OrderFox

Retrieval-Augmented Generation LLM Agent

Bayes Brigade

Ben Ellis, Harald Semmelrock, Felix Schatzl, Sebastian Brunner

Executive Summary

After removing redundant and useless data (such as CSS) from the corpus, we load the data into 3 different databases: vector (ChromaDB), inverted index (using BM25) and extracted named entities into an SQL (sqlite) table. We then query all 3 storages (configurable) for each query, pass the results to a critic LLM, and pass the relevant documents to a response LLM. To use the tool nicely, we provide a simple Web UI.

Parsing & Preprocessing

We removed data by:

- regex pattern matching on the URLs (e.g. .css, javascript)
- removed "_jb_static" and "_static\" URLs
- removed data not encoded in UTF-8 (such as PDFs)
- used MinHash to remove duplicates (such as headers, footers) on a subset of the data (due to time/compute constraints, ~60% effectiveness)

With this we reduced the dataset from 17 GB to 5 GB.

Knowledge Base Design

We used three data storage containers:

- **Vector database**:
    - Used ChromaDB due to simple setup and integration.
    - Used "all-MiniLM-L6-v2" model for embeddings (on GPU)
        - Due to time constraints, no other models could be considered
    - Tried different chunk sizes (512 up to 4096) with sliding windows
        - Final model used 4096 with a step size of 2048
- **Inverted Index**:
    - Uses TF-IDF as an alternative way of retrieving documents
- **Named Entity SQL Table**:
    - Extract domain specific information, such as region, products, client and parent companies etc.

Retrieval System

- Pass the user-query through our openAI o1 mini model, which is prompt engineered to construct a vector database query from the users input to maximize positive results.
- Build in context from conversation history
- Retrieve top-k documents from each database
- Separately returns page URL, filename and page text

Generation Agent
Constructed iterative refinement on answer generation

- **Prompt LLM:** Uses the conversation history and information about our database system to rewrite the user query to retrieve more relevant results
- **Filter LLM**: Judges whether the retrieved pages provide information relevant for answering the user query. If it deems them relevant they are added to the context of the Agent LLM
- **Agent LLM**: Is given the filtered context and a system prompt to answer the user query, depending on whether we use strict-RAG it is requested to make judgements only based on the context or only with the help of it.
- **Critic LLM**: Judges whether the answer to the user Query can be inferred from the information in the context alone, and whether it accurately reflects the information there. In strict-RAG mode it just removes the answer, otherwise it adds a critique to the answer stating that the information does not reflect the database
- **Reporter LLM**: Finally the Reporter concisely states the answer and possible problems present in it that the Critic detected.

Bonus

- **UI:**
  Simple UI, which allows user to interact with a context-aware chatbot and allows the user to augment the retrieval and generation steps in different ways
- **Handling of Uncertainty:**
  When we detect that our context is not enough to infer a valid answer, depending on the strict-RAG setting we either state that no valid conclusion can be made from the data, or that the provided answer is not backed up by our database and may be inaccurate.

Our results were very good! We were able to answer example queries, and many of our constructed queries that ChatGPT would not have appeared in the company's html data. Our agent ran in a reasonable amount of time, and provided very clear and helpful responses.

Key Learnings:

- Importance of data preprocessing real world messy data. If we had been able to reduce our dataset further, or had enough time to run more accurate summarization models on the messy data, constructing our databases would have been much easier.
- Need for fallback logic when retrieval fails
- Importance of prompt engineering to achieve good results

GitHub: https://github.com/Wernerson/datathon-2025