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 Data Creation

1.1 Raw Data Collection

1.1.1 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.

1.1.2 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.

1.2 Annotations

1.2.1 Type and Quantity of Annotations

A total of 90 test annotation pairs were created manually - 31 for the history and events category, 15 for the LTI program handbook category, 21 for the CMU courses category, 14 for the LTI faculty category, and 9 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

*This work was done as part of the 11-711 course at Carnegie Mellon University. The assignment started off with a group of 3 members, but after 21st February, I had to complete the assignment individually. Permission to work alone was taken from the course instructor. Permission to use material that the group had worked on together, prior to splitting up, was also taken from the course instructor. Shared work has been attributed to contributors.

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.

The annotations canbe further classified into two categories based on the number of documents needed to answer the question - single document and multi document questions. This classification is used in the analysis section to evaluate the performance of the models across different types of questions.

1.2.2 Single Document Questions

When was the Carnegie Plan initiated ? This is a simple question whose answers can be found in one document alone. The document itself is distinct from other documents in that it is not a scheduler or a list, but a file containing some history facts in the form of sentences. I expected most models to get this answer correct, as long as the retriever was able to retrieve the correct documend among all the ones it retrieved.

On which days do Kiltie band rehearsals happen during football season ? This is a similar to the previous question and its answer can be found using a single document about the Kiltie band. The number of relevant documents here were greater than 2 since there were multiple mentions of the Kiltie band. But, the specific question is answered directly in the text.

What is the course number of the capstone project for MS-AII students ? This question differs from the previous two in that many more documents (chunks) would have an embedding similarity score close to this, and there are two possible documents that can answer this question - One is the relevant excerpt from the MS AII handbook and another is from the cmu course schedule file.

1.2.3 Multi Document Questions

Which paper did Bertsch, Uri, Matthew R and the instructors of courses 11711 and 10601 publish together ? This is a hard question since at least 3 relevant documents need to be retrieved - The LTI papers chunk and two chunks from the course schedules. Since the schedules chunks were likely largely similar due to the number of overlapping words they had, the retriever would often retrieve the wrong set of documents. Even when the retriever retrieves the correct set of documents, the reranker might not rank the 3 documents needed at the top

1.2.4 Annotation Quality Estimation

As I worked alone on this assignment, I measured inter annotator agreement by ranking all 90 questions twice - at different points in time. Each question was assigned a score of 1-4 based on difficulty, with 4 being reserved for the most difficult questions. I expected to see an agreement of 0.75 across all random samples. I did multiple random sampling with a sample size of 30, and the observed agreement varied between 0.67 and 0.97 and the average agreement scores after running random sampling for 100 iterations was 0.63, indicating moderate to substantial agreement across both sets of annotations.

2 Model Details

2.1 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.

2.2 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-en-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.

2.3 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.

Model	Chunk overlap	F1 Score %	Recall
Llama2	0.2	7.35	18.13
Openchat	0.2	11.81	19.49
Neural Chat	0.2	4.82	13.39
Llama2	0.3	6.72	17.28
Openchat	0.3	8.17	16.86
Neural Chat	0.3	5.03	16.06

Table 1: Evaluation results with Llama2 embeddings

Model	Chunk overlap	F1 Score %	Recall
Llama2	0.2	5.72	24.38
Openchat	0.2	10.89	30.87
Neural Chat	0.2	7.42	22.19
Llama2	0.3	7.60	28.74
Openchat	0.3	11.52	29.41
Neural Chat	0.3	8.46	23.25

Table 2: Evaluation results with BGE embeddings

3 Results

3.1 Embeddings

I compared performance across two embeddings to choose an embedding model - **bge-large-en-v1.5 and llama2-7b**. Both embeddings were created with text documents with a chunk size of 1000. I used F1 score as the primary metric to compare the embeddings, along with recall. No embedding filter was used to rank the retrieved documents to feed to the model as context. The results are shown in tables 1 and 2. The same set of parameters were used across all experiments for a fair comparison.

In most cases, the BGE embedding proved superior, which was as expected. The low F1 scores are likely due to the models generating long answers, and the reference answers being short. The BGE embeddings were chosen for the final model. In terms of signifance, using openchat with BGE embeddings proved superior to using llama2 embeddings across chunk overlaps and F1 and recall scores. The most significant p-value for F1 scores was p=0.155, observed with the open chat model using embeddings with chunk overlap of 0.3. As the p-value is greater than 0.05, we fail to reject the null hypothesis that the F1 scores across the two embeddings are the same, at the 95% confidence level. The results are not signifacant at p = 0.05 confidence level.

3.2 Comparison Between Models

I used the BGE embeddings and attempted to improve F1 score by adding (Compression). I hypothesized that contextual compression (Compression) would pick out the most relevant parts of the retrieved documents. I opted for an embeddings filter (Docs, 2024a) over an llm chain filter (Docs, 2024b), since it is faster. I evaluated metrics on the same set of threee models as before. All the compared models used an embedding chunk size of 500, as that gave marginal peformance improvements over a chunk size of 500 and Top-k sampling with k value of 15. The results are shown in tables 3.

Among the three models, there was no significant difference in recall with most pair wise comparisons resulting in p values of 0.3 to 0.4, favoring Openchat. The F1 scores had more variation with Openchat outperforming the other two models in most cases. As seen in table 3, the most significant p-value for F1 was observed between OpenChat and neural chat, with Openchat being significantly better than neural chat at the p=0.05 level. The performance difference between Openchat and LLama2 was not significant at the p=0.05 level.

4 Analysis

The annotations contained questions that required different types and numbers of documents to answer them. For example, most questions from the history category could be found from the context within one document. Similarly, questions from the schedules category that simply asked where or when a class was held were expected to be easier for the RAG system. Questions from the LTI faculty and LTI research categories were also of two types - ones whose answers could be found within one document or ones that needed collection of information from multiple documents. By document, I refer to the chunked documents here. Questions such as *Which author of the paper titled " Unsupervised Dense Retrieval Training with Web Anchors" did not publish any other paper in 2023 ?* required information from all documents corresponding to LTI papers. This question was not expected to be answered correctly by any of the baseline models. I categorized the question based on a hypothetical difficulty level during annotation creation, and evaluated the peformance of the models I tested across these difficulty levels. The results of this evaluation are given below.

4.1 Single Document Questions

When was the Carnegie Plan initiated ?
Surprisingly, most of the basic models got this question wrong, and none of the models operation on top of a llama2 embedding vector store were able to get it right. Given that any model that is provided the answer somewhere in its context should get this question correct, I investigated the retriever to see if the relevant chunk was being retrieved. Using the BGE emneddings, the chunk was retrieved on occasion, but in general - almost no retriever got the right chunk. This is possibly because the question is generic in the sense that it does not contain a unique structure in terms of what it is asking nor does it use any unique words. To deal with this type of queestion, a better retrieval mechanism is most likely needed.

On which days do Kiltie band rehearsals happen during football season ?
Unlike the previous question, most models got this question correct, since the question is specific to the Kiltie band and football. The retreiver worked well possibly because there were few documents that contained information about the Kiltie band or football. Since this was a very direct question, most models got it right, provided that the retriever retrieved the right chunk.

What is the course number of the capstone project for MS-AII students ?
Most systems using the BGE embedding got this question right, while the ones using llama2 embeddings did not. This is because using the BGE embeddings resulted in the correct chunk being retrieved, while using llama2 embeddings often did not. The main reason for this could be that BGE creates much better sentence level embeddings and can thus match this query to the right chunk from the MS AII handbook documents.

4.2 Multi Document Questions

Which paper did Bertsch, Uri, Matthew R and the instructors of courses 11711 and 10601 publish together ?
Given that I was using sentence level embeddings, I did not expect any of the retrievers I built to retrieve all the documents relevant to this question. This was indeed the case - no retriever retrieved all the correct documents needed to answer this question. The only way to get this correct was if the retriever got all the documents of papers author by the people mentioned, and the reader subsequently made

Models	Chunk overlap	F1 Scores%	p
`Openchat, LLama2`	0.4	13.35, 9.64	0.15
`Openchat, Neural`	0.4	13.35, 8.55	0.02
`Openchat, LLama2`	0.3	13.75, 8.47	0.13
`Openchat, Neural`	0.3	13.75, 7.79	0.05

Table 3: Evaluation results with BGE embeddings and different models

the correct guess about the paper's name. Questions similar to this one constituted 30% of the test set. I had attempted to develop retrievers that could deal with the complexity needed for these questions, but could not succeed in doing so. None of the systems whose outputs I directly checked were able to answer this question correctly.

Acknowledgements

I acklowledge the contributions of the following people to this assignment and report:
Sudeep Agarwal (sudeepag@andreew.cmu.edu) *contributed to the scraping of raw data for academic calendar schedule, LTI faculty and LTI faculty research*
Hugo C (hcontant@andrw.cmu.edu) *contributed to the scraping of raw data for Kiltie band, Scottie and Tartans*

References

Lang Chain Contextual Compression. [link].

Lang Chain Docs. LangChain. [link].

Lang Chain Docs. 2024a. Local retrieval qa. LangChain.

Lang Chain Docs. 2024b. Local retrieval qa. LangChain.

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