Ground the Domain - From Naive RAG to Production Patterns

Executive Summary

The Retrieval-Augmented Generation (RAG) system developed for this assignment implements a question-answering pipeline using the RAG Mini Wikipedia dataset. The RAG Mini Wikipedia dataset comprises 3,200 pre-chunked passages with some length variability, exhibiting a right-skewed distribution. Passage lengths range from 1 to 2,515 characters with a mean of 390 characters and a median of 299, while word counts span 1 to 425 words with an average of 62 words per passage with a median of 48. Approximately 75% of passages contain fewer than 574 characters, indicating a concentration of concise, focused content suitable for single-fact retrieval.

The architecture consists of three primary components: document embedding and storage, semantic retrieval, and language model-based generation. The system was designed to support systematic experimentation with embedding dimensions, retrieval strategies, and prompting techniques. My experiments reveal that RAG provides substantial improvements over no-retrieval baselines, showing over a 124% increase in F1 scores, while 768-dimensional embeddings outperform 384-dimensional alternatives. Persona prompting consistently achieves optimal performance across configurations, whereas Chain-of-thought prompting shows severe degradation in traditional metrics due to answers being more verbose, despite improved answer relevancy scores. The advanced system with confidence scoring and priority-based context assembly achieved competitive performance with top-5 retrieval which is discussed later in the report.

System Architecture

The RAG system establishes a baseline question-answering pipeline without advanced enhancements.
The system implements standard RAG architecture with four processing stages. The document indexing phase processes passages using sentence-transformers to generate dense vector representations. Two embedding models provide different semantic capacity trade-offs: all-MiniLM-L6-v2 produces 384-dimensional vectors, while all-mpnet-base-v2 generates 768-dimensional vectors capturing richer semantic nuances. I used batch processing with size 64 to balance encoding throughput against GPU memory constraints.

For storage and retrieval, I used Milvus Lite implementing a three-field schema: an id field, a passage field containing the original text, and a field storing embeddings with the set dimensionality. The FLAT index provides exact nearest neighbor search through exhaustive comparison. Cosine similarity serves as the distance metric, normalizing for passage length variations and providing an intuitive score interpretation.

Query processing implements a straightforward pipeline: user questions undergo identical embedding encoding to ensure vector space consistency, similarity search retrieves top-k passages ranked by cosine distance, and retrieved contexts are concatenated before generation. The generation component employs Flan-T5-Base that has 248M parameters with sequence-to-sequence architecture, processing concatenated context and question through a maximum 512-token input window. Beam search decoding (beam_size=4) explores multiple generation paths, while temperature sampling (T=0.7) balances coherence against diversity.

Prompting Strategies

I evaluated two different prompting styles for this assignment. Persona prompting establishes a specific role identity with behavioral constraints focusing on accuracy, conciseness, and factual grounding, resulting in direct answers. Chain-of-thought prompting asks the model to articulate its reasoning process through intermediate steps before providing final answers, which improves transparency and logical coherence but generates significantly longer responses that perform poorly on lexical overlap metrics despite demonstrating sound reasoning.

- **Persona Prompting**
  You are a helpful and accurate assistant. Use the provided context to answer the question as accurately and concisely as possible. If the context doesn't contain enough information to answer the question, say so clearly. Do not make up information that is not in the context.

  Context: {context}

  Question: {query}

  Answer:

- **Chain-of-Thought Prompting**
  Think step by step to answer the question using the provided context. First, identify the key information in the context that relates to the question, then reason through to your answer. Only respond with the final concise answer, with no intermediate steps or reasoning in the final output.

  Context: {context}

  Question: {query}

  Let me think step by step:
  1. What is the question asking?
  2. What relevant information is in the context?
  3. How do I connect this information to answer the question?

  Answer:

Naive Implementation Limitations

The naive implementation shows several clear opportunities for enhancement. The system cannot clarify ambiguous questions or expand abbreviated queries, treating all user inputs as-is regardless of specificity. Retrieved contexts receive equal treatment without relevance confidence assessment, potentially elevating marginally-relevant passages to the same status as highly-relevant ones. The fixed context selection strategy applies uniformly across simple and complex queries, missing opportunities to adapt based on question difficulty or retrieval quality. Long contexts risk truncation when combined with prompts and questions, potentially losing critical information at arbitrary cutoff points. These limitations, while establishing a baseline, point toward specific enhancement opportunities that could address real-world deployment challenges.

Experimental Results and Evaluation

As a baseline to compare the efficiency of a RAG approach and also to justify the need for the overheads involved with a RAG system, the model was given the set of questions directly without any context. A simple prompt was also provided along with each question. This represents the model's parametric knowledge alone.

Baseline: No RAG (Flan-T5-Base, all-MiniLM-L6-v2)	
Exact Match	0.1328
F1 Score	0.2847

As seen above, the LLM scored pretty low, with the lack of context that would have been provided by the RAG system severely affecting the capability of the system to accurately answer questions.

The introduction of RAG with 384d embeddings and persona prompting (top-1 strategy) dramatically improved performance to EM: 0.329 and F1: 0.392, representing a 124% F1 improvement and demonstrating the substantial value of retrieval-augmented generation for this task.

384 Embedding Dimensions, Persona Prompting			
	top-1	top-3	top-5
Exact Match	0.3289	0.3028	0.1873
F1 Score	0.3924	0.3854	0.2555

RAGAs Evaluation	Top-1 (Top Strategy)
Faithfulness	0.6900
Answer Relevancy	0.7863
Context Precision	0.6200
Context Recall	0.4400

When using chain-of-thought prompting, we notice a major change. The traditional scoring metrics decline significantly, with exact match and F1 scores going down significantly. This is because the responses generated by the model with this prompt are more verbose, causing a mismatch with the direct answers required. However, the answer relevancy RAGAs metric goes up to 0.9012, showing that while the answers do not match the required formats of the test set, they are very relevant, with the context precision and recall also being higher.

384 Embedding Dimensions, Chain-of-Thought Prompting			
	top-1	top-3	top-5
Exact Match	0	0.0010	0
F1 Score	0.0575	0.0608	0.0566

RAGAs Evaluation	Top-1 (Top Strategy)
Faithfulness	0.5242
Answer Relevancy	0.9012
Context Precision	0.6200
Context Recall	0.4600

Now comparing this with the results from the 768d experiment, we see a 3% improvement in F1 scores, with the RAGAs metrics being significantly better. Top-1 turned out to be the best strategy here as well.

768 Embedding Dimensions, Persona Prompting			
	top-1	top-3	top-5
Exact Match	0.3191	0.2930	0.1732
F1 Score	0.4039	0.3800	0.2389

RAGAs Evaluation	Top-1 (Top Strategy)
Faithfulness	0.7553
Answer Relevancy	0.8286
Context Precision	0.8200
Context Recall	0.7800

768d with chain of thought showed similar results to the 384d test, showing that the prompting style results in responses that are more verbose, even with explicit instructions to only return the final answer. The RAGAs metrics show that even though the results are verbose, they are still contextually relevant to the question posed. Faithfulness is higher than with the persona prompt, with context precision and recall being marginally better as well.

384 Embedding Dimensions, Chain-of-Thought Prompting			
	top-1	top-3	top-5
Exact Match	0	0	0.0021
F1 Score	0.0578	0.0598	0.0527

RAGAs Evaluation	Top-1 (Top Strategy)
Faithfulness	0.6497
Answer Relevancy	0.8707
Context Precision	0.7200
Context Recall	0.5800

Comparative Embedding Dimension Analysis

Systematic comparison between 384d and 768d embeddings using top-1 retrieval and persona prompting revealed consistent performance advantages for higher-capacity embeddings.

Metric	384d	768d	Change
Exact Match	0.392	0.3191	-3%
F1 Score	0.3924	0.4039	+3%
Faithfulness	0.6900	0.7553	+9.5%
Answer Relevancy	0.7863	0.8286	+5.4%
Context Precision	0.6200	0.8200	+32.3%
Context Recall	0.4400	0.7800	+77.3%

The 768-dimensional model shows marginal EM decline but improved F1 and dramatic RAGAs metric improvements. The 32.3% context precision improvement indicates substantially better passage selection, while the 77.3% context recall improvement demonstrates that retrieved contexts better cover ground truth information. These semantic quality gains justify the 2x inference latency cost for production applications prioritizing answer accuracy and grounding.

Retrieval Strategy Evaluation

Evaluation across three retrieval strategies revealed consistent patterns favoring single-context retrieval. As seen in the results from individual experiments, this consistent pattern across embedding dimensions suggests that for this dataset's well-structured passages, the most relevant context typically provides sufficient information while additional contexts introduce noise through contradictory details, irrelevant tangential information, and token budget consumption reducing generation capacity.

Prompting Strategy Evaluation

Comparison between persona and chain-of-thought prompting revealed significant metric divergence. CoT prompting degraded traditional metrics by over 85% while improving answer relevancy by 11-15%. This reflects a fundamental metric misalignment: CoT generates valuable step-by-step reasoning that comprehensively addresses questions but creates minimal lexical overlap with ground truth answers like "1832". The results demonstrate that CoT provides transparency value not captured by overlap-based metrics. These results also highlight the importance of RAGAs values, as it showcases the improvements in relevance and faithfulness that would not have been captured by traditional metrics.

Statistical Significance

The sample size of 918 test queries provides sufficient statistical power for reliable conclusions. The consistency of retrieval strategy findings across both embedding dimensions (384d and 768d showing identical preference for top-1) strengthens confidence in generalizability within this dataset class.

Enhancement Analysis

The enhanced system addresses naive baseline limitations through two techniques: confidence-based adaptive context selection and context window optimization. These enhancements target retrieval quality and efficient context utilization without introducing excessive computational overhead or sacrificing system transparency. The prompt was also tightened up to add stronger checks to be concise and to the point.

Prompt:

```
You are an extremely knowledgeable and helpful assistant. {certainty_note}, answer the question using the provided context. Be extremely concise, only respond with the answer to the question, with no other commentary or information required. If the context doesn't fully answer the question, acknowledge this limitation and say 'I don't know'.
```

`certainty_note`:

`Based on highly relevant information`/`Based on available information with moderate confidence`

Confidence Scoring and Adaptive Context Selection

The confidence scoring mechanism calculates retrieval quality by averaging cosine similarity scores from vector search results, using this metric to adaptively select contexts. When confidence exceeds 0.75, indicating highly relevant top results, the system uses only the 2 most relevant passages, reasoning that quality exceeds quantity in high-confidence scenarios and additional contexts might dilute information with tangential details. When confidence falls at or below 0.75, suggesting no single passage comprehensively addresses the query, the system expands to 4 contexts to piece together answers from multiple partially-relevant sources. This approach provides transparency by modifying prompts to communicate confidence levels: high-confidence responses include "Based on highly relevant information" while lower-confidence responses note "Based on available information with moderate confidence."

Context Window Optimization

The context window optimization enhancement implements sentence-aware truncation to maximize information density within token constraints. Rather than arbitrary cutoffs that might sever sentences mid-thought or eliminate crucial concluding information, the system analyzes combined contexts sentence-by-sentence, preserving complete sentences up to a specified character limit. This ensures coherent, grammatically complete input to the generation model while respecting computational constraints and maximizing the utilization of available context space.

Performance Results and Analysis

The enhanced system with 768d embeddings achieved the following results

Advanced RAG Results			
	top-1	**top-3**	**top-5**
Exact Match	0.2723	0.3050	0.3104
F1 Score	0.3495	0.3972	0.4243
vs Naive Top-1	-14.7%, -13.5%	-4.5%, -1.6%	+3.5%, +7.5%

RAGAs Evaluation	Top-5 (Top Strategy)
Faithfulness	0.6987
Answer Relevancy	0.7588
Context Precision	0.6400
Context Recall	0.7800

The enhanced RAG system shows an unexpected trend when compared to the naive implementation. Performance improves with increasing context count, contrary to naive system patterns where top-1 excels. With top-5 retrieval, the enhanced system achieves competitive F1, beating the best F1 score set by the 768d Persona prompt experiment. RAGAs metrics show slight degradation, while remaining very close to the best values set by the naive implementation.

Implementation Challenges and Insights

I experimented with multiple enhancement options before choosing confidence scoring and context window optimization. I experimented with query reranking and query rewriting on a small sample of the test set. While results were promising, the time required to process all 918 queries was significant. Additionally, the choice of model also affects the improvements the enhancements result in, with Flan T5 Base not offering enough of a boost to justify the additional resource requirements.

The added enhancements significantly improved traditional metrics, proving that the strategy to adaptively select contexts based on confidence scores and also optimize context windows worked. As another test, I ran each enhancement independently without the other, and got performance scores that were very close to and sometimes even lower than the naive implementation. This combination of approaches gave the best improvement in performance. Interestingly, the enhanced system achieves its best performance with top-5 retrieval, suggesting that broader context collection compensates for per-passage information loss through volume.

Production Considerations

Optimal Configuration and Deployment Recommendations

Based on comprehensive evaluation, the recommended production configuration uses the enhanced RAG with all-mpnet-base-v2 (768 dimensions), top-5 retrieval and a strong persona prompt with added instructions. This configuration provides the best results using traditional metrics, while also being very close to the best RAGAs results observed. Another option could be the naive RAG with the same embedding model, top-1 retrieval with persona prompting. This configuration prioritizes semantic quality and answer grounding while maintaining computational efficiency through single-context retrieval.

Scalability

The current architecture can handle 3,200 documents, with 918 questions processed in around 390 seconds (6:30 average for top-1) using a T4 GPU on Google Colab. Scaling this solution could be done with changes to the index used, the implementation of a cache system like Redis for frequent queries, and the vector database being deployed across clusters instead of a local db as is the case with this assignment.

Limitations and Future Work

Dataset-specific findings may not generalize to collections with more fragmented passages requiring multi-document synthesis. The test set's relatively straightforward questions may not represent real-world query diversity that could include vague, multi-part, or contextually-dependent queries. Flan-T5-Base's 512-token context window also constrains information capacity. The enhanced system's performance suggests that context assembly strategies require careful dataset-specific tuning. Future iterations could explore importance-based rather than position-based extraction, potentially using extractive summarization or keyword-guided selection.

This assignment also brought to light metric limitations. Traditional overlap metrics heavily penalize linguistic variation while RAGAs depend on evaluation model capabilities. The divergence between traditional and semantic metrics highlights ongoing challenges in defining "good" performance for open-ended generation.

Key Insights and Recommendations

- The 124% F1 improvement from no-retrieval baseline demonstrates substantial value from context augmentation, validating the RAG approach for this task.
- 768d embeddings justify the increased latency latency cost through 9.5% faithfulness improvement and 32.3% context precision gain, critical for systems prioritizing answer grounding.
- Top-1 retrieval for all naive RAG implementations consistently outperforms multi-context strategies for well-structured passages, suggesting precision exceeds recall for this dataset class.
- Persona prompting was optimal for traditional metrics while CoT provides transparency value that was not captured by these evaluation metrics. This means that application requirements should drive selection of comparison metrics.
- Advanced features do require extensive tuning before deployment. The inverted performance pattern where the enhanced system prefers top-5 vs naive top-1 also indicates fundamental operating point differences between the two approaches, which could require further investigation.

Appendix

AI Usage:

AI was used for generating code snippets: To improve my basic metric calculation function

- Tool Name/Version: ChatGPT 5
- Purpose: To streamline and improve my metric calculation logic.
- Input Query: Streamline this function that calculates the F1 score and Exact Match, return the results exactly as I have done in my function.
- AI Output Modification: I tested and verified every model output before integrating it into my workflow.
- Verification: Manually computed F1 and Exact Match on a set of 10 Questions to validate changes.

AI was used to ideate improvements to the naive RAG implementation

- Tool Name/Version: ChatGPT 5
- Purpose: For ideation.
- Input Query: Help me understand how I can implement enhancements to a naive RAG implementation. How does the enhancement work? What combination of improvements might give good results with Google Flan on a QA task?
- AI Output Modification:I did not use the model output directly, I worked with the model to find a few improvements that I worked on later.
- Verification: No verification needed.

AI was used to debug issues around RAGAs implementation.

- Tool Name/Version: ChatGPT 5
- Purpose: To debug coding issues.
- Input Query: I see this error when implementing RAGAs evaluation. Help me fix it.
- AI Output Modification: I worked with the model to find the bug, and used the proposed solution in my workflow
- Verification: The bug was fixed.

AI was used to improve the final results file I store.

- Tool Name/Version: ChatGPT 5
- Purpose: To store better results after experimentation.
- Input Query: I have saved the following values after running my model. I would like more metrics for comparison later. Help me improve my function.
- AI Output Modification: I tested and verified every model output before integrating it into my workflow.
- Verification: The results as displayed in the notebook were saved in the JSON file as specified.

I used Grammarly for writeup and report grammar correction.

No AI was used to implement the core parts of the RAG system.

Technical Specifications

- All experiments were run on Google Colab using a T4 GPU.
- Dataset: RAG Mini Wikipedia - 3,200 passages, 918 test queries
- Hyperparameters: Batch size 64, beam size 4, temperature 0.7

Reproducibility

Execute all cells in the provided notebook to perform EDA, create the RAG system (Naive and Advanced), run model inference, perform traditional evaluation and also advanced RAGAs evaluation.