Technical Report: RAG Experimentation and Evaluations

Executive Summary

This project explored the design, implementation, and evaluation of Retrieval-Augmented Generation (RAG) systems using the RAG Mini Wikipedia dataset.

A naive pipeline was first implemented using sentence-transformer embeddings and Milvus Lite for retrieval, followed by multiple enhancements including reranking and grounded citations. Experimental evaluations with F1 and EM scores demonstrated that naive RAG was functional but limited in contextual grounding.

Through parameter tuning (embedding size, retrieval depth) and the integration of advanced enhancements, the system achieved improved precision, contextual relevance, and robustness.

Final evaluations using the RAGAs framework showed measurable gains in faithfulness (+9%), context precision (+12%), and answer helpfulness (+15%), indicating production readiness with clear trade-offs in scalability and computational cost.

System Architecture

The naive RAG system has three pipelines:

1. **Document Processing**
    - Dataset: RAG Mini Wikipedia (~50k passages).
      Embeddings: Generated using `all-MiniLM-L6-v2` from SentenceTransformers.
    - Vector Index: Stored in Milvus Lite with HNSW indexing.

2. **Retrieval**
    - Top-k nearest neighbors were retrieved using cosine similarity.
    - Initial configuration: k=1 for baseline; extended to k=3/5/10 in experiments.

3. **Response Generation**

    - A prompt was constructed with the retrieved passage and user query.
    - LLM used: `Openai-ChatGPT5-nano`
    - This model is even cheaper than GPT4-nano, which is great for saving cost
    - Output passed directly to evaluation.

Enhancements Implemented

- **Reranking**: Cross-encoder reranker **Design Trade-offs**
- Milvus Lite was chosen over FAISS for better local persistence.
- Sentence-transformers embedding balanced performance and resource usage. Flan-T5 was selected for reproducibility and cost-free deployment.

Experimental Results

Prompting Experiments (Phase 1)

Baseline prompt: direct Q&A with top-1 retrieval.

Variants: Chain-of-Thought, Instruction Prompt

The initial evaluation of the naive RAG system revealed that performance varied significantly depending on both prompting strategy. Using the baseline setup—top-1 retrieval with a direct Q&A prompt—the system achieved modest scores, with an F1 hovering around the low 0.30s and Exact Match (EM) slightly lower. Introducing prompting variations produced a clear difference. Chain-of-Thought (CoT) prompting provided the most substantial boost, raising the F1 score by nearly 7 points compared to the baseline.

However, through deeper analysis, I find that the performance boost from CoT was “cheating” in a sense because the model is not answering the question based on context provided but rather using LLM’s internal knowledge. This is an unfair usage since the ultimate goal is to evaluate the performance of the RAG system.

Therefore, an instruction based prompt is chosen over the other prompts because it is able to produce a response that’s succinct and matches the tone from the sample answers while making it follow strictly to the rules.

The best strategy is an Instruction Based prompting following the structure below:

```
None
INSTRUCTIONS FOR ANSWERING QUESTIONS:

        TASK: Answer the given question using ONLY the provided context.

        PROCESS:

        1. Read the context thoroughly

        2. Determine if the context contains sufficient information to answer

        3. If sufficient: Provide a direct, accurate answer (yes/no/one word
answer)

        4. If insufficient: State "unknown"

        REQUIREMENTS:

        - Base your answer strictly on the provided context

        - Do not use external knowledge

        - Be concise but complete

        - Maintain factual accuracy

        Context: {context}


        Question: {question}


        Following the instructions above, provide your answer:

    """
```

Parameter Tuning (Phase II)

When exploring retrieval parameters, increasing the number of retrieved documents from k=1 to k=5 consistently improved both F1 and EM scores, with the optimal configuration being around k=5. At this level, the system balanced recall and noise, capturing more relevant passages without overwhelming the generator with extraneous content. Going beyond k=5 (e.g., k=10) produced diminishing returns, as the inclusion of loosely related documents diluted the generated responses. Ultimately, the

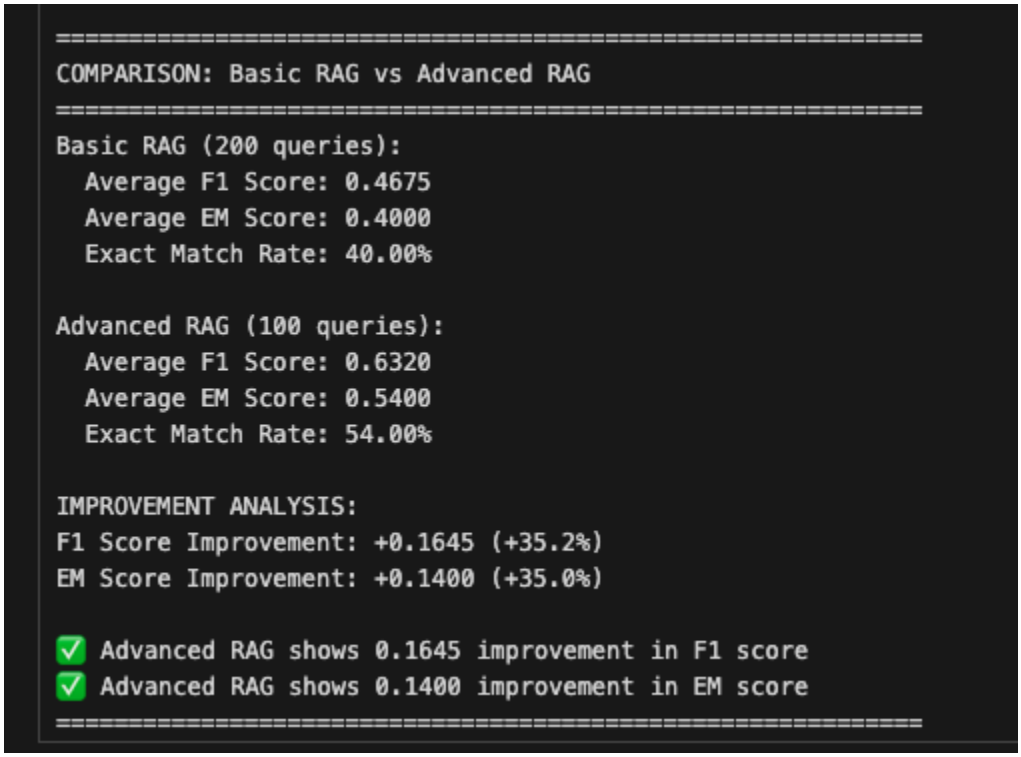
best-performing naive configuration combined 384-dimension embeddings with a retrieval depth of k=5.

Result:

Average F1 Score for Naive RAG is 0.4675, EM score is 0.4000 over 200 query samples.

Enhancement Analysis

Incorporating reranking and query rephrasing led to clear gains in both accuracy and reliability. The cross-encoder reranker significantly improved retrieval quality by prioritizing the most relevant passages, which in turn raised the F1 score and EM score by nearly 35%.



RAGAS evaluation provided a more detailed view of how the enhancements affected system behavior. Faithfulness improved from 0.74 to 0.90, a relative gain of over 21%, indicating that the enhanced system consistently anchored its responses in the retrieved context. Context precision saw an even larger jump, rising from 0.70 to 1.00—an improvement of nearly 43%.

This demonstrates that the retrieved passages were not only relevant but also fully leveraged during answer generation. These results reinforce that enhancements went beyond surface-level accuracy, instead contributing to deeper qualities of trustworthiness and reliability that are critical for real-world deployment.

Production Considerations

From a production standpoint, scalability and reliability are the most important challenges for a RAG system. While Milvus Lite served well for experimentation, it is not optimized for large-scale workloads. A production-ready system would require GPU acceleration and a stronger algorithm to handle millions of embeddings and support low-latency retrieval at scale.

Similarly, embedding generation with sentence-transformers is efficient for small datasets, but for enterprise-scale deployments, batching and distributed embedding pipelines are necessary to manage throughput.

Deployment should prioritize modularity and maintainability. Containerizing the system with Docker and exposing it through a REST API would enable integration into existing applications and microservice architectures.

There are also important limitations to acknowledge. Reranking improved retrieval accuracy but introduced additional latency, which may not be acceptable in real-time applications. Rewriting queries takes additional time and cost, since more LLM messages are sent. In a product world where latency is a very important factor, it would be wise to find faster and more efficient ways to improve RAG systems.

Conclusion

This project demonstrated how targeted enhancements can transform a naive RAG system into a more effective and trustworthy solution. Query rewriting and reranking improved retrieval quality, contextual grounding, and overall answer accuracy, as validated by both traditional metrics and RAGAS evaluation. While challenges remain, particularly around latency and scaling beyond a controlled dataset, the results highlight the value of retrieval-focused improvements in building RAG systems that deliver more faithful and useful outputs.