universität wien

AI – Using a Large Language Model (LLM) together with Retrieval-Augmented Generation (RAG) for answering questions on custom data

University of Vienna | Faculty of Computer Science | Bachelor Thesis Presentation

Ivan Khomich

universität wien

Abstract

This bachelor's thesis focuses on the development of a chatbot application that integrates a large language model (LLM) with Retrieval-Augmented Generation (RAG) to provide accurate and contextually appropriate responses based on custom datasets. The objective is to develop a chatbot that can efficiently retrieve relevant information from uploaded documents and websites and use this data to generate accurate responses.

Motivation and Problem Statement

- **Context:**
  - Use of large language models has advanced beyond generic Q&A into specialized fields.
  - Standard LLMs often struggle with custom, domain-specific, or newly updated info.
- **Retrieval-Augmented Generation (RAG):**
  - Integrates LLMs with a retrieval mechanism to fetch relevant context from specific datasets.
  - Addresses limitations of standard LLMs, which cannot easily adapt to new information in real time.
- **Local LLM Focus:**
  - Avoids ongoing cloud costs.
  - Enhanced privacy – data stays local.

Project Goals

- **Primary Goal:** Implement a chatbot with a local LLM + RAG for domain-specific Q&A.
- **Key Metrics:**
  - **Accuracy:** Generate factually correct answers using external context.
  - **Efficiency:** Handle large datasets with minimal latency.
  - **Adaptability:** No continual retraining required to handle new or updated data.
- **Scope:**
  - RAG retrieves *specific* doc segments.
  - Not designed for complex tasks like extensive summarization or multi-step reasoning.

Process and Methods

- **RAG Pipeline:**
  - Uses vector database to fetch relevant context.
  - LLM uses that context to craft final response.
- **ReActAgent:**
  - Dynamically decides how to solve query.
  - Possible tools: web search, RAG-based doc retrieval, math solver.
- **Streamlit UI:**
  - Upload documents, clean db, and query the chatbot.

universität wien

Related Work

- **Existing Guides/Tutorials:**
  - Mostly illustrate basic usage or toy demos.
- **Issues Observed:**
  - Insufficient accuracy on large or domain-specific data.
  - Lack of robust GUIs or multi-dataset integration.
  - Overly simplistic usage of LlamaIndex or default prompts.
- **Core Reasons:**
  - Inefficient doc parsing or suboptimal embeddings.
  - Poor similarity search methods.

Retrieval-Augmented Generation (RAG)

- RAG = LLM + document-retrieval component.
- Steps:
  1. Retrieve relevant context from data store.
  2. Provide context to the LLM for a more accurate response.

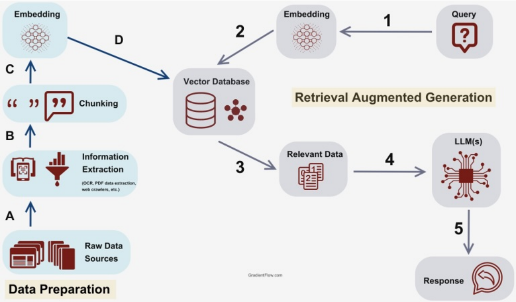


Figure: RAG workflow

Large Language Models (LLMs)

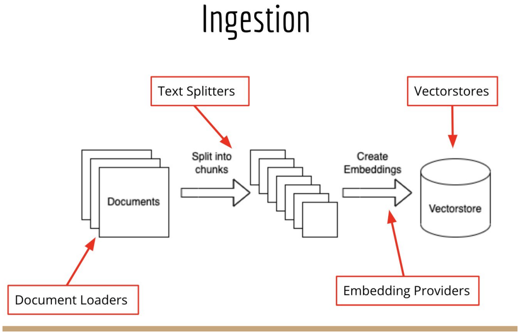
- **Examples:** Llama, GPT, etc.
- **Focus here:** Llama 3.1 + LlamaIndex ReAct agent.
- **Agent-based approach:**
  - Calls external tools for tasks like web search or math solver.

Ollama

- **Open-source platform** for running LLMs locally.
- **Benefits**:
  - Free, minimal license issues.
  - Full control over data, offline usage.
  - Simplifies local LLM integration.

Embeddings & Vector Database

- **Embeddings:**
  - High-dimensional vectors capturing semantic context, this project uses **bge-base-en-v1.5**.
  - Enable efficient similarity search for text, images, or audio.
- **Vector Database:**
  - Stores and queries these high-dimensional vectors.
  - **ChromaDB:**
    - Used for storing embeddings + metadata.
    - Key for text-based similarity search in LLMs.

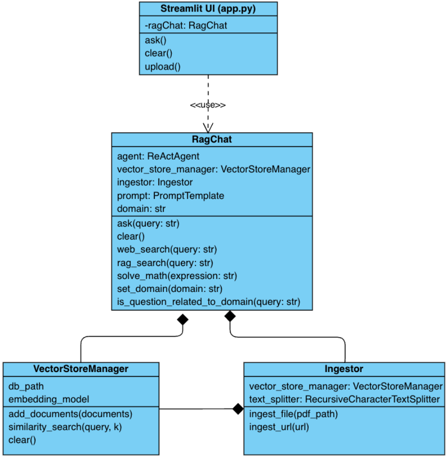


Data Preparation

- **Uploaded PDFs:**
  - `PyPDFLoader` for text extraction.
  - `RecursiveCharacterTextSplitter` for chunking.
- **Webpages:**
  - `requests` + `BeautifulSoup` for parsing.
- **Embeddings in Chroma**:
  - Each chunk vectorized and stored for fast retrieval.

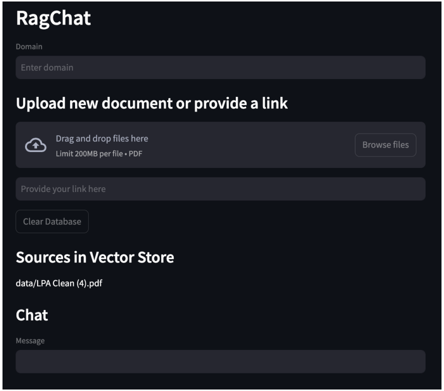
Implementation Overview

- **UI (Streamlit)** for user interaction.
- **LlamaIndex ReActAgent** for orchestration.
- **Document Parsing & Vector DB** for retrieval.



UI (Streamlit)

- Web-based interface:
  - Upload documents, input URLs.
  - Interact with chatbot Q&A.
  - Clear or update database.



LlamaIndex ReActAgent

- **Process:**

1. User Query
2. Agent decides whether to:
   - Retrieve from local docs.
   - Use web search.
   - Use math solver.
3. Synthesizes final answer.

Document Retrieval and Storage

- Embeddings stored in `Chroma`.
- Agent queries top matching chunks for local context.
- Chunks inserted into LLM prompt for context-aware answers.

Data Ingestion Pipeline

1. **Parse Documents:** PDFs or web sources.
2. **Split Text:** `RecursiveCharacterTextSplitter`.
3. **Store Embeddings:** $\rightarrow$ vector DB (Chroma).

universität wien

Repository

- Full implementation with RAG + local LLM.
- Detailed README for setup.

Ref: `https://github.com/grippvh/ragChat`

Test Coverage

- Focus on verifying correct retrieval + context usage.
- **Challenge:** LLM outputs can vary in wording.
- **Strategy:**
    - Ask the model, then re-ask it to validate its answer against expected response.
    - Negative test cases ensure incorrect answers are flagged.
- **No 100% determinism:** LLM variability can produce different valid answers.

Observations

- Combining LLM + RAG can be tricky (embedding quality, chunking).
- **Naive RAG:**
  - Works on small docs.
  - Degrades with larger, complex data if not optimized.
- **Key Factor:**
  - Embeddings heavily influence retrieval quality.
  - Prompt design also matters.

Performance Example

- **Without RAG:** Llama 3.1 returned irrelevant facts.
- **With RAG:** Correctly retrieved exact text from the doc.
- **Conclusion:** RAG essential for domain-specific or fresh content not in LLM training.

TODO: split with screenshots

Summary of Findings

- **Accuracy:** Increases when relevant context is retrieved.
- **Limits:** Not ideal for broad tasks like full summarization.
- **Potential:** Additional modules (knowledge graphs, rerankers) could improve coverage.

universität wien

Lessons Learned

- **Better embeddings** significantly improve retrieval accuracy.
- **RAG is best for factual queries,** less so for deeper or multi-doc comprehension.

Future Improvements

- Smarter query rewriting and rerankers.
- Integrating knowledge graphs.
- Multi-agent approaches for more complex tasks.

Conclusion

- **Local LLM + RAG:** Real-time adaptivity, data privacy, better domain Q&A.
- Even simple RAG **significantly** improves domain-specific Q&A.
- Fine-tuning not always necessary; data retrieval is often enough.