Improving Retrieval-Augmented Generation Systems

Enes Özeren, Julia Broden, Daniel Gloukhman, Ercong Nie

August 16, 2024

Abstract

How can we make LLMs trustworthy in providing accurate information? Retrieval-Augmented Generation (RAG) has emerged as a promising solution to address the knowledge limitations of LLMs. By searching external sources to retrieve relevant information and providing grounded answers, RAG systems offer a way to improve the accuracy of responses. However, RAG still faces challenges such as selecting the most relevant information, reducing question-answering latency, and synthesizing information for complex questions, necessitating further research and development. The Meta Comprehensive RAG Challenge (CRAG) of the KDD Cup 2024 aims to provide a benchmark with clear metrics and evaluation protocols to rigorously assess RAG systems and advance solutions in this field. In the retrieval summarization task, we were provided with five web pages per question, which are likely, but not guaranteed, to be relevant. The objective was to develop a RAG system and evaluate its answer generation capabilities by measuring its ability to identify and condense relevant information into accurate answers. We have performed various experiments to explore the potential improvements of a RAG system and present our results in this report. Our source code and experiment logs are published on this Github Repository.

1 Introduction

The RAG system is fundamentally built from 2 models, embedding model and chat model [1]. For this project we have used the Meta CRAG template. And also to test the RAG system we have used the CRAG dataset. The CRAG dataset includes question-answer pairs that reflect real-world scenarios across five domains: Finance, Sports, Music, Movies, and Encyclopedia Open domain. These domains represent varying rates of information change—rapid (Finance and Sports), gradual (Music and Movies), and stable (Open domain). It is crucial to efficiently retrieve relevant documents from the data source. Several key issues are involved, including improving pre-processing through chunking strategies and selecting the embedding model. After retrieval, it is not ideal to directly input all the retrieved information to the LLM for answering questions. Similar to humans, LLMs tend to focus on the beginning and end of long texts, often neglecting the middle portions. Therefore, we further process the retrieved content with re-ranking, which reorders document chunks to highlight the most relevant results first. Additionally, we performed hyper-parameter tuning to optimize performance.

2 Evaluation of the RAG System

2.1 Evaluation Metrics and Models

One of the biggest challenges of developing a RAG system is evaluating how good the performance of the system is. First of all, there is a need for a good dataset which consists of information sources, questions and ground truth answers. In our project we have used the dataset from Meta CRAG Challenge. For the retrieval summarization task in the challenge, there are 2,706 information source, question, answer pairs. To use our resources efficiently and have a reasonable evaluation duration, we have sampled 333 information source, question, answer pairs and used it for our experiments. The next step for evaluating the RAG system is comparing the ground truth answers and RAG System generated answers. This step can also be non-trivial since LLM generated answers do not necessarily match exactly with the ground truth answers even though they are accurate. After generating the answer with our RAG System, our evaluation algorithm follows these steps:

1. First, we check the exact string match between the ground truth answer and the RAG system generated answer (exact accuracy).

2. Next, we have utilized open-source large language models. We have given both the ground truth answer and the RAG system generated answer to another LLM with a prompt which gives instructions to evaluate the answer as an accurate or not accurate answer (See Appendix 1. for the prompt) (accuracy).

3. If the answer is not accurate then the RAG system hallucinated (hallucination).

4. If the RAG system answer is "I don't know." then it is a miss (missing).

After evaluating each RAG system generated answer, we calculate a general score.

$$\text{Score} = \frac{2 \times (\text{exact accurate answers} + \text{accurate answers}) + \text{miss answers}}{n} - 1$$

where n is the number of questions.

For this evaluation approach we have experimented with Llama 3 8B Instruct and Llama 3 70B Instruct models and we have compared the results with the Open AI GPT4 api results which we obtain from submission to the Meta CRAG challenge.

After comparing the 2 Llama models, we have seen that Llama 3 70B Instruct model evaluation for the same RAG system is closer to the GPT4 model evaluations (Table 1). When we checked some samples, we saw that the Llama 3 8B Instruct model sometimes labels the RAG system generated answer as accurate even though it is clearly not accurate. Therefore, we have used the Llama 3 70B Instruct model for evaluation for the rest of our experiments.

	CRAG Submission	Our Evaluation	
EVAL MODEL	**GPT-4**	**Llama 3-8B-Instruct**	**Llama 3-70B-Instruct**
total samples	336	333 (Random Sample)	333 (Random Sample)
score	-0.08	0.12	0.04
exact_accuracy	2.7%	3.0%	3.0%
accuracy	23.4%	33.6%	29.4%
hallucination	30.9%	21.3%	25.5%
missing	45.6%	45.0%	45.0%

Table 1: Evaluation Model Experiments. For embedding model, all-MiniLM-L6-v2 is used.

2.2 Logging

To keep track of our experiments we have developed a logging structure for evaluation. When the evaluation script executed, all the dataset, hyperparameters of the RAG system, calculated metrics and the duration of the inference for each part of the system logged into a txt file.

3 Optimization Methods in RAG systems - Experiments

3.1 Embedding Model

Retrieval is achieved by calculating the similarity, such as cosine similarity, between the embeddings of the question and document chunks. The semantic representation capability of embedding models plays a crucial role in this process. Our starting point for potential embedding models for our RAG system was Hugging Face's MTEB leaderboard[1], that evaluates embedding models across 8 tasks, covering 58 datasets. We tested six embedding models with different number of parameters, i.e. all-MiniLM-L6-v2, snowflake-arctic-embed-s, stella-base-en-v2, bge-large-en-v1.5, gte-large-en-v1.5, and gtr-t5-xl. The performance metrics of these models are detailed in Table 2. The gtr-t5-xl model was the largest tested model with 1.24 billion parameters. It achieved the best performance for accuracy among the embedding models tested, with an accuracy of 32.1% and an exact accuracy of 4.2%.

[1] https://huggingface.co/spaces/mteb/leaderboard

EMBEDDING MODEL	score	exact accuracy	accuracy	hallucination	missing
all-MiniLM-L6-v2 (23M)	0.045	3.0%	29.7%	25.2%	45.0%
snowflake-arctic-embed-s (33M)	0.012	3.0%	24.0%	22.8%	53.2%
stella-base-en-v2 (55M)	0.057	3.6%	30.0%	24.3%	45.6%
bge-large-en-v1.5 (335M)	0.054	3.6%	30.3%	24.9%	44.7%
gte-large-en-v1.5 (434M)	0.033	3.6%	27.9%	24.6%	47.4%
gtr-t5-xl (1.24B)	0.045	**4.2%**	**32.1%**	27.6%	40.2%

Table 2: Performance metrics of different embedding models. With all embedding models Llama 3-8B-Instruct is used as chat model

3.2 Chat Model

The other important part of a RAG system is the chat model which takes the retrieved documents and generates answers to user queries. In Meta CRAG Challenge every model other than Meta Llama series was limited with 1.5B parameters, therefore we have experimented with Llama 3 8B Instruct and Llama 3 70B Instruct models. For both models we have used half precision for model weights for efficient use of memory and low latency during inference. We have seen an significant improvement when we compare the 2 models. The larger Llama model uses the retrieved information better and follows our instructions therefore, has a higher exact accuracy and lower hallucination rates. (Table 3).

CHAT MODEL	Llama 3 - 8B - Instruct	Llama 3 - 70B - Instruct
total samples	333 (Random Sample)	333 (Random Sample)
score	0.04	0.07
exact_accuracy	3.0%	6.6%
accuracy	29.4%	30.0%
hallucination	25.5%	23.4%
missing	45.0%	46.5%

Table 3: Chat Model Experiments. For embedding model all-MiniLM-L6-v2 is used

3.3 Hyperparameters

For hyperparameter tuning of the RAG system, we evaluated the performance across various values for the number of relevant context sentences, the maximum length (in characters) of each context sentence, and the maximum total length (in characters) of all context references combined. We have also experimented with the inference hyperparameter of the chat model as well. We have tried different temperature values and beam search widths. For the temperature, the lower it is the better for our RAG system performance. When we experimented with different temperature values between 0 and 1, with higher temperatures the hallucination increases and accuracy decreases. We have seen the best overall performance with temperature 0.1.

For beam search width we had some interesting results, having no beam search was the best performing set up. These results suggest that the high-probability sequences are not necessarily the best answers in our RAG system.

3.4 Chunking

Chunking means splitting the source data into smaller "chunks" to increase the granularity of the retrievable information and reduce irrelevant information. Splitting a text into too small parts can corrupt some of the information though, while keeping the text too long for a chunk might cause inefficiencies in processing and retrieval, as well as potential difficulties in extracting relevant and precise answers. Therefore, a nice balance between them should be found for good performance of the system. In addition to that, since the given information is in the html format in our

CRAG dataset, processing the html data was also important. Hence, we are trying to improve our chunking algorithm have focused on 2 main aspects:

1. Processing the html data
   (a) Removing boilerplate
   (b) Not removing boilerplate
2. Chunking unit
   (a) Sentences
   (b) Paragraphs

In the html data from given websites for retrieval, there are a lot of boilerplate contents, such as navigation links, headers, and footers. Since those boilerplate contents also contain some valuable information about the user query, we have tried both removing and not removing the boilerplate contents. For removing the boilerplate content we have used the "justext" python package. The second aspect was about the chunking unit, we have developed 2 chunking strategies, one with sentence based and the other is paragraph based. The chunking algorithms that do not remove boilerplate are called V1 and the other ones are called V2. When we compared the chunking algorithms we have seen that the chunking algorithm which has paragraph units and removes boilerplate (Paragraph V2) has the best performance with lower number of context chunks. Decreasing the number of context chunks gives worse performance for sentence based chunking algorithms (Table 4).

	Paragraph V2	Sentence V1	Sentence V2	Paragraph V1	Paragraph V2
NUM CONTEXT CHUNKS	15	20	20	20	20
total samples	333	333	333	333	333
score	**0.099**	0.096	0.072	0.057	0.096
exact_accuracy	**8.7%**	6.9%	6.9%	5.4%	8.1%
accuracy	32.7%	**36.3%**	31.5%	34.5%	32.7%
hallucination	**22.8%**	26.7%	24.3%	28.8%	23.1%
missing	**44.4%**	36.9%	44.1%	36.6%	44.1%
Response time per query (second)	**5.7**	8.0	6.1	8.7	6.3

Table 4: Chunking Experiments. Response times are measured while running the RAG System on 4 Nvidia A100 GPUs

3.5 Reranking

As explained above our RAG system uses the cosine similarity between the embeddings of the prompt and the available text chunks. The retrieval sorts chunks according to the angle in the embedding space and deems smaller angles as more important. However the chatmodel used for generating the response may not use this order correctly. In fact, empirical observations shows that LLMs may put more emphasis on text chunks in the beginning and the end of the prompt. Thus changing the order of retrieved chunks might improve the generated response. We investigated changing the order by using a pre-trained transformer based cross-encoder to calculate different similarity scores. We opted to use the popular BAAI/bge-reranker-v2-m3 model. Using a cross-encoder to calculate similarity scores makes the cosine similarity scores obsolete. However using a transformer based model is computationally more expensive than using the cosine similarity. To reduce computational cost we introduce a two-step process.

1. Calculate cosine-similarities
2. Calculate the crossencoder based similarities only on the top k chunks from step 1

reranking	no reranking	just reordering			
TOP K BEFORE RERANKING	-	15	100	200	500
TOP K	15	15	15	15	15
score	0.10	0.93	0.11	0.11	0.11
exact_accuracy	9,6 %	9,6%	10,2%	9,9 %	9,9 %
accuracy	32,4 %	32,4%	33,6 %	33.3%	33.3%
hallucination	21,6 %	23,1%	23,1%	22.8%	22.8%
missing	45,9 %	44%	43,2%	43.8%	43.8%

Table 5: Comparison of reranking hyper parameters

Our experiments show that reranking improves our metrics. But for further experiments we do not consider using TOP K BEFORE RERANKING with 100, 200, 500 chunks since it increases the latency of our rag system and makes it infeasible for users. With some further studies this re-ranking can be more efficient and used in the RAG system to improve it.

3.6 Prompts for Chat Model

We have also discovered that the instruction prompt to the chat model can also affect our RAG performance. Therefore we have added some more instructions to the base instructions that were given by Meta CRAG Challenge template. We have instructed the chat model to use the most recent information for the question and to answer 'invalid question' to the questions that have the false premise. With this small update we have seen improvements for the exact accuracy and the hallucination of the model (Table 6).

CHUNKING	Old Prompt	New Prompt
total samples	333	333
score	0.10	0.11
exact_accuracy	8.7%	9.6%
accuracy	32.7%	32.4%
hallucination	22.8%	21.6%
missing	44.4%	45.9%

Table 6: Comparison of Old and New Prompts

4 Results

We have conducted a large amount of experiments to improve our RAG system. Through these extensive experiments, we have gained valuable insights that have significantly contributed to improving the system's ability to provide accurate information with minimal hallucination. With the insights from our experiments we could improved all metrics of our RAG system (Table 7). These gains in accuracy and reduction of hallucinations are key steps toward more trustworthy AI systems. Below, we summarize the best configuration for our RAG system, detailing the key adjustments and optimizations that have led to these improvements.

Best Configurations:

- **Chat & Evaluation Model:** Meta-Llama-3-70B-Instruct
- **Embedding Model:** gtr-t5-xl
- **Hyperparameters**
  - Number of Context Chunks = 15
  - Max Context Chunk Length (in characters) = 2,048
  - Max Merged Retrieval Length (in characters) = 32,000
  - Temperature for Chat Model: 0.1
  - No Beam Search for Chat Model Inference
- **Chunking Algorithm:** Paragraph based and HTML boilerplate removed

	Starting RAG System	Improved RAG System
total samples	333	333
score	0.045	0.108
exact_accuracy	3.0%	9.6%
accuracy	29.7%	32.4%
hallucination	25.2%	21.6%
missing	45.0%	45.9%

Table 7: Comparison of Starting and Improved RAG Systems

5 Future Work

Future work should explore several points to enhance the performance and reliability of Retrieval-Augmented Generation (RAG) systems. A key area is the development of multi-modal RAG systems that integrate text, images, and audio to provide contextually richer answers. This could be particularly beneficial in domains where visual or auditory information is critical.

Additionally, using multiple LLMs (agents) in a collaborative way to evaluate and refine responses before delivering them to users could significantly improve accuracy and reduce hallucinations. This iterative feedback mechanism could leverage the strengths of different models and mitigate individual model weaknesses.

Data security and privacy present another critical challenge. Future research should focus on robust mechanisms to prevent LLMs from revealing sensitive data.

Improving the efficiency of RAG systems, particularly in terms of reducing latency without compromising accuracy, is another essential aspect. Investigating advanced indexing techniques and more efficient search algorithms can contribute to this goal.

Lastly, further research should aim to develop more sophisticated evaluation metrics that better capture the nuances of LLM-generated responses, ensuring a more accurate assessment of system performance.

6 Conclusion

In this study, we have presented extensive experiments and optimizations to improve the performance of a Retrieval-Augmented Generation (RAG) system. By systematically evaluating various components such as embedding models, chat models, hyperparameters, chunking strategies, and re-ranking techniques, we have made significant strides in enhancing the system's accuracy and reducing hallucinations. Our best configuration, which includes the Meta-Llama-3-70B-Instruct chat model, the gtr-t5-xl embedding model, and a refined chunking algorithm, demonstrates notable improvements across all key metrics.

These advancements are crucial steps towards building more trustworthy and reliable AI systems capable of providing accurate and contextually relevant information. The insights gained from our experiments highlight the importance of continuous optimization and evaluation in developing robust RAG systems.

With this project we aim to contribute to the development of AI systems that can seamlessly integrate and synthesize diverse information sources, ultimately enhancing the utility and trustworthiness of LLMs in real-world applications.

References

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

A Appendix

A.1 Evaluation Prompt

```
INSTRUCTIONS = """
# Task:
```

```
You are given a Question, a model Prediction, and a list of Ground Truth
    answers, judge whether the model Prediction matches any answer from the
     list of Ground Truth answers. Follow the instructions step by step to
    make a judgement.
1. If the model prediction matches any provided answers from the Ground
    Truth Answer list, "Accuracy" should be "True"; otherwise, "Accuracy"
    should be "False".
2. If the model prediction says that it couldn't answer the question or it
    doesn't have enough information, "Accuracy" should always be "False".
3. If the Ground Truth is "invalid question", "Accuracy" is "True" only if
    the model prediction is exactly "invalid question".
# Output:
Respond with only a single JSON string with an "Accuracy" field which is "
    True" or "False".
"""
```

```
IN_CONTEXT_EXAMPLES = """
# Examples:
Question: how many seconds is 3 minutes 15 seconds?
Ground truth: ["195 seconds"]
Prediction: 3 minutes 15 seconds is 195 seconds.
Accuracy: True

Question: Who authored The Taming of the Shrew (published in 2002)?
Ground truth: ["William Shakespeare", "Roma Gill"]
Prediction: The author to The Taming of the Shrew is Roma Shakespeare.
Accuracy: False

Question: Who played Sheldon in Big Bang Theory?
Ground truth: ["Jim Parsons", "Iain Armitage"]
Prediction: I am sorry I don't know.
Accuracy: False
"""
```

A.2 Chat Model Instruction Prompts

Old Prompt:

```
You are provided with a question and various references.
Your task is to answer the question succinctly, using the fewest words
    possible.
If the references do not contain the necessary information to answer the
    question,
respond with 'I don't know'.
There is no need to explain the reasoning behind your answers.
```

New Prompt:

```
You are provided with a question and various references.
Your task is to answer the question succinctly, using the fewest words
    possible.
The time of the question is given before the question as Current Time,
Prioritize the most recent information in the references with respect to
    Current Time.
If the references do not contain the necessary information to answer the
    question,
respond with 'I don't know'.
All False Premise questions should be answered with a standard response '
    invalid question'.
There is no need to explain the reasoning behind your answers.
```