Technical Appendix

Appendix A: AI Usage Log

Tool: Claude.ai (Sonnet 4.5)

1. Technical Report Writing and Organization

- **Purpose:** Structuring experimental results into cohesive technical report sections
- **Input:** "I have outputs such as (F1 scores, EM scores, RAGAs metrics), give me a nice structure to put them in my report."
- **Output Usage:** Used as foundation for final report with manual editing and verification.
- **Verification:** Read the entire report to ensure flow was smooth.

2. Code Output Formatting Enhancement

- **Purpose:** Improving test result readability with formatted tables and visual separators
- **Input:** Basic print statements for displaying retrieval results and citations
- **Output Usage:** Implemented box-drawing characters for citation tables and added visual separators
- **Code implemented:**

```
print("┌──────┬──────────┬──────────┐")
print("│ Rank │ Chunk ID │ Relevance │")
print("├──────┼──────────┼──────────┤")
```

- **Verification:** Tested in notebooks; confirmed cosmetic improvement with no functional changes

3. Evaluation Progress Tracking

- **Purpose:** Adding timestamp logging and progress indicators for long-running evaluations
- **Input:** Sequential evaluation loops without progress feedback
- **Output Usage:** Added datetime tracking and status messages
- **Code implemented:**

```
start_time = datetime.datetime.now()
print(f"→ Evaluating {name} on {N_SAMPLES} samples - started at
{start_time.strftime('%H:%M:%S')}")
```

- **Verification:** Improved monitoring of evaluation progress; no impact on results

4. ThreadPoolExecutor Implementation

- **Purpose:** Implementing parallel processing for RAGAs evaluation following TA recommendation
- **Input:** TA's suggestion to use ThreadPoolExecutor for concurrent API calls, original sequential evaluation code
- **Output Usage:** Implemented parallel processing with 10 workers.
- **Code implemented:**

```
with ThreadPoolExecutor(max_workers=10) as executor:
   futures = {executor.submit(process_question, i, system_type): i
           for i in range(len(eval_questions))}
```

- **Verification:** Confirmed time reduction while maintaining identical result accuracy; handled NaN values with np.nanmean()

5. RAGAs Metrics Interpretation

- **Purpose:** Understanding and explaining RAGAs evaluation metrics (faithfulness, answer relevancy, context precision/recall)
- **Input:** RAGAs documentation, evaluation results showing metric scores

- **Output Usage:** Generated explanations for what each metric measures and why improvements occurred
- **Verification:** Cross-referenced with class notes

6. Statistical Calculations

- **Purpose:** Computing relative improvement percentages for report
- **Input:** Raw metric scores (naive vs enhanced system performance)
- **Output Usage:** Calculated improvement percentages (e.g., faithfulness: 56.9%, context recall: 75%)
- **Verification:** Manually verified calculation formulas: (Enhanced - Naive) / Naive × 100%

Appendix B: Technical Specifications

Development Environment

- Platform: Google Colab (free tier)
- GPU: NVIDIA T4 (when available)
- Python: 3.10.x

Core Libraries

- sentence-transformers: 2.x
- transformers: 4.x
- faiss-cpu: 1.x
- pymilvus: 2.x
- datasets: 2.x
- ragas: 0.x
- langchain-google-genai: 0.x
- google-generativeai: 0.x
- pandas, numpy, torch

Model Specifications

Embedding Models:

- all-MiniLM-L6-v2 (384 dimensions)
- all-mpnet-base-v2 (768 dimensions)

Reranking:

- cross-encoder/ms-marco-MiniLM-L-6-v2

Language Models:

- google/flan-t5-small (60M parameters) - Naive system
- google/flan-t5-base (220M parameters) - Enhanced system, query rewriting

Evaluation Judge:

- Google Gemini 2.5 Pro (RAGAs evaluation only)

Appendix C: Reproducibility Instructions

Repository Structure

Five Jupyter notebooks executed sequentially:

1. Data_Exploration.ipynb - Dataset analysis
2. Naive_RAG_&_Evaluation.ipynb - Steps 2-3
3. Parameter_Comparison.ipynb - Step 4
4. Advanced_RAG.ipynb - Step 5
5. Advanced_Evaluation_RAGAs.ipynb - Step 6

Prerequisites

- Google Colab account
- Google Gemini API key (Step 6 only)

Execution Instructions

Steps 1-5:

1. Open notebook in Google Colab
2. Run all cells sequentially
3. Dependencies install automatically from first cell
4. Results display in console output

Step 6 (RAGAs Evaluation):

1. Set GEMINI_API_KEY variable in notebook
2. Run all cells
3. Results save to step6_outputs/ragas_results.csv

Issues

- Config file labels MPNet as "512D" but actual dimension is 768D (model's native output)
- 2-5% of RAGAs samples may return NaN due to API timeouts (handled with np.nanmean)
- Token limit warnings appear during Top-5 evaluation (automatic truncation, no impact on results)