Report for Q&A RAG System

Vinh Bui

01/08/2024

Abstract

While Mistral offers a more affordable solution, Cohere delivers commercial-grade quality. Across all selected metrics, Cohere outperforms Mistral, particularly in the "LLM as judge" metric, where ChatGPT 3.5 consistently favors Cohere's responses.

Introduction

Large language models (LLMs) are a leading AI technology, revolutionizing tasks such as summarization, translation, and question-answering. However, retraining these models is costly and inefficient for updating them with more recent information. Therefore, I developed a Retrieval-Augmented Generation (RAG) system tailored for question-answering using public data from various sources. This system dynamically adjusts language and tone based on the audience by leveraging langchain.

Key Findings

- Effective prompting is crucial for generating high-quality responses from large language models (LLMs).
- Commercial grade LLM, such as Cohere, outperforms open-source LLM Mistral. However, please note that I use quantized Mistral, not the full accuracy one.
- Embedding transformers plays a crucial part in the RAG system. Changing the embedding models can significantly impact the performance because it affects the information that we send to LLM.
- Selecting appropriate chunk size improves efficiency of the system
- For evaluation, I recommend moving beyond N-gram-based approaches like BLEU and ROUGE. Instead, we should use advanced LLMs to assess our RAG system responses, vector space similarity, faithfulness, and relevance provides more meaningful metrics.

Methodology

Technical Approach

Retrieving relevant data is important to help LLM provide meaningful responses that are crucial for a RAG system. Therefore, I decided to test two important aspects that impact this process: embedding model, and chunk size. The embedding models will affect the returned documents to feed in the LLM. Chunk size will affect how much information we give to it.
There are many SBERT transformers for embedding models. However, I decided to test with the three: all-mpnet-base-v2, multi-qa-mpnet-base-dot-v1, and all-distilroberta-v1. The all-mpnet-base-v2 model is an excellent starting point for experimentation due to its training on

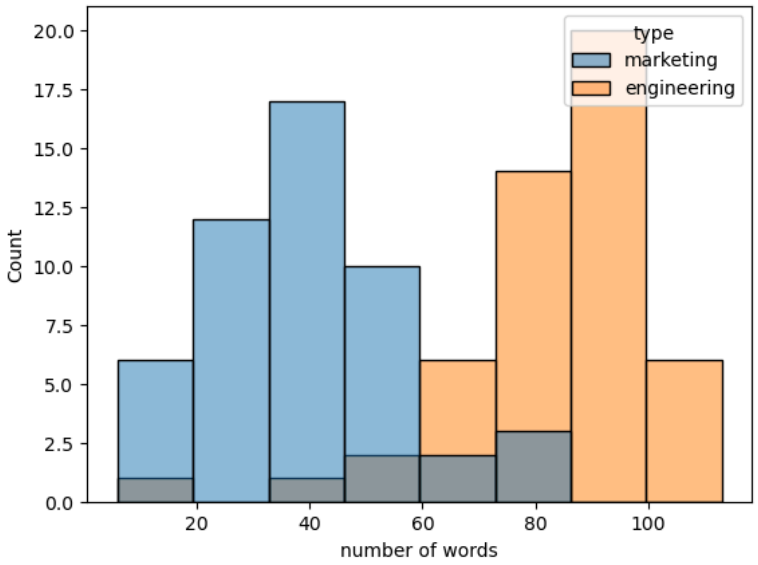
diverse data. This aligns well with our varied datasets, including blog posts, Wikipedia entries, and arXiv papers. Meanwhile, the multi-qa-mpnet-base-dot-v1 model is good for our Question and Answering (Q&A) system. Both models offer unique strengths that can be leveraged to enhance our AI solution effectively. I only tested with all-distilroberta-v1 once because it is slower than the other two.

Additionally, our system needs to accommodate two different target audiences: marketing team and engineering team. To accommodate this requirement, I use langchain, an open source project, to compile a prompt template that has three adjustable variables: audience, context, and question. These variables provide context for the LLM to generate outputs.

- Audience: specifies the type of audience (marketing team or engineering team) and their expectations. We can know the user's persona easily when they login to the system.
- Context: provides relevant information retrieved by a vector database based on the questions.
- Question: represents the input question from users.

Context is the most important aspect for LLMs. With enough context, LLM can provide good answers.

When I investigated the golden dataset, I also noticed that answers for the engineering team are often longer than the marketing team.



As shown in the chart above, the two types of answers belong to distinct distributions with different means. While I have the option to adjust the desired number of words and sentences based on audience type, I believe it's not ideal to do so. If we limit the outputs from the LLM too much, such as suggesting a range of words depending on the user's persona, we will not get the full picture of users' preferences. Instead of prematurely optimizing the prompt, we should get users feedback from the responses and change accordingly.

Testing and Evaluation

I am going to evaluate the responses from Mistral and Cohere on different types of metrics: vector space and using another high performance model to evaluate the results as a judge.

- Vector Space: by transforming the golden answers and their corresponding AI generated answers into a vector space, we can compare their cosine similarity to evaluate how close they are to each other.
- "LLM as a judge": I evaluate the responses based on their faithfulness (measure how much of the contexts that the LLM used to generate response), relevant answers (measures how relevant the answers is to the question), and my own custom metric which is asking chatGPT to choose which generated answers it thinks is more similar to the golden answer.

I have tested the system with different kinds of embedding transformers: all-mpnet-base-v2 and multi-qa-mpnet-base-dot-v1. I also tested with different chunk size configurations.

Results and Findings

Proof of Concept Functionality

Cosine similarity: When evaluating the answers using cosine similarity between golden answers and AI generated answers with chunk size 256 and overlap 128, "all-mpnet-base-v2" performs slightly better. In any case, Cohere is better than Mistral.

Embedding Models	Marketing		Engineering	
	Mistral	Cohere	Mistal	Cohere
all-mpnet-base-v2	0.7589	**0.7862**	0.7832	**0.8037**
multi-qa-mpnet-base-dot-v1	0.7535	**0.7557**	0.7602	**0.7819**

LLM as a judge: For chatGPT's preference, I developed a prompt to instruct chatGPT3.5 to select what it thinks that is more similar to the golden answer that I provided. For faithfulness and answer relevancy, I use RAGAS to evaluate, which uses OpenAI under the hood to calculate those scores.

Metrics	Marketing		Engineering	
	Mistral	Cohere	Mistral	Cohere
Preference	5	**70**	4	**71**

Faithfulness	**0.7063**	0.6480	**0.7137**	0.6487
Relevancy	0.8559	**0.8941**	0.8542	**0.9027**

In faithfulness, Mistral beats Cohere. However, in the preference and relevance metrics, cohere outperforms marketing. It is interesting to see how chatGPT3.5-turbo refers Cohere's answers.

Lessons Learned

- Prompt engineering is crucial to produce quality outputs. More importantly, I always back test a lot when I decide to make changes in the prompts.
- Cost adds up real fast, especially for services such as OpenAI. We need to be strategic about how to evaluate the system more efficiently.
- Even though open-source models may not perform as well as commercial ones, they offer significant advantages like keeping data securely within the company, making them an excellent alternative for organizations handling highly sensitive information.

Challenges and Limitation

Due to resource constraint, I can only use a quantized Mistral model that has far less parameters than cohere. Therefore, it is not surprising that cohere is outperforming Mistral in most of the cases.
Another challenge is that the cohere server refuses to respond to the question that I send if I am sending too fast, especially on the production key. Additionally, RAGAS sometimes randomly throws errors when it needs to evaluate answers of all 75 questions

Next Steps

If there are more resources provided, I could experiment on the bigger Mistral model, which has more parameters and not quantized. Additionally, I could instruct LLM to provide an explanation on how it uses the contexts that we provide to generate answers, so it could improve the performance.

Summary and Recommendations

Given a good scores of faithfulness, relevancy, and highly preferred cohere, I recommend building the RAG system using public dataset and commercial grade LLM. Then, we can collect feedback from the employees on how satisfied they are. Additionally, managers should evaluate how much more productive employees get when they use the system. This evaluation should give us some data to decide whether we should invest more resources to develop a RAG on more sensitive data.
Regarding the capacity of the system, assuming 30% of the employees query the system at the same time, we need the system able to handle approximately 100 requests at the same time. Therefore, I recommend we should start with pay-per-use then change to per deployment as needed.

References

ChatGPT 3.5 is used to improve my original writing in the abstract and introduction more concisely. However, I did make changes, so it is not exactly like what chatgpt wrote
Pretrained Models — Sentence Transformers documentation (sbert.net)
BLEU - a Hugging Face Space by evaluate-metric
ROUGE - a Hugging Face Space by evaluate-metric
Evaluating RAG Applications with RAGAs | by Leonie Monigatti | Towards Data Science