W267 Final Project: Optimizing RAG Through Systematic Evaluations

Sean Sica

2025-04-14

1 Executive Summary

I systematically evaluated RAG systems across multiple configurations to identify optimal parameters for domain-specific question answering. Through a phased experimental approach, I found that embedding model selection has less impact than query template design and retrieval parameters. The multi-qa-mpnet-base-cos-v1 embedding model with chunk size of 1024, chunk overlap of 100, and a similarity-based retriever with top-k=8 consistently delivered the best performance. Engineering-focused query templates consistently outperformed marketing templates regardless of other parameters. Larger chunk sizes and higher overlap improved context relevance, while similarity-based retrieval with appropriate top-k values yielded more faithful and accurate responses than threshold-based approaches.

All code and experimental results can be found on GitHub here.

2 Introduction

This project implemented a RAG system to enhance LLM responses with external knowledge, addressing hallucination issues and improving factual accuracy for domain-specific Q&A. The system ingests technical documentation, processes it into retrievable chunks, and leverages embedding models and retrieval strategies to provide contextually relevant answers. A key feature is its ability to customize responses for different audiences (engineering vs. marketing) through specialized query templates.

3 Key Findings

- Template Engineering Matters Most: The engineering vs. marketing query template had a more significant impact on performance than embedding model selection, with engineering templates consistently outperforming marketing ones across all configurations. Notably, the engineering template consistently outperformed the marketing template, which I attribute to the nature of the RAG corpus and evaluation corpus being fundamentally biased towards technical content (i.e., the marketing template has the added challenge of needing to summarize/translate technical material for a non-technical audience, whereas this is not as much a requirement for the engineering audience).
- Chunking Parameters: A chunk size of 1024 tokens with 100 token overlap provided the best balance between context preservation and retrieval precision, with context relevance showing a strong correlation ($R^2 \approx 0.70$) to these parameters.
- Retriever Strategy: Similarity-based retrieval with top-k=8 consistently outperformed other strategies, while high similarity thresholds (0.8+) often retrieved nothing and collapsed performance, demonstrating that retriever method selection significantly impacts evaluation metrics ($R^2 \approx 0.4 - 0.6$).
- Embedding Model Performance: While multi-qa-mpnet-base-cos-v1 was technically the best performing embedding model for both engineering and marketing contexts, differences between embedding models

were statistically insignificant. This could likely be overcome with a larger evaluation QA corpus, as well as repeated iterations of experiments to increase the draw pool for the random distribution. LangSmith specifically provides an argument for evaluating (and normalizing) metric outcomes against multiple runs, but this drives up the cost linearly.

- Evaluation Metric Sensitivity: Different evaluation metrics showed varying sensitivities to parameter changes, with DeepEval faithfulness being extremely sensitive to retriever method changes, making it an effective signal for optimizing RAG systems.

4 Methodology

4.1 Technical Approach

I followed a three-phase approach to optimize the RAG system:

1. **Embedding Model Selection:** Five embedding models (all-mpnet-base-v2, all-MiniLM-L6-v2, all-distilroberta-v1, multi-qa-mpnet-base-cos-v1, and multi-qa-mpnet-base-dot-v1) were evaluated under constant conditions. This phase established the best semantic representation for the corpus while minimizing the need for costly re-embedding.
2. **Chunking Parameter Optimization:** After selecting the best embedding model, I experimented with different chunk sizes (256, 512, 1024, 2048 tokens) and overlaps (0, 50, 100 tokens). This phase identified a 1024-token chunk with 100-token overlap as optimal for balancing context preservation and retrieval precision.
3. **Retrieval and Generation Refinement:** Advanced retrieval strategies, including similarity-based retrieval with various top-k values (4, 8, 12), Maximum Marginal Relevance (MMR) for enhanced diversity, and similarity score thresholding, were tested. Simultaneously, distinct query templates were tailored for engineering versus marketing audiences. The engineering-focused template consistently delivered superior results.
   In evaluating the RAG generator, comparisons between Cohere(Cohere, 2024) and Mistral models revealed that Cohere not only produced higher-quality outputs but also achieved significantly faster response times, making it the preferred choice despite its network call overhead.

All experiments were executed on an Amazon EC2 instance (g4dn.xlarge) featuring one NVIDIA T4 GPU, 4 vCPUs, and 16 GB of memory.

4.2 Evaluation Process

The evaluation process was designed to ensure reproducibility and enable in-depth analysis. First, evaluation questions were processed and all details were logged to LangSmith, with results also saved locally for offline analysis.

Next, evaluation metrics for each configuration were aggregated into an unweighted, normalized composite score to rank RAG system configurations and identify top performers. (Future iterations could incorporate weighting mechanisms to prioritize critical metrics).

Finally, statistical significance was confirmed through ANOVA and regression analyses to identify correlations between configuration choices and performance outcomes. Comparative analysis between Cohere and Mistral models showed that, despite potential network latency, Cohere delivered higher quality outputs and faster responses—Mistral required ~25 seconds per request versus Cohere's 2-3 seconds—leading to Cohere's selection as the sole generator.

4.3 Testing and Evaluation

System performance was measured using 13 distinct metrics across three categories: (1) Simple Metrics using binary LLM-as-judge approach (Groundedness, Retrieval Relevance, Relevance, and Correctness); (2)

RAGAS Metrics[Es et al. (2023)](Team, 2024) with normalized LLM-as-judge scoring (Faithfulness, Context Relevance, Response Relevance, and Answer Accuracy); and (3) additional metrics including DeepEval G-Eval, DeepEval Faithfulness, BERTScore(Zhang et al., 2020), plus word/character counts for length comparison.

Testing used 78 domain-specific questions with reference answers, assessing statistical significance through ANOVA tests, pairwise t-tests, and regression analyses. GPT-4o served as the judge model for all LLM-as-judge evaluations. While Simple Metrics use a single inference request producing binary scores, RAGAS metrics involve multiple inference calls with normalized scores (0.25 increments), providing more stable means across the evaluation set. The DeepEval G-Eval metric particularly aligned with human judgment through its chain-of-thought reasoning.

5 Results and Findings

5.1 Proof of Concept Functionality

The RAG system successfully demonstrated the ability to answer domain-specific questions with high factual accuracy. Although several embedding models were evaluated, statistical analysis revealed that differences among them were minimal—only about 5% of pairwise comparisons showed statistical significance. The multi-qa-mpnet-base-cos-v1 model performed marginally better; however, the variation in evaluation scores was so slight that the choice of embedding model appears less critical than other configuration parameters.

In contrast, chunking parameters had a substantial impact on performance metrics. Experiments consistently showed that a chunk size of 1024 tokens combined with a 100-token overlap yielded the best results. This combination significantly improved context relevance, as indicated by statistically significant p-values in metrics such as Ragas Faithfulness, DeepEval Faithfulness, G-Eval, and BERTScore(Zhang et al., 2020). Larger chunks with moderate overlap effectively preserved related information, thereby enhancing overall system performance.

Retrieval method selection emerged as the most influential parameter. Similarity-based retrieval with a top-k value of 8 consistently outperformed alternative strategies, though top-k=12 scored very highly as well, with some question sets scoring better. In comparison, methods like Maximum Marginal Relevance produced a more diverse yet sometimes less precise set of results, while similarity score thresholding at 0.8 often retrieved no data at all, leading to a collapse in performance.

Additionally, the configuration of the generator template had a significant impact, with the engineering-oriented template outperforming the marketing version across all experiments. A sizable effect (reflected in an effect size of d=0.70 for relevance metrics) suggests that technical precision in question formulation directly improves retrieval quality, even if the final output is later adjusted for a non-technical audience.

Finally, performance metrics for system resource usage further validated the setup. The Cohere implementation consumed approximately 1.3 GB of memory and produced responses within 2–5 seconds, while vector retrieval operations took roughly 0.25 seconds. Thus, the overall latency is primarily attributable to the language model generation phase.

6 Lessons Learned

Technical Insights:

1. Parameter Interaction: Chunk size and retrieval strategy are collinear: larger chunks benefit from higher top-k values to maintain diversity of information. Interestingly, a chunk overlap of 100 was consistently observed to be the "sweet spot" across all tested chunk sizes, yielding the highest results within its vertical of same (chunk) size experiments.
2. Metric Reliability: DeepEval's G-Eval showed strong correlation with my own human judgment, making it a reliable indicator of system quality. It also conveniently provides a supplemental natural

language explanation for the produced score, and I found that those reasons were always grounded and well reasoned.

3. Statistical Significance: Most observed differences between configurations were within statistical noise. This was to be expected given the relatively small size of the validation set (78 questions). This could possibly have been overcome by running each evaluation more than once, but the existing experimental setup was already cost prohibitive for most. Overall, this finding just highlights the importance of doing statistical evaluations rather than relying on raw scores, because typical RAG evaluations run the risk of misleading interpreters into making false causal relationships between the evalation metric and some variable of interest.
4. Overall, the majority of the RAG optimization I experimented with were not worth the squeeze. Despite some independent variables showing colinearity with the evalation metrics and having statistical significance, most of the potential improvement margins were quite narrow. We might infer that the system is approaching it's theoretical maximum output, qualitatively speaking, given the project constraints. This is certainly possible given the overall coherence and accuracy of the results from a human evaluation perspective. But we can't know for sure without expanding the scope of the project.
5. Audience Adaptation: How questions are formulated dramatically affects retrieval quality, with engineering/technical phrasings yielding better performance.
6. Evaluation Trade-offs: Different metrics emphasized different aspects of performance, revealing the multi-faceted nature of "good" RAG systems. Achieving automated evaluations that are as reliable as human judgement is perhaps even more challenging than building and optimizing the RAG system itself.

7 Challenges and Limitations

The experimental process revealed several key challenges. Different evaluation frameworks sometimes produced conflicting results. DeepEval's faithfulness metric tended to be more lenient than RAGAS assessments. In addition, using LLMs as evaluators introduces subjective biases that can affect score objectivity. Notably, many performance differences did not reach statistical significance, suggesting that either the sample size was too small or the parameter adjustments were too subtle to yield definitive outcomes.

Resource constraints further limited the study. A phased approach was necessary, potentially missing some optimal parameter combinations, and the financial burden from API calls and hardware limitations (such as using an NVIDIA T4 GPU) constrained experimentation. For instance, self-hosted models like Mistral required around 24 seconds per request compared to Cohere's 2–3 seconds. Future work should explore hybrid retrieval strategies, finer parameter tuning, improved prompt engineering, and broader cross-domain testing with additional models.

8 Summary & Recommendations

Based on the experimental results, I recommend implementing a production RAG system with the following configuration:

- Embedding Model: multi-qa-mpnet-base-cos-v1
- Chunk Size: 1024 tokens
- Chunk Overlap: 100 tokens
- Retrieval Strategy: Similarity-based with top-k=8
- Query Template: Engineering-focused template, even for non-technical users

This configuration consistently delivered the best overall performance across metrics and demonstrated robust behavior across different question types. The engineering template's superior performance suggests that technical precision in question formulation benefits retrieval quality, even when the final response will be adapted for non-technical audiences.

The system's modest resource requirements (1.3 GB MB memory, 2-5 second response time) make it viable for research environments with minimal infrastructure. The fact that vector retrieval comprises only a small

portion of the overall latency (0.25 seconds) suggests that optimization efforts should focus on LLM inference rather than retrieval operations.

For future development, establishing a continuous evaluation pipeline would enable ongoing refinement as document collections grow and query patterns evolve. The significant impact of template design also suggests that A/B testing different query formulations could yield substantial performance improvements with minimal implementation costs.

References

Cohere. (2024). Basic rag: Retrieval-augmented generation with cohere. https://docs.cohere.com/page/basic-rag

Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023). Ragas: Automated evaluation of retrieval augmented generation. https://arxiv.org/abs/2309.15217

Team, R. (2024). Ragas: List of available metrics. https://docs.ragas.io/en/latest/concepts/metrics/available_metrics/

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). Bertscore: Evaluating text generation with bert. https://arxiv.org/abs/1904.09675