AskCMU: Advancing Inquiry with End-to-End LLM-Driven Question-Answering Architecture

Jiarui Li[*]
Information Network Institute
jiaruil3@andrew.cmu.edu

Ye Yuan[*]
Information Network Institute
yeyuan3@andrew.cmu.edu

Zehua Zhang[*]
Information Network Institute
zehuazha@andrew.cmu.edu

Abstract

This study investigates the application of Retrieval Augmented Generation (RAG) to improve the factual accuracy of Large Language Models (LLMs) for queries related to Carnegie Mellon University (CMU) and the Language Technology Institute (LTI). Addressing the challenge of LLM hallucinations, we curated a specialized dataset from CMU's extensive resources and employed an automated annotation process to enhance data quality. Our system integrates a RAG pipeline with a Retriever and Generator model, notably utilizing LLaMA 2 for its advanced capabilities in handling complex linguistic patterns. Through experiments with both pretrained and fine-tuned models, we demonstrate the system's effectiveness in generating more accurate and reliable answers to CMU and LTI-specific questions. This research highlights the potential of RAG systems in augmenting LLMs with external datasets for improved performance in knowledge-intensive tasks.

1 Introduction

Large Language Models (LLMs) such as ChatGPT3.5 and Llama 2 have demonstrated their efficacy in question-answering (QA) tasks(Brown et al., 2020; Touvron et al., 2023). These models are trained on vast amounts of text data from various sources, enabling them to understand and generate human-like responses to questions across a wide range of topics. While LLMs acquired such ability through extensive pretraining, it is evident that information of some scale is retained in LLMs' parameters(Li et al., 2023). LLMs are known to hallucinate: the model may generate results that are irrelevant or random outputs that may hold no correlation with inputs or desired outputs (Huang et al., 2023). Hallucination is often unreliable and may be misleading or offensive to the users. Many reasons are proposed to explain LLM hallucination, such as limited model capacity or insufficient pretraining. While pretraining should cultivate the model's emergent ability, it is not expected to help the model remember all the pretraining data accurately. Recently, methods such as In-Context Learning have been investigated to discover LLM's performance on certain tasks such as sentiment classification in a few-shots setting. By providing a few examples related to the desired tasks, the model could solve the expected problem with relatively higher accuracy without the need to change its parameters. Retrieval Augmented Generation (RAG) (Gao et al., 2023) is recently regarded as a promising solution to extend the model's generation capacity on tasks that require high factuality by introducing context information from external datasets. Connecting the models to external datasets, RAG could help LLMs handle knowledge-intensive tasks and allow continuous knowledge updates and integration of domain-specific information.

Our project aims to investigate and utilize the aforementioned RAG pipeline with a customized dataset we created related to Carnegie Mellon University (CMU) and the affiliated Language Technology Institute (LTI). While existing LLMs or QA systems built upon them do not contain sufficient related information, they are likely not able to give out the factually correct answers to the related questions. Thus, by augmenting the question with retrieved relevant documents as context, the model should perform better and give more credible answers.

2 Dataset Creation

2.1 Web Crawler

2.1.1 Data Crawling

To create an accurate and complete dataset that contains enough information to answer questions, we constructed a customized Web Crawler using Sele-

[*]These authors contributed equally to this work.

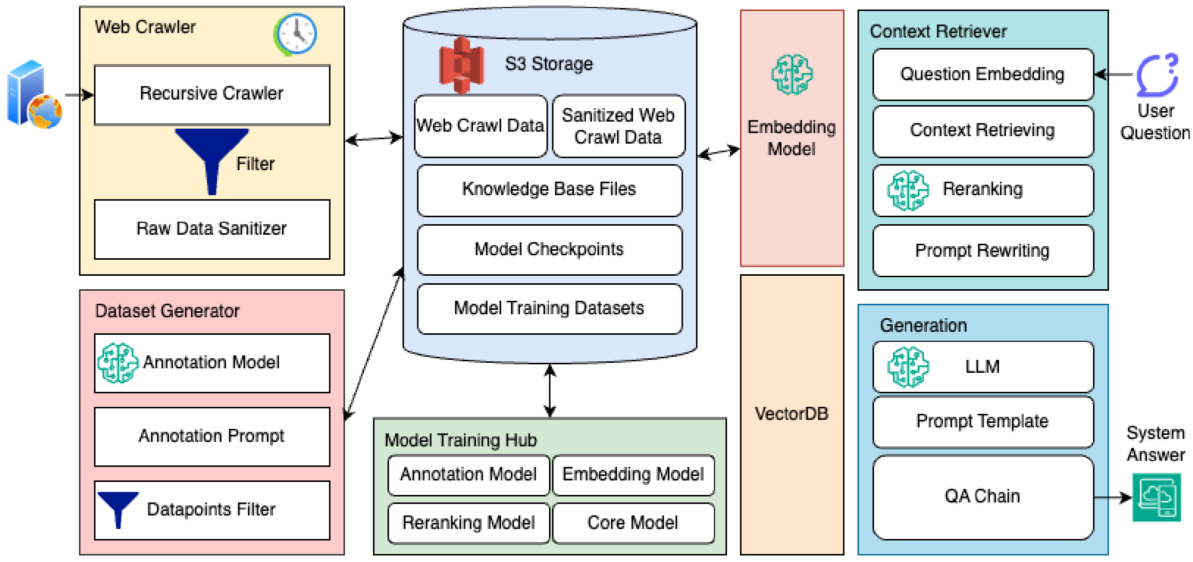

Figure 1: Overview of our system design. The process begins with a Web Crawler, which consists of a Recursive Crawler and a Filter to gather raw data. This raw data is sanitized and stored in an S3 storage bucket. The Dataset Generator relies on an Annotation Model to prompt annotations and filter datapoints, generating finetune datasets for all models in the system, i.e. Annotation, Embedding, Reranking, and Core Models. For query processing, the Context Retriever embeds the user question and retrieves relevant contexts, which are re-ranked and rewritten if necessary. The Generation module then utilizes the Core Model and a Prompt Template to generate a system answer through a QA Chain, resulting in an answer that leverages both retrieved information and generative capabilities for accurate and context-aware responses.

nium and BeautifulSoup libraries to crawl through the CMU websites. Specifically, we first stored all the links provided on the instructions page. We additionally ran a Breadth-First-search of depth 2 sourcing from each webpage, so our program would also visit and store links embedded in it. We then ran the crawler on all links. The crawler would automatically pull all the webpage HTML source codes. Since information useful for potential questions may be sparse, preprocessing including but not limited to removing Javascript codes and HTML tags and removing website text headers is also conducted to ensure data quality.

2.1.2 Data Organization and Post-processing

The crawled data was initially stored in two formats, HTML and PDF, based on the original content format. These were then processed to extract textual information, resulting in a collection of text files organized into html/ and pdf/ directories. Both html pages and pdfs are obtained from the internet. We differentiate the data in this way due to the inherent difference between html pages which usually is messy and short and pdfs which is often organized and lengthy. Additionally, a sample/ directory was created to house a curated set of data points selected for their cleanliness and informativeness, serving as the foundation for initial question-answer pair generation.

To ensure the relevance and quality of our dataset, we implemented a post-processing step where text files lacking keywords related to CMU and LTI (e.g., "cmu", "carnegie mellon university", "tartans") either in their content or titles were removed. We also filtered out files with less than 200 characters and those with "Page_not_found" titles, ensuring that our dataset is both relevant and of high quality. Python package *unstructured* is used to perform the post-processing.

This meticulous process of data creation and curation has equipped our retrieval augmented generation system with a robust knowledge base, enabling it to deliver precise and informative responses pertaining to the Language Technology Institute and Carnegie Mellon University.

2.2 Research Papers

On the other hand, since a majority of problems may correlate with the research conducted in CMU and LTI, we handcrafted the LTI faculty name list as a reference and used it in the Semantic Scholar API to search for and filter all open-sourced papers published in 2023. All fetched papers are downloaded. Paper PDF files, along with other PDF files we downloaded from CMU websites, are converted into plain texts and stored.

Text files generated from academic papers often have lengths of more than 10,000 words, and they are often too long for the LLM context windows. Thus, all raw data are split into chunks of 1000 words that could be processed by most LLMs after tokenization.

2.3 Annotation Automation

After the construction of the external dataset, we proceed with the annotation. We have 304 html documents, 12 pdfs, and 245 papers. While human annotation could achieve the highest accuracy, we unfortunately do not have sufficient resources to annotate our data in this fashion. Thus, we wish to explore using pretrained LLMs as data annotators. Three of our options are GPT4All(GPT, 2023), Llama 2 (Touvron et al., 2023), and WizardLM(Xu et al., 2023), which are all open-sourced on HuggingFace. We selected some data points from each data catagory, including, html, pdf, and papers, that are more informative and cleaner to be the representatives. Initial QA pairs are generated on this selected subset of all the samples. This is meaningful in the following two aspects: (1) We perform a manual examination of sample QA pairs generated by the three models, and choose WizardLM, the best annotator. (Sample outputs A) (2) We use the human-validated QA pairs as few-shot examples to direct the future QA generations on the larger dataset, enhancing the overall annotation quality.

Overall, for each data chunk, we use WizardLM(Xu et al., 2023) to help generate 10 QA pairs. In total, we have generated 34,781 QA pairs, where 27,824 pairs are used as training data and 6,957 pairs are used as testing data after random split.

3 Question-Answering Pipeline

As shown in Figure 2, we present our curated RAG QA pipeline. We first assembled a customized dataset containing CMU and LTI-related information, of which we have elaborated on in section 2. Aside from the external dataset, our RAG pipeline includes a Retriever model that would retrieve the most relevant information from the dataset and return them as context for later generation. A generator model, oftentimes an LLM, would take the returned information and append it into the QA

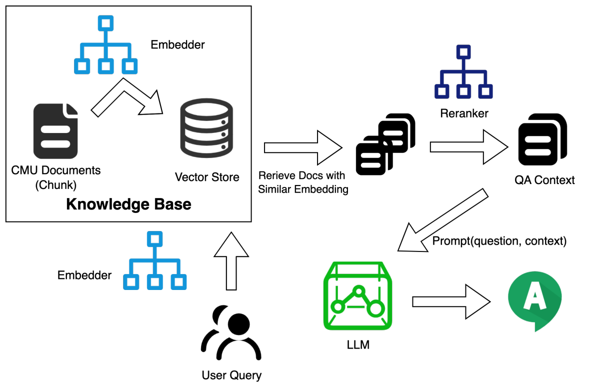

Figure 2: Overview of RAG QA Pipeline, which can be divided into retrieval phase and generation phase. In the retrieval phase, the retriever fetches top 5 reference chunks with maximum similarity in terms of `mmr` score, which is then sent to reranking model to prioritizing the most relevant information for the given user inquery. In the generation phase, the generative model takes the rewritten prompt as input and completes the answer.

prompts as context to generate answers. All modules are trained end-to-end, and we will state our model specification below.

3.1 Embedding Model

For the retrieval part, we used the pretrained embeddings model from Mixedbread.ai (HuggingFace Model ID: mxbai-embed-large-v1)(Lee et al., 2024) for its compact size and high performance on MTEB benchmark(Muennighoff et al., 2022), which measures embeddings models across tasks like classification, clustering, reranking and so on. It benefits from the proprietary dataset that contains over 700 Million pairs using contrastive training, despite an additional 30 million triplets for finetuning. The great coverage of data domains ensures its performance on both pretrained and fine-tuned settings. Thus, aside from the pretrained model, we also experimented with fine-tuning the embedding model on the train split of our QA pair dataset for 17390 steps with a batch size of 8.

3.2 Reranking Model

In our project, we enhance query result relevance through a custom reranker named BgeRerank, which employs the BAAI/bge-reranker-large model based on a CrossEncoder architecture. This model intricately assesses pairs of queries and documents, assigning relevance scores that reflect each document's alignment with the query's intent. By leveraging advanced natural language processing techniques, BgeRerank reorders the initial broad set of documents, prioritizing the top-N (top-5 out of 10 documents retrieved in our experiments) most pertinent ones to the user's query. Our integration of this state-of-the-art reranking process into the retrieval pipeline ensures improved accuracy and user satisfaction by delivering results that closely match the query's semantic nuances, streamlining the path to the most relevant information.

3.3 Core Model

For our core generation model, we chose to use LLaMA 2(Touvron et al., 2023) as the base model. This state-of-the-art large language model, developed by Meta AI, has demonstrated impressive performance across a wide range of natural language processing benchmarks. Pretrained on 2 Trillion tokens and further fine-tuned on 100k human annotation data, LLaMA 2 could capture linguistic patterns and domain knowledge, enabling it to generate highly fluent and coherent text. Importantly, LLaMA 2 is an open-source model released under a permissive license, which enables our continuous training and experiments with it. LLaMA 2 has been optimized for scalability and efficiency. It used group query attention (GQA)(Ainslie et al., 2023). Reducing the number of transformer heads, GQA significantly decreases the number of key-value pairs within the transformer blocks and thus lowers the memory requirement. This feature could further help us efficiently fine-tune the baseline with our customized dataset. Also, its context windows doubled to 4096 in comparison to LLaMA 1.

4 Experiments

4.1 Setup

In our experiments, we extensively tested the performance enhancements brought by finetuning the LLaMA 2 model and embedding models on a meticulously constructed QA pair dataset. For the core model finetuning, leveraging the meta-llama/Llama-2-7b-chat-hf checkpoint from HuggingFace, we embarked on a finetuning journey with our QA dataset comprising questions, answers, and reference texts. The finetuning process was optimized by quantizing the model to INT4 and incorporating LORA with a rank of 16 to maintain computational efficiency. The training regimen spanned 5 epochs with 1000 maximum steps, a batch size of 8, and a learning rate of 2e-4, ensuring a balance between performance and training time.

Parallelly, for the embedding model identified as mixedbread-ai/mxbai-embed-large-v1, we adopted a novel approach to finetuning using the diverse set of QA pairs extracted from our dataset. This method focused on enhancing the model's understanding of context relevance through SentenceTransformer and MultipleNegativesRankingLoss. The embedding model was fine-tuned over 10 epochs, with a specific emphasis on warmup steps to acclimate the model gradually to the task, thereby refining its ability to discern and encode nuanced semantic relationships within the data.

Both finetuning processes were critical in tailoring the models to our specific dataset and objectives, resulting in a significant uplift in the models' ability to generate contextually relevant and accurate answers.

4.2 Ablation Study

Our comprehensive analysis centered on the effects of finetuning both the embedding and core models on our specifically curated question-answer-context training dataset. Through a meticulous ablation study, we sought to unravel the individual and combined impact of these finetuning processes on our model's performance. Key performance metrics—recall, exact match (EM), and F1 score—were meticulously calculated to provide a holistic view of the improvements. This methodical approach allowed us to identify the configurations that significantly enhanced our model's ability to understand and process complex queries accurately.

According to table 1, the analysis revealed that finetuning the embedding model and core model individually and in combination led to distinct performance trajectories across all metrics. Finetuning the embedding model yielded improvements in understanding the nuanced semantic relationships within the data, as evidenced by the uplift in recall and F1 scores. Similarly, refining the core model with our dataset enhanced its capacity for generating precise and contextually relevant answers, reflected in the improvement of exact match and F1 scores. The cumulative effect of finetuning both models was synergistic, showcasing the highest performance uplift across all evaluated metrics, underscoring the efficacy of our finetuning strategy in crafting a robust retrieval-augmented generation system.

Configuration	Recall	Exact Match	F1 Score
Raw Pipeline	0.418 (0.062)	0.0	0.232 (0.033)
+ Embedding Finetune	0.366 (0.039)	0.0	0.255 (0.031)
+ Core Model Finetune	-	-	-
+ Embedding & Core Model Finetune	-	-	-

Table 1: Performance metrics of different finetuning configurations. Both score and standard deviation from 5 independent runs. Note that exact match is zero since we measures it as exact match between the output and reference.

4.3 Case Study

Below we provide examples from the model with both fine-tuned embeddings model and core generation model.

```
Question: What is Graham Neubig's
    research goal?

Answer: Graham Neubig's long-term
    research goal is to break down
    barriers in human-human or human-
    machine communication through the
    development of natural language
    processing (NLP) technologies.
```

References

2023. Gpt4all.

Joshua Ainslie, James Lee-Thorp, De Jong Michiel, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions.

Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. 2024. Open source strikes bread - new fluffy embeddings model.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Von Werra Leandro, and De Vries Harm. 2023. Starcoder: may the source be with you!

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark.

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, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions.

A QA Pair Generation examples

We eyeballed the example QA pairs generated using three different LLMs: Llama 2(Touvron et al., 2023), WizardLM(Xu et al., 2023), GPT4All(GPT, 2023). GPT4All may generate questions that are not closely related to the context, such as in QA pair 2. On the other hand, Llama 2's outputs are often overly simple and short.

A.1 WizardLM examples

QA pair 1:

```
Question: What is Graham Neubig's
    research goal?

Answer: Graham Neubig's long-term
    research goal is to break down
    barriers in human-human or human-
    machine communication through the
    development of natural language
    processing (NLP) technologies.
```

QA pair 2:

```
question: What are the potential
    applications of advances in core NLP
     technology according to Graham
    Neubig's research?

Answer: Advances in core NLP technology
    can help improve the accuracy with
    which we can analyze, translate,
    generate, or reply to textual inputs
    .
```

A.2 Llama 2 exmaples

QA pair 1:

```
Question: What are Graham Neubig's
    research interests?

Answer: Graham Neubig's research
    interests include machine
    translation and natural language
    processing."
```

QA pair 2:

```
Question: What is the name of the
    program that the author has worked
    extensively on?

Answer: Speech models.
```

A.3 GPT4All exmaples

QA pair 1:

```
Question: What is Graham Neubig's
    research goal?

Answer: To break down barriers in human-
    human or human-machine communication
     through the development of natural
    language processing (NLP)
    technologies.
```

QA pair 2:

```
Question: What is the first major area
    of research in NLP technology?

Answer: Advances in core NLP technology.
```

B Prompt details

B.1 Dataset generation

```
f"""
### Instruction:
You are a smart assistant designed to
    help high school teachers come up
    with reading comprehension questions
    .
Given a piece of text, you must come up
    with {args.num_qas} question and
    answer pairs that can be used to
    test a student's reading
    comprehension abilities.
The questions you generated should be
    specific to the text and should not
    be too general.
When coming up with question/answer
    pairs, you must respond in the
    following format:
```
[
    {{{{
        "question": "$YOUR_QUESTION_HERE
    ",
        "answer": "$THE_ANSWER_HERE"
    }}}},
    {{{{
        "question": "
    $YOUR_SECOND_QUESTION_HERE",
        "answer": "
    $THE_SECOND_ANSWER_HERE"
    }}}}
]
```
Everything between the ``` must be valid
     array.

Please come up with {args.num_qas}
    question/answer pairs, in the
    specified JSON format, for the
    following text:
----------------
{{text}}
### Response:
"""
```

B.2 Core model

```
f"""
[INST]<<SYS>> You are an assistant for
    question-answering tasks. Use the
    following pieces of retrieved
    context to answer the question. If
    you don't know the answer, just say
    that you don't know. Use 50 words
    maximum and keep the answer concise
    .<</SYS>>
Question: {question}
Context: {context}
Answer: [/INST]
"""
```