Advanced RAG Pipeline: Hierarchical Retrieval & Evaluation

Language Models and Agents Major Project: Mid Eval Report

Saideekshith Vaddineni (2022101110)
Akshith Reddy (2022102048)

International Institute of Information Technology, Hyderabad

October 25, 2025

Abstract

This project implements and evaluates an advanced Retrieval-Augmented Generation (RAG) pipeline for a "Geography - Natural Resources" document dataset scraped from the internet. The core of this project is a hierarchical chunking strategy (2048 $\rightarrow$ 512 $\rightarrow$ 128) to provide deep context to the language model. We evaluate multiple retrieval strategies and models to find the most effective combination.

1 Pipeline Overview

1.1 Ingestion & Preprocessing

- Loads various document types (`.pdf`, `.docx`, `.txt`, `.md`, `.pptx`).
- Cleans text by normalizing whitespace, removing non-informative lines (e.g., "—"), and converting to lowercase.
- Performs exact and near-duplicate (Jaccard similarity > 0.92) detection to ensure a clean dataset.

1.2 Hierarchical Chunking

- Documents are first split into large 2048-token "parent" chunks.
- Each 2048-chunk is then split into 512-token "child" chunks.
- Finally, each 512-chunk is split into 128-token "grandchild" chunks.
- Each chunk stores its `parent_chunk_id` in its metadata, creating a traceable tree structure.

1.3 Indexing

- All three levels of chunks (2048, 512, 128) are indexed.
- Two separate embedding models are used for comparison:
  - `all-mpnet-base-v2`
  - `BAAI/bge-base-en-v1.5`
- This results in 6 total vector indexes (2 models x 3 chunk sizes) stored in FAISS.

1.4 Retrieval & Reranking

- A Top-Down (2048$\rightarrow$512$\rightarrow$128) retrieval strategy is implemented.
- This is augmented with a final reranking step using a Cross-Encoder (`BAAI/bge-reranker-base`) to improve the relevance of the final, small chunks.

1.5 Evaluation

- A formal evaluation (`eval_retrieval.py`) is run to measure the performance of each "flat" (non-hierarchical) index.
- Metrics: Hit Rate@10, MRR (Mean Reciprocal Rank), and nDCG@10.

2 Qualitative Results: The Reranker's Impact

The logs for the "Top-Down" search (`hierarchical_search`) clearly show the value of the final reranking step. For the query "Impact of mining on the environment," we can directly compare the results.

Query: "Impact of mining on the environment"

2.1 `all-mpnet-base-v2` (Test 1 vs. Test 3)

With Reranker (Test 1):

- ...pollution of land, soil, water and air...
- ...water directly kills living organisms...
- ...reduced water quality...

Without Reranker (Test 3 - Vector Search Fallback):

- ...pollution of land, soil, water and air... (Good!)
- ...social impacts associated with... mining... (Less relevant)
- ...destruction of forest resources... (Good, but #2 is off-topic)

Analysis: The vector-only search (Test 3) found a "social impact" chunk at Rank 2. The reranker (Test 1) correctly identified this as less relevant to the environmental query and replaced it with a highly relevant chunk about water quality and dead organisms. The reranker improves thematic precision.

2.2 `BAAI/bge-base-en-v1.5` (Test 2 vs. Test 4)

With Reranker (Test 2):

- ...effects of mining operations on forest and tribal people...
- ...water directly kills living organisms...
- ...reduced water quality...

Without Reranker (Test 4 - Vector Search Fallback):

- ...effects of mining operations on forest and tribal people...
- ...destruction of forest resources...
- ...disturb the land...
- ...water directly kills living organisms... (Pushed to Rank 4!)

Analysis: This is a perfect example. The vector-only search (Test 4) found a highly relevant chunk ("...water directly kills living organisms...") but ranked it at #4. The reranker (Test 2) correctly identified its high relevance and promoted it to Rank 2, pushing out less direct impacts.

Table 1: Evaluation Summary: Flat retrieval performance across models and granularities.

Model	Granularity	Hit_Rate	MRR	nDCG_K
`all-mpnet-base-v2`	2048	1.0	1.000	0.945
`BAAI/bge-base-en-v1.5`	2048	1.0	0.900	0.911
`all-mpnet-base-v2`	512	1.0	0.900	0.881
`BAAI/bge-base-en-v1.5`	512	1.0	0.650	0.753
`BAAI/bge-base-en-v1.5`	128	0.8	0.490	0.552
`all-mpnet-base-v2`	128	0.8	0.212	0.404

3 Quantitative Results: Model & Granularity Evaluation

The `eval_retrieval.py` script performed a "flat" search (simple vector search, not hierarchical) against each of the 6 indexes to find the best base retriever.

3.1 Metric Definitions

- **Hit Rate@10:** What percentage of queries found at least one relevant document in the top 10 results? (1.0 = 100%)
- **MRR (Mean Reciprocal Rank):** How high up is the first relevant result? A score of 1.0 means the first relevant result was always at Rank 1.
- **nDCG@10:** The overall quality of the ranking. It rewards putting multiple, highly-relevant documents at the very top. (1.0 = perfect ranking)

3.2 Analysis of Quantitative Results

- **Larger Chunks are Better for "Finding":** The most striking result is that 2048-token chunks are the best for the initial retrieval step.
  - `g128` chunks performed poorly (80% Hit Rate, very low MRR/nDCG). This is because broad, topical queries (like "impact of mining") are a bad vector match for small, specific 128-token "nuggets."
  - `g2048` chunks performed best. Their vectors capture the entire topic, making them an excellent match for broad queries.
- **`all-mpnet-base-v2` is the Best "Finder":**
  - The `all-mpnet-base-v2` model paired with 2048-chunks was the undisputed winner, achieving a perfect 1.0 MRR. This means for every query, its first result was a relevant one.
  - `BAAI/bge-base-en-v1.5` was a strong second but clearly less effective at this "topic-finding" task.

4 Final Conclusion & Recommended Strategy

Combining all our findings leads to a clear, data-driven recommendation.

The quantitative evaluation (`eval_retrieval.py`) proves that the best model for the initial search (finding the right topic) is `all-mpnet-base-v2` on 2048-chunks.

The qualitative logs (`hierarchical_search`) prove that a Reranker is essential for the final step (finding the best answer).

Therefore, the optimal strategy for this RAG pipeline is the **Top-Down Reranked Hierarchy**:

1. **Step 1 (Find Topic):** Use `all-mpnet-base-v2` to perform a fast vector search on the 2048-chunk index (as our eval showed it's the best at this).

2. **Step 2 (Filter Context):** Use the `parent_chunk_ids` to find all the 512-chunks and 128-chunks within those top topics.
3. **Step 3 (Find Answer):** Use the `BAAI/bge-reranker-base` model to do a final, "smart" re-ranking on the small 128-chunk candidates.

This hybrid approach gives us the best of all worlds:

- The broad, topical accuracy of `all-mpnet-base-v2` at the 2048-level.
- The high-precision, relevance-based selection of the BGE Reranker at the 128-level.