EmPuAssistant: Virtual Assistant for EmPULIA Documentation

Martina Capone[1,*]

[1] *University of Bari Aldo Moro Bari, Apulia, Italy*

Abstract

This paper presents EmPuAssistant, a virtual assistant designed to answer procedural questions using institutional documentation published on the EmPULIA platform. The system is based on a two-phase pipeline: (1) the construction of a knowledge graph from PDF manuals using a controlled text reformulation and triple extraction process; (2) the deployment of a Retrieval-Augmented Generation (RAG) architecture that leverages this graph to generate grounded, context-aware answers. The pipeline employs the LLaMAntino-3-ANITA model both for content processing and for generating responses during RAG, with an experimental comparison to Mistral. A Gradio-based interface enables intuitive user interaction. Evaluation on 52 domain-specific queries demonstrates the system's effectiveness in providing reliable, semantically grounded answers.

Keywords
LLaMAntino-3-ANITA, Knowledge Graph construction, RDF triple extraction, RAG

1. Introduction and Motivations

Public administration platforms such as EmPULIA [1] host numerous documents detailing procedures, participation rules, and technical functionalities for procurement processes. However, this information is typically provided in PDF format, which is difficult to parse automatically due to inconsistent structure, visual elements, and textual redundancies.

On the EmPULIA platform, relevant documents are published under the following sections:

- *Guide pratiche nuova piattaforma*
- *Nuove Guide dedicate agli utenti SA*
- *Nuove Guide dedicate agli Operatori Economici*

These manuals contain formal procedural language, structured lists, and domain-specific terminology, targeting both institutional users and economic operators. As of today, a total of 38 documents have been collected and processed as the foundational dataset for this work.

To address the challenges of accessing this content semantically, we propose EmPuAssistant, a semantic assistant that enables users to query such documents via natural language. The system revolves around a modular and reusable pipeline that extracts, restructures, and organizes EmPULIA content into a knowledge graph (KG), used within a retrieval-based QA system powered by local language models.

2. Related Work

The integration of Large Language Models (LLMs) with Knowledge Graphs (KGs) has recently emerged as a promising paradigm for structuring and accessing information extracted from unstructured documents. Several works have explored the automatic generation of KGs from heterogeneous sources, the use of RDF triples in Retrieval-Augmented Generation (RAG) architectures, and the application of domain-specific ontologies for semantic enrichment.

Docs2KG presents a unified pipeline for constructing knowledge graphs from various document types, including PDFs, by leveraging LLMs for content extraction and structuring. This approach is particularly relevant to the problem of transforming semi-structured procedural content into reusable, semantic knowledge bases [2]. Similarly, the DO-RAG framework demonstrates how domain-specific knowledge graphs can be combined with RAG models to support question answering from technical documentation, integrating both symbolic and neural components [3].

Several works focus on enriching RAG architectures with graph-based representations. For instance, OG-RAG proposes the use of ontologies to ground the retrieval process, improving the relevance and consistency of the context provided to the language model [4]. GraphRAG explores how graphs built from document content can improve the grounding of generative responses and mitigate hallucination in LLM outputs [5].

Surveys such as "Unifying LLMs and Knowledge Graphs: A Roadmap" highlight current efforts to bridge parametric knowledge stored in LLMs with explicit symbolic representations in KGs [6]. These studies support the idea that hybrid systems—where LLMs are used both to construct and query semantic graphs—can provide more explainable and controllable outputs.

3. Proposed Approach

3.1. Description of the solution and dataset

The proposed solution is centered on a modular pipeline designed to extract and structure procedural content from PDF documents. Its goal is to transform textual instructions into a machine-readable knowledge graph that supports semantic querying via large language models.

The architecture is composed of two main phases:

1. **Knowledge graph construction**: based on text extraction, reformulation, and RDF triple generation;
2. **Query answering**: leveraging the knowledge graph within a Retrieval-Augmented Generation (RAG) framework.

The following sections describe each component of the pipeline in detail, starting from document acquisition and proceeding through text preprocessing, triple extraction, graph assembly, and finally the interactive question-answering mechanism.

3.1.1. Document Processing and Text Reformulation

The process begins with the scraping of PDFs directly from the EmPULIA website. Each document is converted into plain text using PDF parsing techniques. However, raw conversion introduces a variety of issues such as duplicated content, broken formatting, and artifacts from index structures. To address this, each text block is passed through a LLaMAntino-3-ANITA, a fine-tuned model developed by the University of Bari, for semantic reformulation. This step ensures that the resulting text is concise, readable, and suitable for knowledge extraction.

After reformulation, the same LLM is prompted to extract RDF triples that capture factual or procedural knowledge present in the text. Since LLM-generated triples may vary in structure, a post-processing step using regular expressions is applied to validate the format and ensure compatibility with standard KG structures. Triple responses were validated post-hoc, and when hallucinations or malformed structures were detected, the model was prompted again to regenerate them. This validation is critical, as the accuracy of the triples directly impacts the quality and navigability of the resulting knowledge graph.

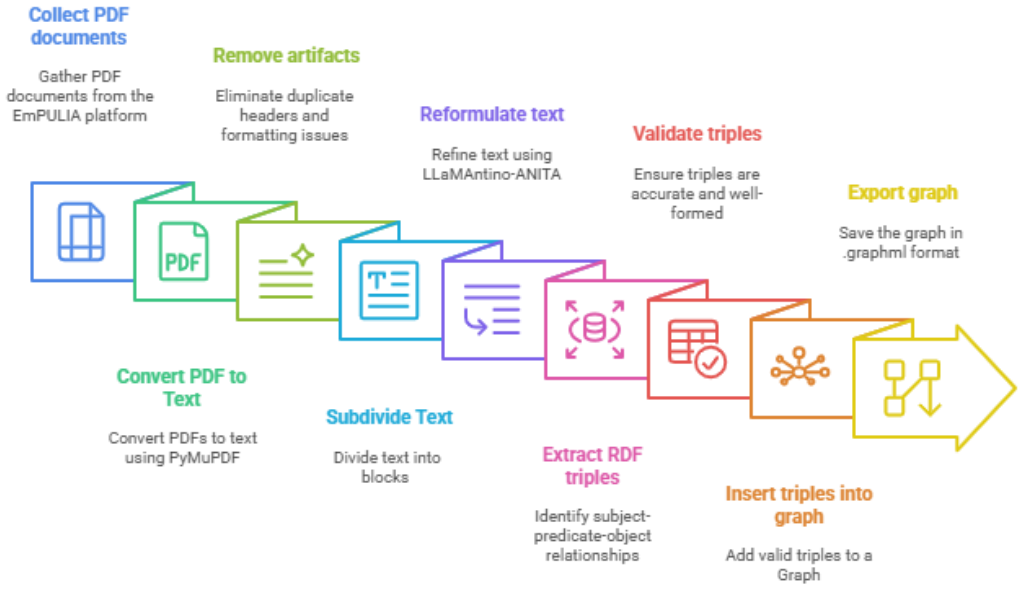

Figure 1 Overview of the pipeline used to extract and structure procedural content into a knowledge graph.

3.1.2. Semantic Querying and Answer Generation

Once the knowledge graph has been constructed, it is employed as an external source of structured knowledge to augment responses generated by a local language model. This architecture enables a controlled, verifiable interaction framework, in which generated answers remain grounded in factual information extracted from institutional documents.
To handle user queries in natural language, the system first identifies the most semantically relevant triples from the knowledge graph. Each user query is tokenized and filtered using

an Italian stopword list. Relevance scoring is performed by matching keywords against the subject, predicate, and object components of each triple. A weighted heuristic assigns higher scores to matches on subject and object nodes than on the relation label, reflecting their greater importance in semantic alignment.

The top-ranked triples are selected and used to build a contextual prompt that informs the language model of relevant facts. This lightweight retrieval mechanism provides a balance between precision and computational efficiency and ensures that the model is exposed only to contextually meaningful graph fragments.

The selected triples are passed to a instruction-tuned language model—either Mistral-7B or LLaMAntino-3-ANITA—through a structured prompt that includes the user query and the selected triples. Both models were tested under identical conditions. The prompt instructs the model to respond in Italian using clear and accessible language, suitable for non-technical users.

By grounding the model's response on curated graph information, the system significantly reduces the risk of hallucinated content. Furthermore, since the retrieved triples originate from a manually verified knowledge graph, the assistant provides answers with higher factual reliability and traceability.

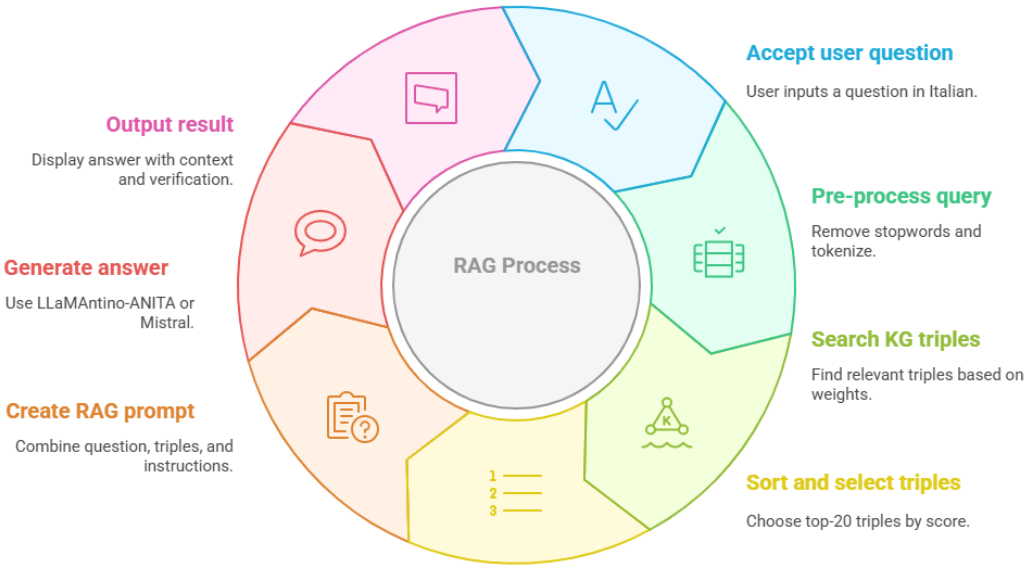

Figure 2 The RAG-based interaction flow: from user query to answer generation grounded on the kg.

3.1.3. Gradio Interface

The final system is deployed as a web-based chat interface using Gradio. A custom dark-themed UI presents the assistant as a conversational agent capable of handling open-ended queries about procedures, access methods, and participation criteria on the EmPULIA platform. The interface supports rapid experimentation, direct user interaction, and qualitative evaluation by domain experts.

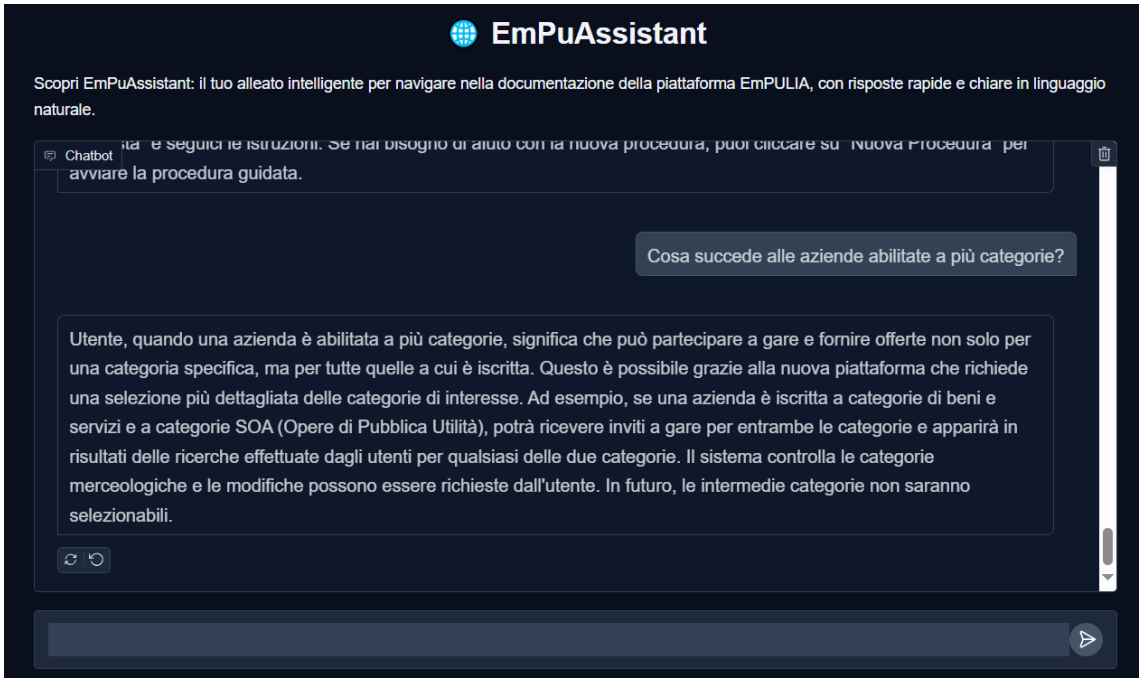

Figure 3 Example interaction between a user and EmPuAssistant.

3.2. Other information useful to replicate the approach

The EmPuAssistant system is implemented in Python 3.11 and organized into a modular pipeline. All dependencies are listed in requirements.txt, and the project includes a ready-to-use DevContainer (.devcontainer/devcontainer.json) to facilitate deployment without manual configuration.
The system relies on a minimal set of libraries:

- requests, beautifulsoup4: for scraping procedural documents from the EmPULIA website
- PyMuPDF: to extract raw text from PDFs
- huggingface_hub, transformers, torch: for local or API-based model loading
- llama-cpp-python: for running quantized GGUF models (e.g., ANITA and Mistral) locally
- tiktoken: for tokenizing and chunking text
- sentence-transformers, spacy, nltk: for token cleaning, stopword removal, and text preprocessing
- networkx: for RDF graph construction
- gradio: for the interactive chat interface

The project is organized as follows:

- data/PDF: original EmPULIA manuals in PDF format
- data/TEXT: raw text extracted from PDFs
- data/REPHRASE_TEXT: semantically reformulated text

- data/TRIPLES: extracted RDF triples in CSV format
- knowledge_graph/kg.graphml: the final RDF graph
- models/ANITA and models/mistral: folders for local GGUF model files
- src/: source code organized into scraping, processing, graph construction, prompt building, and UI modules
- pipelines/preprocessing.py: builds the knowledge graph
- EmPuAssinstant_mistral.py, EmPuAssinstant_ANITA.py: scripts for launching the assistant using the respective models

To test the assistant:
1. Download the ANITA model using the helper script:
   python src/utils/download_model.py --model anita
2. If the knowledge graph has not been generated, run:
   python pipelines/preprocessing.py
3. Run the system using one of the following:
   python EmPuAssinstant_mistral.py
   python EmPuAssinstant_ANITA.py

The assistant will launch in a Gradio-based chat interface accessible in the browser.

4. Evaluation

To assess the effectiveness of the proposed pipeline in generating accurate responses grounded in EmPULIA's official documentation, a benchmark set of **52 domain-specific questions** was manually created. These questions were derived from PDF manuals published on the EmPULIA platform and were designed to reflect realistic scenarios involving procurement procedures, platform functionalities, document submission, and evaluation processes.

4.1. Experimental Setup

Each question was processed using a retrieval-augmented generation (RAG) approach. Relevant triples were extracted from the knowledge graph and embedded within a structured prompt, which was then passed to the language model. Two models were compared under identical inference conditions:

- LLaMAntino-ANITA
- Mistral

Both models were evaluated with the same context length, temperature, and formatting constraints to ensure consistency.

4.2. Evaluation Methodology

Given the domain-specific nature of the content, a fully authoritative evaluation of factual accuracy would require input from subject-matter experts (SMEs). In the absence of such an annotation layer, a qualitative analysis of the generated responses was conducted, using the following criteria:

- **Factual consistency** with the content in the extracted triples and source documents.
- **Completeness** in addressing the key aspects of the input question.
- **Fluency and formality** of the generated text, particularly in an institutional context.
- **Presence of hallucinations**, such as fabricated acronyms, commands, or roles.
- **Language appropriateness**, focusing on adherence to Italian.

4.3. Observed Behaviors

The LLaMAntino-ANITA model consistently delivered coherent and well-structured responses. Notable characteristics include:

- Clear and often summarized answers, providing both overview and detail.
- A polite and institutionally appropriate tone, aligning with the expected user-facing role.
- Inclusion of a disclaimer in highly specific cases, stating that the answer is generated based on the knowledge graph and may not reflect official or complete information—an effective measure for encouraging user verification.
- Absence of hallucinations or fabricated terminology.

While Mistral exhibited some inconsistencies—such as occasional hallucinations or unnatural phrasing—it was often more detailed than LLaMAntino-ANITA in elaborating on procedural steps. However, this verbosity did not always correlate with factual accuracy, and in some cases led to overgeneration or inclusion of speculative content.

Model	Factual Accuracy	Completeness	Fluency & Tone	Hallucinations	Language Adherence
LLaMAntino-ANITA	High	High	Clear, Formal	None observed	Fully in Italian
Mistral	Moderate	Partial	Inconsistent	Frequent	Occasionally English

Table 1 Summarizes the qualitative comparison between LLaMAntino-ANITA and Mistral models across five key dimensions.

5. Conclusion and limitations

This work demonstrates the effectiveness of integrating instruction-optimized LLM into a structured pipeline for knowledge extraction from procedural documents. During the triple extraction phase, both ReBEL and spaCy, two widely used solutions for relation extraction and NER, were tested. However, LLaMAntino-3-ANITA-8B proved to be significantly more effective in this use case, due to its instruction optimization for Italian tasks. Its specialization in the Italian language was crucial for handling formal expressions and domain-specific vocabulary present in EmPULIA manuals.

Limitations:

- Incomplete coverage of edge cases with fuzzy terminology
- Manual curation is still required for knowledge base construction
- No evaluation of fact consistency across the graph and model generations

Future directions:

- Using semantic retrievers for graph search
- Integrating automatic KB updating
- Extending the assistant with memory-enabled dialogue to support multi-turn interaction

References

[1] "EmPULIA - Guide pratiche." Accessed: Jun. 30, 2025. [Online]. Available: http://www.empulia.it/tno-a/empulia/Empulia/SitePages/Guide%20pratiche.aspx

[2] Q. Sun *et al.*, "Docs2KG: Unified Knowledge Graph Construction from Heterogeneous Documents Assisted by Large Language Models," *Proceedings of (Under Review)*, vol. 1, Jun. 2024.

[3] D. O. Opoku, M. Sheng, and Y. Zhang, "DO-RAG: A Domain-Specific QA Framework Using Knowledge Graph-Enhanced Retrieval-Augmented Generation," May 2025, Accessed: Jun. 30, 2025. [Online]. Available: https://arxiv.org/pdf/2505.17058

[4] K. Sharma, P. Kumar, and Y. Li, "OG-RAG: Ontology-Grounded Retrieval-Augmented Generation For Large Language Models," Dec. 2024, Accessed: Jun. 30, 2025. [Online]. Available: https://arxiv.org/pdf/2412.15235

[5] H. Han *et al.*, "Retrieval-Augmented Generation with Graphs (GraphRAG)," Dec. 2024, Accessed: Jun. 30, 2025. [Online]. Available: https://arxiv.org/pdf/2501.00309

[6] J. Z. Pan *et al.*, "Large Language Models and Knowledge Graphs: Opportunities and Challenges," vol. 000, no. 42, p. 30, Aug. 2023, doi: 10.1234/0000000.00000000.