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 148% 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

Note:
Evaluation was performed using all 918 queries for traditional metrics. I used 50 questions and their outputs to generate RAGAs metrics, as doing it for all 918 queries is computationally expensive, and would also use up a lot of OpenAI credits. I chose 50 as it was a good middle ground, resulting in measurable and comparable metrics while balancing costs and computation needs.

To establish a meaningful baseline and justify the computational overhead of implementing a RAG system, I first evaluated Flan-T5-Base's performance without any retrieval augmentation. This configuration tests the model's parametric knowledge alone. That is its ability to answer questions based solely on information encoded in its weights during pre-training.

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

These results reveal significant limitations in the model's standalone question-answering capability. An exact match rate of 13.28% means the model produces precisely correct answers for only about 1 in 8 questions. The F1 score of 0.2847, which measures partial credit through token overlap, indicates that even when answers aren't perfectly correct, they share less than 30% similarity with ground truth responses on average. This poor performance stems from the model's inability to access specific factual information beyond what was memorized during training, demonstrating why external knowledge retrieval is essential for factual question answering.

The introduction of retrieval-augmented generation with 384-dimensional all-MiniLM-L6-v2 embeddings produced great improvements.

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

The top-1 configuration represents a 148% improvement in exact match and a 38% improvement in F1 score compared to the no-RAG baseline. This validates the core value proposition of RAG that providing models with relevant external context dramatically improves factual accuracy.

The RAGAs metrics provide deeper insight into system quality beyond simple answer correctness: Faithfulness measures grounding in provided context, indicating that roughly 69% of answer content can be directly attributed to retrieved passages while 31% represents hallucination or inference beyond the context. Answer Relevancy evaluates whether answers actually address the question asked, confirming the model generally understands queries and responds directly. Context Precision measures the proportion of retrieved passages that are actually relevant, suggesting about 2 in 3 passages contain useful information while 1 in 3 is marginally relevant or irrelevant. Context Recall evaluates whether retrieved contexts cover all information needed to answer completely, with the relatively low score indicating that retrieved passages often miss important supporting details that would enable more comprehensive answers.

Chain-of-thought (CoT) prompting produces what initially appears to be a major failure, with exact match scores near zero and F1 scores dropping 85% compared to persona prompting. However, this interpretation is misleading as the problem lies not with answer quality but with evaluation metric incompatibility.

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

CoT prompts instruct the model to reason through questions step-by-step before answering. Despite explicit instructions to provide only the final answer, Flan-T5-Base consistently generates verbose responses. Traditional metrics find minimal lexical overlap, severely penalizing the response.

The RAGAs metrics however tell a different story. Answer Relevancy increases to 0.90 indicating that the lengthy explanations are highly relevant to the questions asked. Context Precision remains stable at 0.62, showing retrieval quality is unaffected by prompt style. Context Recall improves to 0.46 suggesting the reasoning process helps the model extract more information from context.

This reveals a fundamental evaluation challenge: CoT generates answers that are arguably superior in transparency and reasoning quality but perform worse on metrics designed for concise responses. The 15% improvement in answer relevancy despite 85% degradation in F1 score demonstrates that different prompting strategies optimize for different definitions of "good" performance.

The 768-dimensional all-mpnet-base-v2 embeddings show some trade-offs. Exact match actually decreases slightly by 3%, but F1 score improves 3%. More significantly, RAGAs metrics show substantial gains.

Faithfulness improves 9.5%, meaning answers are better grounded in retrieved context with less hallucination. Answer Relevancy improves 5.4%, indicating more directly relevant responses. Context Precision improves 32.3%, showing dramatically better passage selection. The retrieval system now identifies relevant passages 4 out of 5 times rather than 3 out of 5. Context Recall improves 77.3%, indicating retrieved passages now cover most of the information needed for complete answers rather than less than half.

The 768-dimensional model's superior semantic representation enables more precise retrieval. While encoding time does increase, the quality improvements justify this latency cost for production systems prioritizing answer accuracy and grounding over response speed.

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

The pattern observed with 384d embeddings repeats with 768d with CoT prompting. CoT produces verbose reasoning that performs poorly on traditional metrics but shows strong answer relevancy. Interestingly, faithfulness is lower with CoT, suggesting that the step-by-step reasoning process occasionally introduces inferences beyond the provided context.

768 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%

This comparison clarifies the value proposition of higher-dimensional embeddings. The slight exact match decrease likely results from random variation given the small sample size of 918 queries, while the consistent improvements across all semantic metrics demonstrate real quality gains. The dramatic context recall improvement is particularly significant, as 768d embeddings retrieve passages that contain 78% of necessary information versus only 44% for 384d embeddings.

Retrieval Strategy Evaluation

The performance degradation with top-3 and especially top-5 retrieval warrants detailed analysis. While retrieving more passages to provide more potential information intuitively seems beneficial, several factors could explain why additional context hurts performance in this configuration:

With Flan-T5-Base's 512-token context window, adding passages means each passage receives less representation in the concatenated context. Critical information from the most relevant passage may be pushed further from the question. Lower-ranked passages could also contain tangential or contradictory information. And finally, Language models must distribute attention across all provided contexts. More passages mean the model's attention to the most relevant information becomes diluted by less relevant material.

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

With 918 test queries, the results provide strong statistical power. The consistency of findings across both embedding dimensions, with both showing top-1 > top-3 > top-5 for persona prompting strengthens confidence that these patterns reflect genuine system behavior rather than random fluctuation.

However, generalization limitations must be acknowledged. The RAG Mini Wikipedia dataset consists of well-structured, focused passages averaging 62 words. Real-world document collections often contain longer, more fragmented text where multi-passage synthesis becomes necessary. The evaluation also uses relatively straightforward factual questions rather than complex multi-part queries or requests requiring synthesis across topics.

Enhancement Analysis

The enhanced system addresses naive baseline limitations identified in the naive RAG baseline through two complementary techniques: confidence-based adaptive context selection and sentence-aware context window optimization. These enhancements target the core challenge of balancing information quantity against quality, recognizing that not all queries benefit equally from the same retrieval strategy and that poorly-truncated context can be worse than shorter but coherent context. Unlike approaches requiring additional model calls like query rewriting and reranking, these enhancements add minimal latency while directly addressing context assembly quality.

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'.
```

The certainty note dynamically adapts based on retrieval confidence: "Based on highly relevant information" signals strong confidence when average similarity exceeds 0.75, while "Based on available information with moderate confidence" communicates uncertainty when similarity is lower. This transparency allows users to calibrate trust appropriately.

.

The "Be extremely concise" instruction strengthens the directive from the naive prompt's "accurately and concisely," addressing Flan-T5-Base's tendency toward verbosity. The "only respond with the answer" clause attempts to prevent explanatory preambles. The explicit "I don't know" fallback encourages the model to acknowledge uncertainty rather than hallucinating when context is insufficient.

Confidence Scoring and Adaptive Context Selection

The confidence mechanism calculates retrieval quality by averaging cosine similarity scores across retrieved passages. Cosine similarity ranges from -1 to 1, with higher values indicating stronger semantic alignment between query and passage embeddings. By averaging scores from top-k results, the system obtains a holistic measure of retrieval quality that considers both the best match and the quality of alternative passages.

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.

Context Window Optimization

Naive concatenation of retrieved passages risks arbitrary mid-sentence truncation when the combined text exceeds token limits. 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 approach maximizes information density within constraints, every included character contributes to complete, grammatically coherent thoughts rather than partial fragments that provide minimal value. It also prioritizes information from top-ranked passages, since sentences are included in retrieval order.

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 results reveal an unexpected pattern: the enhanced system shows inverse performance characteristics compared to the naive implementation. While naive RAG performed best with top-1 retrieval and degraded with additional contexts, the enhanced system performs worst with top-1 and progressively improves with top-3 and top-5.

This inversion provides insight into how the enhancements fundamentally alter system behavior. The confidence-based selection mechanism rarely triggers high-confidence mode because the 0.75 threshold is stringent. When working with top-1 retrieval in the enhanced system, low confidence usually triggers, but the system is constrained to only one available passage. This mismatch between intended behavior and actual capability creates suboptimal performance.

With top-5 retrieval, the enhanced system has flexibility to execute its adaptive strategy: it can use 2 passages in high-confidence scenarios or 4 passages in low-confidence scenarios, selecting from the 5 available options. The sentence-aware truncation also benefits from larger passage pools as it can construct more coherent, information-dense contexts by cherry-picking complete sentences from multiple sources rather than being constrained to a single passage's structure.

The enhanced system's best F1 score of 0.4243 exceeds the naive system's best of 0.4039 by 5%, validating that the enhancements provide measurable improvement when given sufficient retrieval breadth to operate as designed. However, RAGAs metrics show slight degradation. This suggests a quality-versus-correctness trade-off, the enhanced system produces answers with better lexical overlap with ground truth but slightly less semantic grounding and relevance.

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.

The decision to pursue confidence scoring and context optimization reflected certain constraints: these techniques add minimal computational overhead while directly addressing the most severe limitations of the naive system. The choice also reflects model constraints, with Flan-T5-Base's limited capacity meaning that sophisticated enhancements show diminishing returns, whereas more powerful models like GPT-3.5 or Claude might benefit substantially from query rewriting or reranking.

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. Only the combination of both techniques achieved the 5% improvement over naive baseline, demonstrating that the enhancements address complementary weaknesses that must both be resolved for measurable gains.

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 148% 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.