A Knowledge Graph-Based RAG Approach for Question Answering: A case study on EmPULIA Regulations

Mattia Curri[1,*,†]

[1]*Dipartimento di Informatica - Università degli Studi di Bari Aldo Moro, Italy*

Abstract

The proliferation of Large Language Models (LLMs) offers significant potential for information access, yet their application in specialized domains such as public administration is obstructed by issues of hallucination, lack of up-to-date, domain-specific knowledge, and poor verifiability. This paper proposes an approach that integrates Knowledge Graphs (KGs) with Retrieval-Augmented Generation (RAG) to overcome these challenges. We detail the construction of a comprehensive KG from the regulatory documents of EmPULIA, an Italian e-procurement platform. This KG serves as a structured, factual foundation for a RAG pipeline, enabling the system to retrieve relevant, interconnected information to ground the LLM's responses. By doing so, our approach aims to provide accurate, verifiable, and context-aware answers to complex regulatory queries. We present the complete workflow, from document processing and KG construction to the generation of a specialized question-answering dataset, EmPULIA-QA, and evaluate the system's performance, demonstrating the effectiveness of combining structured knowledge with generative models for domain-specific applications.
Code is available at https://github.com/mattiacurri/NLP.

Keywords

Retrieval Augmented Generation, Public Amministration, Semantic Search

1. Introduction and Motivations

The advent of Large Language Models (LLMs) has opened new avenues for information retrieval and knowledge synthesis. However, their direct application to specialized domains like public administration often faces critical limitations: LLMs can suffer from "hallucinations" and may not possess the specific, up-to-date knowledge required. Furthermore, the opacity of their internal decision-making processes makes it difficult to verify the source and reasoning behind their outputs, a crucial requirement for accountability in public sector.

This paper addresses these challenges by proposing and evaluating an approach that leverages the strengths of Knowledge Graphs (KGs) in conjunction with Retrieval-Augmented Generation (RAG) techniques. KGs excel at representing structured and connected knowledge; the RAG paradigm, on the other hand, enhances LLMs by grounding their responses in retrieved, relevant information from the knowledge base, thereby mitigating hallucinations and ensuring factual accuracy.

The scenario in which this approach is applied is EmPULIA regulations. EmPULIA is an e-procurement Apulia platform that aggregates demand, helps rationalize regional public spending, and simplifies the relationship between entities and economic operators [1].

2. Related Work

Our research builds upon the rapidly evolving intersection of Large Language Models (LLMs), Knowledge Graphs (KGs), and advanced information retrieval techniques. The predominant approach to mitigate LLM "hallucinations" and ground them in factual, domain-specific data is Retrieval-Augmented Generation (RAG), which has been extensively surveyed [2][3]. This method is particularly relevant for the public administration sector, where accessing accurate and up-to-date information is critical.

NLP Course 2024-2025
[*]Corresponding author.

m.curri8@studenti.uniba.it (M. Curri)
https://mattiacurri.github.io/ (M. Curri)

However, standard RAG often falls short when dealing with complex queries that require understanding intricate relationships within data. To address this, recent work has focused on integrating KGs, which excel at representing structured, interconnected knowledge, thereby enhancing the reasoning capabilities of LLMs and further reducing the risk of factual errors [4] [5]. This synergy has given rise to GraphRAG [6], an approach that leverages a graph-based index to navigate relationships between information chunks, improving retrieval for complex questions. Our work applies these principles by constructing a KG from the regulatory texts of the EmPULIA e-procurement platform, creating a Knowledge Graph-based RAG system specifically tailored to answer questions within this domain.

3. Proposed Approach

In this section we present the adopted workflow. In Figure 1, we show the high-level pipeline, which consists of the following steps:

- Document Collection
- Document Parsing
- Document Chunking
- Knowledge Graph Construction
- Relation Clustering
- Triples/Entities Encoding

At **inference time**, the pipeline consists of the following steps:

- User Query Encoding
- Semantic Search to create the Context
- The answer is generated on the final query, made by the context and the original User Query.

For the **dataset generation**, the following steps are performed:

- Storing the graph in a Neo4j database
- Generating training samples from the graph
- Saving the training samples in a file

3.1. Workflow

3.1.1. Document Collection -> Document Parsing -> Document Chunking

For document collection, we scraped the texts from [7]. We parsed the PDFs using `Docling` [8], leveraging its `HybridChunker` to split each document into semantically coherent segments. `HybridChunker` combines hierarchical, document-based partitioning with tokenization-aware refinements (using the `Qwen3-Embedding-0.6B` tokenizer [9]) to ensure consistent chunk boundaries. Finally, to balance granularity and retrieval efficiency, we aggregated every three consecutive chunks into a single unit.

3.1.2. Knowledge Graph Construction

To construct the Knowledge Graph, we employed `Gemini-2.5-Flash`[1] [11] using a structured output, with `temperature=0` and `thinking_budget=0` (all other parameters at their API defaults). For each document chunk, the model extracted entities, relations, and triples of the form `(entity1, relation, entity2, source)`, where `source` indicates the part of a chunk from which the triples has been extracted. Finally, we merged all triples across chunks to assemble a single Knowledge Graph.

[1]All the (system) prompts are stored in the repository. They are built following the Gemini Prompting Guide 101 [10], exploiting Gemini itself to write Power Prompts.

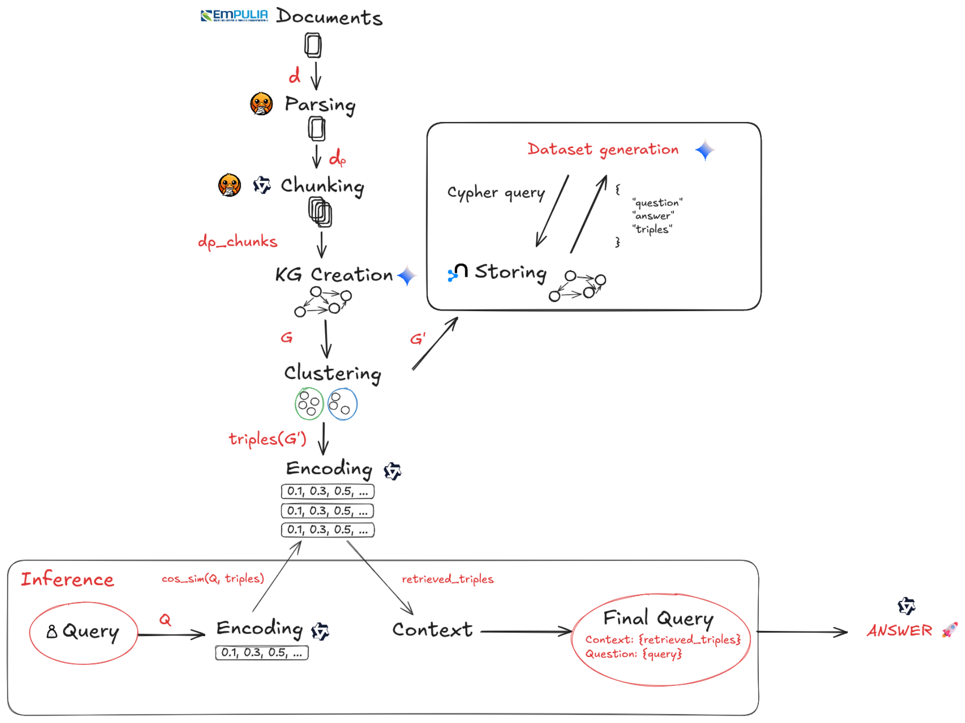

Figure 1: High-level pipeline of the proposed approach.

3.1.3. Relation Clustering

To obtain a more cohesive and compact graph, we cluster relation labels by embedding each name into a vector space and applying DBSCAN [12]. We measure similarity via the cosine similarity between relation embeddings and set $\epsilon = 0.05$ to ensure that only highly similar labels - e.g., `e'_approvato`, `e'_approvata`, and `sono_approvati` - are merged into the same cluster.

3.1.4. Neo4j Graph Storing

To generate the dataset, we stored the unified graph in a Neo4j database. Each entity was represented as a node, and each triple was implemented as a directed relationship: the relationship type corresponds to the original predicate label, and the source document is recorded as a property on that relationship.

3.1.5. Triples/Entities Encoding

To enable semantic search over the Knowledge Graph, we embedded each triple using `Qwen3-Embedding-0.6B` [9]. We represented triples as the concatenated string `entity1 relation entity2: source` for embedding, although we also evaluated an alternative that embeds only the core triple `entity1 relation entity2`, but still passing to the LLM the full string for context. Then, at inference time, the search was done using a `cosine_threshold=0.6` and `top_k=546`, that is, the number of triples in the biggest connected component of the graph. This high value of k is aimed at testing the model capability to filter out irrelevant triples. Finally, the embeddings are generated, saved in a file, and used with FAISS [13].

3.2. Dataset Generation

To generate the dataset, we leveraged Neo4j capabilities. We stored the graph in a Neo4j database and then generated training samples from the graph. In particular, we divided the dataset (from now EmPULIA-QA) into several difficulty levels, based on the kind of question and hops needed. In detail:

- **Isolated Nodes**: Questions answerable by retrieving a single, unconnected node, testing the system's handling of standalone facts.
- **Single Hop**: Questions requiring exactly one edge traversal, assessing retrieval of directly related information.
- **Two Hops**: Questions that span two consecutive edges, evaluating the system's ability to follow multi-step relationships.
- **Hubs**: Queries centered on high-degree nodes, testing aggregation, summarization, and counting over densely connected graph regions.
- **Out-of-Domain**: Questions unrelated to EmPULIA regulations, measuring the system's capacity to detect and reject irrelevant queries

We generated eight questions for the first three categories, and six questions for the remaining one, for a total of 36 queries. These questions are designed to thoroughly assess both the retrieval and generation performance of our approach.

All question-(answer, analysis, source) pairs were created using `Gemini-2.5-Flash` with `temperature=0` and `thinking_budget=0`, based on the Cypher queries presented in Table 1.

Isolated Nodes	1 Hop	2 Hop	Hubs
		MATCH (s)-[r]->(o)-[r1]->(e)	
		WHERE	
		count {(o)-->()} = 1	
MATCH (s)-[r]->(o)		AND	
WHERE		rand() < 0.1	
count {(s)-->()} = 1	MATCH (s)-[r]->(o)		
AND	WHERE	WITH s,	MATCH (hub)
count {(s)<--()} = 0	count {(s)-->()} = 1	head(collect({r: r,	WITH
AND	AND	o: o,	hub,
count {(o)-->()} = 0	rand() < 0.1	r1: r1,	count {(hub)--()}
AND		e: e}))	AS degree
rand() < 0.1	WITH s,	AS path_data	ORDER BY degree DESC
	head(collect({r: r,		
WITH s,	o: o}))	RETURN	WITH hub
head(collect({r: r,	AS path_data	elementId(s)	
o: o}))		AS source_id,	MATCH (hub)-[r]-(o)
AS path_data	RETURN	s.id	WHERE o.id IS NOT NULL
	elementId(s)	AS entita1,	RETURN
RETURN	AS source_id,	path_data.r.type	elementId(hub)
elementId(s)	s.id	AS relation,	AS hub_id,
AS source_id,	AS entita1,	path_data.r.properties	hub.id
s.id	path_data.r.type	AS source,	AS hub_name,
AS entita1,	AS relation,	path_data.o.id	collect({
path_data.r.type	path_data.r.properties	AS entita2,	relation: r.type,
AS relation,	AS source,	elementId(path_data.o)	source: r.properties,
path_data.r.properties	path_data.o.id	AS target_id,	connected_node_name: o.id,
AS source,	AS entita2,	path_data.r1.type	connected_node_id: elementId(o)
path_data.o.id	elementId(path_data.o)	AS relation1,	}) AS connections;
AS entita2,	AS target_id	path_data.r1.properties	
elementId(path_data.o)		AS source1,	
AS target_id		path_data.e.id	
		AS entita3,	
		elementId(path_data.e)	
		AS target_id_1	

Table 1
Generated queries for dataset creation.

3.3. Inference Workflow

3.3.1. User Query -> Context Generation -> Final Query

At inference time, we evaluated three retrieval strategies to construct the context and formulate the final input to the generator:

- **Direct Retrieval:** Encode the user's original query and perform semantic search over the triple embeddings to retrieve the top-K relevant triples.
- **Multi-Query Retrieval:** Automatically generate multiple paraphrases of the original query, execute each against the KG, and aggregate the retrieved results to improve coverage.

- **Structured Extraction:** Analyze the query to extract candidate triples, then do semantic search over the KG to find relevant context.

We found that Multi-Query Retrieval and Structured Extraction are far from performing well already in the retrieval component. Therefore, our final evaluation focuses exclusively on the Direct Retrieval strategy.

3.3.2. Answer Generation

We perform inference with `Qwen3-4B` [14], with `temperature=0`, `contextlength=8192` and the rest as the default parameter offered by the `Ollama` implementation (`repeat_penalty=1`, `min_p=0`, `top_p=0.95`, `top_k=20`, `quantization=Q4_K_M`). Each generated response is structured into three parts:

- **Answer**: the LLM's reply to the user's query, grounded in the retrieved context.
- **Analysis**: a concise explanation of the reasoning steps and how the context was leveraged.
- **Sources**: the specific context elements or triples cited by the LLM, provided for user verification.

In this way, a user could verify the answer reading the analysis provided by the model itself, and look up the sources to check the correctness of the answer.

The LLM is instructed to use a specific answer in case it cannot find any relevant information in its context.

4. Evaluation

In this section we present the evaluation of the proposed approach. We first describe the dataset generation, then we present the evaluation metrics used and finally we report the results of the experiments conducted on the `EmPULIA-QA` dataset.

4.1. Evaluation Metrics

For the *Retrieval component*, we used the following metrics:

- **Context Precision**: It evaluates whether the relevant ground truth contexts are ranked higher in the list of retrieved contexts. It is computed as follows:

$$CP = \frac{\sum(Precision@k)}{\text{Number of relevant items in top K}}$$

  where Precision@k = $\frac{\text{True Positives@K}}{(\text{True Positives@}k) + (\text{False Positives@}k)}$;
- **Context Recall**: It indicates the proportion of claims from the ground truth that the context successfully captures. It is computed as follows:

$$CR = \frac{\sum(Recall@k)}{\text{Number of relevant items in top K}}$$

  where Recall@k = $\frac{\text{True Positive@K}}{(\text{True Positives@}k) + (\text{False Negatives@}k)}$

For the *Generation component*, we used the following metrics:

- **Faithfulness**: We say that the answer $a_s(q)$ is faithful to the context c(q) if the claims that are made in the answer can be inferred from the context. To estimate faithfulness, we first use `Gemini-2.5-Flash` with temperature=0 and thinking_budget=0 to extract a set of statements, $S(a_s(q))$. For each statement $s_i \in S$, `Gemini-2.5-Flash` determines if s_i can be inferred from c(q) using a verification function $v(s_i, c(q))$, carried out using a prompt. The final faithfulness score, F, is then computed as $F = \frac{|V|}{|S|}$, where |V| is the number of statements that were supported according to the LLM and |S| is the total number of statements [15]. We measured the **Context Faithfulness** and the **Analysis Faithfulness**:
  - **Context Faithfulness**: Measures the faithfulness of the context to the answer, i.e., whether the context supports the claims made in the answer.
  - **Analysis Faithfulness**: Measures the faithfulness of the analysis to the answer and to the context, i.e., whether the analysis correctly explains how the context was used to generate the answer and if the used context is actually the one retrieved and not hallucinated.
- **Answer Accuracy**: Measures the agreement between a model's response and a reference ground truth for a given question, scored as 0 (inaccurate), 2 (partial alignment), or 4 (exact alignment), using a dual-prompt evaluation approach where two instances of `Gemini-2.5-Flash` (same settings as above) rate the response and reference from swapped perspectives, with the final score being the average of valid ratings, making an LLM jury. [16]

4.2. Results

In this section we report the result of the experiments conducted on the `EmPULIA-QA` dataset. In particular, we tried to use the embedding of the triples alone, and the embedding of the triples along with the source.

In Table 2 are reported the retrieval results, while in Table 3 are reported the generation results, with highlighted the best result for each metric.

EmPULIA-QA						
Category	**Precision**			**Recall**		
	Source	w/out Source	Δ	Source	w/out Source	Δ
Single Hop	**0.539**	0.394	↓ 0.145	0.875	0.875	– 0.000
Two Hop	**0.659**	0.520	↓ 0.139	**1.000**	0.750	↓ 0.250
Isolated	0.567	**0.660**	↑ 0.093	**1.000**	**1.000**	– 0.000
Hubs	**0.463**	0.415	↓ 0.048	**0.108**	0.061	↓ 0.047
Out of Domain	1.000	1.000	– 0.000	1.000	1.000	– 0.000

Table 2
Retrieval Evaluation on `EmPULIA-QA` dataset. Precision and recall for each category, comparing *source* vs *nosource*. Best values in bold. Arrows indicate variation (nosource - source).

EmPULIA-QA									
Category	**Context Faithfulness**			**Analysis Faithfulness**			**Answer Accuracy**		
	Source	w/out Source	Δ	Source	w/out Source	Δ	Source	w/out Source	Δ
Single Hop	0.288	**0.438**	↑ 0.150	0.625	**0.740**	↑ 0.115	0.250	0.250	- 0.000
Two Hop	**0.769**	0.583	↓ 0.186	**0.924**	0.913	↓ 0.011	0.375	**0.406**	↑ 0.031
Isolated	**0.627**	0.552	↓ 0.075	0.862	**0.958**	↑ 0.096	0.531	**0.594**	↑ 0.063
Hubs	0.565	**0.644**	↑ 0.079	0.844	**0.921**	↑ 0.077	0.333	**0.500**	↑ 0.167
Out of Domain	1.000	1.000	– 0.000	1.000	1.000	- 0.000	1.000	1.000	- 0.000

Table 3
Generation Evaluation on `EmPULIA-QA` dataset. Faithfulness and answer accuracy per category. Best source values in bold. Arrows indicate variation (nosource - source).

5. Conclusions and Limitations

In this paper, we introduced a novel pipeline that integrates a Knowledge Graph with Retrieval-Augmented Generation to enhance question answering over public administration regulations. By constructing a structured Knowledge Graph from EmPULIA's regulatory texts and grounding an LLM in retrieved, domain-specific context, our approach delivers accurate, verifiable, and context-aware answers. Evaluation on the `EmPULIA-QA` dataset demonstrates the benefits of combining structured knowledge with generative models in a specialized domain.

Despite promising results, several limitations warrant further investigation:

- **Dataset coverage:** Some regulatory facts may lie outside the current graph or retrieval scope, leaving certain queries unanswerable.
- **Graph topology:** Our current embedding strategy treats triples independently; exploiting the graph structure could improve retrieval relevance.
- **Source validation:** The generation and attribution of source citations have not been rigorously evaluated, which may introduce hallucinated or misattributed references.
- **User feedback:** Integrating interactive feedback could refine context selection and increase answer fidelity over time.
- **Prompt refinement:** Drawing on the model's strong instruction-following ability (evidenced by its Out-of-Domain performance), we can more precise, structured prompts to further improve answer accuracy, relevance, and consistency.
- **Graph enhancemnt:** Neo4j's allows to enhance nodes and relations with additional properties. This could be useful to create a better knowledge graph.

In conclusion:

- **Data Preprocessing (still) matters**: throwing a large language model at a problem does not guarantee good results. The quality of the input data and the preprocessing steps are crucial for achieving high performance.
- **Knowledge Graphs are powerful (if carefully crafted)**: A well-designed knowledge graph can significantly enhance the retrieval and generation capabilities of language models by providing structured, domain-specific context.
- **Small Language Models are not enough (yet)**: even giving the LLM the context, they still struggle to use that context effectively.

References

[1] EmPULIA, Empulia platform, 2025. URL: http://www.empulia.it/tno-a/empulia/SitePages/Home.aspx.

[2] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, H. Wang, Retrieval-augmented generation for large language models: A survey, arXiv preprint arXiv:2312.10997 2 (2023).

[3] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, Q. Li, A survey on rag meeting llms: Towards retrieval-augmented large language models, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 6491–6501.

[4] G. Agrawal, T. Kumarage, Z. Alghamdi, H. Liu, Can knowledge graphs reduce hallucinations in llms?: A survey, arXiv preprint arXiv:2311.07914 (2023).

[5] A. Hogan, X. L. Dong, D. Vrandečić, G. Weikum, Large language models, knowledge graphs and search engines: A crossroads for answering users' questions, arXiv preprint arXiv:2501.06699 (2025).

[6] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, J. Larson, From local to global: A graph rag approach to query-focused summarization, arXiv preprint arXiv:2404.16130 (2024).

[7] EmPULIA, Empulia regulations documents, 2025. URL: http://www.empulia.it/tno-a/empulia/Empulia/SitePages/Normativa.aspx.

[8] D. S. Team, Docling Technical Report, Technical Report, 2024. URL: https://arxiv.org/abs/2408.09869. doi:10.48550/arXiv.2408.09869. arXiv:2408.09869.

[9] Y. Zhang, M. Li, D. Long, X. Zhang, H. Lin, B. Yang, P. Xie, A. Yang, D. Liu, J. Lin, F. Huang, J. Zhou, Qwen3 embedding: Advancing text embedding and reranking through foundation models, arXiv preprint arXiv:2506.05176 (2025).

[10] Google, Gemini for Google workspace prompting guide (v1.0.1), 2024. URL: https://services.google.com/fh/files/misc/gemini-for-google-workspace-prompting-guide-101.pdf.

[11] G. Team, Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL: https://arxiv.org/abs/2507.06261. arXiv:2507.06261.

[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: kdd, volume 96, 1996, pp. 226–231.

[13] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, H. Jégou, The faiss library (2024). arXiv:2401.08281.

[14] Q. Team, Qwen3 technical report, 2025. URL: https://arxiv.org/abs/2505.09388. arXiv:2505.09388.

[15] S. Es, J. James, L. E. Anke, S. Schockaert, Ragas: Automated evaluation of retrieval augmented generation, in: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, 2024, pp. 150–158.

[16] P. Verga, S. Hofstatter, S. Althammer, Y. Su, A. Piktus, A. Arkhangorodsky, M. Xu, N. White, P. Lewis, Replacing judges with juries: Evaluating llm generations with a panel of diverse models, arXiv preprint arXiv:2404.18796 (2024).