11-711 Assignment 2 : Retrieval Augmented Generation System

Rajeev Veeraraghavan *

rveerara@andrew.cmu.edu

Abstract

This report summarizes the work done to build a stand alone Retrieval augmented generation system to answers questions on CMU (Carnegie Mellon University). The system uses publicly available data to build a knowledge base, and LLama2-7b model to generate answers to questions.

1 Introduction

2 Data Creation

2.1 Raw Data Collection

2.1.1 Which Data was Collected ?

All collected raw data were from the publicly available sources mentioned on the assignment's github repository (Neubig, 2024). All the documents listed on the page were used because they broadly covered the topics that would be asked in the question set.

2.1.2 Data Extraction

LTI Faculty The LTI faculty page was scraped using BeautifulSoup to get the names of the faculty members and their research interests. The scraped text was stored in raw fashion with one file per faculty member. These files were transformed into a single JSON file and also into a descriptive text file where each sentenced contained exactly one faculty member's information.

LTI Faculty Research Semantic Scholar was used to get the research papers (and abstract, metadata) of the faculty members. The data was parsed into a JSON file which had a list where each member contained information about one paper.

Academics @ LTI The old LTI website landing page was scraped using BeautifulSoup to get gists of the courses offered by LTI. The scraped text was stored in raw fashion with one file per course, and fed directly to the embedder. The program handbooks were downloaded as pdfs and were then parsed using PyPdf to generate text files. The PDFs were directly used in one set of embeddings while the text files were used in another set of other embeddings.

CMU Courses The CMU course schedule catalog was scraped using BeautifulSoup to get the schedules of all courses. The scraped data was converted into a JSON file and into a combined text file where each sentence contained a worded form of one course's information. The academic calendars for 2022-23 and 2023-24 were downloaded as PDFs and parsed into text files. Both formats were fed into different sets of embedders.

History, Athletics and Events The relevant pages were scraped with BeautifulSoup, and PDFs were downloaded where available. As the documents here had different inherent structures, most of the data was manually copied into text files and parsed with separate scripts to convert the raw data into sentence form.

2.2 Annotations

2.2.1 Type and Quantity of Annotations

A total of 115 test annotation pairs were created manually - 31 for the history and events category, 15 for the LTI program handbook category, 30 for the CMU courses category, 20 for the LTI faculty category, and 19 for the LTI faculty research category. This split covered the different types of data collected, and the different types of questions that could be asked on them. No train annotations were created, since there was no fine tuning or few shot prompting being performed. The annotations

---

were used to select the best models and embeddings. All the reference answers were kept down to 5 words with multiple variations being accepted (eg - 11711; 11-711; 11711 course; 11-711 course; 11711 course at CMU; 11-711 course at CMU etc). Reference answers were kept as short as possible with the goal of selecting a model that gave concise answers (to maintain the F1 score), while maintaining recall. This likely resulted in low f1 scores, but the idea behind this is to select a model that is concise. As I worked alone on this assignment, I could not calculate an inter annotator agreement score.

3 Model Details

3.1 Constraints

The RAG system was built on a GCP compute instance - n1-highmem-4 with 4 core CPU, 1 NVIDIA T4 GPU, 26 GB RAM, and 160 GB SSD. This influenced the choice of inference and embedding models. Langchain (Docs) was used to build the embedder and retriever modules due to its ease of use and integrations available.

3.2 Embedding Models and Vector Store

I chose to use the bge-large-en-v1.5 BGE embedding model as it is an established model in the top 15 of the MTEB leader board. Further, the v1.5 model update embedding model: release bge-*-v1.5 embedding model alleviates the issue of the similarity distribution and has ehanced retrieval ability without instruction, compared to earlier bge models.
I experimented with the chunk sizes for both embeddings to find out the which size fits best. I used recursive chunking for greater granularity and variety of text, varying the chunk overlap between 0.1 and 0.4 times the size of the chunk size to see what effect the overlap had on the retrieval results. The embeddings were stored in a Chroma vector database due to its support for multiple formats and ease of use for a first time user.
I also evaluated llama2-7b embeddings and compared the two sets of embeddings to see which one gave better results. My hypothesis was that the llama2-7b embeddings along with the llama2-7b model for inference would provide comparable results to using the BGE embedding with llama2-7b and similar sized models.

Embedding + Model	F1 Score %
Llama2 Embedding + Llama2	5.6
Llama2 Embedding + Mistral	5.9
Llama2 Embedding + Open chat	7.8
BGE + Lllama2	11.6
BGE + Mistral	10.5
BGE + Openchat	21.1

Table 1

3.3 Inference Models

I used the following suite of open source models available with Ollama: llama2-7b, mistral, openchat and neural-chat. I expected the 4 models to give different performances across the metrics measured while all being light weight enough to run locally with a reasonable infernce time. Among the 4 models, I expected to choose 2 for the final unseen test set submission, based on their relative peformance F1, recall and exact match. I did not expect any of these models to do very well at exact matching as they are all chat models.

4 Results

As shown in the table 1, the BGE embeddings gave better results than the llama2-7b embeddings across different inference models. The BGE embeddings had a higher F1 score, with a significant difference. Further, the BGE embeddings with the Openchat model gave the best results, with a 21.1% F1 score.

4.1 Comparison of Models

5 Analysis

Acknowledgements

References

Lang Chain Docs. LangChain. [link].

Graham Neubig. 2024. nlp-from-scratch-assignment-spring2024. github repository.