Sai Pranav Madineni
smadinen

Technical Report – RAG Evaluation

Executive Summary

This project implements a retrieval-augmented generation (RAG) system over a mini-Wikipedia corpus and evaluates two pipelines: a naïve baseline and an advanced configuration. I keep the stack local by default—SentenceTransformers for embeddings (MiniLM-L6-v2, 384-dim), FAISS/Milvus for similarity search, and FLAN-T5-base for answer generation—so no external API keys are required. The corpus is chunked into ~600-character passages and indexed once; queries are then encoded, retrieved, and summarized strictly from retrieved context.

Parameter sweeps in an earlier phase examine retrieval depth (top-k) and selection strategy (concatenation vs MMR). On a quick n=50 slice, concatenation consistently beats MMR for MiniLM, e.g., at top_k=3: EM 38.0, F1 45.43; with MMR at the same k: EM 32.0, F1 37.88. In the final Phase-5 setup, I compare a "basic" top-1 pipeline with a "ranked" pipeline that adds multi-query expansion and cross-encoder reranking. The ranked version substantially increases evidence recall (macro/micro ≈0.24 vs 0.16) but lowers evidence precision (≈0.048 vs 0.16), which matches the design goal of pulling in more potentially relevant passages before answer generation.

From an engineering standpoint, the system is deployable for small-to-mid collections on a single node with FAISS, and can graduate to Milvus/HNSW for scale. Recommended next steps are answer-aware context trimming, an IVF/HNSW index, and larger-n evaluation with confidence intervals.

System Architecture

Data & EDA. I use `rag-datasets/rag-mini-wikipedia` with a `text-corpus` split for passages and a `question-answer` split for evaluation. Light normalization preserves named entities.

Chunking. The baseline uses fixed-width character chunking at ~600 characters with no overlap. This is deterministic and fast to compute, and it yielded ~1.3k chunks in my runs. (A sentence-aware packer exists in the advanced notebook but is not required for the reported numbers.)

Embeddings. I encode chunks with SentenceTransformers **all-MiniLM-L6-v2** (384-dim), normalized for inner-product search. Batch size is 64 in the advanced notebook; query embeddings are normalized on the fly.

Indexes.

- **Naïve path (Steps 2–3):** Milvus Lite with a **FLAT** index and **IP** metric. This keeps dependencies consistent with the course's earlier steps.

- **Advanced path (Step 5): FAISS IndexFlatIP** for exact search. At the current corpus size, exact search is fine; for larger corpora I would switch to IVF/HNSW.

Retrieval.

- **Basic:** Single query vector → top-k neighbors (often top-1 for a tight sanity pass) → simple concatenation.
- **Ranked: Multi-query expansion (MQE)** creates three query variants; I retrieve ~20 candidates total and rerank with **cross-encoder/ms-marco-MiniLM-L-6-v2**; the top-5 are kept. Context is assembled up to a 2,000-character budget with lightweight inline citations.

Generation. FLAN-T5-base via Hugging Face pipelines produces answers **strictly from context**. An optional switch exists for an OpenAI model, but all results in this report use the local model.

Evaluation. I use (i) small-n EM/F1 sweeps to understand retrieval choices and (ii) an **ID-overlap** proxy that measures context precision/recall at the chunk level. This separates "did I fetch the right evidence?" from "did the generator express the answer well?"

Experimental Results

I focus on two complementary views: answer-level quality (EM/F1) from a quick sweep, and evidence-level quality (context precision/recall).

1) Retrieval strategy matters.
On a 50-question slice, **concatenation** outperforms MMR at the same depth for MiniLM. At **top_k=3 (MiniLM)**:

- **Concat:** EM **38.0**, F1 **45.43**
- **MMR:** EM **32.0**, F1 **37.88**

This pattern repeats at larger k (not all numbers shown here): as I increase k, concatenation's F1 improves modestly, while MMR trails. The implication is that, for this corpus and question style, "more neighbors" helps more than "diversity of neighbors." MMR likely drops some near-duplicate—but still helpful—evidence that improves generation confidence.

2) Adding MQE + reranking trades precision for recall.
For the **basic** pipeline (single-vector, top-1), the **ID-overlap** proxy gives:

- precision_macro **0.160**, recall_macro **0.160** (micro metrics match at **0.160**), n=50.

For the **ranked** pipeline (MQE → 20 candidates → **CE** rerank to top-5), I see:

- precision_macro **0.048**, recall_macro **0.240** (micro: **0.048** / **0.240**), n=50.

This is the expected movement: multi-query retrieval plus CE reranking **widens** the candidate set, so I catch more of the gold passage(s) (higher recall) but also include extra context (lower precision). Whether this helps EM/F1 depends on how the generator uses the longer context.

3) Why EM/F1 and overlap can disagree.
The overlap proxy measures whether the gold evidence appears in the final context, not whether the generator states the answer exactly. A recall gain without a precision guard can still produce similar or only slightly better EM/F1 if the generator is conservative or if the signal is diluted. This is the rationale for the enhancement plan below: push **answer-aware trimming** after reranking so the final context is both **covering** and **compact**.

4) Limitations of this pass.
The n=50 slice is intended for quick iteration. I would scale to n≥200 and report bootstrap CIs for publication. Also, character-based chunking can fragment sentences; a sentence-aware packer should reduce CE friction and likely lift precision without sacrificing recall.

Enhancement Analysis

Multi-Query Expansion (MQE). I generate three query variants per question (e.g., direct, elaborated, and "contextual meaning" prompts). This helps when the initial query underspecifies the entity or relation; different vectors land in slightly different semantic neighborhoods. MQE is cheap at my scale but multiplies retrieval calls; batching helps keep latency reasonable.

Cross-Encoder (CE) reranking. The CE (ms-marco-MiniLM-L-6-v2) meaningfully reshapes the candidate set; however, I used it only to **rank** passages, not to **filter** the final context. The observed precision drop suggests an extra CE pass—or a threshold on CE scores—should prune marginal passages before concatenation. A common improvement is **two-stage CE**: (1) rerank to top-k; (2) re-score each candidate w.r.t. the **predicted answer span** (short answer first), then keep only those with high answer affinity. This tends to lift EM and faithfulness while preserving recall.

Context assembly. A **2,000-character** budget is simple but blunt. I plan to migrate to **sentence-aware packing** with a redundancy penalty (MMR at the sentence level). This keeps entities intact and reduces repeating facts that don't contribute to the answer. In practice, sentence-level packing plus an answer-aware CE filter can recover much of the precision lost to MQE while keeping the recall benefit.

Embeddings and indexing. MiniLM-L6-v2 is a solid latency-optimized choice; mpnet-base-v2 may yield higher headroom at the cost of throughput. For indexing, **IndexFlatIP** is exact but O(N); I will move to **IVF/HNSW** with tuned `nlist/nprobe` (or `ef` for HNSW) as the corpus grows. The CE and trimming steps make this robust to small recall losses from approximate search.

Generation. FLAN-T5-base works well offline. If policy permits, swapping in a stronger generator often compresses context better and produces crisper answers; however, I prefer to first fix **evidence precision** so gains are attributable to retrieval, not model prior.

Production Considerations

Scale & performance.

- Up to ~1–5M chunks: FAISS with **IVF_FLAT** or **HNSW** on CPU; consider **float16** vectors to halve memory.
- Larger or multi-tenant: **Milvus** (or Zilliz) for distributed shards and background compaction.
- Budgeting latency: Query embed (MiniLM) ~milliseconds; MQE triples that; ANN search on IVF/HNSW is ~5–15 ms at 100k vectors; CE rerank for 20 candidates is ~10–40 ms on CPU (batching helps); FLAN-T5 generation (≤256 tokens) is tens of ms on a modest GPU or ~100 ms on CPU.

Reliability & observability.
I log retrieved IDs, CE scores, chosen context length, and the final answer string. A **no-answer** path (explicit "I don't know") is kept when coverage is poor. For safety, answers are constrained to provided context.

Limitations & risks.
Character chunking can split sentences and degrade CE signals. The overlap proxy is evidence-level; I treat it as a retrieval health indicator rather than a final quality metric. The reported EM/F1 at n=50 is directional; I will expand test size and attach CIs before any claims.

Recommendations.

1. Switch to **IVF/HNSW**,
2. add **answer-aware CE filtering** before concatenation,
3. adopt **sentence-aware packing** with redundancy penalties, and
4. expand evaluation to n≥200 with bootstrap CIs. These steps should raise precision without losing the recall gains from MQE.

Appendices

A. AI usage log

I used ChatGPT to help draft this report's prose and to generate inspection scaffolds that normalize exports between notebooks (e.g., mapping `chunks[*].content` → `text`), but all implementation, experiments, and decisions reported here were mine. No generative model was used to fabricate metrics; values reflect notebook outputs.

Sai Pranav Madineni
smadinen

B. Technical specifications

- **Corpus:** `rag-datasets/rag-mini-wikipedia` (text-corpus, question-answer)
- **Chunking:** ~600 chars, no overlap (≈1.3k chunks)
- **Embeddings:** SentenceTransformers **all-MiniLM-L6-v2**, 384-dim, normalized
- **Index:** Milvus FLAT/IP (naïve), FAISS IndexFlatIP (advanced)
- **Retrieval:** basic top-k; advanced MQE (3 variants) → 20 candidates → CE rerank → top-5
- **Reranker:** `cross-encoder/ms-marco-MiniLM-L-6-v2`
- **Generation:** `google/flan-t5-base` (offline)
- **Context budget:** 2,000 chars
- **Evaluation:** EM/F1 small-n; evidence precision/recall (ID-overlap) at n=50

C. Reproducibility

1. Run notebooks in order: **EDA → Naïve (2&3) → Param Eval (4) → Advanced (5) → Evaluation (6)**.
2. Ensure HF models are cached; no OpenAI key is needed.
3. Confirm chunk schema (`chunks[*].content`) and normalize to `text` if orchestrating across notebooks.
4. Build the vector index (FAISS or Milvus) and verify dimension=384 and vector count ≈ chunk count.
5. Reproduce the EM/F1 sweep at n=50 (or larger); then run the overlap metrics to obtain precision/recall.
6. For scale tests, switch to IVF/HNSW and re-run Step 5/6 to compare latency and accuracy.