W267- Assignment 5 Report

Catherine Liao

Executive Summary

A retrieval-augmented generation (RAG) system was developed using the LangChain framework as a proof of concept project to enhance search and question-answering capabilities. The system integrated an embedding model, a vector store, prompt templates, and a large language model (LLM) to generate responses. Each component was iteratively selected and fine-tuned based on experimental results to optimize overall system performance. Evaluation metrics were carefully chosen to capture both retrieval effectiveness and answer generation quality, using a golden reference question-answer dataset for comparison. Results indicated strong semantic alignment between generated and reference answers, though further improvements in retrieval, particularly around chunk selection, are needed. Techniques such as re-ranking, varying the vector store search methods, and tuning chunk size and overlap were tested, with each contributing to retrieval quality to varying degrees. Prompt engineering was implemented with success in increasing answer generation quality though further improvement is still recommended.

Introduction

The company is exploring new approaches to improve internal search and question-answering capabilities to support its 300 engineers and 40-person marketing team. These improvements aim to accelerate engineering workflows and content creation. To evaluate the potential of RAG systems in this context, a mini proof of concept was conducted with a goal to assess whether a RAG-based approach could improve document retrieval and question answering across technical and non-technical domains.

The RAG system is composed of two primary components: the retriever and the generator. The retriever uses an embedding model and a vector store to search and retrieve relevant content, while the generator employs a large language model to synthesize answers based on the retrieved chunks. The system architecture allows for the evaluation of different configurations, including the tuning of hyperparameters for the embedding model, chunk sizes, overlap settings, and vector store parameters. Various document types, with data sources from web articles, academic papers, and Wikipedia entries, were used in the proof of concept to assess the system's ability to handle both engineering and marketing-related queries. Numerous experimental iterations were run to test different configurations of the system. These included varying the embedding models, LLMs, chunk sizes, overlap sizes, vector store parameters, and additional additions such as re-rankers. The experiments aimed to identify the optimal combination of components and hyperparameters to deliver the most effective performance. Evaluation metrics were carefully chosen to reflect different aspects of the RAG system. A golden reference dataset containing 78 high-quality responses from the engineering and marketing teams was also used to assist evaluation of system performance.

Key Findings

1. **Strong Semantic Alignment:** Generated answers showed high BERTScore and semantic textual similarity, indicating good overall meaning alignment.
2. **Some Reranking Improves Performance:** Implementing reranking of retrieved chunks using the Cohere re-ranker led to noticeable improvements in generated answer quality. This was reflected in higher factual correctness and semantic textual similarity scores.
3. **Answer Generation Needs Further Improvement:** Prompt engineering improved answer generation, though challenges in getting more overlap of statements between generated answers and the reference answers still remained since factual correctness scores are still low(0.2-0.4)
4. **Retriever Performance Needs Further Improvement:** Low to moderate context recall scores (0.25–0.6) suggest that the retriever component needed improvement in returning supporting information with sufficient similarity to the reference answers.
5. **Engineering and Marketing answer performance was similar:** Overall evaluation results throughout experiments showed there was no clear indication of whether the RAG system consistently performed better on either team's answers.

Methodology (Implementation and Evaluation Approach)

The full RAG pipeline is made of two main components, retrieval and answer generation.

Retrieval: The retriever's key components include the embedding model and the vector store. Relevant hyperparameters include the chunk size, overlap size, and vector store settings such as the search type and the number of contexts returned. The data sources stored in the vector store included a fixed set of web articles, academic papers, and Wikipedia entries. The vector store used is from Qdrant. Five embedding models were evaluated using the maximum allowed chunk size per model and zero overlap. The five models seemed to show similar performance on the evaluation metrics (Appendix F) so the chosen embedding model was selected based on whether the model could produce more coherent and contextually appropriate sentence segments compared to the alternatives. Additionally, vector store search type was set to MMR to reduce duplicate chunks from being returned. The retrieved chunks from the vector store then goes through a Cohere re-ranker to get the top 5 ranked contexts.

Answer Generation: Once relevant chunks are retrieved from the vector store, they are passed to either a Mistral model or a Cohere model for answer generation. In the final pipeline the Mistral model was chosen over the Cohere model due to higher scores in faithfulness compared to the Cohere model (Appendix C). Answer generation is facilitated using the LangChain framework, which handles the formatting of the input chunks and any necessary parsing of the model's output. Crucially, LangChain also integrates the prompt templates that inject retrieved context into structured prompts for the language model. These prompt templates were customized by each team to best suit their use case.

Chosen models and hyperparameters to the final RAG system is in Appendix A

Evaluation Approach:

Experiment Sampling Size: To evaluate the pipeline's performance, the full test set of 78 questions was initially used to establish baseline metrics. To reduce evaluation time while maintaining consistent performance, it was determined that running 20 randomly selected questions from the original 78 was

sufficient to produce metric values that closely approximated those from the full run. This random sampling ensures diversity in evaluation across different runs.

Evaluation Metrics: With the pipeline in place, the next step was to select evaluation metrics that would effectively assess model performance. A reference dataset containing "best" answers for the marketing and engineering teams served as the gold standard for comparison. The chosen evaluation metrics included:

- **RAGAS Metrics** (gpt-4o-mini):
    - **Context Recall**: Measures the overlap of statements between retrieved context chunks and the reference answers. A higher score indicates that more relevant statements from the reference answers are present in the retrieved chunks.
      Faithfulness: Evaluates how well the generated answers align with the retrieved context. A high score implies that the generated response closely adheres to the provided context.
    - **Factual Correctness**: Compares the generated answers with the reference answers to assess factual alignment.
- **BERTScore**:
  Provides token-level similarity between generated and reference answers, and includes precision, recall, and F1 components to evaluate semantic similarity on a token level.
- **BLEU Score**:
  A word-overlap-based metric that measures how closely the generated answers match the reference answers in terms of exact word choice and order.
- **Semantic Textual Similarity**:
  This metric, computed using models from the sentence-transformer library, evaluates how similar the overall meaning of the generated answers is to the reference answers at the sentence level.
- **Manual Evaluation:**
  In addition to automated metrics, a manual review was conducted on three randomly selected questions: question 0, question 18, and question 67. For each, both the retrieved context and the generated answers were closely analyzed. Findings from this manual inspection informed adjustments to various hyperparameters, improving both retrieval quality and answer generation.

Results and Findings

Metrics Results: (Appendix B)

- **RAGAS Metrics**:
    - **Context Recall**: Final engineering dataset results showed a context recall of **0.3857**. Final Marketing dataset results showed a context recall of **0.5729**. Overall context recall throughout experiments ranged from **0.25 to 0.6**, indicating that while relevant chunks were being retrieved, there was still significant room for improvement in retrieving more contextually appropriate material aligned with the reference answers.
      Faithfulness: Final engineering dataset results showed a faithfulness of **0.6762**. Final marketing dataset results showed a faithfulness of **0.6684**. Overall faithfulness throughout experiments scores were between **0.5 and 0.7**, suggesting that the generated answers were generally aligned with the retrieved context, though not always a perfect match.

    - **Factual Correctness**: Final engineering dataset results showed a Factual Correctness of **0.3144**. Final marketing dataset results showed a Factual Correctness of **0.3219**. Overall Factual Correctness throughout experiments remained lower, between **0.2 and 0.4**, highlighting ongoing challenges in producing answers that closely matched the factual content of the reference responses.
- **BERTScore**: Final engineering dataset results showed a Bertscore F1 of **0.8678**. Final marketing dataset results showed a Bertscore F1 of **0.8788**. Overall Bertscore F1 throughout experiments was consistently high, ranging from **0.8 to 0.9**, indicating strong semantic similarity at the token level.
- **BLEU Score** : Final engineering dataset results showed a Bleu score of **0.2156**. Final marketing dataset results showed a Bleu score of **0.1375**. Overall Bleu score throughout experiments remained low throughout the experiments, between **0.1 and 0.23**, reinforcing the idea that exact word-level matches were less common, though also potentially less indicative of response quality.
- **Semantic Textual Similarity** : Final engineering dataset results showed a Semantic Textual Similarity of **0.7424**. Final marketing dataset results showed a Bertscore F1 of **0.7341**. Overall Bertscore F1 throughout experiments scores ranged from **0.67 to 0.78**, reflecting reasonably good alignment in overall meaning between generated and reference answers, though with potential for refinement.

Challenges:

Reducing Irrelevance and Redundancy in Retrieval: Initial challenges observed was that the retrieved chunks often contained extraneous information, such as long lists of author names, which detracted from the relevance of the content. Chunk size and overlap settings were tuned in hopes of reducing irrelevant information from being retrieved. Very small chunks led to performance degradation, while medium to large chunks performed similarly. Although reducing chunk size helped surface more targeted information, it did not fully resolve the issue, highlighting a limitation of basic chunking strategies. Another key issue was the frequent retrieval of duplicate chunks. To mitigate this, Maximal Marginal Relevance (MMR) was implemented as the vector store search strategy. Manual inspection confirmed that MMR effectively reduced redundancy, improving the diversity and relevance of the retrieved context.

Further Improving Retrieval Quality: To further improve retrieval quality, re-rankers from Cohere as well as the Sentence Transformer library were experimented with in hopes of getting better retrieved chunks. The Cohere model showed better overall metrics and was integrated to reorder retrieved chunks. This proved to be a useful change, resulting in improvements in metrics like semantic textual similarity and RAGAS factual correctness. However, the context recall metric remained largely unchanged, indicating that while the quality of selected chunks improved, the overall breadth of coverage did not shift significantly.

Improving Answer Quality Through Prompt Engineering (Appendix D, Appendix E): Prompt engineering proved to be important. Enhancing prompts with more detailed and specific instructions led to noticeable gains in answer quality, particularly reflected in factual correctness and semantic similarity scores. The prompt templates were inspired by Mollick-style frameworks, which clearly define the role of the LLM, specify what it should and shouldn't do, and provide structured guidance. These templates were iteratively revised to strike a balance between being concise and informative, ensuring they included the

necessary instructions without overwhelming the model. Revisions to the prompt templates consistently led to increasing factual correctness scores, indicating that the generated answers were becoming more aligned with the reference answers. However, the overall low baseline of this metric suggests that there is still significant room for improvement in the quality and reliability of answer generation.

Limitations: The RAG system implementation faced practical limitations due to API call limits with Cohere's reranking models. To mitigate this, the LangChain-based pipeline was adapted to handle requests more efficiently by adding sleep statements to slow down API call rates as well as processing the dataset by segments to run the full dataset. Specifically, 15 questions were run in each of the first four batches, and 18 questions were run in the final batch. Final performance metrics were computed by averaging the results across these five runs. However, eventually a paid production key was used to continue generating experimental results. Additionally, a hybrid search using both dense and sparse vectors to perform similarity search the vectorstore was planned however due to time constraints, the implementation was not fully added into the pipeline.

The code and documentation References used in this project are in Appendix G.

Link to Assignment 5 Google Colab notebook.