Question & Answer (Q&A) pipeline
– Report –

Balázs Menkó

November 10, 2025

Overview

This project implements a Question & Answer (Q&A) pipeline based on the Retrieval-Augmented Generation (RAG) approach. The goal is to combine information retrieval and language generation to deliver accurate, context-aware answers from a knowledge base built from Wikipedia articles.

Description

The project is implemented in Python 3.12. The following tools and libraries were used:

- **Vector Store:** Chroma
- **Pretrained LLM:** Llama3.2 (3B) from Ollama
- **Interface:** Streamlit
- **Containerization:** Docker

Project #1: Wikipedia Q&A Pipeline

The pipeline extracts keywords from the user's question to retrieve relevant Wikipedia articles. Initially, Wikipedia's search engine was used, then `llama3.2` generated searchable keywords via the `extract_keywords_from_question()` function. Articles are fetched with the `wikipedia` and `wikipedia_api` packages (`fetch_wikipedia_articles()`), saved as `.txt` files, and stored as embeddings in a Chroma vector store (`create_chroma_collection()`).

Semantic search uses multi-qa-MiniLM-L6-cos-v1 from `sentence_transformers`, mapping sentences to 384-dimensional vectors. Finally, `llama3.2` generates answers using the top five most relevant contexts.

Project #2: Board Game Manuals Q&A

To evaluate the pipeline on a known dataset, 11 board game manuals were used:

- Terraforming Mars
- Dead of Winter: A Crossroads Game
- Bang!
- Kill the Unicorns
- Terra Mystica
- Dune: Imperium
- We Didn't Playtest This at All!

- Sushi Go Party!
- Azul
- UNO
- Catan

The same pipeline steps from Project #1 were applied. A `Streamlit` interface was created for usability: streamlit_board_game_qa.py.

A test mechanism was implemented using Mistral 7B to evaluate answer similarity against expected results. Results are in pytest_result.txt.

Further Improvements

- Add the ability to refresh the database dynamically
- Allow users to ask multiple questions in a single session
- Experiment with other LLMs and support different languages (e.g., Hungarian)

Results

Based on the PyTest, four questions were answered correctly out of five. The test took 15 minutes and 2 seconds, which means an average query can be solved in about three minutes. This seems somewhat long, but the test was run on a CPU-only machine (Intel Core i7 - 2.4 GHz), which may explain the slowness.

Fortunately, the failure was not a hallucination. The RAG pipeline correctly identified that the question was not answerable based on the dataset, even though it could theoretically be answered.

References and Sources

Websites

- HuggingFace Models
- RAG Tutorial GitHub
- Ollama Search Model

YouTube Tutorials

- How to containerize Python applications with Docker
- The Only Docker Tutorial You Need To Get Started
- Chroma Python Basics

- Build a RAG in 10 minutes! | Python, ChromaDB, OpenAI
- Python RAG Tutorial (with Local LLMs): AI For Your PDFs

ChatGPT was used to improve development efficiency.