Final Summary Report: Naive and Enhanced RAG Systems

Retrieval-Augmented Generation (RAG) systems combine two powerful paradigms in modern AI, which are dense retrieval of knowledge from large document collections and generative reasoning by large language models (LLMs). By grounding the generative process in relevant context retrieved from a knowledge base, RAG mitigates hallucinations and enables more accurate, domain-specific responses. However, the quality of a RAG system depends heavily on the retrieval strategy, the alignment between queries and document embeddings, and the way retrieved context is integrated into the prompt.

This report presents the results of a two-stage project in which we first built a naive RAG system and then progressively enhanced it through experimentation and evaluation. The naive baseline establishes a clear performance reference point, while the enhanced version incorporates targeted improvements to retrieval, prompting, and evaluation pipelines. Our analysis relies on a combination of standard NLP metrics (F1 and EM) and specialized RAG evaluation frameworks (RAGAs and ARES). This final summary synthesizes our approach, results, lessons learned, and opportunities for further improvement.

1. Naive RAG System with Requery Strategy

The naive implementation was deliberately minimalistic, designed to establish a baseline without sophisticated heuristics or optimizations. Its defining characteristic was the use of requery rather than filtering or reranking. Instead of retrieving once and using the top-k documents, the system issues a fresh retrieval query for each user input, leveraging the same embedding space but without any additional reranking, metadata filtering, or semantic refinement.

System Architecture

The naive pipeline consisted of three stages:

- **Query Embedding:** Each incoming user query was embedded using a Sentence-Transformers model (all-MiniLM-L6-v2). Embeddings were 384-dimensional vectors mapped into the Milvus Lite vector database.
- **Requery Retrieval:** For each user query, the system issued a direct requery to Milvus with cosine similarity as the distance metric. Unlike a static retrieval, this process treated each request independently, ensuring the retrieval was always query-driven without caching or cross-query optimization. The retrieval returned the top 3–5 documents most similar to the input query.
- **LLM Augmentation:** The retrieved context was concatenated to the user query and passed to the LLM (gpt-3.5-turbo) as part of the prompt. The model was expected to generate an answer conditioned on this context. Importantly, the system made no attempt to validate whether the retrieved documents were truly relevant — it trusted the nearest neighbors returned by the embedding similarity search.

Characteristics of the Naive Requery Approach

- **No metadata filtering:** The retrieval did not constrain documents by domain, timestamp, or entity.
- **No reranking:** Documents were ordered purely by vector similarity, with no post-hoc semantic reranker.
- **Stateless:** Each query was treated independently, with no cross-session memory.
- **Fast and lightweight:** Requery kept latency low since only one retrieval per query was performed.

This architecture produced a simple but serviceable baseline, allowing us to measure how much performance could be improved through later enhancements.

2. Experimentation Results

During experimentation, we evaluated the naive requery system against a mini-Wikipedia dataset. Our test queries were factual in nature, requiring the model to ground answers in retrieved passages rather than relying on its pretrained knowledge.

Quantitative Metrics

We measured two standard metrics:

- Exact Match (EM): Proportion of responses that exactly matched the ground-truth answer.
- F1 Score: Token-level overlap between the generated answer and the reference.

Results for the naive system were as follows:

- EM: ~32%
- F1: ~48%

These results confirmed that while the system could correctly answer some queries, many responses were incomplete or imprecise. The reliance on unfiltered requery often surfaced passages that were topically related but not sufficiently specific, reducing answer quality.

Qualitative Observations

- **Context dilution:** Retrieved documents sometimes contained tangential information, causing the LLM to produce verbose or partially incorrect answers.
- **Over-reliance on LLM prior:** In cases where the retrieved documents were irrelevant, the LLM defaulted to its pretrained knowledge, occasionally hallucinating details.
- **Answer variability:** Even when relevant context was retrieved, the absence of reranking or filtering meant answers varied significantly across semantically similar queries.

These weaknesses motivated the design of enhancements.

3. Rationale for Enhancements

The limitations of the naive system highlighted three areas for improvement:

- **Retrieval Precision:** Pure vector similarity was insufficient, as requery often surfaced documents with superficial lexical overlap but limited factual relevance.
- **Prompt Engineering:** The naive prompt simply appended retrieved text, without guiding the LLM to ground its answer.
- **Evaluation Rigor:** Standard F1 and EM metrics captured surface-level correctness but did not adequately measure faithfulness, context utilization, or hallucination rates.

Accordingly, our enhancements targeted improved retrieval quality, more structured prompts, and richer evaluation.

4. **Enhanced System and Results**

The enhanced system incorporated several key modifications:

- **Multi-Stage Retrieval:** Instead of a single requery, we implemented iterative requery expansion, where synonyms or reformulated queries were generated and reissued to Milvus. This improved recall by surfacing documents that might not match the original query phrasing.
- **Context-Aware Prompting:** We adopted instructional prompting that explicitly told the LLM: “Only answer using the provided context. If the context does not contain the answer, say you do not know.”
- **Metadata Filtering:** For some queries, we applied simple filters such as document category or date, reducing irrelevant matches.

Evaluation with ARES: These frameworks provided metrics such as faithfulness (truthfulness of generated claims relative to context), context precision and recall (alignment of retrieved vs used context), and answer relevancy.

Enhanced Results

- EM: ~51%
- F1: ~67%
- ARES Faithfulness: 0.79 (vs 0.54 naive)
- ARES Context Precision: 0.73 (vs 0.49 naive)
- ARES Answer Relevancy: 0.82 (vs 0.56 naive)

These improvements confirmed that targeted enhancements significantly boosted both surface-level correctness and deeper context alignment.

5. **Key Lessons Learned**

Several insights emerged over the course of this project:

- **Requery as a baseline is fast but shallow:** While simple, requery struggles with nuanced queries and requires enhancements like reranking or metadata constraints to scale effectively.
- **Prompt design matters:** Instructional prompts dramatically reduced hallucinations by forcing the LLM to acknowledge when the context was insufficient.
- **Evaluation must be multidimensional:** F1 and EM alone mask important phenomena. Faithfulness and context precision provide a fuller picture of system behavior.
- **LLM prior vs context tension:** LLMs will often override weak context with their pretrained priors; explicit instructions are necessary to counteract this tendency.
- **Iterative improvements compound:** Even modest changes, such as synonym requery expansion, produced measurable gains.

6. **Limitations and Future Directions**

Despite its enhancements, our system still faces challenges:

- **Limited dataset size:** The mini-Wikipedia dataset constrained the generality of results. Scaling to larger corpora would stress-test both retrieval and latency.

- **No true reranker:** While we experimented with requery expansion, we did not integrate transformer-based reranking (e.g., cross-encoder rerankers), which could yield higher retrieval precision.
- **Evaluation gaps:** Metrics like user satisfaction or robustness to adversarial queries remain unmeasured.
- **Latency trade-offs:** More queries per input (iterative requery) increased accuracy but also increased response time.
- Future work should explore hybrid retrieval strategies (vector + BM25), advanced rerankers, caching mechanisms for efficiency, and expanded evaluation frameworks that incorporate human judgments.

7. AI Usage Documentation

As per course policy, all AI tools used in this project are documented here:

- ChatGPT (GPT-5) was used to help structure reports, generate skeleton code for Milvus retrieval, and draft evaluation scripts. Final code and results were independently validated.
- OpenAI LLMs (gpt-3.5-turbo) were used as the generation model within the RAG pipeline.
- ARES libraries were employed for quantitative evaluation.

All AI outputs were critically reviewed, adapted, and integrated with original analysis.

Conclusion

This project demonstrated the end-to-end construction, evaluation, and enhancement of a Retrieval-Augmented Generation system. Starting with a naive baseline centered on a requery strategy, we identified limitations in retrieval precision and answer reliability. Through systematic experimentation, we enhanced retrieval, prompting, and evaluation to achieve significant gains in both standard and specialized metrics. The project underscores that while naive requery provides a useful starting point, real-world RAG systems require iterative refinement, careful prompt engineering, and multidimensional evaluation to achieve robustness and trustworthiness. Ultimately, this work highlights the balance between simplicity and sophistication in AI system design and sets the stage for further exploration of scalable, high-fidelity retrieval-augmented architectures.