End-to-end Retrieval Augmented Generation

Raunak Sood, Grant Ovsepyan, Nitya Mathur
Advanced NLP
Carnegie Mellon University
{rrsood, govsepya, nityam}@andrew.cmu.edu

Abstract

Retrieval augmented generation (RAG) has improved natural language processing (NLP) systems by including relevant context when prompting large language models (LLMs) [5]. In this project, we built an end-to-end question-answering system that uses retrieval to answer questions about Pittsburgh and Carnegie Mellon University. This paper outlines the data preparation, model development, and results of our project. All of our work can be found here: https://github.com/Gamgrant/RAG/tree/main

1 Data Creation

1.1 Data curation

For each link provided in the README.md we scraped the data and divided it in a few categories and wrote separate scripts for each:

- Calendar Events with a static link, where the events could be found on a static page number associated with the link of the event table page
- Calendar events that required the dynamic search and expansion of more events on the page where static links were not present
- PDFs, which required separate scraping packages (see below)
- Web pages with primary textual information

Note that that each event had the following members: a clear title, and date, location (optional), and description of the event (optional). The depth of web scraping captured not only the information from the specified page, but also from embedded pages (increasing the depth of scraping by one level) for the following websites, where upon the inspection the home page wasn't as informative as we hoped it to be:

- https://www.visitpittsburgh.com/ (homepage only provides the general list of the events this fall without describing each individual event, so we decided to go one level deeper
- https://www.pittsburghpa.gov/ (homepage only provides the names of the open government activities, citiparks, city council meetings but doesn't describe them in detail and how to reach them)
- https://www.cmu.edu/about/ (homepage only provides a generic statistical information about the ranking, number of students and Alumni, but doesn't delve deeper into the history of CMU, CMU's traditions, and awards
- https://bananasplitfest.com/ (homepage only mentions the event, the dates but you need to search one level deeper to find the data about the vendors of the event, sponsors, and activities of the event)
- https://www.picklesburgh.com/ (similarly to the previous point, the main page only describes the generic information about the event and location, but doesn't give a detailed break down of the schedule and 3 main competitions held at the festival

1.1.1 Excluded data

Manual data cleaning was kept to a minimum, however City of Pittsburgh governmental web page (https://www.pittsburghpa.gov/) contained repetitive information about 311 response center, redundant department menu list of link tags, and repetitive policies with regards to the job positing and internships. That information was manually removed from the dataset.
From Encyclopedia Brittanica's page, the advertisement tags and partial advertisement information was scraped, which was removed from the dataset due to its irrelevance.

From other websites, repetitive information like "Contact Us" as well as associated text was removed, but the emails, phone numbers as well as social links were preserved in the .txt files

1.1.2 Tagging data

Each link, provided in the README, was scraped and wrapped between clearly defined tags to create a well-structured '.txt' file with boundaries for future document generation. This structure also enables efficient extraction of data for tasks such as generating vectorized embeddings using Pinecone's multilingual-e5-large model. Each '.txt' file contained information within sections marked by `=section_start=`, `=section_name=` and `=section_end=`. Events were tagged with `--`, and tables were enclosed by `=== Table ===` and `=== End of Table ===` tags. All textual data, events, and tables were saved within these section tags. This tagging system facilitated the transition from raw data dumps to a well-organized document format, ensuring that each section, event, and table was categorized and tagged for efficient retrieval and further analysis.

1.1.3 Tools for extraction

A variety of browser automated tools, python libraries and http clients were used in our project:

1. **Selenium**
   Type: Browser Automation Tool
   Extraction Type: Dynamic Content Extraction

2. **BeautifulSoup (from bs4)**
   Type: HTML Parser
   Extraction Type: Static Content Extraction

3. **Requests**
   Type: HTTP Client
   Extraction Type: Static Content Extraction

4. **lxml**
   Type: HTML and XML Parser
   Extraction Type: Static Content Extraction (XPath)

5. **WebDriverManager**
   Type: Driver Manager
   Extraction Type: Tool Setup/Configuration

6. **urllib3**
   Type: HTTP Client
   Extraction Type: Static Content Extraction

For dynamic content extraction with expandable elements for calendar events where loading content that requires user interaction we used Selenium. For the static content we used BeautifulSoup and Requests to parse static HTML pages for extracting text, tables, and structured content. For the difficult cases when BeautifulSoup wasn't sufficient, we used XPath (lxml), where the xpath was extracted from the manual inspection of the website.

1.2 Data Annotation

We annotated 150 Pittsburgh/CMU related question/answer pairs for testing based on the provided links. Our approach was to annotate a high-quality data set that, even though small due to time and annotator constraints, could still distinguish between accurate and inaccurate models. All of our annotations were done completely by hand since we wanted to ensure that our testing data was of the highest quality. We split our test set evenly into three categories:

1. General Pittsburgh knowledge (52 questions)

2. Specific Pittsburgh/CMU trivia knowledge (83 questions)

3. Very specific event/time-based information (15 questions)

The first category serves as a sanity check that our RAG pipeline is at least as good as a strong language model. We used the LLaMA-3 API to generate these types of questions as they were more general and easier for language models to ask; however, we annotated the ground-truth answers by hand to avoid spurious answers from the model. The second and third categories provide a more challenging test of whether the language model has enough information to answer specific questions that general pre-trained models would not have. This was to ensure that our retrieval method was effective as most baseline models would not have the knowledge to answer these questions. All the data in these categories were created by reading through the various scraped links and coming up with challenging questions and answers.
The largest category Pittsburgh/CMU Trivia Knowledge is a niche information, which is not commonly known or included in the language model's pre-trained data, which is why the majority of question fall into this category to evaluate RAG's performance. Furthermore, the diversity of

the questions span history, culture, events, sports, academia topics, to ensure that the breadth of system's retrieval is tested contributing to the quality of out results. Although the annotation size seems to be small, it was intentionally limited to ensure the high quality through thorough annotation.

For training data that wasn't annotated, we didn't use any extra data. For quality of your annotation estimation, please look at the Analysis section.

1.2.1 Quality of annotations

Since the quality of annotations affects the retrieval process, we performed inter-annotator analysis (IAA). Two team members annotated the same subset of data (around 50 samples) separately and the annotations were compared. We calculated the IAA score to be 0.713, indicating a substantial amount of agreement according to Cohen's kappa statistic [6]. Hence, we can reasonably say that the annotations were of high quality. Purely comparing the F-1 score and the recall for Mistral model, it is clear that the subannotated set has a lower f-1 and recall values compared to the fully annotated dataset. Although the sizes of those two datasets aren't identical, we might concur that fully annotated dataset might still have the higher quality.

2 Model Details

2.1 Model selection

In selecting the LLMs for our project, we considered various factors including model size, instruction-tuning and training data. Based on these factors, we decided to use the following three HuggingFace models:

1. meta-llama/Meta-Llama-3-8B-Instruct [2]
2. mistralai/mistral-large-2407 [8]
3. deepseek-ai/deepseek-llm-7b-chat [1]

We justify our model selection in the following subsections.

2.1.1 Model size

Initially, we wanted to use a much larger model (70b parameters) as we believed that the question-answering capabilities would be significantly better. However, we quickly realized that the inference cost for larger models wasn't entirely feasible with limited computing and time budgets. For instance, we experimented with the LLaMA3-70b model for our baseline; however, this needed four A6000 GPUs which we couldn't reliably get a hold of. Hence, we aimed to use models in the 7-15b parameter range as they comfortably fit on a single GPU. Moreover, the performance of the 70b parameter model over the smaller 8b Llama3 model wasn't significant enough to justify its use. The only exception to this is mistral-large-2407, which has 123b parameters but we found the way to use API without hitting the rate limit

2.1.2 Instruction Tuning

We also experimented with instruction vs non-instruction tuned models before deciding which type to use [9]. In our initial experiments, we tried prompting non-instruction-tuned models to generate answers to simple questions about Pittsburgh. However, we noticed that it is more challenging to direct the model towards a desired answer (i.e short, direct answers are preferred as opposed to vague long answers). With an instruction-tuned model on the other hand, we could explicitly state that we wanted the answer to be one sentence long, for instance. Accordingly, we decided that the instruction-tuned models were more appropriate for the question-answering task and decided to use these types of models in our system.

2.1.3 Training data

We aimed to use LLMs that were trained on large portions of the internet for reliable question-answering capabilities. We also looked into LLMs that were specifically fine-tuned for question-answering tasks.

2.2 Fine-tuning vs Inference Time Procedures

Another design decision we had to make was whether to fine-tune our RAG system or just use retrieval at inference time. We ended up using an inference-time procedure for the following reasons: 1.) We did not collect a large enough data set, based on LLM scaling laws [3], for fine-tuning due to time budget. We wanted to ensure that our data was high-quality and human-annotated and thus, it was not feasible to create a data set large enough for training 2.) Fine-tuning is a resource and time-intensive process that we believe would not be necessary if we constructed an inference-time procedure carefully.

2.3 RAG Procedure

In implementing RAG, we used the LangChain toolkit to create vector embeddings, find similar

vectors and retrieve the relevant context. We used Pinecone to create a large vector database based on these processed documents. PineconeEmbeddings (model: multilingual-e5-large) generated the document embeddings, which were indexed in the Chroma vector store. Then we used a vector similarity search to retrieve the context that most closely matches the given question. We then prompted the LLM, using vLLM [4], with the question and retrieved context and compared the generation to the without-context case. The final results were written to a JSONL files for analysis.

```
prompt_template_retrieval = """
Answer the question in one sentence based on the context below.

Context:
{context}

Question:
{question}
"""
prompt_template_no_retrieval = """
    Answer the question in one sentence.

    Question:
    {question}
    """
```

Figure 1: Prompt templates used for RAG.

The data preprocessing was handled by a custom DynamicHierarchicalTextLoader class that loaded and structured documents into sections, events, and tables, which were tagged (See section 1.1.2). DynamicHierarchicalTextLoader dynamically chunked large text content using a RecursiveCharacterTextSplitter to ensure that the documents remained within the required token limit for vector embedding. Each chunk, whether a text section, table, or event, was tagged with metadata (such as category, section name, and content type) to provide more accurate retrieval during question answering.

3 Results

In this section, we evaluate the baseline and RAG models on several metrics. We look at BLEU [7] score and exact match to get a general sense of the model's capability of generating accurate results. We also hand-evaluate the "correct answer" by manually reading the model generation and deciding whether it is the correct answer based on the ground truth annotation. Finally, we look at F1 score and Recall, which gives us a better sense of the model's accuracy in answering the questions.

3.1 Baseline Results

	DeepSeek	LLaMA	Mistral
BLEU-2	16.7%	18.7%	14.9%
Match	1.37%	2.07%	1.37%
Correct	30%	35%	29%
F-1 score	0.26	0.28	0.24
Recall	0.41	0.45	0.32

Figure 2: Baseline results when prompting models without retrieved contexts.

3.2 RAG Results

	DeepSeek	LLaMA	Mistral
BLEU-2	18.1%	19.6%	16%
Match	2.07%	2.75%	2.07%
Correct	33%	36%	31%
F-1 score	0.42	0.45	0.41
Recall	0.53	0.55	0.52

Figure 3: Results when prompting models with relevant retrieved contexts.

3.3 Statistical Significance

The best-performing model was LLaMA-3, which dominated on all metrics; we now determine whether the improvements seen by RAG on LLaMA-3 are statistically significant. Since a key metric to determine model success is the number of questions it answers correctly, we conducted an A/B test to determine whether the increase in number of answers LLaMA-3 gets right is statistically significant. We calculated that the improvement was statistically significant with a confidence of 95% and a p-value 0.0082.

4 Analysis

4.1 Analysis based on metrics

In this section we compare the performance of the RAG models with the baseline (LLM's generation without a context) by comaring the BLEU score, exact match, correctness, F1 score, and Recall. To begin with, BLEU-2 and exact match metrics are less relevant for performance evaluation. Exact match is a very strict metric, since the model can output the correct answer that will not match the reference answer exactly. By examining the baseline and RAG results, it is clear that the Exact match is in the range of 1.37% - 2.75%, which is not indicative of the performance. With regards to BLEU-2 score, it measures the extent of matching of texts based on

overlapping 2-grams (or n in generic case). BLEU score does not account for the semantic similarity, but instead focuses on lexical overlap between the reference and generated text. The BLEU score can be low even when different words convey the same meaning, despite the answer being correct, which is why we refrain from heavy reliance on this metric. However, BLEU-2 is still higher for all the RAG models compared against the baseline by 0.9% - 1.4%.

The "Correct" metric is an indicator of the manual evaluation of correctness. LLaMA model answered the questions with the highest 'correctness' with no context (35%), while DeepSeek and Mistral - with 30% and 29% correspondingly. with provided context of relevant documents, LLaMA still is the highest performer - 36% correctness (1% improvement) against DeepSeek model (33% with improvement of 3%) and Mistral (31% with improvement of 2%). DeepSeek has the hightest improvement % of correctness (3%).

All the model showed substantial improvements in F-1 scores. DeepSeek improved the F-1 score from 0.26 to 0.42 (61.5% improvement), LLaMA - from 0.28 to 0.45 (60.7% improvement) and Mistral - from 0.24 to 0.41 (70.8% improvement). It is clear that Mistral achieved the highest relative improvement, implying that the context significancy improved the balance of precision and recall, while LLaMA and DeepSeek closely matched in relative improvement. This implies that supplying the relevant information can significantly boost models performance, and the effectiveness diminishes more without the context for Mistral model than for other models. Note that F-1 score is the highest for LLaMA model in both settings, which means that LLaMA model is better at correctly identifying true positives (tokens that are present in both generated and reference answer) while minimizing false positives (tokens that are predicted but not in the correct answer) and false negatives (tokens that are in correct answer but missing from predicted answer).

Recall measures the ability of the model to retrieve all relevant instances from the dataset. and we can see that it improves for all the models with RAG. DeepSeek improves recall from 0.41 to 0.53 with a relative improvement of 29.3%), LLaMA improves from 0.45 to 0.55 (with 22.2% improvement), and Mistral - from 0.32 to 0.52 (62.5% improvement). Mistral model benefits greater in terms of relative improvement in recall than any other models, however LLaMA has the highest recall in both settings. High recall in baseline case for LLaMA means that model is inherently better at generating correct answers without the context. And in RAG case, highest recall for LLaMA model (with an improvement in recall) suggests that model can effectively utilize the additional context information and generate more correct answers, with a smaller chance of committing important details (minimizing false negatives).

4.2 Hidden Test Set

Based on the results and our analysis, we decided to use the LLaMA-3 model with RAG in our submission on the hidden test cases because it seems to perform the best on our test set.

4.3 Examples

Question: Who is the current director of finance of the city of Pittsburgh?
Baseline Response: As of my last update, the director of finance for any city, including Pittsburgh, can change over time due to appointments, elections, or other administrative changes.
RAG Response: Jennifer Gula is the current director of finance of the city of Pittsburgh.

Question: What opera is being performed by an all-Japanese and Japanese American creative team at the Benedum Center in March 2025?
Baseline Response: As of my last update, I do not have specific information about future opera performances scheduled for March 2025 at the Benedum Center or any details about the creative teams involved.
RAG Response: The opera being performed by an all-Japanese and Japanese American creative team at the Benedum Center in March 2025 is Madama Butterfly.

Question: When does Pittsburgh Restaurant Week Winter 2025 end?
Baseline Response: Pittsburgh Restaurant Week typically runs for a week in January for the winter edition, but the exact dates can vary each year.
RAG Response: Pittsburgh Restaurant Week Winter 2025 ends on Sunday, January 19.

In these examples, we can clearly see that the model with RAG is able to answer certain questions much better than the baseline. These questions are more specific and require information

that the model was likely not trained on to answer correctly. Accordingly, our RAG pipeline is able to retrieve the relevant context and help the LLM answer the question correctly.

References

[1] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

[2] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

[3] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

[4] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

[5] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

[6] Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica vol. 22,3*.

[7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

[8] Mistral AI Team. 2024. Mistral large 2. Available at: `https://mistral.ai/news/mistral-large-2407/`.

[9] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.