Article Retrieval System

Weronika Hilaszek

April 11, 2024

1 Introduction

The Article Retrieval System is a comprehensive tool designed to leverage cutting-edge technology for the purpose of efficient article retrieval and insightful response generation. The primary goal of this system is to process a large corpus of data science articles from the "Towards Data Science" publication on Medium, enabling users to query specific topics and receive contextually relevant responses.

2 System design

The system consists of several key components:

- Data loading and preprocessing: The dataset is loaded into a Pandas DataFrame and preprocessed to generate insights using the YData Profiling library.
- Document splitting: Articles are split into smaller chunks using the RecursiveCharacterTextSplitter to facilitate efficient processing.
- Semantic indexing: FAISS is used to index documents based on their semantic similarity, enabling fast and accurate retrieval of relevant articles.
- Integration of the language model: GPT-3.5-turbo is integrated into the system to generate responses based on the retrieved articles.

3 Selected technologies

- FAISS (Facebook AI Similarity Search): Chosen for its efficiency in performing similarity search on high-dimensional data, FAISS handles the indexing and retrieval of articles based on content similarity.
- OpenAI's GPT-3.5-turbo: This language model generates responses based on the context provided by the retrieved articles. GPT-3.5-turbo was selected for its advanced natural language understanding and generation capabilities, which allow it to produce coherent and contextually relevant text based on the input it receives.
- YData Profiling: Used to generate a profile report of the dataset
- BAAI/bge-base-en-v1.5: I chose this embedding model after examining MTEB Leaderboard. Despite its small size (consuming only 0.41 GB of memory), it ranks relatively high compared to others.
- Langchain: It was incorporated into this project to enhance language processing capabilities, providing syntactic parsing, semantic understanding, and sentiment analysis, which synergize with GPT-3.5-turbo for more accurate and insightful responses to user queries.

4 Challenges

- Selecting tools: Before settling on the current technologies, I experimented with a variety of embedding models. I also tested ChromaDB and several other language models before finalizing my choice.
- Integration: The integration of FAISS with GPT-3.5-turbo posed certain difficulties, especially in ensuring a smooth data transition between the article retrieval segment and the response creation unit.
- API rate limits: The reliance on OpenAI's API brought about limitations concerning usage limits and expenses.
- Lack of resources: At first, I used Google Colab equipped with a T4 GPU, but after several hours of project work, I was no longer able to access it.

5 Future development

- Enhanced customization: Future versions could offer users more control over the retrieval and response generation processes, such as adjustable parameters for response length, complexity, and style.
- Expansion of dataset: Including articles from additional sources or expanding the current dataset to cover more diverse topics could significantly improve the utility of the system.
- Improved UI: Developing a more sophisticated user interface and possibly integrating web technologies.
- Other models: In the future, I would like to try different vector embeddings as well as different language models.