experimental results in a real-world LOD context. Specifically, we demonstrate the effectiveness of RAG-enhanced few-shot learning and Low-Rank Adaptation (LoRA) for generating SPARQL queries over LOD on cultural assets published by the Italian Ministry of Culture. Our results, based on the evaluation of the

RAGE-KG 2025: The Second International Workshop on Retrieval-Augmented Generation Enabled by Knowledge Graphs, co-located with ISWC 2025, November 2–6, 2025, Nara, Japan
*Corresponding author.
†These authors contributed equally.

Notus 7B-v1 model, show that even lightweight LLMs can support reliable access to cultural heritage data for general users.

The rest of this paper is structured as follows. Section 2 presents the state of the art. Section 3 details our methodology for the development of the proposed KGQA system. Section 4 presents the preliminary results obtained using the open ArCo knowledge graph. Finally, Section 5 concludes the paper by outlining future work.

2. Related Work

Research on KGQA has evolved significantly, spanning from early rule-based approaches to LLM methods [3]. Prominent datasets like QALD [4], LC-QuAD [5], and Mintaka [6] have been foundational for evaluating these systems, with the latter emphasizing natural and multilingual queries for complex SPARQL generation. Neural KGQA systems have increasingly incorporated pre-trained language models to improve generalization and linguistic flexibility [7]. However, most solutions rely on heavyweight LLMs (e.g., GPT-3, T5), which limit accessibility due to computational costs. Our work employs LoRA [8] to enable fine-tuning of smaller models, like Notus 7B-v1[1], without full retraining. Retrieval-Augmented Generation (RAG) [9] has also proven effective for enhancing few-shot prompting [10], and it is gaining traction in semantic QA to provide context from KGs rather than from unstructured text [11]. Our system combines this with prompt engineering and lightweight adaptation for a resource-conscious deployment scenario.

3. Methodology

The methodology we follow consists of the following steps: (1) analysis of the domain ontology of reference. This step serves as the basis for defining relevant user questions and corresponding SPARQL queries; (2) analysis of open source LLMs. This analysis is meant to select models that are capable and lightweight enough to allow us to carry out the necessary fine-tuning operations with limited computational resources, while still obtaining acceptable results; (3) construction of the training and test dataset; (4) fine-tuning of the selected LLMs based on the dataset.

3.1. Methodology application to the Italian LOD on cultural heritage

As mentioned earlier, we have applied the methodology in the real-case context of the Italian cultural heritage domain. To this regard, there exists a network of ontologies that have been developed over years; namely, Cultural-ON[2] on cultural institutes and events [12] and ArCo[3] on single cultural properties [13]. These ontologies have been used to build a knowledge graph of cultural properties, both movable and immovable.

Question Design In this context, we have started from the Cultural-ON ontology, selecting two main classes of it: `CulturalInstituteOrSite` and `CulturalHeritageObject`. These classes are richly represented in the data and semantically interconnected with other entities. In fact, from their definition, we have chosen a set of linked entities such as locations, ticket pricing, opening conditions and visual depiction. These ontology's elements allow us to model user questions and SPARQL queries.

Testing and LLM selection Open-source LLMs vary in size and capabilities. At the time of this study, models in the 7B parameter range have been considered the smallest that could be reliably used in real-world application scenarios. These models typically support input contexts of 2048 to 4096 tokens, sufficient for few-shot learning but not for full ontology ingestion. For the sake of conciseness,

[1] https://huggingface.co/argilla/notus-7b-v1
[2] https://dati.cultura.gov.it/cultural-ON/ENG.html
[3] http://wit.istc.cnr.it/arco/primer-guide-v2.0-en.html

we do not report the comparative analysis of the models, also employing RAG techniques, we have conducted. However, based on it, we have selected Notus 7B-v1 that outperformed alternatives under similar conditions (i.e. LLaMa 2 7B and 13B, Falcon 7B, Zephyr 7B, Mistral 7B).

Dataset Construction To construct our training and test dataset, we have been inspired by the Mintaka dataset [6], which contains 20.000 natural language questions/answers pairs. These pairs are divided in eight complexity categories; among them we have selected the following ones: counting, superlative, comparative, multi-hop, yes/no, difference, and a generic one. However, building a dataset of natural language questions and relative SPARQL code can be a daunting challenge. To this end, we leverage GPT-3 in creating SPARQL code in response to user competency questions, these latter formulated when developing the target ontology. The result is an initial set of 30 Italian NLQs targeting the Cultural-ON ontology and data, each tagged with relevant entities and complexity categories. The following is an example of a manually crafted triple (translated into English for clarity) where *count* is the complexity category.

Tags	**NL Question**	**SPARQL**
[count, sites, city]	*How many cultural institutions are there in Florence?*	`SELECT (COUNT ...`

Several combinations of these tags were automatically created by a script and used as input to GPT-3 to generate more complex NLQs and corresponding NLQ-SPARQL pairs, thereby expanding the dataset. Overall, we guided the generation process by supplying GPT-3 with the initial 30 examples, selecting target categories, combining up to five other tags per instance and then letting GPT-3 generate the question-SPARQL pairs based on this input.

The resulting dataset, consisting of 383 instances, was manually validated and corrected; nonetheless, during evaluation (section 4), we observe that, in some cases, the expected query (ground truth) still contains mistakes (see Evaluation Code 3 in Table 1), revealing residual imperfections despite the initial validation. To further augment the dataset, each question was paraphrased either automatically via GPT-3 or manually, effectively doubling its size. The final dataset contains 766 triplets of tags, NLQs, and SPARQL queries. The complexity of the questions varies by design, with most examples combining three to five tags (excluding the complexity category).

The dataset was partitioned into training and test subsets using an 80/20 random split. To ensure robust model evaluation and prevent data leakage, each example and its corresponding paraphrased version were constrained to reside within the same subset. Both subsets were stored as a pair of JSONL text files to be later ingested by the LLM finetuning script, while only the training set was converted into a collection of 768-dimensional vector embeddings that were stored inside a Qdrant[4] DB instance. Specifically, to support a vanilla RAG approach, each NLQ was embedded using a BERT-based model named `nickprock/sentence-bert-base-italian-uncased`[5]. The latter was chosen both because of its compatibility with the *sentence-transformers*[6] Python library and because its author fine-tuned it on an Italian sentence similarity dataset.

LLM fine tuning with LoRA To keep our setup lightweight and reproducible, we fine-tuned Notus 7B-v1 using LoRA [8] Quantized Low-Rank Adapters (QLoRA) [14] . This approach freezes the original model weights and trains only a small number of additional weight matrices inserted into specific transformer layers. Although LoRA is commonly applied to large models, recent research [15] shows that it performs well even with smaller architectures. With just a tiny fraction of the original parameters involved in the training, LoRA significantly reduces resource requirements; this allows us to successfully run the fine-tuning on a single consumer 8 GB GPU.

[4] https://github.com/qdrant/qdrant
[5] https://huggingface.co/nickprock/sentence-bert-base-italian-uncased
[6] https://github.com/UKPLab/sentence-transformers

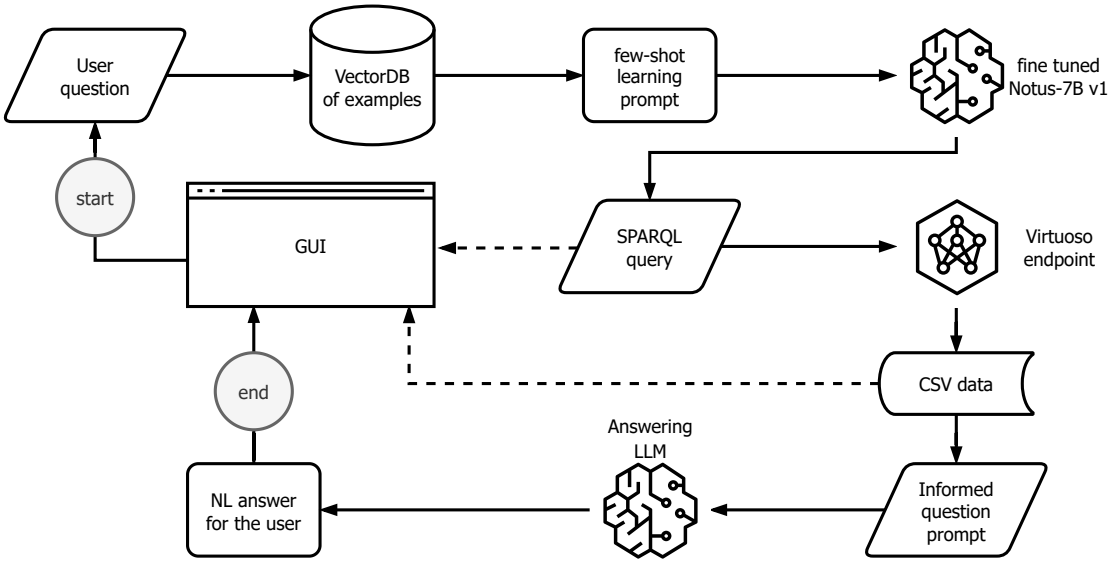

Figure 1: Demo Pipeline: User input is converted into a SPARQL query executed on the SPARQL endpoint, and interpreted to generate the final answer in NL

3.2. Demo development

To showcase the capabilities of our KGQA system in practice to also people of the Italian Ministry of Culture, we have developed ASK ArCo, a GUI-based demo built as a Python *gradio*[7] app with a focus on the Cultural-ON ontology. Users input questions in natural language, assisted by example prompts matching the training data style. Each question is wrapped in a RAG-based few-shot prompt and passed to the LLM to generate a SPARQL query. If the query is valid, it is executed on a SPARQL endpoint and returns a CSV-formatted result. A second LLM prompt then synthesizes a final answer from the CSV. The interface progressively displays, as they get generated, the SPARQL code, a table showing the raw CSV response data, and the final answer (see Figure 1). Failures are diagnosed and flagged depending on whether they occurred during query generation, execution, or interpretation.

4. System Evaluation and Discussion

To assess the SPARQL query generation performance of our system, we conducted four distinct evaluation tests, comparing both untrained and fine-tuned models, each with and without the integration of the RAG technique.

In our RAG pipeline, the retrieval step does not directly extract ontology fragments (e.g., classes or properties). Instead, it retrieves validated natural language question-SPARQL pairs from a vector database. During both runtime and evaluation, the SPARQL generation prompt is dynamically constructed as follows. Using the same model cited in Section 3.1, the user's input question is firstly embedded as a 768-dimensional vector, which is then used to query the vector database for the top eight most similar questions based on cosine similarity. These retrieved questions, together with the corresponding ground-truth SPARQL queries, are subsequently incorporated into the prompt as contextual demonstrations. This approach ensures that the language model (LLM) is consistently exposed to ontology-aligned query patterns, avoiding the need to ingest large ontology fragments that could easily exceed the model's context window.

The evaluation dataset consists of 154 unseen questions: 77 original user questions and their corresponding paraphrases. These were not included in the training data and were never used in any

[7] https://github.com/gradio-app/gradio

RAG context. Each generated SPARQL query and its output was manually categorized into one of ten evaluation result types, as detailed in Table 1. These categories support multiple analysis perspectives depending on whether the focus is on query validity or the correctness of the answer.

Table 1
Evaluation codes assigned to each test case, based on the generated SPARQL query and its data results.

ID	Definition
1	Generated query is functionally identical to the expected one.
2	Generated query shows potential improvements over the expected one (e.g. it adds a DISTINCT clause that we instead forgot to include).
3	Generated query is correct, whereas the expected one contains mistakes.
4	Generated query only answers the user's question indirectly, as the final result must be inferred or computed (e.g. when asked to make a comparison, it provides a list of values to be compared instead of selecting the best one right away).
5	Lexical error (e.g. issues in the natural language strings used for regex or exact matching).
6	Logical or ontological error (e.g. wrong manipulation of the data to be retrieved; non-pertinent, wrongly named or made up properties and classes).
7	Minor syntax error, easily reparable mistake (e.g. unbalanced parenthesis, keyword typos).
8	Major syntax error, irreparable mistake (e.g. usage of undefined IRI prefixes, wrong punctuation, invalid keyword positioning).
0	Total hallucination, gibberish or empty LLM response.

To systematically evaluate our results, we organized the evaluation codes into two groupings that assess different aspects of query generation performance.

- **Grouping 1** in Table 2 distinguished between formally valid and invalid SPARQL code. This dichotomy is fundamental to assess the model reliability in code production, and to find out what might cause bad SPARQL generation, regardless of the query semantics.
- **Grouping 2** in Table 3 distinguishes fully acceptable results from those that could potentially be improved using post-processing techniques, and from those that are considered unusable for answering the user's question. The second group includes outputs or code that could be repaired, for instance, by prompting the LLM to correct its own errors or by applying automated SPARQL parsing and correction. While these techniques may incur some runtime cost, they require minimal implementation effort, and we therefore track these cases as potentially positive outcomes. The third group, on the other hand, comprises outputs that we do not expect to easily recover.

In the following we briefly analyze each of the four tests we conducted on the test dataset.

Test 1 analysis (no train, no RAG) The first test yielded a 0% success rate across all groupings, an expected outcome given the model's lack of SPARQL and domain knowledge. Notably, all outputs were classified as evaluation code 0 (hallucinations), confirming our hypothesis that the untuned model lacks the necessary grounding to generate meaningful and domain-aware queries.

Test 2 analysis (no finetuning, with RAG) The second test evaluates the Notus 7B-v1's capability to fare well when relevant examples are presented inside its context window. Results are summarized in Table 4. Data shows good results in producing functioning SPARQL code (58,2%), but most of these successes contain logical flaws or ontology-specific errors (36,60%). Thus, results are usable only in ~17% of the cases with a maximum of ~30% if actions to fix queries or data were applied at the end of the pipeline. Even with limited adherence to our query style (~15%), the results are encouraging.

Table 2
Evaluation groupings with respect to the validity of the SPARQL query.

Grouping 1 - Validity	Evaluation code
formally valid SPARQL	1 - identical to ground truth 2 - better than ground truth 3 - correct, ground truth is wrong 4 - indirect answer 5 - lexical error 6 - logical or ontological error
formally invalid SPARQL	7 - minor syntax error 8 - major syntax error 0 - hallucination

Table 3
Evaluation groupings with respect to the quality of the SPARQL query results.

Grouping 2 - Quality	Evaluation code
totally admissible query	1 - identical to ground truth 2 - better than ground truth 3 - correct, ground truth is wrong
fixable query	4 - indirect answer 7 - minor syntax error
totally inadmissible query	6 - logical or ontological error 5 - lexical error 8 - major syntax error 0 - hallucination

Table 4
Test results across validity (Grp 1) and quality (Grp 2) dimensions, including detailed percentages for each Evaluation code (E) listed in Table 1.

Grp 1	T2	T3	T4	E	T2	T3	T4	Grp 2	T2	T3	T4	E	T2	T3	T4
valid	58,2	65,6	87,6	1	15,0	16,2	58,2					1	15,0	16,2	58,2
				2	0,6	0,7	4,0	admis.	17,6	19,5	68,7	2	0,6	0,7	4,0
				3	2,0	2,6	6,5					3	2,0	2,6	6,5
				4	2,0	1,3	7,8	fixable	13,1	3,3	14,3	4	2,0	1,3	7,8
				5	2,0	7,1	5,2					7	11,1	2,0	6,5
				6	36,6	37,7	5,9					5	2,0	7,1	5,2
not valid	41,8	34,4	12,4	7	11,1	2,0	6,5	not admis.	69,3	77,3	17,0	6	36,6	37,7	5,9
				8	12,4	19,5	3,3					8	12,4	19,5	3,3
				0	18,3	13,0	2,6					0	18,3	13,0	2,6

Test 3 analysis (finetuned model, no RAG) With the trained model we noticed progress in the provision of SPARQL code (~65% vs. ~58% of working SPARQL queries) but no significant progress with their results (still ~20%) (see Table 4). We notice a significant improvement in terms of hallucinations against RAG testing (~-6%), seemingly compensated with a similar increase in major code errors (~+7%). Adherence to our style shows little improvement (+1%). Surprisingly, only ~2% of the queries ended up being affected by minor code errors. This also means that there is little room for improvement in recovering slightly wrong queries or results with subsequent techniques. In other words, fine-tuning the model proved more decisive than the few-shot learning technique, but only in terms of the SPARQL syntax. The model now either fails greatly in writing a functioning query, or does not fail at all, with little in-between. Even so, knowledge of the ontology is still insufficient, as errors of type 6 are pretty much invariant, as well as the quality of the data retrieved.

Test 4 analysis (finetuned model, with RAG) This test highlights the power of combining fine tuning with RAG for our purposes (see Figure 2). Working SPARQL queries are produced in ~88% of the test cases, and their results are usable in ~69% of the cases, with ~14% cases possibly recoverable for a potential total success rate of ~83% (see Table 4). Specifically, both hallucinations and major code errors are significantly reduced, as well as all other error categories, while adherence to our writing style shows a massive boost (~58% compared to ~15.5% of previous experiments). While the observed improvements are promising, a careful reflection on the scope and potential limitations of these results is necessary before reaching final conclusions.

Figure 2: Comparison of Test 2, 3 and 4 performance across validity, and quality dimensions.

Dataset Limitations Our artificially generated questions, created through the combinatorial pairing of tags, sometimes lack a natural feel. While this may limit the system's ability to handle diverse user queries, it provides significant advantages for internal knowledge graph testing and development. A uniform SPARQL coding style, while limiting generalizability, is essential for maintaining the consistency needed for our lightweight model's performance. Additionally, we have carefully designed our queries to handle variations in the knowledge graph's data quality, such as inconsistent date formats and special characters.

Production Feasibility Finally, our findings indicate that modest scaling may enable production-ready deployment. The lightweight approach using 7B models and LoRA fine-tuning offers cost-effective deployment for public administrations, researchers, and businesses.

5. Concluding remarks and future work

In this paper we presented a KGQA system to automatically produce SPARQL queries. This study shows that fine-tuned lightweight LLMs, when combined with RAG techniques, can effectively bridge the gap between general users and LOD. Specifically, our system achieved a 69% success rate in generating correct SPARQL queries, supporting accessibility to KGs in the cultural domain for non-technical users. Although dataset limitations exist, the system provides a foundation for developing production-level systems that can democratize access to LOD across domains.

The combination of LoRA fine-tuning and RAG-enhanced prompting offers a scalable solution for organisations, also of the public sector, seeking to make their semantic data more accessible while maintaining technical and economic feasibility. The system's modularity enables adaptation to other intuitive graphs with similar entity-relationship structures, potentially broadening access to various LOD sources.

Future work includes improving dataset realism through user-sourced questions and structured input interfaces, such as drop-down-based builders or controlled natural languages like SQUALL [16]. We plan to evaluate our approach on established datasets and benchmarks such as QALD, LC-QuAD, and Mintaka to better assess its generalizability and comparability. Potential technical enhancements include incorporating SPARQL-specialized language models, leveraging automatic query repair, extending the RAG method to include retrieval of ontology snippets (such as class or property definitions) to further refine query generation and minimize logical errors and improving data access through structured graph exploration.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. Quarati, R. Albertoni, Linked open government data: Still a viable option for sharing and integrating public data?, Future Internet 16 (2024) 99. URL: https://doi.org/10.3390/fi16030099. doi:10.3390/FI16030099.

[2] T. Soru, E. Marx, D. Moussallem, G. Publio, A. Valdestilhas, D. Esteves, C. B. Neto, SPARQL as a foreign language, CoRR abs/1708.07624 (2017). URL: http://arxiv.org/abs/1708.07624. arXiv:1708.07624.

[3] C. Ma, Y. Chen, T. Wu, A. Khan, H. Wang, Large language models meet knowledge graphs for question answering: Synthesis and opportunities, CoRR abs/2505.20099 (2025). URL: https://doi.org/10.48550/arXiv.2505.20099. doi:10.48550/ARXIV.2505.20099. arXiv:2505.20099.

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

[5] P. Trivedi, G. Maheshwari, M. Dubey, J. Lehmann, Lc-quad: A corpus for complex question answering over knowledge graphs, in: 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. URL: https://doi.org/10.1007/978-3-319-68204-4_22. doi:10.1007/978-3-319-68204-4_22.

[6] P. Sen, A. F. Aji, A. Saffari, Mintaka: A complex, natural, and multilingual dataset for end-to-end question answering, in: Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022, International Committee on Computational Linguistics, 2022, pp. 1604–1619. URL: https://aclanthology.org/2022.coling-1.138.

[7] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, J. Leskovec, QA-GNN: reasoning with language models and knowledge graphs for question answering, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021, Association for Computational Linguistics, 2021, pp. 535–546. URL: https://doi.org/10.18653/v1/2021.naacl-main.45. doi:10.18653/V1/2021.NAACL-MAIN.45.

[8] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, W. Chen, Lora: Low-rank adaptation of large language models, CoRR abs/2106.09685 (2021). URL: https://arxiv.org/abs/2106.09685. arXiv:2106.09685.

[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: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[10] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, E. Grave, Few-shot learning with retrieval augmented language models, CoRR abs/2208.03299 (2022). URL: https://doi.org/10.48550/arXiv.2208.03299. doi:10.48550/ARXIV.2208.03299. arXiv:2208.03299.

[11] C. Mavromatis, G. Karypis, GNN-RAG: graph neural retrieval for large language model reasoning, CoRR abs/2405.20139 (2024). URL: https://doi.org/10.48550/arXiv.2405.20139. doi:10.48550/ARXIV.2405.20139. arXiv:2405.20139.

[12] G. Lodi, L. Asprino, A. G. Nuzzolese, V. Presutti, A. Gangemi, D. R. Recupero, C. Veninata, A. Orsini, Semantic Web for Cultural Heritage Valorisation, Springer International Publishing, Cham, 2017, pp. 3–37. URL: https://doi.org/10.1007/978-3-319-54499-1_1. doi:10.1007/978-3-319-54499-1_1.

[13] V. A. Carriero, A. Gangemi, M. L. Mancinelli, A. G. Nuzzolese, V. Presutti, C. Veninata, et al., Pattern-based design applied to cultural heritage knowledge graphs, SEMANTIC WEB 12 (2021) 313–357.

[14] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, Qlora: Efficient finetuning of quantized

llms, CoRR abs/2305.14314 (2023). URL: https://doi.org/10.48550/arXiv.2305.14314. doi:10.48550/ARXIV.2305.14314. arXiv:2305.14314.

[15] S. Sun, D. Gupta, M. Iyyer, Exploring the impact of low-rank adaptation on the performance, efficiency, and regularization of RLHF, CoRR abs/2309.09055 (2023). URL: https://doi.org/10.48550/arXiv.2309.09055. doi:10.48550/ARXIV.2309.09055. arXiv:2309.09055.

[16] S. Ferré, SQUALL: A controlled natural language for querying and updating RDF graphs, in: T. Kuhn, N. E. Fuchs (Eds.), Controlled Natural Language - Third International Workshop, CNL 2012, Zurich, Switzerland, August 29-31, 2012. Proceedings, volume 7427 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 11–25. URL: https://doi.org/10.1007/978-3-642-32612-7_2. doi:10.1007/978-3-642-32612-7_2.