Enhancing Factual Accuracy of LLMs through Retrieval-Augmented Generation on Pittsburgh and CMU Knowledge Bases

Xinru Li[*]
Carnegie Mellon University
xinrul@andrew.cmu.edu

Shengnan Liang[*]
Carnegie Mellon University
shengna2@andrew.cmu.edu

Muyang Xu[*]
Carnegie Mellon University
muyangxu@andrew.cmu.edu

Abstract

Retrieval-Augmented Generation (RAG) has emerged as an effective approach for addressing knowledge limitations in large language models (LLMs), particularly for domain-specific factual question-answering tasks. In this project, we design and implement an end-to-end RAG system tailored specifically for answering questions related to Pittsburgh and Carnegie Mellon University (CMU). Our approach involves extensive data collection and preprocessing from a variety of publicly available sources, followed by careful annotation and quality evaluation using Inter-Annotator Agreement (IAA) metrics. Additionally, we explore and compare the capabilities of large-scale QA models (e.g., Llama and Qwen) and smaller-scale QA models to investigate their suitability for knowledge-intensive NLP tasks. We also perform qualitative analyses to examine how retrieval methods and prompt engineering affect the accuracy and relevance of generated answers. Our study highlights the importance of retrieval mechanisms and carefully designed prompts in effectively augmenting language models tailored to specific domain knowledge. Our code is available at *ThreeRiversRAG*.

1 Introduction

Recent advancements in large language models (LLMs) have significantly improved their performance across various natural language processing (NLP) tasks, notably in factual question-answering (QA) scenarios (Touvron et al., 2023). However, despite these advancements, standalone LLMs still frequently suffer from hallucination (generating plausible but incorrect information) and outdated knowledge, particularly in specialized domains or topics requiring up-to-date facts (Lewis et al., 2021a).

To overcome these limitations, Retrieval-Augmented Generation (RAG) has been proposed as a promising solution, combining the generative capabilities of language models with external knowledge retrieval systems (Gao et al., 2024). RAG systems augment language model outputs with carefully retrieved documents relevant to a given query, thereby significantly enhancing answer accuracy and reliability (Lewis et al., 2021a). Recent works demonstrate the effectiveness of RAG across multiple knowledge-intensive NLP tasks, including domain-specific question-answering and fact verification (Touvron et al., 2023).

In this project, we build upon these advances by developing an end-to-end Retrieval-Augmented Generation system tailored specifically for answering factual questions related to Pittsburgh and Carnegie Mellon University (CMU). Such a targeted domain poses unique challenges due to the necessity of detailed local knowledge, including historical events, cultural information, time-sensitive details, and upcoming events, many of which are not covered comprehensively by general-purpose large language models.

Our methodology involves the following steps:

- **Data Collection and Preprocessing:** We compiled a targeted knowledge base by gathering publicly available information related to Pittsburgh and CMU. Sources included Wikipedia pages, official city websites, local event calendars, and historical archives. We utilized static and dynamic web crawlers (BeautifulSoup4, Selenium) to extract relevant textual content from HTML pages and employed pdfplumber and pdf2image for PDF documents.

- **Question-Answer Data Annotation:** To develop a robust evaluation framework, we generated question-answer pairs for annotation

[*] These authors contributed equally.

through a hybrid approach combining automated generation by LLMs with manual validation. This strategy ensured the relevance, clarity, and coverage of our evaluation dataset. Furthermore, we assessed annotation quality using Inter-Annotator Agreement (IAA) metrics to confirm the reliability of our data.

- **Retrieval-Augmented Generation (RAG) Pipeline:** We implemented a Retrieval-Augmented Generation (RAG) system composed of three key components: document embedding, retrieval, and generation. Specifically, we utilized embedding models and performed vector-based retrieval with Chroma. For answer generation, we experimented with both large-scale models (e.g., Qwen (Bai et al., 2023)) and smaller models to evaluate scalability. We further refined system performance through targeted prompt engineering and careful selection of text splitting strategies to optimize accuracy and relevance.

In summary, our contributions are as follows:

- We constructed a comprehensive, well-annotated dataset specifically tailored for factual question-answering about Pittsburgh and CMU.
- We developed an effective Retrieval-Augmented Generation pipeline, thoroughly evaluated across multiple retrieval strategies and configurations.
- We performed comparative analyses highlighting the role of model size, retrieval strategies, and prompt engineering in optimizing model performance for specialized, knowledge-intensive NLP tasks.

2 Data Creation

2.1 Raw Data Crawling

We constructed a comprehensive knowledge base tailored for our Retrieval-Augmented Generation (RAG) system by initially compiling 77 manually selected seed URLs. These sources primarily cover topics related to Pittsburgh and Carnegie Mellon University (CMU), including local history, cultural events, sports, and academic life. To further expand coverage systematically, we employed a breadth-first search (BFS) approach with a maximum crawl depth of 2, retrieving content from each seed URL and their immediate sublinks. To ensure relevance, retrieved sublinks were filtered using a domain-specific keyword list ("cmu", "carnegie", "mellon", "university", "tartans", "lti", "scotty", "pittsburgh", "pitts", "pit", "carnival", "trustarts", "event").

We systematically extracted textual content from webpages and PDF documents using multiple techniques. For HTML pages, we utilized two strategies based on the webpage structure:

If the webpage content could be retrieved via static HTTP requests (i.e., returned status code 200), we applied the BeautifulSoup4 parser to extract the textual content. For dynamically rendered pages (detected when static crawling returned status code 403), we employed Selenium to simulate browser interaction and extract complete text content.

Additionally, we manually collected relevant PDF files and extracted their textual content using the pdfplumber and pdf2image libraries. Specifically, pdfplumber handled text extraction directly, while pdf2image was utilized as an auxiliary method for processing PDF pages that contained embedded or image-based text.

In total, our crawling and extraction procedure yielded 21,312 webpages and 164 PDF documents. The extracted data underwent rigorous manual inspection and cleaning processes, including standardizing text by lower-casing and removing irrelevant or noisy segments, to maintain data consistency and quality across the corpus.

This comprehensive and systematically processed dataset established a solid foundation for developing and evaluating our retrieval-augmented QA system.

2.2 Data Annotation

2.2.1 Annotation Strategy

For this project, the annotated data for testing and training was drawn from a knowledge resource designed for an RAG system covering topics related to Pittsburgh and Carnegie Mellon University (CMU). The data was categorized into five primary topics: general information and history, events, music and culture, sports, and a distinct category for food-related events. Approximately 14,000 URLs were extracted, from which we primarily selected around 80 websites for the annotation. These websites were chosen based on recommendations provided in the assignment requirements to ensure comprehensive coverage and diversity across all topics. This approach allowed us to target a balanced representation of content types, ensuring that each topic was adequately represented for both training and testing purposes while maintaining the diversity necessary for robust model performance.

For each website source, we annotated between 5 and 10 QA pairs—the exact number depended on the length of the document, with shorter sources yielding fewer pairs to ensure each question remained relevant to Pittsburgh and CMU. We initially collected 698 raw QA pairs and, after filtering out those with hallucinations or where the model failed to produce correct answers, finalized **564** QA pairs for testing our RAG system. This refined dataset is consistent with established QA evaluation practices. For example, the SQuAD dataset (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017) demonstrate that even a few hundred high-quality QA pairs can provide a robust basis for evaluating QA systems, supporting the validity of our testing dataset.

2.2.2 Annotation Interface

For each website source, we leveraged the Qwen2-7B-Instruct language model (Yang et al., 2024) to generate QA pairs. We carefully designed prompts with detailed instructions and set the model temperature to 1.0 to enforce strict adherence to these guidelines. Our annotated QA pairs are stored together with references to their respective source data, enabling subsequent evaluation of annotation quality and ensuring traceability.

Our prompt strategy is designed to guide the model in generating precise, diverse, and relevant QA pairs from web-crawled documents about Pittsburgh and CMU. First, the role and context are clearly defined—ensuring that the model understands its task is to produce reading comprehension questions based solely on the provided content. Then, the instructions emphasize the need for specific and focused questions that cover various aspects, such as events, people, dates, and locations, while discouraging broad or vague inquiries. Additional guidelines prevent the generation of ambiguous questions (e.g., questions that generically refer to the document or leave key subjects unspecified) and enforce strict output formatting, including succinct answers limited to several keywords and precise responses for yes/no questions. Illustrative examples clarify the expected format and content further, ensuring that the model adheres closely to the detailed requirements.

2.2.3 Annotation evaluation

We adopt inter-annotator agreement (IAA) measurement to evaluate the quality of our test data. From our final set of 564 gold standard QA pairs, we randomly selected 50 pairs for evaluation. Two annotators, Muyang and Xinru, independently answered the questions and rated each one on a 1-5 scale based on its suitability for the RAG task. According to our evaluation standards, a question must be extremely specific—clearly stating the events and subject details—and must exclude any hallucinated information. A rating of 1 indicates that the question is extremely poor, while a rating of 5 represents the best quality. Questions rated 3 or higher are deemed reasonable for the model to answer; those rated below 3 are considered unsuitable.

The Cohen's Kappa score for the annotators' categorical answers was **0.612**, and the score for their ratings was **0.830**. In addition, the semantic cosine similarity between the annotators' answers, computed using TF-IDF vectorization, is **0.711**. These metrics demonstrate that our generated test questions are both consistent and of high quality, making them well-suited for the RAG task.

3 Model details

We implement a standard Retrieval-Augmented Generation (RAG) pipeline to build our ThreeRiversRAG question-answering system, as shown in figure 1 The process begins with preprocessing the raw text data, which we segment into chunks with certain overlap. Each chunk is then transformed into a dense vector representation using a pre-trained embedding model, and the resulting embeddings are stored in a vector database for efficient similarity search.

During the inference (or testing) phase, we load this vector database to support real-time question answering. For each incoming question, we first compute its embedding using the same embedding model. This query embedding is then passed to a retriever module, which performs a nearest-neighbor search to retrieve the top-k most relevant documents or text chunks from the vector database.

These retrieved chunks serve as external knowledge and are combined with the original question and a prompt template to form the full input to the generation model. The generation model then synthesizes this information to produce a final, contextually grounded answer.

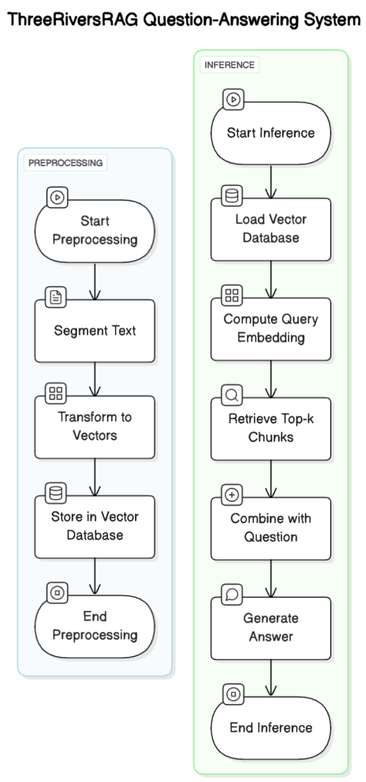

Figure 1: ThreeRiversRAG Question-Answering System pipeline

3.1 Baseline

As a baseline, we evaluate a closed-book question answering model that operates without any external retrieval mechanism or embedding model. Specifically, this model relies solely on the internal knowledge encoded in the parameters of the large language model (LLM), without access to the original text corpus during inference. This baseline experiment helps demonstrate the effectiveness of our RAG system by highlighting cases where required information is not present in the LLM's internal knowledge alone.

To ensure a fair comparison, we use the same LLM architecture as employed in our ThreeRiversRAG system. The only difference is the removal of the retriever and vector store components—there is no retrieval of relevant documents or contextual information based on the input question. Instead, the model is prompted with the question alone and generates an answer based purely on its pre-trained knowledge.

We also use the same set of annotated question-answer pairs for evaluation, allowing us to isolate the impact of retrieval-augmented context on performance. This closed-book baseline provides a clear reference point for understanding the added value of our RAG pipeline in terms of factual accuracy, relevance, and completeness of generated answers.

3.2 Embedding Models

We experiment with two pre-trained sentence embedding models: `sentence-transformers/all-mpnet-base-v2` (Khashabi et al., 2021) and `sentence-transformers/all-MiniLM-L6-v2` (Lewis et al., 2021b). Our initial choice, `all-mpnet-base-v2`, offers strong embedding quality but generates 768-dimensional vectors, leading to increased memory and disk usage. This added resource consumption can limit the memory available for LLM generation during inference.

To mitigate this issue, we adopt `all-MiniLM-L6-v2`, which produces 384-dimensional embeddings—half the size of the previous model. This switch reduces memory overhead and enables more resources to be allocated to the generation model. Furthermore, it improves system efficiency by accelerating both the construction of the vector database and the retrieval process.

3.3 Retrievers

We experimented with both FAISS and Chroma as vector stores and retrieval backends for our RAG pipeline. These tools are responsible for storing high-dimensional embeddings and supporting efficient similarity search during both indexing and inference.

FAISS (Facebook AI Similarity Search) (Johnson et al., 2019) is a library developed by Meta for efficient nearest neighbor search in large-scale vector spaces. It supports both exact and approximate indexing methods and performs well, especially when working in memory. Chroma (Team, 2023) is a newer open-source vector database built with LLM use cases in mind. It combines vector search with document and metadata storage, and primarily uses approximate search to improve speed.

In our pipeline, after preprocessing the raw documents, we chunk the text and generate embeddings using the selected embedding model. These embeddings are stored in the vector database, which can be cached locally and reloaded during inference to avoid rebuilding from scratch.

Through empirical testing, we observed notable differences in the underlying similarity search behavior of the two systems. Chroma typically uses approximate nearest neighbor algorithms to speed up retrieval, whereas FAISS—depending on the index type—can offer more precise and deterministic similarity matching. On our evaluation dataset, FAISS consistently outperformed Chroma in both retrieval accuracy and runtime efficiency, leading to better overall performance in the downstream question-answering task.

3.4 Generation Models

To evaluate how model size affects the performance of our Retrieval-Augmented Generation (RAG) system, we experiment with two language models: FLAN-T5-Base and Qwen2-7B-Instruct.

FLAN-T5-Base (Lacoste et al., 2019) is a lightweight variant of Google's T5 model, instruction-tuned to improve zero-shot and few-shot performance. With approximately 248 million parameters, it is resource-efficient and capable of producing concise answers without requiring elaborate prompt engineering.

Qwen2-7B-Instruct (Yang et al., 2024) is a much larger instruction-tuned model with 7 billion parameters. It offers stronger reasoning capabilities and improved language understanding but benefits from more carefully crafted prompts to achieve optimal performance.

By comparing these two models, we aim to understand how scaling model size influences the RAG system's answer quality and resource trade-offs.

Prompt Design. To ensure concise and focused answers, we used the following instruction prompt for both models:

> *"Answer the question based only on the provided context in just one sentence. Each answer must be as concise as possible, extremely succinct—limited to just several keywords—and should not repeat the question."*

To enforce stricter formatting for yes/no questions, we added a secondary instruction:

> *"If a question is a yes or no question, the answer must be exactly 'yes' or 'no' without any additional information and without punctuation."*

However, we observed that applying the second instruction to **T5** negatively impacted its behavior, causing it to generate yes/no responses even for non-yes/no questions. Therefore, we used this instruction only with the Qwen model.

Few-Shot Learning. We also experimented with few-shot prompting to further improve answer quality. The following examples were prepended to the prompt for both models, to make sure it generates concise answers:

```
Example 1:
Context:      Carnegie    Mellon
University    was    founded    in
Pittsburgh in 1900.
Question:   When   was   Carnegie
Mellon University founded?
Answer: 1900


Example 2:
Context: UPMC Shadyside is part
of the University of Pittsburgh
system.
Question: Is UPMC Shadyside part
of the University of Pittsburgh
system?
Answer: Yes
```

4 Results

In this section, we present the results of a series of experiments designed to evaluate the key components of our Retrieval-Augmented Generation (RAG) system. Specifically, we examine the impact of using a retriever (open-book) versus a closed-book approach, compare the performance of different generation models (small vs. large language models), test alternative retrievers (FAISS vs. Chroma), embedding models (MiniLM vs. MPNet), and assess the effect of varying chunk sizes for document splitting.

Our evaluation uses three standard metrics: Exact Match, F1 Score, and Answer Recall. Across all experiments, we observe consistent and substantial performance improvements when moving from a closed-book setup to an open-book (RAG) system. Similarly, larger language models tend to outperform smaller ones, although they are also more sensitive to prompt design. Many of these observed differences are statistically significant, highlighting the importance of both model scale and retrieval-based augmentation in enhancing the accuracy and reliability of generated answers.

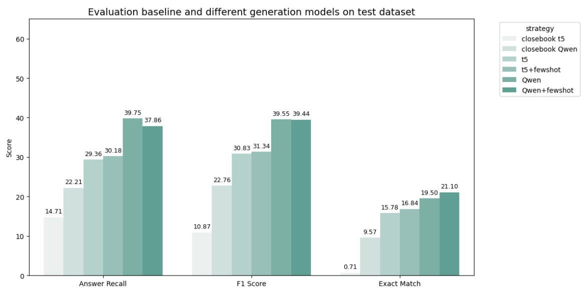

Figure 2: Performance of different generation strategies on the test set. RAG methods outperform closed-book baselines. Qwen generally outperforms T5, while few-shot prompting improves T5 but slightly degrades Qwen's performance.

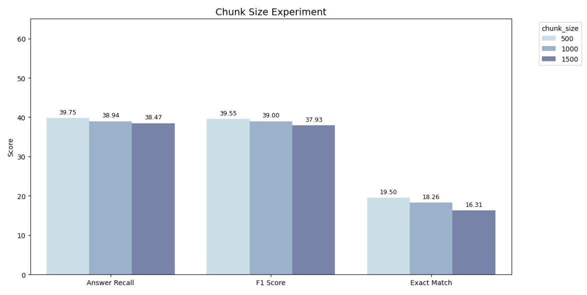

Figure 3: Performance comparison of different chunk sizes (500, 1000, and 1500 tokens) on the test set. Smaller chunk sizes lead to better scores across all metrics, with 500 tokens yielding the best performance.

4.1 Baseline & Generation Algorithms

We evaluate the performance of our closed-book baseline, small language model (SLM), and large language model (LLM) under different generation strategies. As shown in Figure 2, our RAG-based methods consistently outperform the closed-book baselines across all evaluation metrics—Answer Recall, F1 Score, and Exact Match. Among the models, the Qwen-based LLM achieves overall better performance than the T5-based SLM. Interestingly, introducing few-shot examples improves the performance of the T5 model, particularly in Answer Recall and F1 Score. In contrast, few-shot prompting slightly reduces performance for the Qwen model, suggesting that larger models may be more sensitive to prompt structure or redundancy in simpler tasks.

4.2 Embedding Models and Retrievers

We evaluate the impact of different embedding models and retrievers on RAG performance, as summarized in Table 1. Specifically, we compare two sentence embedding models: `all-MiniLM-L6-v2` and `all-mpnet-base-v2`, along with two vector stores: FAISS and Chroma.

Interestingly, `all-MiniLM-L6-v2`, despite having a smaller embedding dimension (384 vs. 768), consistently outperforms `all-mpnet-base-v2` across all metrics. This may be due to its efficiency and better generalization in lower-dimensional spaces.

In terms of retrieval backend, FAISS achieves better results than Chroma for both embedding models. This suggests that FAISS's indexing and similarity search mechanisms provide more effective nearest-neighbor retrieval in our setup, likely contributing to more relevant context for the generator.

Model	Exact Match	F1 Score	Answer Recall
FAISS + all-MiniLM-L6-v2	**19.50**	**39.55**	**39.75**
FAISS + all-mpnet-base-v2	17.20	36.23	36.68
Chroma + all-MiniLM-L6-v2	18.44	38.92	39.21
Chroma + all-mpnet-base-v2	16.48	33.59	33.82

Table 1: Performance comparison of different combinations of retrievers (FAISS vs. Chroma) and embedding models (MiniLM vs. MPNet). The best-performing score for each metric is shown in bold.

4.3 Chunk Size

We explore the impact of chunk size on RAG performance by evaluating three configurations: 500, 1000, and 1500 characters, while the overlap is correspondingly 100, 150 and 200. As shown in Figure 3, smaller chunk sizes generally lead to better results across all evaluation metrics. The 500-token setting achieves the highest scores in Answer Recall, F1 Score, and Exact Match, suggesting that shorter chunks offer more focused and relevant context for the retriever and generation model. As chunk size increases, performance gradually declines, indicating that overly long chunks may dilute the relevance of retrieved content or reduce the retrieval granularity.

5 Analysis

5.1 Event & Date Related Questions

During our experiments, we found that event-based questions posed particular challenges for the RAG system. One key difficulty is that relevant information is often hard to extract from official event websites, many of which rely on dynamic content rendering or employ tools like reCAPTCHA to block web crawlers. Additionally, even when relevant data is retrieved, large language models strug-

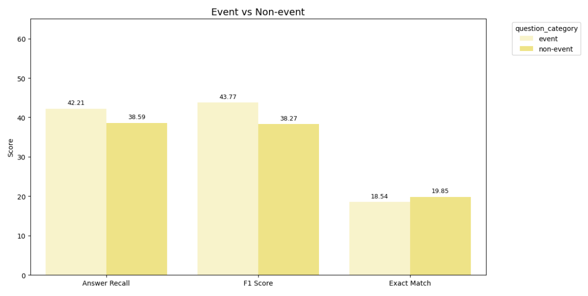

Figure 4: Performance comparison between event-related and non-event-related questions across three evaluation metrics. Despite challenges in date formatting, the RAG system performs slightly better on event-based questions in terms of F1 Score and Answer Recall.

gle with consistent formatting of date-related answers—for instance, varying in whether the year is included, the format used (e.g., "1-Mar" vs. "March 1"), or the inclusion of the weekday. Table 2 shows examples that illustrate these challenges.

To further analyze this issue, we categorized our test questions into two groups: 151 event-related questions and 413 non-event-related questions. Surprisingly, as shown in Figure 4, the RAG system performs slightly better on event-based questions across both Answer Recall and F1 Score, despite their formatting complexity. This may be attributed to the fact that many event questions include concrete, fact-based content that can be directly retrieved from context, while non-event questions are often more abstract or require synthesis across multiple documents.

5.2 RAG vs Closed-book Method

We conduct a detailed comparison between our RAG-based system and the closed-book baseline to evaluate their effectiveness across different types of questions. As shown in Table 3, for general knowledge questions—such as the founding year of Carnegie Mellon University—both the closed-book model and RAG variants (Qwen and T5) provide accurate answers. However, for more domain-specific or localized questions, such as those involving the number of CMU-affiliated startups or details about local events, the closed-book model struggles to retrieve relevant facts and often produces hallucinated or outdated information.

In contrast, both RAG variants are able to retrieve more accurate responses grounded in retrieved evidence. Among the two, the Qwen-based RAG model tends to offer slightly more precise or up-to-date answers, particularly when dealing with ambiguous or loosely structured event data. The T5-based RAG system, while slightly less precise, still outperforms the closed-book baseline in most cases. These results demonstrate the advantage of retrieval-augmented generation, especially when answering questions that go beyond the LLM's pre-trained knowledge.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

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.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *Preprint*, arXiv:1702.08734.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Preprint*, arXiv:1705.03551.

Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. 2021. Gooaq: Open question answering with diverse answer types. *Preprint*, arXiv:2104.08727.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *Preprint*, arXiv:1910.09700.

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. 2021a. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021b. Paq: 65 million probably-asked questions and what you can do with them. *Preprint*, arXiv:2102.07033.

Question	RAG	Ground Truth
What is the date of the Opera in Red Gala?	Saturday, May 17, 2025	Saturday, May 17, 2025
When is the Whitney Tribute Concert with Dwayne Fulton?	15-Mar-23	Sat., March 15.
When is the Kara Walker exhibition opening?	1-Mar-25	1-Mar

Table 2: Examples of RAG-generated answers for event-based questions. While the RAG model often retrieves the correct date, inconsistencies in date formatting (e.g., year inclusion, weekday naming, or style) can lead to mismatches against the ground truth.

Question	Closed-book Qwen	RAG - Qwen	RAG - T5	Ground Truth
When was Carnegie Mellon University founded?	1990	1990	1990	1990
How many startups linked to CMU have raised over $7 billion?	Over 20 startups.	Over 400 startups.	400	Over 400
When is the Kara Walker exhibition opening?	April 15, 2023.	1-Mar-25	March 9, 2025	1-Mar

Table 3: Comparison of answers generated by closed-book Qwen, RAG with Qwen, and RAG with T5 across different question types. While all models correctly answer general knowledge questions, the RAG-based systems—especially with Qwen—consistently outperform the closed-book baseline on domain-specific and local event queries.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *Preprint*, arXiv:1606.05250.

Chroma Team. 2023. Chroma: the ai-native open-source embedding database. https://github.com/chroma-core/chroma.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.