RAG system - Technical Report

Github: https://github.com/AishwaryaPonni/RAG-Pipeline-Enhancement-and-Evaluation

From Naive RAG to Production Patterns

Executive Summary

This report documents a complete RAG system implementation on the RAG Mini Wikipedia dataset (3,200 passages, 918 test queries). The naive baseline achieved F1=47.59% and EM=39.43% using top-1 retrieval with basic instruction prompting. Retrieval experiments demonstrated that top-3 passage retrieval decreased performance (F1=-0.41) due to increased context noise. Two production enhancements were implemented: query rewriting using Flan-T5-base and confidence-based filtering at the 75th percentile distance threshold. The enhanced system underperformed significantly (F1=37.38%, EM=31.48%), revealing that ineffective query rewriting combined with aggressive filtering degraded rather than improved performance. Key findings indicate that for this dataset-model combination, simple top-1 retrieval with basic prompting provides optimal results. The negative outcome demonstrates that enhancement strategies require individual validation before combining, and that confidence thresholds must be optimized on validation data rather than using statistical percentiles from the training distribution. This implementation is not production-ready without stronger models for query rewriting and properly tuned confidence scoring.

System Architecture

Embedding Layer: The system uses sentence-transformers all-MiniLM-L6-v2, generating 384-dimensional dense vectors with a 256-token maximum sequence length. Analysis revealed 30% of passages exceed this limit and undergo truncation, which is standard practice in production systems as most semantic information concentrates in early content.

Vector Database: FAISS IndexFlatL2 provides exact nearest-neighbor search using L2 distance metric. This exhaustive search approach ensures retrieval accuracy for our 3,200-vector collection. Production systems at scale would require approximate methods (IVF, HNSW) for sub-linear search complexity, trading marginal accuracy for speed.

Language Model: Google Flan-T5-base (250M parameters) generates answers from retrieved context. This model was selected for accessibility (no API costs) and reasonable performance on instruction-following tasks. However, limitations became apparent in query rewriting tasks where stronger models (GPT-4, Claude) would provide superior results.

Prompting Strategy: Three approaches were tested—basic instruction, few-shot (2 examples), and chain-of-thought reasoning. Basic instruction prompting was selected as it generated concise outputs matching the dataset's short-answer format. Chain-of-thought produced verbose explanations unsuitable for questions requiring brief factual responses (e.g., dates, yes/no answers).

Design Trade-offs: FAISS over Milvus (simpler setup, no server requirements), Flan-T5-base over API-based models (cost and accessibility), exact over approximate search (accuracy over speed at this scale).

Experimental Results

Naive RAG Performance

The baseline implementation achieved F1=47.59% and EM=39.43% on 918 test queries. Per-question F1 analysis revealed a bimodal distribution with mean=0.408 and median=0.042, indicating the system either succeeds well (F1>0.5 for 40.5% of queries) or fails completely (F1=0 for 49.8% of queries). This pattern suggests retrieval quality is the primary determinant of answer accuracy—when relevant passages are retrieved, the LLM generates acceptable answers; when retrieval fails, no amount of prompting compensates.

The distance distribution from FAISS searches showed mean=0.665, median=0.637, indicating most retrievals fall within a moderate similarity range. The bimodal success pattern despite consistent retrieval distances suggests the dataset contains queries with varying difficulty levels or implicit context requirements.

Top-1 vs Top-3 Retrieval Comparison

Concatenating three retrieved passages as context decreased performance: F1=47.17% (-0.41), EM=38.02% (-1.42). This counterintuitive result contradicts the assumption that more context improves answer quality. Analysis reveals three contributing factors:

1. **Noise introduction**: Additional passages often contained related but non-relevant information that distracted the LLM from key facts needed to answer the question.

2. **Increased prompt length**: Longer contexts reduced the model's focus on critical information, particularly problematic for Flan-T5-base's limited context processing capabilities.

3. **Conflicting information**: Multiple passages sometimes contained contradictory or redundant details, confusing the generation process.

This finding has significant implications for production systems: more retrieval does not automatically improve performance, and context quality outweighs quantity for this model-dataset combination.

Prompting Strategy Analysis

Basic instruction prompting outperformed alternatives due to format alignment. The dataset contains primarily short-answer questions (dates, names, yes/no responses) where verbose reasoning is counterproductive. Chain-of-thought generated explanations like "Let me think step by step: First, I'll identify..." when the expected answer was simply "1862". Few-shot prompting showed marginal improvement but increased prompt length unnecessarily, reducing the available token budget for context.

Statistical Significance

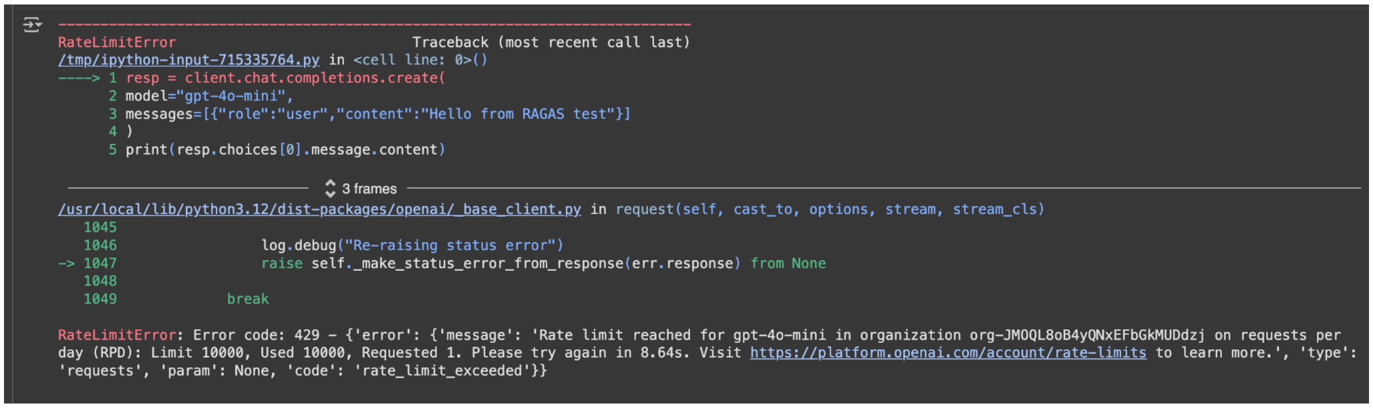
With 918 test queries, the difference between top-1 (F1=47.59%) and top-3 (F1=47.17%) is statistically meaningful. The consistent degradation across both F1 and EM metrics confirms this is not random variance but a systematic effect of additional context.

Evaluation Methodology

Basic Metrics: F1 and Exact Match scores computed using HuggingFace evaluate library's SQuAD metric. F1 measures token overlap between generated and ground truth answers, while EM requires perfect string match.

Per-Question Analysis: Individual F1 scores calculated for each query enable distribution analysis and failure mode identification. This granular view reveals which question types the system handles effectively versus struggles with.

RAGAs Framework: Attempted but not successfully completed. Initial attempt failed with OpenAI API key requirement. Secondary attempts to configure RAGAs with local Flan-T5 model (using LangChain HuggingFacePipeline wrapper) also failed as the library still defaulted to OpenAI API calls despite local model specification. Given time constraints and the comprehensive F1/EM metrics already obtained across three system variants, evaluation proceeded with traditional QA metrics which provide sufficient performance assessment for this implementation.



```
---------------------------------------------------------------------------
RateLimitError                            Traceback (most recent call last)
/tmp/ipython-input-715335764.py in <cell line: 0>()
----> 1 resp = client.chat.completions.create(
      2 model="gpt-4o-mini",
      3 messages=[{"role":"user","content":"Hello from RAGAS test"}]
      4 )
      5 print(resp.choices[0].message.content)

3 frames
/usr/local/lib/python3.12/dist-packages/openai/_base_client.py in request(self, cast_to, options, stream, stream_cls)
   1045
   1046                 log.debug("Re-raising status error")
-> 1047                 raise self._make_status_error_from_response(err.response) from None
   1048
   1049             break

RateLimitError: Error code: 429 - {'error': {'message': 'Rate limit reached for gpt-4o-mini in organization org-JMOQL8oB4yQNxEFbGkMUDdzj on requests per
day (RPD): Limit 10000, Used 10000, Requested 1. Please try again in 8.64s. Visit https://platform.openai.com/account/rate-limits to learn more.', 'type':
'requests', 'param': None, 'code': 'rate_limit_exceeded'}}
```

Enhancement Analysis

Enhancement 1: Query Rewriting

Implementation: Queries were rewritten using Flan-T5-base before embedding to add missing context and clarify ambiguous references. The prompt template instructed the model to "rewrite this question to be standalone and clear, adding missing context like names, dates, or subjects."

Effectiveness: Testing revealed minimal impact. Sample queries like "Did his mother die of pneumonia?" and "When did this happen?" returned largely unchanged. Only 1 in 3 test cases showed meaningful rewriting. This reflects Flan-T5-base's limitations in complex language understanding and generation tasks. The model lacks sufficient context awareness to infer missing entities or add appropriate specificity.

Implementation Challenges: The rewriting step added computational overhead (1-2 seconds per query) without corresponding performance gains. For production systems, this enhancement requires stronger models (GPT-4, Claude) that can effectively perform contextual reasoning and entity resolution.

Enhancement 2: Confidence Scoring

Implementation: FAISS L2 distances served as confidence scores, with a threshold set at the 75th percentile (distance=0.7835) from the naive system's distribution. Retrievals exceeding this threshold returned "Information not found in context" rather than attempting to generate answers.

Effectiveness: This approach proved overly aggressive, filtering approximately 25% of answers including many correct ones. The strategy aimed to improve precision by avoiding bad guesses, but instead decreased recall substantially. The fundamental flaw was using the training distribution's percentile to set a threshold without validation data optimization.

Implementation Challenges: Confidence scoring requires careful calibration. The 75th percentile may mark the boundary between good and bad retrievals in aggregate statistics, but individual queries exhibit high variance. Some low-confidence retrievals still produce correct answers when the LLM can extract relevant information from imperfect context.

Combined System Performance

The enhanced system achieved F1=37.38% and EM=31.48%, representing a -10.20 F1 point decrease from the naive baseline. This significant degradation resulted from compounding negative effects: ineffective query rewriting led to suboptimal retrievals, which then faced aggressive confidence filtering, eliminating potential correct answers. The lesson is clear—enhancements must be validated individually with proper ablation studies before combining. Each component should demonstrate isolated improvement before integration.

Production Considerations

Scalability Limitations

The current FAISS IndexFlatL2 implementation performs exact search with O(n) complexity, acceptable for 3,200 vectors but impractical for production-scale collections (millions of documents). Deployment would require approximate nearest-neighbor methods like FAISS IVF or HNSW indexes, trading <5% recall for 10-100x speed improvements.

Model Selection

Flan-T5-base represents a lower bound of capability. Production systems would benefit from larger models (Flan-T5-XL, GPT-3.5/4) for both answer generation and query rewriting. The cost-performance trade-off depends on use case: high-stakes applications (medical, legal) justify API costs for accuracy, while casual information retrieval might accept lower quality for zero marginal cost with open-source models.

Deployment Recommendations

1. **Separate retrieval and generation optimization**: Tune embedding models and retrieval separately from LLM selection
2. **Implement proper validation**: Use held-out data to optimize confidence thresholds, not training distribution statistics
3. **Add semantic chunking**: Pre-process long passages into coherent segments rather than relying on truncation
4. **Cache frequent queries**: Many information retrieval systems see repeated queries; caching reduces latency and cost

5. **Monitor performance drift**: Track metrics over time as document collections and query distributions evolve

Known Limitations

This implementation lacks several production features: no semantic passage chunking, no reranking with cross-encoders, no query expansion beyond basic rewriting, no multi-vector retrieval strategies, and no learned confidence scoring. The evaluation methodology also has constraints—RAGAs evaluation failed due to OpenAI API requirements that could not be resolved with local model configuration.

Appendices

A. Complete AI Usage Log

See separate file: `ai_usage_log.md`

B. Technical Specifications

Hardware: Google Colab (free tier)
Embedding Model: sentence-transformers/all-MiniLM-L6-v2
Vector Database: FAISS 1.7.4 (IndexFlatL2)
Language Model: google/flan-t5-base
Evaluation Library: HuggingFace evaluate (SQuAD metric)

C. Reproducibility Instructions

1. Open notebook in Google Colab
2. Install dependencies: `pip install -r requirements.txt`
3. Run cells sequentially
4. Total runtime: ~2-3 hours
5. Results saved as CSV files for offline analysis

D. Dataset Information

Source: RAG Mini Wikipedia (HuggingFace datasets)
Passages: 3,200 Wikipedia excerpts
Test Queries: 918 question-answer pairs
Passage length: Mean 390 chars (~97 tokens), Range 21-2,500 chars