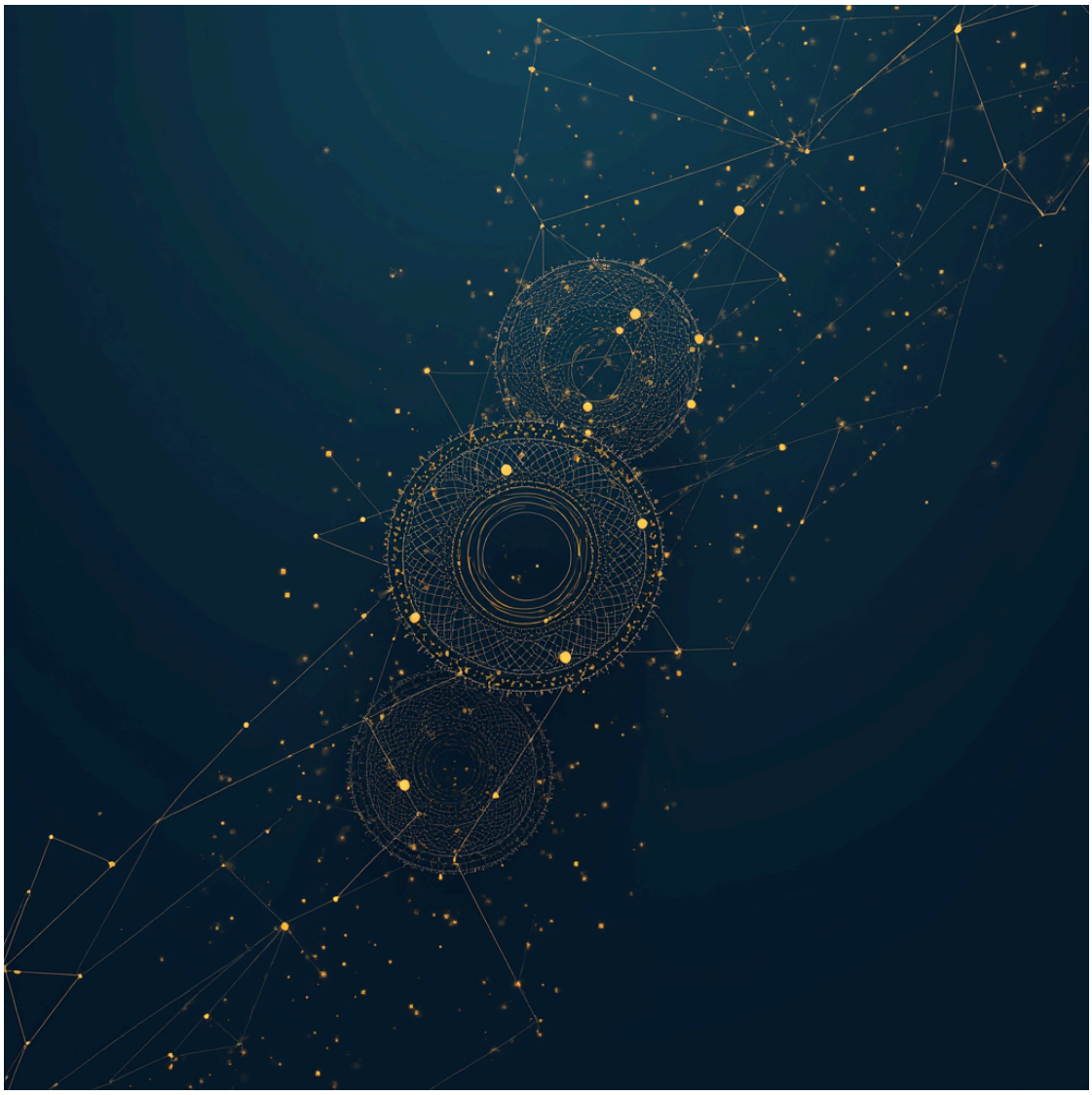
Report: Morocco Investment Knowledge Base (RAG-Based QA System)



Imane FJER

18.12.2024

COLLEGE OF COMPUTING

Summary

Investors seeking opportunities in Morocco face a significant challenge: critical market information is dispersed across numerous government portals, regulatory bodies, and industry websites, making comprehensive research time-consuming and potentially incomplete. To address this challenge, we developed an intelligent Question-Answering system using Retrieval-Augmented Generation (RAG) technology that transforms how investors access and interpret Moroccan market information. Our solution combines advanced web scraping techniques with Ollama's Llama3.2 language model to create a dynamic knowledge base that delivers accurate, context-aware responses to investor queries.

The system stands out through its ability to ground responses in real-time data from authoritative Moroccan sources, significantly reducing the risk of AI hallucinations common in traditional language models. Our web application features an intuitive interface where users can query investment-specific information. Through careful prompt engineering and ethical data collection practices, we've created a reliable tool that demonstrates marked improvement over standard LLM approaches, particularly in accuracy and source verification.

Introduction

In today's rapidly evolving global investment landscape, accessing accurate and comprehensive market information about Morocco presents a significant challenge for potential investors. While the country offers numerous opportunities across sectors such as renewable energy, automotive manufacturing, and aerospace, crucial information remains fragmented across multiple government portals, investment agencies, and industry websites. Traditional approaches either require time-consuming manual research or rely on Large Language Models (LLMs) that may generate responses based on outdated training data, potentially leading to inaccurate or incomplete information.

To address this challenge, we developed a Retrieval-Augmented Generation (RAG) system that bridges the gap between scattered information sources and investors' needs by combining the analytical capabilities of LLMs with real-time retrieval from authoritative Moroccan sources, ensuring responses are both current and factually grounded in official documentation.

Project Objectives

1. **Contextualized Q&A:** Enable investors to submit targeted questions and receive responses grounded in relevant Moroccan investment sources.
2. **Efficient Data Retrieval:** Automate scraping and indexing of authoritative websites, while respecting `robots.txt` guidelines.
3. **RAG Integration:** Combine retrieved source material with an LLM to enhance factual accuracy and reduce hallucinations.
4. **User-Friendly Interface:** Present an intuitive web interface for querying information and managing sources.

Data Pipeline

1. **Source Identification:**
   Identify key Moroccan investment websites known for authoritative information.
2. **Ethical Scraping & Compliance:**
   Check `robots.txt` before scraping. Respect any disallowed paths and focus on permitted content.

3. **Web Scraping & Text Extraction:**
   Use `BeautifulSoup` and a custom loader to extract text from approved webpages, filtering out non-relevant elements.
4. **Chunking & Preprocessing:**
   Apply a `RecursiveCharacterTextSplitter` to break text into ~1200-character segments. This segmentation enhances retrieval granularity and context relevance.
5. **Vectorization & Storage:**
   Convert each text chunk into vector embeddings using Ollama-compatible embeddings. Store these embeddings in a ChromaDB vector database for efficient semantic similarity searches.

Retrieval-Augmented Generation (RAG) Implementation

Concept:
RAG introduces a retrieval step before LLM inference, providing the model with topically relevant context. This ensures that answers do not rely solely on the model's built-in knowledge.

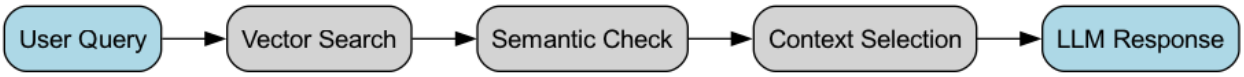
Chosen Technologies:

- **LLM: Ollama's Llama3.2:**
  Runs locally, avoids external API costs, and maintains privacy. Offers robust capabilities suitable for processing the retrieved content.
- **LangChain Framework:**
  Orchestrates prompt construction, retrieval steps, and LLM invocation, streamlining the integration of multiple components.
- **Chroma Vector Store:**
  Stores and retrieves vector embeddings of text chunks, enabling quick similarity searches.

Process:

1. Embed the user query and retrieve top-k relevant chunks from the vector store.
2. Construct a prompt that introduces the model's advisory role, includes retrieved context, and specifies instructions to rely strictly on that context.

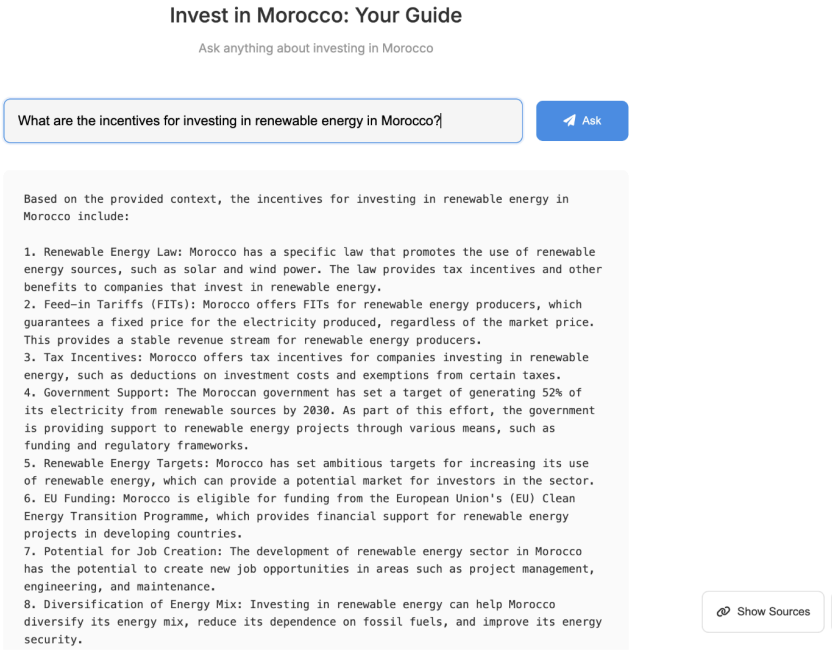
3. Instruct the model to return a fallback message if the requested information is not found.
4. The LLM (Llama3.2) then produces a factual, context-grounded answer.



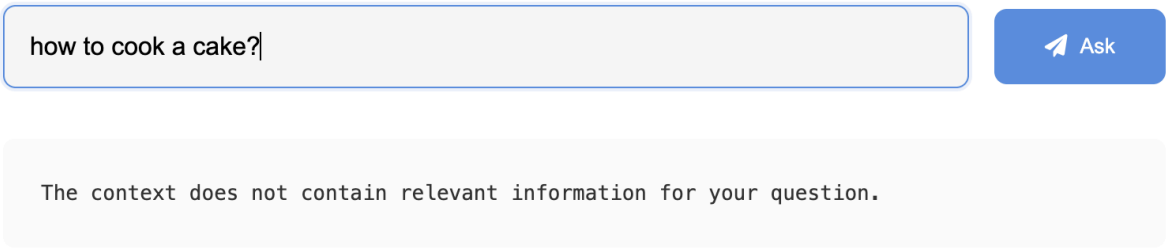
User Interface & Experience:

The web application's interface is built with Flask, HTML, CSS, and JavaScript. Key features include:

- **Query Input Field:**
  Enables submission of investment-related questions (e.g., "What incentives exist for foreign investors in the automotive sector in Morocco?").
- **Ask Button:**
  Triggers the RAG pipeline, returning a concise, context-informed response.
- **Sources List:**
  Displays indexed sources for transparency, showing the origin of retrieved information.



If no context is available at all, the system may display another fallback or error message:



Comparing RAG vs. LLM-Only Responses

Example Query:
"What tax incentives are available for foreign investors in Morocco's automotive sector?"

- ***LLM-Only Output:***
  Without retrieval, the LLM might produce vague or incorrect details, as it relies solely on pre-trained knowledge and may invent information.
- ***RAG-Enhanced Output:***
  With retrieval, the LLM references the provided chunks. If tax incentives are mentioned, the model states them factually. If not, it explicitly acknowledges the lack of information.

Key Challenges and How They Were Addressed:

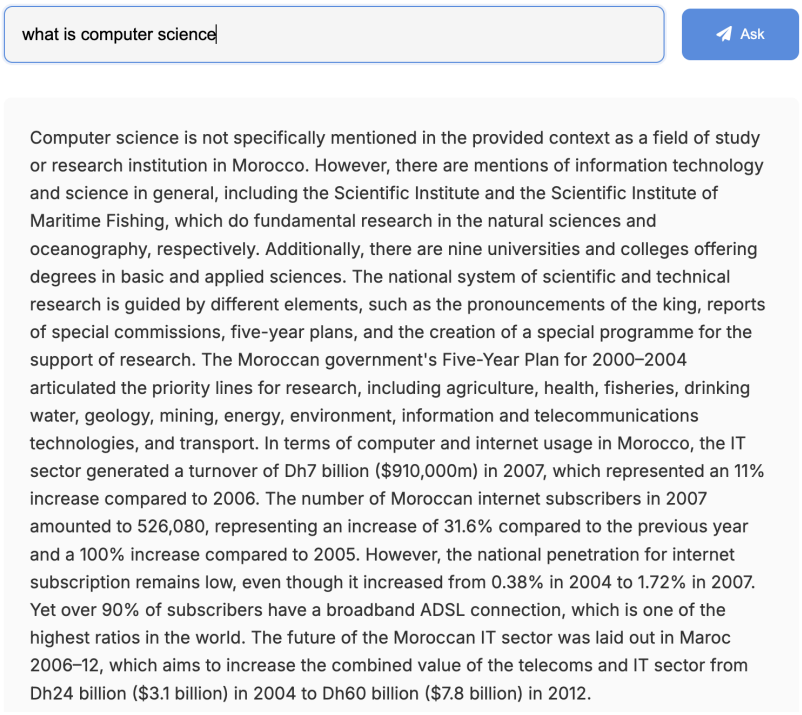
➔ **Prompt Engineering:**
Tested various prompt structures. Determined that an optimal approach involves:
- ◆ Explaining the model's role.
- ◆ Supplying the context.
- ◆ Presenting the question.
- ◆ Instructing the model to refuse unsupported answers by stating a fallback response when data is absent.

```
  prompt = f"""You are an investment advisor for Morocco.
IMPORTANT: Only use information that is explicitly present in the provided context. Do not add any
external knowledge or make assumptions.
Instructions:
- Only use information explicitly mentioned in the context.
- Do not elaborate beyond the given information.
- If the information isn't explicitly  in the context, say : "I cannot find this specific information
in the available data."
- Keep responses short and professional unless the user specifically requests more detail.
- Do not make predictions or assumptions.
- Keep the response focused, factual and to the point.
Question: {question}
Context: {context}
Response:"""
```

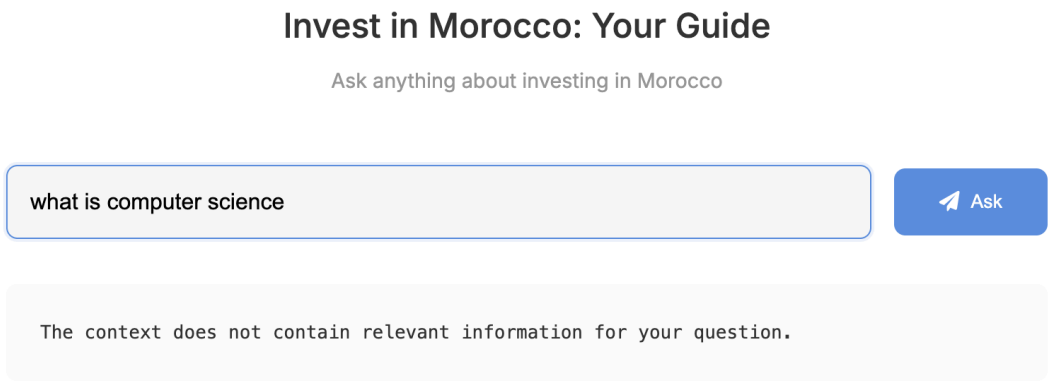
- ➔ **Accuracy & Hallucination Mitigation:**
  Setting a low temperature and emphasizing reliance on retrieved data minimized unsupported claims. This approach grounded the LLM in actual documents, reducing hallucinations.
- ➔ **Ensuring Data Relevance:**
  Selecting reputable websites, segmenting documents into coherent chunks, and storing embeddings in ChromaDB improved the precision of retrieved context.
- ➔ **Dual-Stage Relevance Check:**
  To enhance response accuracy and prevent irrelevant answers, the system implements a two-stage relevance verification process:
  - ◆ **Initial Vector Similarity Search:**
    - Uses ChromaDB's vector similarity search to identify the top 5 most relevant document chunks.
    - Efficiently narrows down the search space from the entire knowledge base.
    - Operates at the document level for quick filtering.
  - ◆ **Semantic Similarity Verification:**
    - Uses OllamaEmbeddings (with "all-minilm" model) for semantic similarity calculation
    - Calculates cosine similarity between the question and retrieved context.
    - Enforces a minimum relevance threshold of 0.2
    - Prevents responses when the context isn't sufficiently relevant.

This dual-stage approach was implemented after observing that the system sometimes provided responses using tangentially related context. For example, when asked about computer science, the system would attempt to construct responses from marginally related business context. The semantic similarity check now ensures that responses are only generated when the retrieved context is genuinely relevant to the query.

Before:



After:



➔ **Ethical Data Collection:**
Compliance with `robots.txt` and attention to site permissions maintained ethical standards. The approach acknowledges that real-world data acquisition

must respect content owners' restrictions.

Leveraging Generative AI Tools in Development:

Throughout this project, generative AI tools played a valuable supporting role in enhancing various aspects of the development process:

- **Prompt Engineering Optimization:**
  - Used ChatGPT and Claude to experiment with different prompt structures and formats
  - Iteratively refined prompts based on AI-generated suggestions to improve response quality
  - Tested various instruction patterns to better control the model's output style and format
- **Code Development Debugging:**
  - Identify and fix bugs in the web scraping implementation
  - Used AI-powered code review to ensure best practices and maintain code quality
- **Documentation & Reporting**
  - Enhanced technical documentation by correcting grammatical and syntax errors, improving sentence structure, and ensuring consistent terminology
  - Refined the project report structure and content organization

The use of generative AI tools complemented human expertise and accelerated the development process while maintaining high quality standards. However, all AI suggestions were carefully reviewed and validated before implementation to ensure accuracy and appropriateness for the project's specific requirements.

Future Enhancements:

Several paths exist to improve performance, reliability, and coverage:

- **Expanded Data Sources:**
  Integrating additional official Moroccan government and industry websites can increase data quality and coverage, further enhancing answer accuracy.
- **Support for Diverse Formats:**
  Incorporating PDFs, tables, and other document types would broaden the range of retrievable information. This capability would allow the system to answer more

complex queries that rely on structured or semi-structured data.

- **Hardware Acceleration:**
  Running Ollama's Llama3.2 model on a GPU rather than a CPU would reduce response latency, improving user experience. Harnessing GPU capabilities can make the system more responsive.

Conclusion:

The RAG-based application for Moroccan investment queries demonstrates a substantial improvement over an LLM-only approach. By anchoring answers in retrieved, authoritative sources and employing careful prompt engineering, the system significantly reduces hallucinations and increases factual accuracy.

Current capabilities streamline context-anchored Q&A, enhance accuracy, and maintain ethical data sourcing. Future enhancements include expanding official sources, adding support for diverse data formats, and leveraging GPU resources for improved performance. Such ongoing improvements can ensure that the platform remains a trusted, versatile tool for accessing and understanding Morocco's investment information.