Assignment 2: RAG pipeline Analysis Report

AndrewID: hsintunl

Executive Summary

This project developed, tested, and enhanced a Retrieval-Augmented Generation (RAG) pipeline using the Mini-Wikipedia dataset and a Milvus vector database. The naive implementation, built on all-MiniLM-L6-v2 embeddings (dim=384) with top-1 retrieval, produced modest baseline scores (F1 ≈ 29.7, EM ≈ 22.5). Iterative experimentation across embedding sizes, retrieval depths, and prompting strategies identified the best baseline: all-mpnet-base-v2 (dim=768), k=5, and persona prompting.

Two advanced enhancements—query rewriting and re-ranking—were then integrated. While traditional evaluation (F1/EM) showed limited uplift over the best baseline, RAGAs metrics revealed meaningful improvements: faithfulness rose from 0.69 → 0.83, context precision from 0.70 → 0.87, and context recall from 0.58 → 0.78. These results show the enhancements reduced hallucinations and improved evidence grounding, even when token-level metrics plateaued.

These improvements demonstrate that advanced retrieval optimization and prompt design not only improve answer quality but also reduce hallucinations by grounding responses more effectively in retrieved passages. With proper caching and selective reranking, the system is suitable for production deployment.

System Architecture

The RAG pipeline is composed of three primary stages: ingestion, retrieval, and generation.

Ingestion

The knowledge base consisted of 3,200 passages, ranging from 4 to 428 words (27 to 2,542 characters). To handle this variability, the Milvus schema included a max_length equals 3000 field for passages, ensuring storage without truncation. As for the embeddings, two models from *sentence-transformers* were compared:

- all-MiniLM-L6-v2 (dim=384): efficient and fast, suitable for shorter contexts.
- all-mpnet-base-v2 (dim=768): richer semantic representations, at the cost of higher latency.

Retrieval

Passages were indexed in Milvus for scalable vector search. Retrieval depth (k) was varied (1, 3, 5). Larger k improved recall but risked including irrelevant passages, motivating the later use of reranking.

Generation

Retrieved passages were injected into prompts for the LLM model Flan-T5-Base. This model was chosen because it is open-source, lightweight enough for efficient local execution, and fine-tuned for instruction following, making it a practical option within the resource constraints of this project. Three prompting strategies were tested: naive, chain-of-thought (CoT), and persona.

Persona prompting consistently balanced factuality and fluency, making it the default for the best baseline.
It should be noted that there are design trade-offs between latency and accuracy that shaped key choices. While MiniLM was faster but less precise, MPNet achieved stronger semantic alignment at a higher cost. Hence, reranking was introduced to mitigate the noise from higher-k retrieval.

Experimental Results
Performance was evaluated across prompting strategies, embedding models, and retrieval depths using 120 stratified queries. Stratification was based on query length, ensuring balanced coverage of short, medium, and long questions (40 each). This approach preserved representativeness, enabled statistical stability, and reduced runtime to manageable levels while maintaining fairness across naive, baseline, and enhanced pipelines.

Baseline Findings

- **Prompt Strategy:** Persona consistently outperformed Naive and CoT, regardless of embedding choice. For example, Persona + MPNet + k=5 achieved the overall best score (F1 ≈ 49.86, EM = 40.0). Persona's concise, context-grounded style aligned well with the dataset's short reference answers, while CoT often generated verbose reasoning that diluted lexical overlap.
- **Retrieval Depth (k):** Increasing k from 1 → 3 → 5 consistently improved F1, particularly with persona prompting. This indicates that retrieving more passages raises the likelihood of including the correct answer span, though it also introduces noise.
- **Embedding Model:** While MiniLM (384-dim) and MPNet (768-dim) showed comparable trends, MPNet consistently provided slight gains, particularly on longer or more complex queries, confirming the value of higher-dimensional semantic representations.

Enhanced System
According to the results, two key challenges were observed:

- **Prompt Sensitivity (Problem A):** Persona prompting significantly outperformed both Naive and CoT, showing the model's high sensitivity to input formulation. This motivated query rewriting to improve clarity and grounding, especially when original queries were vague.
- **Context Precision Issue (Problem B):** Retrieval at k=5 performed better than k=1, indicating that retrieving only one passage often missed the answer. However, retrieving five passages introduced additional noise, creating a need for reranking to identify the most relevant evidence among retrieved candidates.

Hence, query rewriting and reranking were the enhancements selected. However, after integrating both, F1/EM did not significantly increase beyond the best baseline (F1 ≈ 48.32 vs. 49.6; EM ≈ 28.33 vs. 40.0). Possible explanations include:

- In some cases, the rewritten queries did not yield stronger retrieval terms; occasionally, rewrites were even less specific than the original, leading to weaker retrieval. Because many original queries were already simple—particularly among the short queries sampled—rewriting often made little difference and sometimes worsened performance.

- Since baseline Top-5 retrieval was already relatively strong, the cross-encoder reranker could only make marginal adjustments. When retrieval quality is already high, reranking alone cannot meaningfully boost scores.

These findings highlight a realistic lesson: not all enhancements improve every dataset or domain. Even sophisticated techniques such as query rewriting and reranking may deliver limited benefits if the baseline pipeline is already well-optimized.

Retrieval_K	**Prompt_Strategy**	**Embedding_Dim**	**Avg_F1**	**Avg_EM**
1	naive	all-MiniLM-L6-v2	29.73	22.5
1	cot	all-MiniLM-L6-v2	36.8	25.83
1	persona	all-MiniLM-L6-v2	38.21	30.83
3	naive	all-MiniLM-L6-v2	35.8	26.67
3	cot	all-MiniLM-L6-v2	37.22	25.83
3	persona	all-MiniLM-L6-v2	46.56	37.5
5	naive	all-MiniLM-L6-v2	40.44	30.83
5	cot	all-MiniLM-L6-v2	38.77	25
5	persona	all-MiniLM-L6-v2	49	39.17
1	naive	all-mpnet-base-v2	27.71	22.5
1	cot	all-mpnet-base-v2	32.64	23.33
1	persona	all-mpnet-base-v2	34.53	29.17
3	naive	all-mpnet-base-v2	36.73	28.33
3	cot	all-mpnet-base-v2	37.03	25.83
3	persona	all-mpnet-base-v2	44.69	35.83
5	naive	all-mpnet-base-v2	41.77	32.5
5	cot	all-mpnet-base-v2	38.41	26.67
5	persona	all-mpnet-base-v2	49.86	40
5	enhanced-persona+qr+rerank	all-mpnet-base-v2	48.32	38.33

Figure 1: Comparative Evaluation Metrics. Naive is clearly weakest, best baseline achieves the highest F1/EM, while the enhanced pipeline shows comparable scores only.

Enhancement Analysis

While F1 and EM captured surface-level lexical accuracy, RAGAs evaluation was essential to measure deeper semantic alignment. Four complementary metrics were emphasized:

- Faithfulness: Whether generated answers remained consistent with the retrieved evidence.
- Answer Relevancy: The degree to which the generated answer directly addressed the original question.
- Context Precision: The proportion of retrieved passages that were useful and relevant.
- Context Recall: Whether all necessary supporting evidence was successfully retrieved.

Naive vs. Best Baseline

- Faithfulness improved significantly (0.698 → 0.819).

- Context recall also rose sharply (0.583 → 0.767).

This shows that moving from k=1 with naive prompting to k=5 with persona prompting + MPNet represented a qualitative leap. The system retrieved more relevant contexts and produced answers that were much more faithful to the evidence.

Best Baseline vs. Enhanced

- Faithfulness improved slightly further (0.819 → 0.835).
- Answer relevancy also increased (0.731 → 0.771).
- Context precision improved markedly (0.730 → 0.868).
- Context recall remained roughly stable (0.767 → 0.783).

This indicates that while the Enhanced pipeline (query rewriting + reranking) did not produce large improvements in token-level metrics (F1/EM), it performed strongly on RAGAs metrics, especially context precision. The selected passages were more relevant and less noisy.

In summary, the naive system, limited by k=1 retrieval, suffered from low context recall, often leaving answers incomplete. Expanding to k=5 in the best baseline improved recall by covering more relevant contexts, though precision remained limited due to added noise. The enhanced pipeline, through query rewriting and reranking, effectively filtered irrelevant passages, yielding higher precision along with gains in faithfulness and answer relevancy. While F1/EM scores showed little improvement, RAGAs metrics demonstrated clear benefits in semantic quality and grounding—especially in context handling—indicating stronger reliability for real-world deployment.

	faithfulness	answer_relevancy	context_precision	context_recall
Navie	0.698	0.747	0.708	0.583
Baseline	0.819	0.731	0.731	0.767
Enhanced	0.835	0.771	0.868	0.783

Figure 2: Comparative RAGAS Metrics. While the naive system lags significantly, the best baseline delivers a step-change improvement in recall, and the enhanced pipeline excels in precision.

Production Considerations

Appendices

AI Usage Log

- **Tool**: ChatGPT (GPT-5, OpenAI)
- **Purpose**: Assisted in debugging and clarifying issues related to RAG pipeline implementation and evaluation.
- **Input**:
  - How to configure and load the OpenAI API key for RAGAs evaluation.
  - How to correct RAG input type mismatches (e.g., turning a string field back into dict type for questions and answers).
  - Clarification of how to handle row-level evaluation with batch size = 1.
- **Output Usage:** Code snippets recommendation were given (e.g., fixing dataset preparation for RAGAs, including "reference" and "retrieved_contexts" fields).

- **Verification**: All AI-suggested code was manually tested in the Jupyter Notebook environment to ensure functionality.

File Structure

The project directory is organized into three main folders—data, results, and src—to ensure separation of datasets, experimental outputs, and implementation code.

```
Assignment2/
├── data/
│   ├── processed/
│   │   ├── rag_min_wiki.csv
│   │   └── rag_min_wiki_qa.csv
│   └── evaluation/
│       ├── all_experiments_generations.csv
│       ├── all_experiments_for_ragas.csv
│       └── enhanced_experiments_generations.csv
│
├── results/
│   ├── evaluation_f1_em.csv
│   └── ragas_eval_results.json
│
├── src/
│   ├── step0_data_exploration.ipynb
│   ├── step1_naive_rag.ipynb
│   ├── step2_evaluation.ipynb
│   ├── step3_enhanced_rag.ipynb
│   ├── step4_enhanced_evaluation.ipynb
│   ├── step5a_add_contexts.ipynb
│   ├── step5b_RAGAS.ipynb
│   ├── rag_wikipedia_mini.db
│   └── rag_wikipedia_enhanced.db
│
└── Evaluation Report.docx
```

Technical Specifications

Category	**Details**
Environment	JupyterLab (Conda base), Python 3.10, macOS (local execution)
Core Libraries	transformers, sentence-transformers, langchain_huggingface, ragas, evaluate, pandas, numpy , tqdm, ast
Vector DB	Milvus Client
Embedding Models	all-MiniLM-L6-v2 (384-dim), all-mpnet-base-v2 (768-dim)
Generator Model	Flan-T5-Base (max_new_tokens=100)
Retrieval Settings	k = {1, 3, 5}

Prompting Strategies	naive, chain-of-thought (CoT), persona
Enhancements	Query Rewriting, Reranking
Evaluation Metrics	F1, Exact Match (EM), Faithfulness, Answer Relevancy, Context Precision, Context Recall
Query Set	120 stratified queries (40 short, 40 medium, 40 long)