arXiv Chatbot

Erdos Deep Learning Boot Camp (June 7 - Aug 29 2024)

Team: Tantrik Mukherji, Ketan Sand, Xiaoyu Wang, Tajudeen Mamadou Yacoubou, Guoqing Zhang
Github: https://github.com/xywang2017/erdos-arxiv
App: https://erdos-arxiv-chatbot.streamlit.app/

Overview - Why Retrieval-Augmented Generation (RAG)?

RAG, in simpler terms, is a technique used to enhance the accuracy and reliability of generative AI models by using information from external sources.

It has several advantages over generalized chatbots like ChatGPT. To name a few:

- Pull in real-time data, hence most up-to-date information
- Reduces chances of incorrect information or hallucinations which generalized chatbots are prone to.
- Can cite sources for responses it generates.
- Can answer more niche questions effectively

None of this is more useful than the field of research. Where one requires in-depth information with citable sources for a query.

arXiv.org is the largest open database available containing nearly 2.4 million research papers, spanning 8 major domains covering everything there is to understand from the tiniest of atoms to the entire cosmos. A large language model (LLM) having access to such a dataset will make it unprecedented in generating updated, relevant, and, more importantly, precise information with citable sources.

This is exactly what we have done in this project. We have refined the capabilities of Google's Gemini 1.5 pro LLM by building a customized RAG pipeline that has access to the entire arXiv database. We then deployed the entire package into an app that mimics a chatbot to make the experience user-friendly.

Stakeholders - Academics, Universities, All companies R&D department ranging from Medicine to Computer Science and even economics.

Pipeline Description

- **Embedding Model -** Load a pre-trained sentence embedding model (`all-MiniLM-L6-v2`) from `SentenceTransformers`, to encode both user inputs and documents. In contrast to traditional keyword-based queries, the model encodes the semantics for similarity comparisons.
- **User Input -** Asks a user for a topic of interest or keywords to look for.
- **Data Loading and Extraction -** Upon user input, the `ArxivLoader` queries arXiv.org for up to `100` document summaries based on this topic. If documents are found, metadata (`Title`, `Authors`, `Entry ID`) and content are extracted, combined into strings, and stored in lists (`metadata`, `page_content`, `doc_identifiers`) for further processing. The SentenceTransformer model is used to encode the above documents and create a vector database for semantic query.
- **Display Retrieved Documents -** For a preview the code displays 5 random articles. For each sampled document, the title and arXiv specifier (unique ID) are shown, alongside their relevance score (0-1), measured by the cosine similarity between encoded user input and document summaries.

- **User query -** The user can now ask specific queries relevant to the topic provided initially.
- **Query embedding and Document Ranking -** The user's query combined with original topic is encoded into a new embedding vector. A cosine similarity score is used to estimate the similarity between this query embedding and document embedding. The top five most relevant documents are selected and presented to the user, along with their similarity scores.
- **RAG context generation and Prompt construction -** The metadata and content from the top five documents are used to build the `rag_context`, including document excerpts tied to their arXiv specifiers. This context is incorporated into a prompt template, which includes the user's query, topic, and the retrieved document context. The completed prompt is then sent to Google GeminiLLM to generate a detailed, context-aware response.
- **Model Response and Display:** With this narrowed down dataset, the model analyzes relevant information and returns a response for the query, which is then displayed to the user.

Results

- We stress tested our RAG+LLM pipeline on a broad spectrum of topics covered by arXiv.org, such as linguistics, condensed matter physics, astrophysics, and so on. We included a few demos in the examples/ folder. The results are generally satisfactory in the eyes of team members who are domain experts, and more quantitative metrics described as follows.
- The Document Ranking stage of the pipeline generally retrieves relevant documents based on the user query, with relevance scores in the 0.5 to 0.8 range (with 1 being the highest and 0 the lowest). Higher scores are typically obtained if the user query contains keywords that are also in the document summaries.
- The final generated response is contextualized with the retrieved documents, providing accurate answers while also citing relevant sources. A generic language model clearly would not have been able to achieve this, therefore demonstrating the success of our pipeline.
- The pipeline is deployed as a web app at https://erdos-arxiv-chatbot.streamlit.app/, which has a clean user interface, and instructions on how to use it.

Future Work

- The future work will revolve around two aspects: improvements to the core RAG module, and adding new features.
- Pre-built vector database - In the Document Loading stage, the app retrieves 100 document summaries based on the traditional keyword matching, implemented in the `ArXivLoader` module of `LangChain`. We shall improve on this by building a vector database on the entirety of arXiv metadata, which is public domain knowledge. This shall replace keyword search by semantic search in the entirety of our pipeline.
- Customized sentence LLM - In the Document Ranking stage, we used a pre-trained LLM in `SentenceTransformer` to perform semantic search based on user queries. We tested the capability of the pretrained LLM prior to deployment in our app, with some of the testing results provided in `playground/transformer_play_around.ipynb`. While it works well in general language processing (e.g., "tell me an outdoor activity to do" and "hiking in the Appalacians" have a high relevance score), its performance is worse on queries specific to arXiv.org. For example, "show me recent works on topic XYZ" does not appear to rank the retrieved documents by publication date. A domain specific sentence transformer model trained on sentence pairs from arXiv.org will significantly improve the query results.

- Whole document ingestion - to save on user query response time, currently our pipeline utilizes only the document summaries while leaving the handling of the main text to Google Gemini. We shall add a whole document ingestion feature to a future version of the app, which includes the entire textual document and figures. This can be achieved by supplementing fast document summary search with a more refined single document search backend.
- Bibliography construction - one of the most time-consuming tasks in paper writing is to find the most relevant publications and cite them appropriately. We will implement an automation bibliography generation feature where a researcher can create a bibliography based on topic and customized query.