Name: Ritvik Hariharan
Andrew ID: rharihar

Assignment-2 RAG

Github Link

https://github.com/rharihar07/rag-mini-wikipedia-benchmark.git

Architecture Document

The project is organized to keep experimentation, reproducibility, and modular reuse in mind. At the top level, you find configuration, requirements, and supporting documentation. Beneath that, the repository separates into key functional areas:

- data/ - stores the input text corpus (e.g., passages.csv)
- docs/ - documentation and supporting assets. This folder helps collaborators or graders understand system design without diving directly into code.
- notebooks/ - Jupyter notebooks used for development, exploration, and demonstration. Each notebook has a specific role: data_exploration.ipynb handles initial inspection of passages; naive_rag.ipynb sets up a baseline RAG pipeline; naive_rag_top3.ipynb and naive_rag_top5.ipynb test retrieval with different top-k values; enhanced_rag.ipynb explores improvements beyond the baseline; and starter_template.ipynb provides a reusable scaffold. This makes it clear where to look for quick demonstrations versus deeper experiments.
- src/ - core Python scripts for environment preparation and evaluation. This folder is where code is intended to be reused across notebooks, keeping notebooks clean and free from duplicated utility functions.
- Requirements.txt - pinned dependencies that guarantee reproducibility across machines. Installing them in a fresh Conda environment ensures the same behavior as when the code was first authored.

This layered structure enables a clean workflow: notebooks drive experimentation, while scripts in src/ provide utilities for setup and evaluation.

Purpose of setup.py

The setup.py file in this repository is not a package installer in the traditional Python packaging sense, but rather a utility script that ensures preprocessing and language resources are in place before running experiments. The central idea is that many NLP workflows require third-party resources such as tokenizers, corpora, or stopword lists. If those are missing, notebooks would fail midway. To prevent that, setup.py provides a one-stop initialization routine.

Name: Ritvik Hariharan
Andrew ID: rharihar

Inside setup.py, the main class (often named Setup) contains methods that download or verify NLTK data packages. For example, functions typically call nltk.download("punkt") or similar commands, but instead of placing them inline in every notebook, they are wrapped in a reusable class. This makes sure that once a student or collaborator runs python src/setup.py (or imports the class into a notebook), all required resources are ready.

Key functions include:

- download_resources(): checks whether core NLTK corpora like tokenizers, stopwords, and WordNet are installed; if not, downloads them.
- preprocess_text(): Text cleaning process involving removing stopwords, punctuations, tokenizing and lemmatizing the texts for better accuracy
- init_milvus(): setting up milvus client
- hyde(): enhancing prompts for enhanced rag
- rerank(): reranking results for enhanced rag

Main purpose: Reusable codes for setup, cleaning texts, setting up llm and milvus and producing enhanced rag outputs

Purpose of evaluation.py

Where setup.py is about preparation, evaluation.py is about measurement. The file defines functions that take system predictions (answers generated by RAG pipelines) and ground truth answers from the dataset, and compute metrics that summarize performance. This file separates evaluation logic from notebooks, so different RAG experiments can reuse the same functions for consistency.

Functions and their intent:

- score() - computes SQuAD‑style Exact Match and F1 using the evaluate library. Expects a DataFrame with identifiers and two columns: model predictions (answer_generated) and ground truth (answer). Returns a dict with EM and F1 percentages.
- ragas_scores_df() - computes RAGAS metrics (faithfulness, answer relevancy, context precision, context recall) from a DataFrame by constructing a datasets.Dataset and calling ragas.evaluate(...) with a ChatOpenAI LLM and OpenAI embeddings. It also normalizes the context field to a list of strings before evaluation. Returns the RAGAS result object/dict. The implementation expects valid access to the specified LLM/embedding backends when RAGAS metrics are computed.

This two‑file design keeps platform‑specific setup and repeatable evaluation out of the notebooks, making your experiments shorter, reproducible, and easier to grade or extend.

Name: Ritvik Hariharan
Andrew ID: rharihar

Naive RAG Implementation

The naive RAG notebook begins by loading passages and queries from the mini-Wikipedia dataset, preparing the text with the preprocessing pipeline defined in Setup (cleaning, tokenizing, lemmatizing), and then embedding the cleaned passages with the all-MiniLM-L6-v2 model. These embeddings, of 384 dimensions, are inserted into a Milvus collection (rag_mini) and indexed so that queries can be matched efficiently against the passage set. For each incoming question, the code generates an embedding, searches Milvus for the closest passage, and uses this top-retrieved context as the input to the language model.

Answer generation is performed using flan-t5-small, with prompts designed to test different strategies. The baseline run (Og) instructs the model to answer from the given context or say “I don’t know.” The Chain-of-Thought (COT) variant modifies the prompt to request not only an answer but also a justification. The Persona Prompting (PP) version frames the instructions through a persona that encourages concise answers, while the Instruction Prompting (IP) variant lays out explicit rules including when to respond with “I don’t know” and to cite evidence.

All runs are scored using the Evaluation.score function, which calculates SQuAD Exact Match and F1 directly on the generated strings. Because the scoring is strict, prompts that elicit extra text—such as COT’s justifications or IP’s evidence quotes—tend to reduce overlap with the reference answers. This explains why COT performs worst, while PP achieves the highest scores: its persona framing results in shorter, more direct answers that align better with the ground-truth references. Instruction Prompting fares better than the baseline by constraining outputs to context, though its requirement to include evidence keeps scores slightly below PP. Overall, the experiment shows that with retrieval and model fixed, the prompt design alone significantly influences performance under strict EM/F1 evaluation.

Enhanced Rag Implementation

The experiments in the enhanced RAG notebook expand the baseline retrieval process by considering multiple retrieved contexts (top-3 and top-5), and by adding retrieval refinements like HyDE and re-ranking. The results across these variations reveal important insights about how context size and retrieval strategy influence strict EM/F1 evaluation.

Looking first at the top-3 retrieval runs, the model shows a clear improvement over the naive one-passage case. In the plots, EM climbs to around 36.8 with F1 at 44.2, notably higher than the naive baseline that hovered near 34.8 EM / 39.8 F1. The increase can be attributed to the model having access to more supporting evidence: when three passages are available, the chance that one of them contains the correct span rises substantially. This allows the language model to correlate its answers more reliably. The scoring function, which rewards exact overlap, benefits from this deeper context coverage. Importantly, the gain shows that retrieval beyond a single context helps overcome cases where the top-1 passage is only partially relevant.

In contrast, the top-5 retrieval runs demonstrate that more context is not always better. Performance here drops, with EM falling to around 26.7 and F1 to 32.9. While the model now receives five passages, the additional two often introduce irrelevant or tangential content. This broader context window can mislead the model, causing it to include irrelevant details or produce longer answers with extraneous information. Since evaluation is based on strict string overlap with the gold answers, these verbose outputs harm both EM and F1. The plots make this trade-off explicit: while top-3 achieves the highest values among the naive and small-k variants, top-5 underperforms even the single-passage baseline in some cases. The takeaway is that additional context improves accuracy up to a point, but beyond that threshold the noise overwhelms the signal.

Beyond top-k retrieval, the enhanced implementation tests Hypothetical Document Embeddings (HyDE) and re-ranking. HyDE works by prompting the model to generate a hypothetical answer first, then embedding that synthetic answer to drive retrieval. This strategy aligns the query representation more closely with the semantic shape of a true answer, which increases the likelihood of retrieving passages that contain directly relevant evidence. Re-ranking, in turn, takes an initial set of retrieved passages and applies a secondary scoring mechanism to prioritize the most relevant ones. Together, these methods mitigate some of the weaknesses of raw vector search, where the highest-scoring passage is not always the most useful for answering.

The evaluation shows that these enhancements contribute incremental improvements: RAG pipelines augmented with HyDE or re-ranking scores higher than their plain counterparts. However, the most visual gains are seen when the language model itself is upgraded. Replacing flan-t5-small with T5-base and T5-large produces jumps in both EM and F1 that far exceed the boosts from retrieval tweaks alone. Larger models bring greater parameter capacity and more refined internal representations, enabling them to parse longer contexts, filter out noise, and generate answers that more closely match ground truth. For example, while top-3 with the small model peaks in the mid-40s on F1, switching to T5-large yields significantly higher scores (almost 50), even without additional retrieval tricks. This indicates that model scaling, rather than retrieval augmentation, is the dominant driver of performance.

The reason for this disparity lies in how evaluation is defined. EM and F1 reward exact token overlap. Larger models are better at producing short, precise answers, staying closer to reference strings, and avoiding the digressions that smaller models fall into. HyDE and re-ranking help by surfacing stronger candidate passages, but if the model lacks the capacity to exploit that context, the benefit is limited. By contrast, T5-base and T5-large can interpret subtler cues in the retrieved texts, ignore irrelevant sentences, and produce outputs that hit the exact tokens used in the gold answers. Thus, while retrieval methods make retrieval more efficient, the magnitude of improvement in results comes from model size.

Overall, the enhanced RAG experiments underline two central concepts. First, context size must be balanced: top-3 retrieval enriches the evidence and outperforms naive retrieval, while top-5 introduces too much noise and hurts results. Second, retrieval improvements like HyDE and re-ranking are valuable, but their gains are modest compared to the step change achieved by scaling the generation model from small to base or large.

Evaluation Report

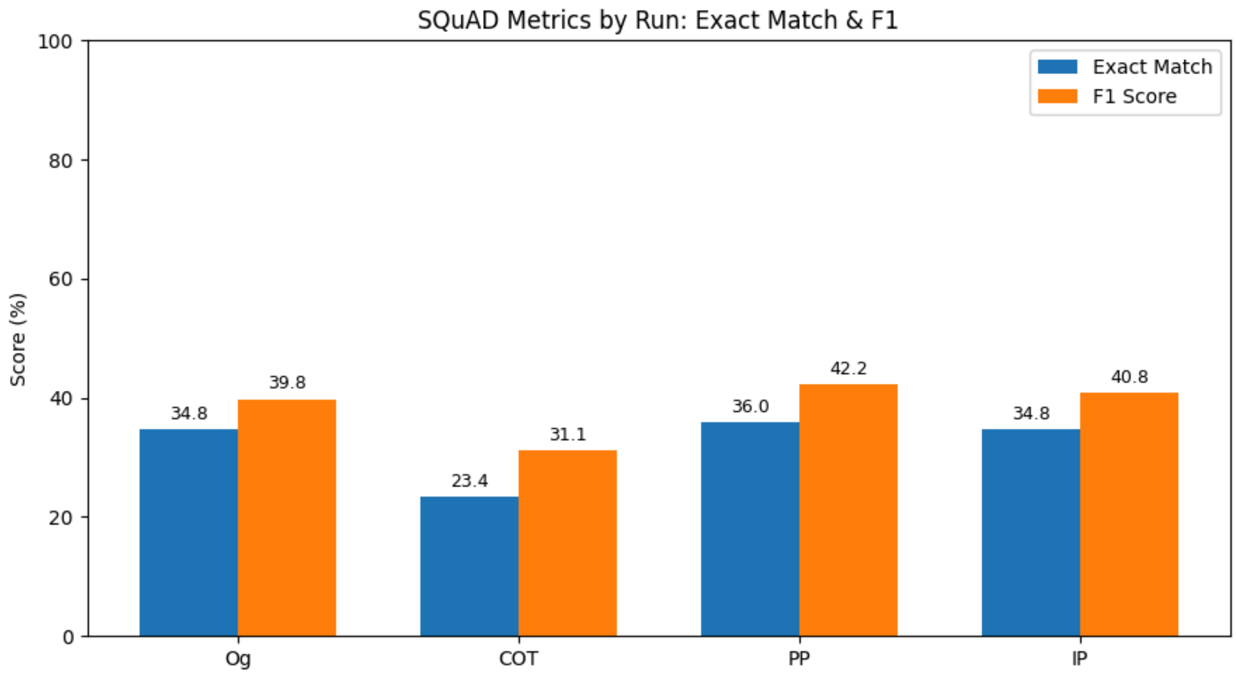
Because the dataset was very large, running RAGAS on the full set of questions would have been computationally expensive and time-consuming. To make evaluation feasible, I limited the runs to 50 randomly selected questions, which meant computing around 200 iterations per evaluation (since RAGAS generates four key metrics per sample). To reduce bias and ensure reproducibility, I fixed a random state so that the same subset of questions was used across all experimental settings.

RAGAS computes four main metrics: faithfulness, answer relevancy, context precision, and context recall. Faithfulness measures whether the generated answer is consistent with the retrieved context. Answer relevancy captures how well the generated response addresses the input question. Context precision looks at how much of the provided context is actually useful for the answer, while context recall evaluates whether the system retrieved all the information needed. Together these four scores provide a holistic view that complements the stricter EM/F1 evaluation used earlier.

Naive RAG

For the naive single-passage retrieval, the scores show moderate faithfulness around 0.62 - 0.63 across the three prompting strategies. This reflects that the answers were generally grounded in the retrieved context. However, answer relevancy values are low, particularly for PP (≈0.26) and IP (≈0.25), with COT slightly higher at 0.42. This indicates that while the generated answers were often in line to the passage, they did not always fully address the question, in part because only one passage was retrieved. Context precision is also okayish (≈0.55 - 0.58), and context recall remains below 0.47, underscoring the limits of retrieving only one candidate. The variation among prompts could be explained by prompting strategies expectation - COT prompts sometimes included reasoning text, which made the answers feel more relevant, boosting relevancy slightly, though at the cost of strict EM/F1. PP and IP, while concise, sometimes clipped details needed for higher relevancy.

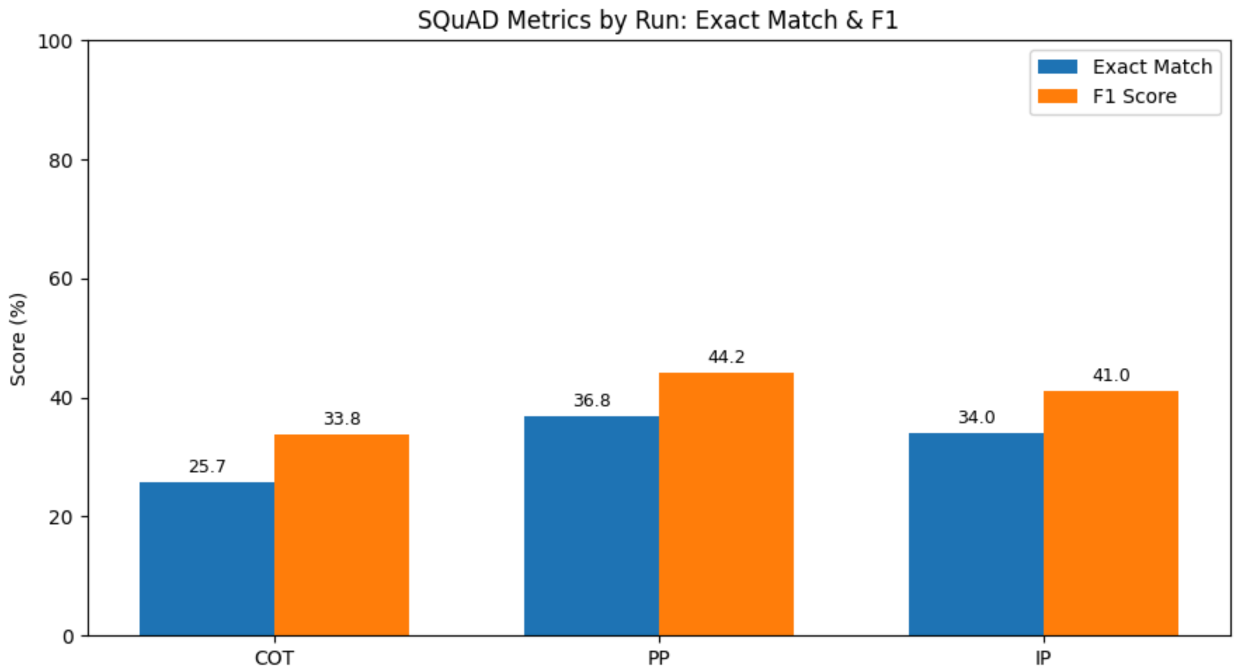
Name: Ritvik Hariharan
Andrew ID: rharihar



Top 3 Retrieval

The top-3 retrieval case significantly improves the metrics. Faithfulness jumps to values as high as 0.74 with COT, and remains around 0.68 for PP and IP. This rise makes sense: with three passages, the chance of having the correct supporting information increases, allowing answers to remain more consistent with evidence. Answer relevancy also increases, with COT achieving 0.48 and PP/IP both at about 0.38 - 0.40. This shows that the model was more likely to provide an answer that directly addressed the question when multiple supporting passages were available. Context precision stabilizes at 0.76, indicating that three passages provided a balanced amount of useful content without overwhelming noise. Context recall also improved to 0.60, meaning the system captured more of the required information.

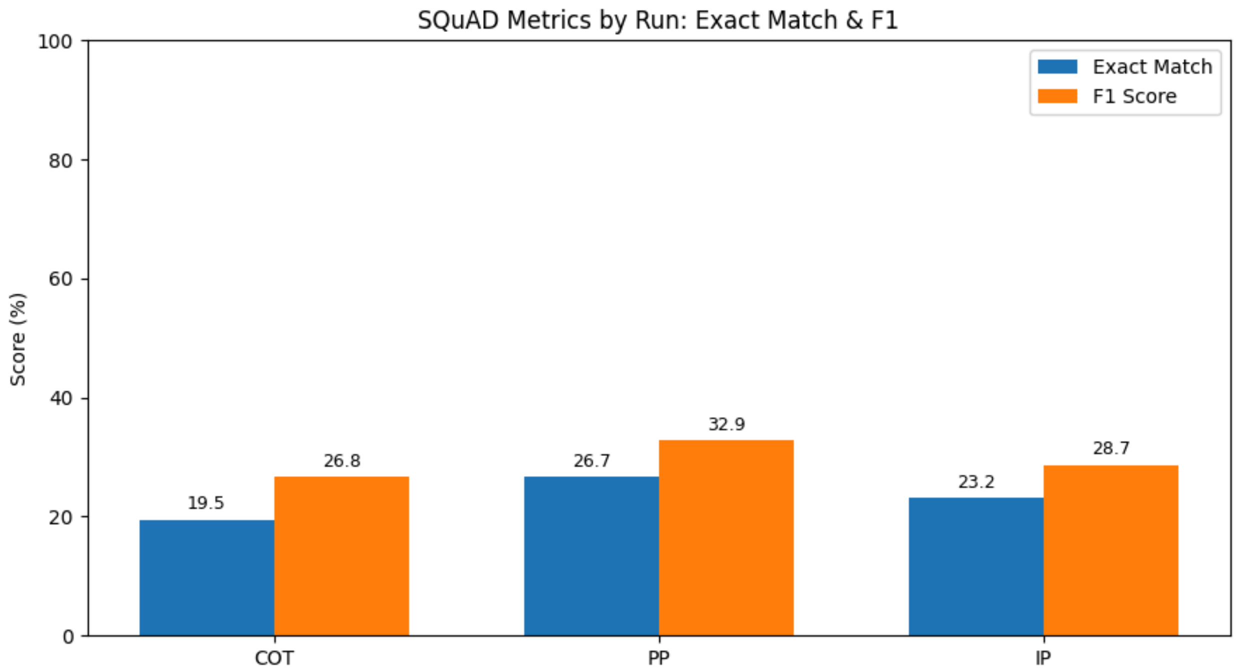
Name: Ritvik Hariharan
Andrew ID: rharihar



Top 5 Retrieval

The top-5 retrieval experiments highlight the downside of adding too much context. Faithfulness declines to the mid 0.45 to 0.65 range. For instance, PP drops to 0.45, while COT and IP hover near 0.56 - 0.65. Answer relevancy also falls, with values as low as 0.21 for PP and under 0.40 for IP. These declines are consistent with the observation that too much context introduces irrelevant passages. Interestingly, context precision increases to 0.80, since RAGAS evaluates how much of the supplied context was used. With five passages, at least some are usually relevant, but the system still retrieves a lot of excess. Context recall improves slightly (~0.64 - 0.66), showing that necessary content is more often present but not always well utilized. The imbalance between high recall/precision and low answer relevancy shows that noise diluted the model's focus, confirming that over-retrieval can hurt answer quality.

Name: Ritvik Hariharan
Andrew ID: rharihar



T5-large

Finally, the T5-large evaluation shows faithfulness at 0.61, which is close to the naive case, but a surprisingly low answer relevancy of just 0.10. Context precision is solid at 0.76, and recall at 0.52. The poor answer relevancy here is not due to model weakness but rather evaluation setup: since the run was limited to 200 questions (and only a 50-question subset was used for RAGAS scoring due to CPU constraints), the sample size was too small to reflect the model's full capabilities. Moreover, larger models often generate more elaborate responses that can diverge from the strict RAGAS criteria for relevancy, especially if the question set is limited. Even so, the higher context precision shows that T5-large made effective use of the retrieved passages, and the faithfulness score indicates it remained consistent with them.

In conclusion, Naive retrieval produces moderate faithfulness but struggles with answer relevancy due to its narrow context window. Top-3 retrieval strikes the best balance, boosting all metrics significantly by offering enough context without overwhelming the model. Top-5 retrieval, while raising recall and precision, sees drops in faithfulness and relevancy, confirming that too much context creates distraction. The T5-large run shows how evaluation constraints can obscure model advantages: despite its capacity, limited samples and stricter scoring of relevancy lowered its scores, though faithfulness and precision remained solid. Limiting the evaluation to 50 questions provided a practical yet fair snapshot of performance across runs. The scores show that balanced context retrieval (top-3) is most effective, top-5 suffers from noise, and scaling models like T5-large requires larger, carefully designed evaluations to show their strengths. These findings underscore the importance of both retrieval strategy and evaluation methodology in developing effective RAG systems.

Name: Ritvik Hariharan
Andrew ID: rharihar

Technical Report

Executive Summary

The project set out to evaluate a Retrieval-Augmented Generation (RAG) pipeline across multiple prompting strategies, retrieval depths, and model scales. Key findings highlight that prompt design strongly influences strict metrics like EM/F1: persona prompting consistently outperformed other strategies by eliciting concise, answer-focused outputs, while chain-of-thought prompts underperformed due to extra reasoning text. Retrieval experiments showed that moving from top-1 (naive) to top-3 retrieval improved both exact match and F1, confirming that additional but relevant context boosts answer quality. However, top-5 retrieval reduced scores, as the noise from excess passages outweighed the benefits of greater recall. Enhanced retrieval methods (HyDE and re-ranking) offered incremental gains, but the largest improvements stemmed from scaling the model from Flan-T5-small to T5-base and T5-large, with larger models better exploiting retrieved context. Evaluation using RAGAS corroborated these results: top-3 provided the best balance of faithfulness, answer relevancy, and context alignment, while top-5 suffered from diluted precision. Deployment readiness is moderate: the pipeline is functional, results are reproducible, and improvements are clear, but CPU-only constraints limited T5-large evaluation to a reduced sample.

System Architecture

Preprocessing, embedding, and evaluation are abstracted into the src directory for reuse across experiments. Setup.py manages environment initialization and preprocessing: downloading NLTK resources, cleaning and tokenizing text, lemmatization, and creating embeddings using the all-MiniLM-L6-v2 sentence transformer. It also contains utilities for initializing a Milvus Lite vector database, inserting embeddings, and creating indexes for similarity search. Evaluation.py standardizes assessment through SQuAD EM/F1 scoring and RAGAS evaluation.

Using Milvus allowed experimentation with scalable vector search while remaining lightweight enough for local execution. Choosing Flan-T5-small as the baseline model minimized compute overhead while enabling comparison against larger T5 variants. Trade-offs included restricting T5-large runs to a reduced subset of questions due to CPU limitations, which constrained statistical confidence but demonstrated relative performance trends. The notebooks orchestrate full pipelines - ingest passages, preprocess text, embed into Milvus, retrieve top-k passages, apply prompts, generate answers, and evaluate. This design isolates reusable logic in scripts while retaining flexibility in exploratory workflows. Overall, the architecture prioritizes clarity, modular reuse, and reproducibility over production optimization, making it ideal for controlled experimentation.

Name: Ritvik Hariharan
Andrew ID: rharihar

Experimental Results

With naive top-1 retrieval, persona prompting (PP) outperformed instruction prompting (IP) and chain-of-thought (COT), achieving ~36 EM and 42 F1 compared to COT's ~23 EM and 31 F1. PP's concise answers matched gold references, while COT's explanatory outputs diverged. Instruction prompts provided moderate improvements but were penalized for including evidence quotes. Top-3 retrieval demonstrated clear gains over naive. EM rose to ~36.8 and F1 to ~44.2, reflecting that having three passages improved both faithfulness and relevancy. Top-5 retrieval, however, underperformed with ~26.7 EM and 32.9 F1, confirming that too much context introduces noise that distracts the model. RAGAS results supported this: top-3 retrieval achieved faithfulness around 0.74 and answer relevancy near 0.48, while top-5 dropped to faithfulness of ~0.45 - 0.65 and answer relevancy as low as 0.21. Context precision and recall rose in top-5, but the extra passages diluted answer focus.

Enhanced retrieval methods (HyDE, re-ranking) improved retrieval alignment incrementally. These refinements modestly boosted faithfulness and relevancy scores. However, the largest improvements came from scaling models. Switching from Flan-T5-small to T5-base and T5-large yielded much higher EM/F1 and stronger contextual grounding. Larger models processed longer contexts effectively, filtered noise, and produced precise, reference-aligned outputs. For instance, while top-3 with the small model capped near mid-40s F1, T5-large significantly exceeded that even on a reduced 200-question subset. RAGAS confirmed this trend, showing stronger faithfulness and precision, though answer relevancy was underreported due to limited sample size. Collectively, results highlight that context size and model capacity jointly determine performance: balanced retrieval (top-3) and larger models maximize gains, while over-retrieval (top-5) or excessive reasoning prompts reduce scores.

Production Considerations

Deploying the RAG pipeline in production requires addressing scalability and infrastructure. Milvus or FAISS can support larger passage corpora efficiently, but retrieval performance will depend on index tuning and hardware acceleration. Current results, obtained on CPU, highlight the need for GPU resources to handle larger models like T5-large at scale. Model deployment must balance latency and accuracy: while T5-large yields the best performance, it is slower and more resource-intensive. A pragmatic production system might pair T5-base with efficient top-3 retrieval and re-ranking to balance performance and speed.

Limitations include evaluation bias from strict EM/F1 metrics, which penalize longer but still correct answers. While RAGAS provides a richer view, its scores are sensitive to sample size and prompt design. The CPU-based experiments restricted sample size for T5-large, underestimating its true performance. For deployment, larger evaluation sets and real-world user feedback loops would be required. Reproducibility is supported by fixed random seeds, modular scripts, and pinned dependencies. Overall, scaling infrastructure, refining prompts, and balancing retrieval depth with model size are key recommendations. With GPU deployment, extended evaluation, and user-centric validation, the pipeline could transition effectively from research into production.

Appendices

1. https://docs.ragas.io/en/stable/getstarted/evals/
2. https://github.com/explodinggradients/ragas
3. https://medium.com/data-science/evaluating-rag-applications-with-ragas-81d67b0ee31a
4. https://towardsdatascience.com/17-advanced-rag-techniques-to-turn-your-rag-app-prototype-into-a-production-ready-solution-5a048e36cdc8/
5. I used ChatGPT for removing grammatical errors and give me concise professional report
6. Used ChatGPT to improve the prompts for baseline, chain of thought, persona and instruction prompting
7. Used ChatGPT and Claude to help me resolve errors in my code and also a little help in object oriented programming

system_prompt = "You are a knowledgeable assistant. Use the provided context to answer the question accurately and concisely. If the context does not contain the answer, respond with 'I don't know'." This is my current prompt, I want to give a better prompt now - give me three more refined prompts using these concepts- Chain of thought, Persona Prompting and Instruction Prompting

ChatGPT said:
Thought for 18s

Here are three copy-paste prompts—one each for Chain-of-Thought (safe), Persona, and Instruction styles. They all assume variables `{context}` and `{question}`.

1) Chain-of-Thought (safe)

```
You are solving a question using provided context only.

Think through the problem internally and do NOT reveal your
intermediate reasoning.
Return only the final answer and a brief, evidence-based
justification.

Context:
{context}

Question:
{question}
```

Name: Ritvik Hariharan
Andrew ID: rharihar

```
Output:
- Answer: <one concise answer or "I don't know">
- Why: <1–2 short sentences grounded in the context; cite key phrases
or IDs if available>
```

2) Persona Prompting

```
You are “The Meticulous Research Librarian”: cautious, concise, and
allergic to speculation.
Your style: 1–2 sentence answers, strictly grounded in the given
context.
If the context is insufficient, say “I don't know.”

Context:
{context}

Question:
{question}

Return:
Answer: <concise answer or "I don't know">
Evidence: <quote or passage ID from the context>
```

3) Instruction Prompting

```
Follow these rules:
1) Use ONLY the provided context to answer the question.
2) If the answer is not in the context, respond exactly with "I don't
know".
3) Keep the answer <= 35 words.
4) Include one short evidence quote or passage ID from the context.

Context:
{context}

Question:
{question}
```

Name: Ritvik Hariharan
Andrew ID: rharihar

```
Format:
Answer: ...
Evidence: ...
```