Erdös Deep Learning 2024 Bootcamp Aware Project Executable Summary

Afsin Ozdemir, Enhao Feng, Ness Mayker Chen

May 3, 2024

1 Overview

Retrieval Augmented Generation (RAG) is a framework of enhancing the output of generative models by providing external knowledge during generation. In the setting of question answering, RAG provides a large language model with a set of documents related to the specific query, allowing the model to generate answer that is more accurate and comprehensive.

In this project, we build a RAG pipeline that takes in a query from user, retrieves relevant information from the database, and output a summarization by a large language model. More specifically, we focus on building a model that can retrieve the information fast (sub-seconds) and can also filter out irrelevant information after retrieval. We also provide a framework using RAGAs to evaluate our model.

2 Dataset

Our dataset is the Reddit submissions and comments dataset. The topics mainly include opinions of employees from various company including Starbucks, Walmart, USPS, and etc. The text is highly noisy from a natural language perspective, so we preprocess the data by removing the emojis and correcting abbreviations into dictionary phrases.

3 Methodology

Our pipeline can be roughly divided into the following four units:

Documents Embedding

After the data preprocessing, for each single Reddit post, we concatenate the entire comments to a single context and divide it into smaller document. We also implement a sliding window chunking methods to capture better relationships between the comments.

We use text embedding models (all-MiniLM-L12-v2, GTE-Large, nq-distilbert-base-v1) to embedd each document into high dimensional vector space to obtain a dense representation. We selected GTE-Large model based on the metric:

$$\text{Accuracy} = \frac{|\text{relevant retrieval}|}{\min(|\text{retrieval}|, |\text{total relevant documents}|)}$$

We store the vector embeddings in vector store powered by Facebook AI Similarity Search (FAISS). We also associate each document with its metadata such as its topic and post type.

Retrieval

Given a user query, we use text embedding models to obtain the query vector. We then send the query vector to our database and use K-Nearest-Neighbor search to find the top N documents with the highest similarity scores (cosine-similarity) with our query vector.

Re-ranking

After we obtained the top N documents associated with our query, we concatenate the query with each retrieved document to produce N contexts. We use the cross encoder by CohereRerank in LangChain to compute similarity scores for the N contexts and reorder our documents based on these scores. We can then select a smaller set of retrieved documents based on the original and the new ranking.

Fine-tuning the ms-marco-MiniLM-L-v2 model with around 10000 training data on walmart and starbucks for 2 epochs using contrastive learning loss achieved roughly 0.3% increase in the percentage of relevant documents among top 20 out of 40 total retrievals.

Generation

We create a prompt with the query and retrieved documents to feed into the LLM (GPT-3.5-turbo, Mistral, Llama 2). We also pass the metadata associated with the documents to the LLM to enhance generation. Moreover, we also make use of the knowledge of the LLM to further determine the relevancy of the document. The model will not provide hallucinated answer if none of the retrieved documents is related to the query.

4 Evaluation

We based our evaluation on the RAGAs framework. Specifically, we retrieve the documents based on a query, rerank the documents, and evaluate the answer generated by the language model. The metrics we used are faithfulness, answer relevancy, context recall, and context precision. We see that answer relevancy and context precision have improved with the reranking process. For example, answer with large words overlap but less logical relevancy is ranked lower after reranking. However, reranking can make result worse based on the other metric. In future works, we would analyze different metrics more closely and apply a better fine-tuned coranker to the pipeline.

Acknowledgements

We thank Erdös institute for hosting this bootcamp and Aware for proposing the project.