LangChain Library:

LangChain stands out in the AI community for its focus on bridging large language models with RAG networks. It facilitates easy connections with multiple language models and offers numerous pre-defined methods for indexing and document retrieval.

Document Indexing Strategy

Given the diversity in length and content of the scientific articles related to computer science, a strategic approach to document indexing was necessary. The decision to employ the RecursiveCharacterTextSplitter with a chunk size of 1500 was influenced by the need to provide substantial context to the language model post-retrieval. This splitter enhances document division by preferentially breaking text at more natural ending points, thus minimizing mid-sentence splits. This approach not only improves readability but also optimizes the chunk length by creating smaller, more contextually isolated pieces.

Additionally, each document is enriched with metadata containing the article's title. This is particularly crucial for long documents, such as those containing code samples, where discerning the subject matter based solely on content can be challenging. The inclusion of titles in metadata provides essential context, aiding in the more effective retrieval of documents.

Embedding and Storage

For embedding the articles, the open-source model bge from Hugging Face is utilized. This model is well-suited for embedding articles due to its performance and compatibility with a variety of documents. The embeddings are stored in a vector database called "chroma," which is noted for its standalone functionality and easy integration with LangChain. This database allows for the efficient use of embeddings and facilitates their storage on disk as files. Unlike the vector database FAISS, which was initially used, chroma supports integration with the SelfQueryingRetriever, offering enhanced querying capabilities and improved retrieval performance.

SelfQueryingRetriever Implementation

To enhance the retrieval of relevant document segments, particularly from longer articles where central passages may not directly appear connected to the topic, I employed the SelfQueryingRetriever. This method proved superior due to its unique capability of leveraging a large language model (LLM) to assess not just the content of the documents but also their metadata. This approach is exceptionally effective in using the article title as a criterion for determining the relevance of a text fragment to the queried subject. The LLM utilized for this task was the freely available general-purpose model from the Hugging Face platform.

In instances where the LLM is unavailable, the system employs the max_marginal_retrieval_search mechanism. This technique strives to select appropriate documents while ensuring that the same information is not repetitively retrieved, thus maintaining the diversity and richness of the returned results.

Challenges Encountered

One significant challenge encountered was related to the initial vector database used, FAISS, which could not support the SelfQueryingRetriever. Through thorough debugging and research into the issue, I identified the cause of the error, which ultimately necessitated switching to another vector database—specifically, chroma. This change was pivotal in resolving the compatibility issues, allowing the retrieval system to fully utilize the capabilities of the SelfQueryingRetriever for enhanced document retrieval accuracy and efficiency.

Areas of improvement

- Expand the use of metadata beyond titles to include abstracts, keywords, and author information, which can provide more nuanced signals for retrieval accuracy.
- Implement dynamic chunk sizing based on the content complexity or the nature of the document instead of using a fixed size.
- exploring additional algorithms or customizing existing ones might provide better performance, especially in handling edge cases or highly specific queries.