End-to-end NLP System Building: Retrieval Augmented Generation

Shanru Lin Zhixian Zhang Xuanang Zhou
Carnegie Mellon University
{shanrul, zhixianz, vincentz}@andrew.cmu.edu

Abstract

In this project, we present an end-to-end Retrieve Augmented Generation (RAG) system designed to provide accurate, up-to-date information about Pittsburgh and Carnegie Mellon University across multiple domains, including sports, art, culture, and history. The system integrates a custom-built corpus, retriever, embedder, reader, and optional reranker to optimize both information retrieval and response generation. We systematically evaluated several configurations of the pipeline and achieved notable improvements over the baseline. The optimal configuration, using the BGE-M3 embedder and Llama-2-7b-chat-hf reader, yielded a precision of 0.3804, recall of 0.6640, and F1-score of 0.4737, representing improvements over the baseline of 853%, 724%, and 799%, respectively. These results demonstrate the system's efficacy in handling diverse informational queries.

1 Introduction

Large Language Models (LLMs) have transformed Natural Language Processing (NLP) in recent years, especially with the release of ChatGPT in 2022, which highlighted their potential. However, standard LLMs are limited by their reliance on static, outdated data. Retrieve Augmented Generation (RAG) addresses this issue by using an up-to-date corpus to enrich responses, allowing the model to provide more accurate and relevant information (Gao et al., 2024).

The RAG model combines key components for efficient information retrieval and generation. The corpus provides a large data repository, from which the retriever pulls relevant text snippets. The embedder transforms these snippets into vector representations for fast similarity calculations. The reader then interprets both the retrieved information and user query to generate answers, while an optional reranker refines the ranking for improved accuracy (Lewis et al., 2021). In our project, we used this standard pipeline.

Our project aims to build a RAG-based search system for Pittsburgh-related information, allowing users to query a wide range of topics accurately. We created the corpus through custom web scraping to gather comprehensive local data. Embedder, reranker and reader models were selected for their lightweight design and strong performance. To evaluate and refine the system, we systematically annotated a diverse dataset, establishing a solid foundation for assessment.

Through this project, we contribute to the development of RAG systems tailored for domain-specific information retrieval, providing valuable insights for localized knowledge access and enhancing user interactions with city-specific data sources.

2 Data Creation

This section outlines the methodology for data creation, including the compilation of knowledge sources, web scraping techniques, and data processing steps.

2.1 Knowledge Source Compilation

The knowledge resource was constructed by curating relevant documents from selected root websites. The inclusion of documents was determined based on their relevance to the research objective. Raw data was extracted primarily from the textual content present in the main sections of the websites. The extraction process was conducted using standard web scraping tools, ensuring accurate and structured retrieval of textual data.

2.1.1 Web Scraping

The web scraping process involved treating the suggested websites as root domains. From each root website, the following content was extracted:

Text from the main content. All relevant second-level URLs from the menu bar and main content, referred to as "non-trivial" URLs. A non-trivial URL was defined as a link originating from the root website itself, including relevant external links such as restaurant listings from the Pittsburgh Restaurant Week website. External links unrelated to the main content, such as Instagram or LinkedIn, were excluded. For each identified second-level URL, the same extraction process was applied, capturing relevant third-level URLs and their associated textual content. This approach ensured comprehensive coverage of the information available at each level of the website.

2.1.2 Tools

The web scraping process utilized requests for retrieving web content, BeautifulSoup for HTML parsing, and Selenium to handle dynamic content generated by JavaScript. Additionally, PyPDF2 was employed for extracting text from PDF documents when necessary.

2.1.3 Processing

During the processing phase, two main issues were addressed:

1. **Reorganizing Incomplete Sentences:** Content that was incomplete or presented as fragmented, such as schedule tables, was restructured into coherent, grammatically correct sentences. This step was essential to maintain readability and ensure the extracted content was consistent in form (see Appendix for an example).

2. **Reinforcing Context in Dispersed Information:** When related information was spread across different sections or rows, we ensured that important context was repeated to avoid loss of meaning during the embedding process. This was particularly important in cases where rows of data (e.g., a table of vendors) lacked clear contextual references (see Appendix for an example).

2.2 Data Annotation

The dataset includes 54 manually annotated questions, covering three topics (General Information about Pittsburgh and CMU, Sports, and Music and Culture) and six question types (who, what, where, when, which, how). Each combination had three questions, resulting in 54 questions in total. The variety in topics and question types ensured balanced coverage for the corpus and limiting each combination in three questions ensures the dataset being manageable and training/testing efficiency. We randomly select 4 of them for few-shot training and utilize the remaining 50 for testing.

Inter-Annotator Agreement (IAA) was employed to measure the quality of our data annotation, assessing the consistency between annotators when classifying the same items. We measured IAA using Cohen's Kappa (κ), which is calculated as:

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{1}$$

where p_0 is the observed agreement among annotators, and p_e is the probability of chance agreement based on the data.

In our study, 3 annotators evaluated 54 question-answer pairs across two criteria: correctness (correct/incorrect) and relevance (relevant/irrelevant). The results showed substantial agreement, with an average Kappa score of $\kappa = 0.95$ for correctness and $\kappa = 0.79$ for relevance. These high Kappa values indicate strong consistency in the annotations, demonstrating reliable agreement on both the correctness of answers and the relevance of questions.

3 Model details

3.1 The RAG Pipeline

We chose **LangChain** (LangChain, 2023) for the RAG pipeline due to its modularity, ease of integration, and support for hybrid retrieval. LangChain simplifies managing components while allowing seamless experimentation, making it an ideal choice for building flexible, scalable pipelines with minimal development complexity.

3.2 Embedding Model

The embedding model converts queries and documents into vector representations. Given our limited computing resources and budget, along with the need to process a large 18.8MB text corpus, we selected two models to balance efficiency and performance (see Table 1 for comparison):

BGE-M3 (Chen et al., 2024) was chosen for tasks requiring high accuracy and handling large-scale data. It supports dense, multi-vector, and sparse retrieval, processes up to 8192 tokens. Its hybrid retrieval capability combines embedding-based and sparse methods without extra computational cost, making it suitable for large datasets despite our resource constraints.

GTE-large (Li et al., 2023) leverages the BERT architecture and is trained on a vast dataset of

relevance-based text pairs. It delivers 1024-dimensional embeddings with a 512-token limit, offering an efficient solution for environments with limited computational resources. It was designed to supports various applications like information retrieval, semantic similarity, and text reranking across diverse domains and tasks.

Model Name	Dimension	Sequence Length
BAAI/bge-m3	1024	8192
thenlper/gte-large	1024	512

Table 1: Embedding model parameter comparison.

3.3 Retriever

We used **FAISS** as the retriever to efficiently identify relevant information from the corpus. The corpus was divided into small chunks using Recursive Text Splitter, with overlapping applied to ensure completeness. By embedding these chunks and indexing them with FAISS's similarity search algorithms, we enabled fast retrieval of the most relevant text in response to user queries. For each query, the top 5 text chunks were retrieved to capture all pertinent information.

3.4 Re-ranker

After retrieving all the related documents from the knowledge base, we integrate a reranker to re-rank the documents based on their similarity with the question. This process can rearrange the priority of each document input to the reader model, and help the reader to generate better answers. The model we used for the reranker is **bge-reranker-v2-m3** (Chen et al., 2024), which is designed to be a powerful and efficient reranker for multilingual contexts. It can be used to rerank the top-k documents retrieved by an embedding model, such as the bge-m3 model.

3.5 Reader

Reader is responsible for generating text answers based on input, which is prompting combination of retrieved chunks and user query in our projects. Considering factors of lightness and performance, we choose HuggingFaceH4/zephyr-7b-beta and Llama-2-7bchat-hf as our reader.

Zephyr-7b-beta (Tunstall et al., 2023) is chosen as the reader in our RAG system due to its effective design as a lightweight, GPT-like model fine-tuned for high-quality conversational responses. Zephyr-7B-beta, a 7-billion-parameter model, leverages Direct Preference Optimization (DPO) on synthetic datasets to enhance helpfulness and accuracy in information-rich tasks. Its strong performance on benchmarks like MT-Bench and AlpacaEval highlights its capability for precise response generation, even compared to larger models.

Llama-2-7bchat-hf (Touvron et al., 2023) is another reader model we use for the RAG system due to its capability in integrating the information from these documents into coherent and contextually accurate responses, leveraging its understanding of complex language patterns and domain-specific knowledge. While larger models might offer marginally better accuracy, they also require significantly more computational resources. The 7B version provides a good compromise, delivering robust performance without the computational overhead of the largest models in the LLaMA series.

To enhance the reader's performance, we fine-tuned several hyperparameters. We enabled do_sample=True with a low temperature setting of 0.2 to increase output determinism. Additionally, we applied a repetition penalty of 1.5 to avoid redundant outputs, ensuring concise responses. To limit the range of candidate words, we set top_k=10 and restricted the cumulative probability with top_p=0.8, both of which help in selecting the most relevant words for generation. We also set max_new_tokens=50 to keep outputs brief and focused. Lastly, we employed beam search with num_beams=3 to find the optimal answer, further improving the quality of generated responses.

3.6 Variations

In developing our RAG system, we explored several variations of the pipeline by combining different models for the embedding, reader, and reranker components (see Table 2 for comparison). We established a baseline by using zephyr-7b-beta as the reader without any reranker, allowing us to measure the incremental benefits of integrating more sophisticated models and reranking strategies. Combining bge-m3 with zephyr-7b-beta and bge-reranker-v2-m3 provided a setup aimed at maximizing accuracy. The bge-m3 embedding model's dense, multi-vector, and sparse retrieval capabilities, coupled with the refined selection from bge-reranker-v2-m3, resulted in high precision in document retrieval and answer generation. The combination of gte-large with zephyr-7b-beta focused on balancing computational efficiency with performance. The other

variance we tried was bge-m3 together with Llama-2-7b-chat-hf. The deep contextual understanding provided by Llama-2-7b-chat-hf resulted in better results in our annotated dataset.

4 Experiments and Results

We performed experiments on different combinations of our RAG system components. The experiment results are shown in the Table 2. The combination of bge-m3 and Llama-2-7b-chat-hf, without a reranker, achieved the best performance with a precision of 0.3804, recall of 0.6640, and F1-score of 0.4737. We selected this configuration as our RAG system to generate test answers.

To assess the performance of our RAG system, we evaluated each configuration across three key metrics: precision, recall, and F1 score. Given that we tested various configurations with two embedding models, two reader models, and an optional reranker, these metrics help us determine the optimal combination for accurate and relevant information retrieval.

Precision: Precision is the ratio of relevant items among those retrieved, reflecting the accuracy of retrievals.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

Recall: Recall measures the ratio of relevant items retrieved out of all available relevant items, indicating coverage.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3}$$

F1 Score: The F1 score is the harmonic mean of precision and recall, providing a balanced measure when both metrics are important.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

5 Analysis

5.1 Quantitative Analysis

As shown in Table 2, the model with bge-m3 as embedder and Llama-2-7b-chat-hf as reader, significantly outperformed the baseline. These models are better suited for handling the complex and nuanced language in the test set, resulting in substantial improvements in precision by 853%, recall by 724%, and F1-score by 799% over the baseline. Moreover, compared with other models with different combination of components (like gte-large as embedder and/or zephyr as reader), the bge and llama2 models are more advanced. Their novel architectures and larger parameter counts allows them to better handle complex language and extract precise meanings.

The inclusion of a reranker (bge-reranker-v2-m3) into the model enhances the system by assigning higher relevance to retrieved chunks, thereby refining the information passed to the reader. However, in the case of the bge-m3 embedder and Llama-2-7b-chat-hf reader combination, the inclusion of the reranker led to a minor reduction in F1-score from 0.4737 to 0.4596. This drop in performance could be due to the nature of the multi-step reasoning tasks in the test set, where overly similar retrieved texts may have confused the generator.

5.2 Quanlitative Analysis

To demonstrate the effectiveness of our RAG pipeline, we compared it against a closed-book version (i.e., using only the reader model without retrieval). We evaluated both models using the same metrics as in previous experiments, with the results shown in 2. The RAG model consistently outperformed the closed-book model across all metrics, with significant improvements in precision, recall, and F1-score. This difference in performance highlights the advantages of incorporating retrieval into the model, confirming that the RAG system provides a substantial benefit over the closed-book approach in handling complex real-time queries.

To compare the system performance in detail, we took answer examples from two of the systems we built. Namely, the baseline system and the {bge-m3, Llama-2-7b-chat-hf} pipeline.

Question: Where does the annual Banana Split Festival take place?
Baseline model: Based on the provided context, I am unable to determine the answer to your question. Please provide more context or clarify your question for me to assist you further.
bge-m3, Llama-2-7b-chat-hf: Based on the information provided in the context, the annual Banana Split Festival takes place in Latrobe, Pennsylvania.

As for the baseline model, the output is generated without any contextual information, thus the language model cannot have a answer closely related to the question. The model integrating bge-m3 and Llama-2-7b-chat-hf can answer the question correctly.

Embedding Model	Reader Model	Reranker Model	Precision	Recall	F1-Score
/	zephyr-7b-beta	/	0.0399	0.0806	0.0527
bge-m3	zephyr-7b-beta	/	0.3551	0.5892	0.4214
bge-m3	zephyr-7b-beta	bge-reranker-v2-m3	0.3345	0.6583	0.4355
gte-large	zephyr-7b-beta	/	0.3171	0.5760	0.3924
gte-large	zephyr-7b-beta	bge-reranker-v2-m3	0.3482	0.5886	0.4174
bge-m3	Llama-2-7b-chat-hf	/	**0.3804**	**0.6640**	**0.4737**
bge-m3	Llama-2-7b-chat-hf	bge-reranker-v2-m3	0.3713	0.6345	0.4596

Table 2: Comparison of RAG Pipelines with Different Models and Their Evaluation Scores

6 Conclusion

This project successfully developed a Retrieve Augmented Generation (RAG) system tailored for detailed queries about Pittsburgh and Carnegie Mellon University. Integrating a comprehensive corpus with advanced retrieval frameworks and embedding models, such as zephyr-7b-beta and Llama-2-7b-chat-hf, paired with bge-m3, has significantly enhanced response precision across diverse domains. Our experiments confirm the efficacy of these dynamic retrieval mechanisms, achieving high precision, recall, and F1 scores, particularly in configurations without the reranker. These findings suggest that simpler configurations can sometimes yield better performance, especially in straightforward query scenarios. However, the inclusion of a reranker, while generally beneficial by refining document selection, can sometimes complicate responses in multi-step reasoning tasks. Comparative analyses with closed-book models further demonstrated the substantial benefits of incorporating a dynamic retrieval process, clearly outperforming models without such mechanisms. This underscores the value of continuous corpus updates and advanced retrieval techniques in improving the relevance and accuracy of responses.

References

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

LangChain. 2023. Langchain: Building a retrieval-augmented generation (rag) application. Accessed: 2024-10-25.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *Preprint*, arXiv:2308.03281.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *Preprint*, arXiv:2310.16944.

A Appendix: Examples of Processing

Case 1: Reorganizing Incomplete Sentences

Original:

> 2024 Entertainment Schedule Friday, Aug. 30 DJ Stage 6 p.m.-10 p.m. – DJ Kode Wred Main Stage 4 p.m. – Dr. James Johnson & Pittsburgh Jazz Allstars 6 p.m. – Theresa Hawthorne Band 8 p.m. – F.L.Y. 9 p.m. – Tweet Saturday, Aug. 31 DJ Stage 2 p.m.-6 p.m. – DJ Wink 6 p.m.-10 p.m. – DJ Rok Main Stage 3 p.m. – The Flow Band 5 p.m. – The House of Soul Band 7:30 p.m. – Keith Washington 9 p.m. – Ronnie Laws Sunday, Sept. 1 DJ Stage 2 p.m.-6 p.m. – DJ Tee Jay 6 p.m.-10 p.m. – DJ Schizo Main Stage 3 p.m. – Soul Raydio Band 5 p.m. – Bill Henry Band 8 p.m. – Sevyn Streeter 9 p.m. – Jadakiss

Processed:

> The 2024 Entertainment Schedule kicks off on Friday, August 30. On the DJ Stage, DJ Kode Wred will perform from 6 p.m. to 10 p.m. Meanwhile, the Main Stage lineup starts at 4 p.m. with Dr. James Johnson & the Pittsburgh Jazz Allstars, followed by the Theresa Hawthorne Band at 6 p.m., F.L.Y. at 8 p.m., and Tweet at 9 p.m.
>
> On Saturday, August 31, DJ Wink will take over the DJ Stage from 2 p.m. to 6 p.m., with DJ Rok performing from 6 p.m. to 10 p.m. The Main Stage will feature The Flow Band at 3 p.m., The House of Soul Band at 5 p.m., Keith Washington at 7:30 p.m., and Ronnie Laws at 9 p.m.
>
> Sunday, September 1, will bring DJ Tee Jay to the DJ Stage from 2 p.m. to 6 p.m., followed by DJ Schizo from 6 p.m. to 10 p.m. On the Main Stage, the Soul Raydio Band will perform at 3 p.m., followed by the Bill Henry Band at 5 p.m., Sevyn Streeter at 8 p.m., and Jadakiss closing the night at 9 p.m.

Case 2: Reinforcing Context in Dispersed Information

Original:

> After Dark Illustrations offers spooky pickle jar stickers that glow in the dark, along with a Picklesburgh balloon print and hand-drawn Pittsburgh-themed illustrations in prints, pins, coasters, and puzzles at Market Square.

Processed:

> After Dark Illustrations, *a vendor at Picklesburgh*, offers spooky pickle jar stickers that glow in the dark, along with a Picklesburgh balloon print and hand-drawn Pittsburgh-themed illustrations in prints, pins, coasters, and puzzles at Market Square.

In this case, we added context ("a vendor at Picklesburgh") to each entry to ensure clarity when the information was embedded into chunks of dispersed text, avoiding ambiguity about the vendor's connection to the event.