Generative AI Assistant for Generative AI Topics

Name: Brian Lin
Date: 8/1/2024

Abstract

In this project, we built a QA system to accelerate Gen AI education within the company. The results from our POC are promising. The system is able to provide helpful, accurate answers to the user, despite currently falling short compared to the gold standard answers. With additional refinement of the retrieval process and prompts, the product will provide substantial value to the company.

1. Introduction

The goal of this project was to develop a Question Answering system tailored specifically for our tech company specializing in Gen AI products. There are two user groups that this system is catered to: the engineering team and the marketing team. To achieve this functionality, two distinct RAG pipelines were created. One pipeline is optimized for engineers, providing detailed, technically nuanced responses, while the other caters to the marketing team, offering more generalized, high-level answers.

The foundation of this QA system is built upon the repository of knowledge from pre-trained Language Models, and further leverages additional documents sourced from ARXIV research papers, Wikipedia, and various blog posts. The additional documents provided to the system gives the LLM with information that it may not have been trained on, which is particularly useful in answering questions around bleeding edge advancements. The system is able to answer questions across fundamental Gen AI concepts and techniques, specific models within the Gen AI domain, and current industry standards. In order to have the system generate accurate and appropriate responses for the two different audience groups, we experimented with varying prompts, LLMs, hyperparameters, and chunking techniques, before creating two final pipelines for the system.

2. Methodology

2.1. Retrieval Strategy

The first step in building the system was coming up with the retrieval strategy. More specially, I experimented with chunking strategies and the embeddings used in the vector store. My experimentation included the following:

1. Custom Chunkers for Document Types: Different document types have varying ways of formatting and separating text. We can create better chunks by accounting for this using custom chunkers.
2. Chunk Size and Overlap: I found that small chunk sizes led to the lack of information and coherence, while large chunks led to the system struggling in retrieving finer details in the chunk. Adjustments were made to chunk sizes and overlaps to balance these tradeoffs.
3. Combination of Semantic and Character-Based Chunking: Only using semantic based chunking led to large chunks that were not performant. This led to the using a semantic chunker first, before reducing the chunk sizes through using a character based chunker.

4. Embedding Model Exploration: Various embedding models were tested for their suitability in the retrieval process. The GIST embedding model was able to capture and return specific phrases and concepts more effectively.

The final parameters of the retrieval process were as follows: use of GIST embedding model, semantic chunking and then the LatexTextSplitter with a chunk size of 650 and overlap of 50 was used for Arxiv papers, the MarkdownTextSplitter with a chunk size of 650 and overlap of 50 for blog posts, and the RecursiveTextSplitter with WikiText specific separators and a chunk size of 650 and overlap of 50 for wikipedia documents.

2.2. Prompt Engineering

The main strategy I used for prompt engineering was to create two sets of prompt components, one specific to engineers and one specific to marketing. Then I tested the different permutations of the components, and selected the final prompts that had the highest weighted evaluation score.

My design philosophy for what to include in the prompt focused on a couple of main aspects. First, I tried to emphasize the audience part and tested different ways to describe the purpose behind why the user was asking the question. Second, I tried to include different output characteristics, emphasizing the word choice of the response being high level for marketing vs technical for engineering. For output characteristics, I also included details on the desired output length. Engineering had slightly longer responses than marketing in the validation set, so I wanted the generated responses to reflect this.

2.3. Model Selection and Hyperparameter Tuning

For model selection and hyperparameter tuning, I employed a similar strategy as for prompt engineering. I created a set of varying values for temperature, top p, and repetition penalty, before I tested the different permutations of the hyperparameters with the Mistral and Cohere model. The hyperparameter values chosen lean toward having a more deterministic output. Our use case does not need the model to have creative outputs, rather it should give factual information that it is certain about. This meant experimenting with lower temperatures and higher top p values. We also want the answer to be concise, so we experimented with higher repetition penalties.

2.4. Testing and Evaluation

For my main metric, I chose to use a weighted evaluation score, consisting of two parts: similarity to the reference answer and general RAG performance. To evaluate the similarity to the evaluation metric, I chose to use the BERTScore and cosine similarity between generated and reference answers. I decided against including other traditional NLP metrics like BLEU and ROUGE due to their limitation from word matching. The closeness of generated answers to reference material is important, thus these metrics were weighted at 65%. Additionally, the Ragas framework was utilized to evaluate general RAG performance, focusing on answer relevance, faithfulness, and context relevance, and was weighted at 35%.

The final equation for computing the weighted evaluation score is as follows:

0.65 * ((BERT Score + Cosine Similarity) / 2) + 0.35 * ((Faithfulness + Answer Relevance + Context Relevance) / 3)

For the two pipelines, I ran three key runs each. The first was to find the prompt that resulted in the highest weighted evaluation score. The second run was using the Mistral model with the best prompt and varying hyperparameter sets, and the third run was using the Cohere model with the best prompt and varying hyperparameter sets. Lastly, I compared the results from the second and third run to determine my final system architecture. Ideally, we would test each of the prompts with the different models and sets of hyperparameters, however, we are limited by computation costs and time.

2.4.1. Finding "Best" Prompt

This first run evaluated 36 engineering prompt permutations, and 12 marketing prompt permutations against the validation subset of 10 questions. The main goal is to find the prompts that yield the highest weighted eval score. For this run, we used the Mistral model with the default hyperparameters. It is important to note that this is a limitation. Different models react differently to prompts, so we can not be sure that the best prompt we found here would be the best for the Cohere model. However due to our constraints, we will first find the prompt that performs best on the default Mistral model, and then test that prompt with different sets of hyperparameters. The final prompt used for each pipeline is shown in the Appendix.

2.4.2. Model Selection and Hyperparameters

After finding the best prompt, we performed the second run which tested 12 different hyperparameter sets with the Mistral model. For temperature, we tested with a value of 0.1, 0.3, and 0.6. For top p, we tested with a value of 0.9 and 0.95. For repetition penalty, we tested with a value of 1.2 and 1.5. In our third key run, we tested the same hyperparameter values but using the Cohere model instead.

For the engineering pipeline, the model that performed the best in terms of the weighted evaluation score was the Cohere model with a temperature of 0.1, top p of 0.95, and repetition penalty of 1.5. For the marketing pipeline, the model that performed the best in terms of the weighted evaluation score was the Cohere model with a temperature of 0.1, top p of 0.95, and repetition penalty of 1.2.

3. Results and Findings

Below are the results of running the final pipelines on the set of 75 validation questions:

	Weighted Score	BERT Score	Cosine Sim	Answer Relevance	Context Relevance	Faithfulness
Engineering Pipeline	0.78184	0.88355	0.76097	0.71799	0.50517	0.88545
Marketing Pipeline	0.78240	0.89462	0.76106	0.72145	0.53343	0.84056

The evaluation of our RAG system reveals mixed results. The engineering and marketing pipelines demonstrate comparable performances when looking at the weighted score, suggesting the system performs adequately overall. Notably, the BERT score and faithfulness metrics

exhibit high values, suggesting that the generation component of the RAG system is working well in producing accurate and faithful responses. However, the system struggles with context relevance, indicating that the retrieval portion is likely the system's weak link. Furthermore, while the answer relevance and cosine similarity metrics are satisfactory, they appear to be likely impacted by the LLM's reliance on poorly retrieved contexts. These observations indicate the need for refining the retrieval process to enhance the system's overall performance.

When looking at a subset of the generated answers, it appears that the system is able to provide helpful, accurate answers to the user. However, it often falls short in providing the same level of details as the reference answer, for both the engineering and marketing pipeline. Additionally, there are several cases where the generated engineering and marketing answers are nearly identical. This indicates a need for further refinement on prompting.

3.1. Challenges and Lessons Learned

One lesson learned through this project was that getting good chunks is difficult. I spent a good chunk of my time on this aspect, but the performance was still poor. Also, I learned that getting the LLM to exhibit specific behavior is tricky. I experimented with several prompts, attempting to get the right level of detail and stop the LLM saying things like "based on the context", but had little success. I was also surprised that my longer prompts did not result in better performance.

3.2. Limitations

The main limitations stemmed from cost and time. As mentioned in the testing section, we can not confidently say that the prompt and hyperparameters chosen were the optimal. The validation subset was small and we did not test each prompt with the different models. Evaluating using Ragas was expensive and I did not want to continue paying out of pocket.

3.3. Next Steps

For next steps, I would recommend refining the retrieval process and prompts. Implementing a sentence aware chunker and experimenting with vector store parameters would likely pay great dividends, as this is the aspect our system is struggling with the most. Additionally, I would also experiment with incorporating an "audience score" into the evaluation metrics, possibly through having an LLM evaluate the response.

4. Summary & Recommendations

The results of the POC are promising. The system is able to provide helpful, accurate answers to the user, despite currently falling short compared to the gold standard answers. With additional refinement of the retrieval process and prompts, the product will be able to add substantial value to the company. The system will essentially become the go to expert on Gen AI topics, and while extremely valuable, it also comes with risks. I would strongly caution against users blindly relaying the LLM response to our clients, or implementing a product based on the response without further research. After refining the RAG system, I would suggest a pay per use deployment, as there are a low number of potential users (350 in the company), but a formal comparison should be done after the architecture is finalized.

5. Appendix

Final Engineering Prompt:
[INST] You are a question-answering assistant for Generative AI topics. Your task is to provide factual, concise, and informative answers to Generative AI related questions. Use information in the provided context to answer the question. You will be answering questions for a team of engineers working at a startup creating new Generative AI products. The engineers are new to the Generative AI field. The engineers are asking questions in order to gain the technical knowledge necessary to build new Generative AI products. Your response should include clarity, technical depth, and precision. Do not use information that is not in the provided context to answer the question. Do not use more than 7 sentences in your answer. Do not say 'Based on the context'. You are very capable. Answer the question based on the following context: {context} Question: {question} [/INST]

Final Marketing Prompt:
[INST] You are a question-answering assistant for Generative AI topics. Your task is to provide factual, concise, and informative answers to Generative AI related questions. Use information in the provided context to answer the question. You will be answering questions for the marketing team working at a startup creating new Generative AI products. The marketing team is new to the Generative AI field. The marketing team is asking questions in order to gain the high level knowledge necessary to promote their new Generative AI products. Your response should be high level, concise, and nontechnical. Do not use information that is not in the provided context to answer the question. Do not use more than 5 sentences in your answer. Do not say 'Based on the context'. You are very capable. Answer the question based on the following context: {context} Question: {question} [/INST]