RAG Pipeline Project Progress Report

Topic: Self-Help and Psychology
Team Members: Ashutosh Kumar, Suraj Prajapati

Objective:

The purpose of this project is to develop a Retrieval-Augmented Generation (RAG) pipeline for analyzing and extracting meaningful insights from content related to self-help and psychology. This initiative aims to enhance query-based information retrieval while ensuring contextual and accurate response generation.

Task Allocation

Team Member	Task	Status
Ashutosh	To create an initial RAG Pipeline and to get feedback from others.	Completed RAG pipeline and analyzed approx 100 question answers manually for feedback.
Suraj	To Analyze output by trying different model combinations. Implemented an evaluation matrix for retrieval and generational for model outputs.	Completed initial testing and analysis by trying different models.

Content Analyzed

- Think and Grow Rich by Napoleon Hill
- Atomic Habits by James Clear
- Power of the Subconscious Mind by Joseph Murphy

Result Analysis

Chunking:

We created total 697 chunks, For chunking we experimented with LangChain libraries such as:
- SemanticChunker

- RecursiveCharacterTextSplitter

The SemanticChunker produced better results because it captured the complete context from the text. It identified coherent, meaningful chunks that align well with the semantic structure of the document, whereas RecursiveCharacterTextSplitter, though useful for fixed-length splits, often interrupted the flow of contextual information.

Results:
- SemanticChunker: More context preservation (measured by coherence in retrieved chunks)
- RecursiveCharacterTextSplitter: Less context preservation (due to breaking context mid-sentence)

Embedding:

For embedding generation, we tested two models:
- SBERT
- BAAI/bge-base-en-v1.5

After evaluation, we found BAAI/bge-base-en-v1.5 provided better results for our use case. This model produced smaller, faster embeddings without compromising on accuracy. SBERT performed well but was slower and more resource-intensive due to its larger embedding size.

Results:

- BAAI/bge-base-en-v1.5:
  - Embedding size: 512 dimension
- SBERT:
  - Embedding size: 768 dimensions

BAAI/bge-base-en-v1.5 was faster than SBERT.

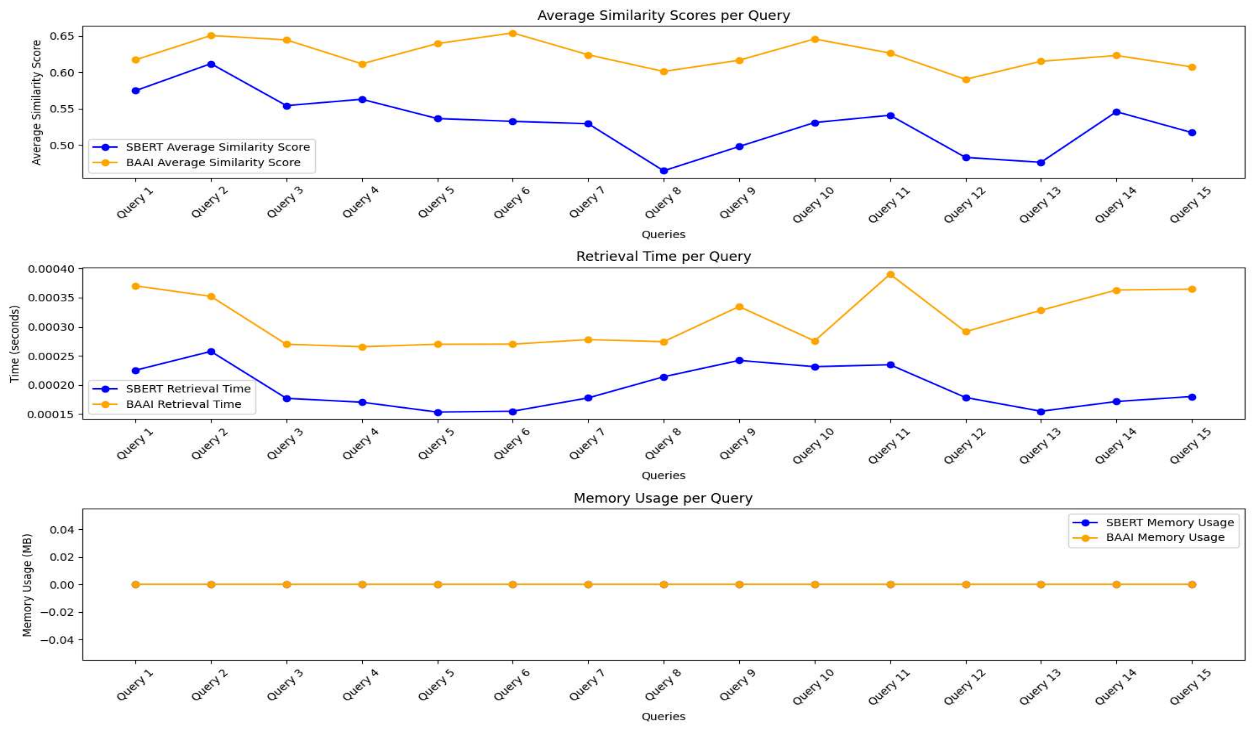
Retrieval:

For retrieval, we compared:
- FAISS (Flat L2)
- ChromaDB

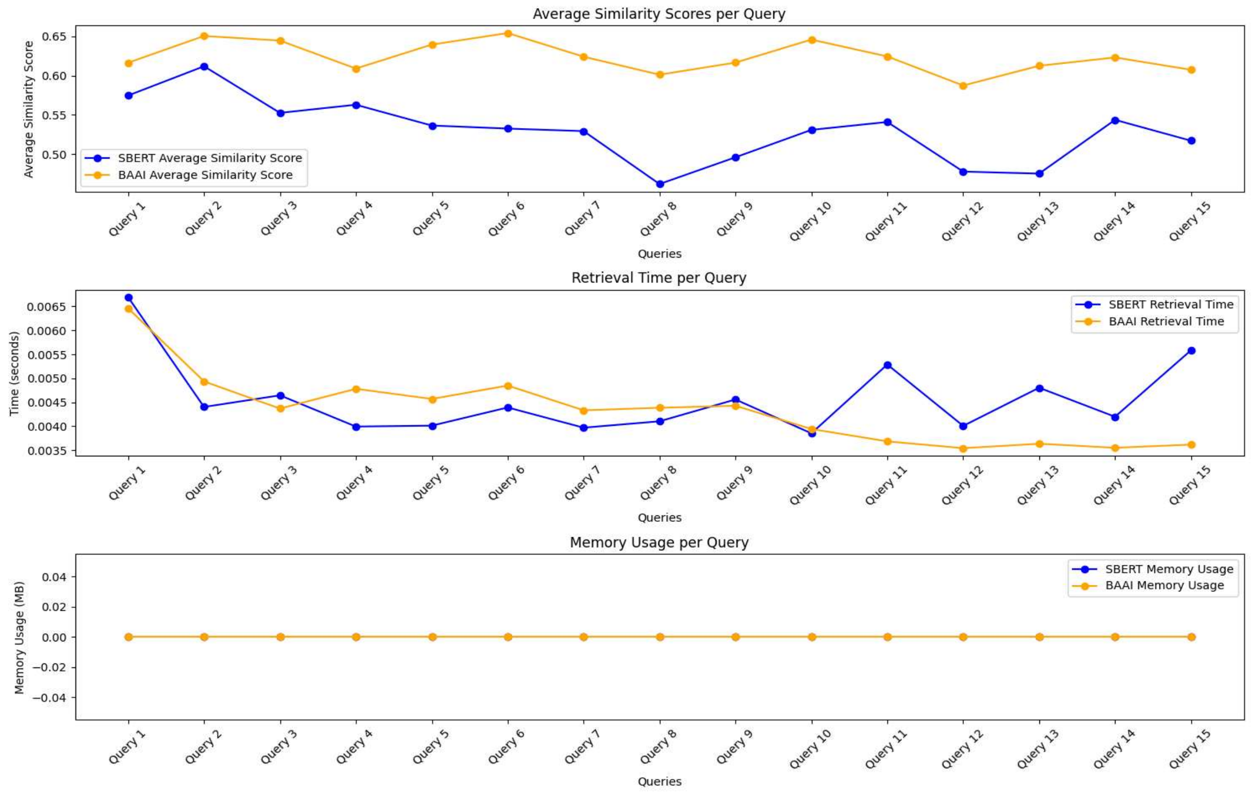
In our testing, ChromaDB yielded better results in terms of accuracy and relevance of retrieved chunks, particularly for complex queries. However, FAISS was notably faster and more scalable, making it ideal for real-time or large-scale retrieval. In terms of overall performance, ChromaDB emerged as the better option due to its superior semantic search capabilities.

Results:

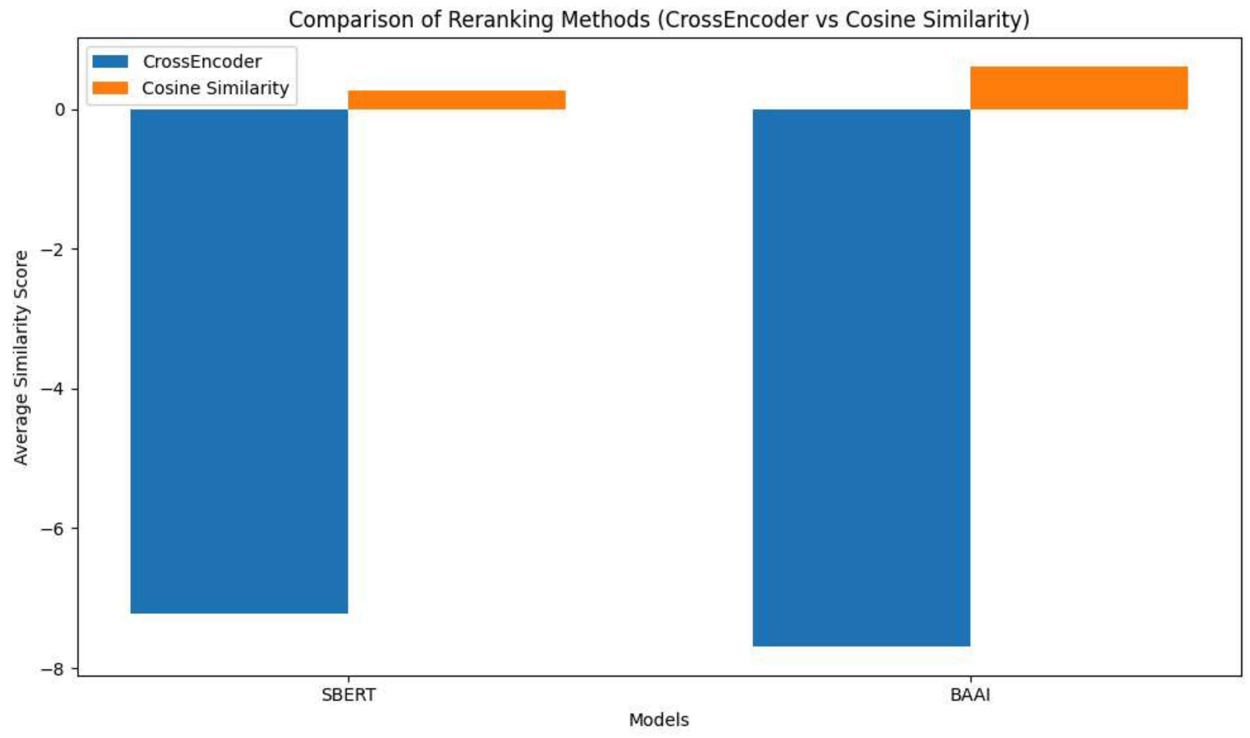
For FAISS:

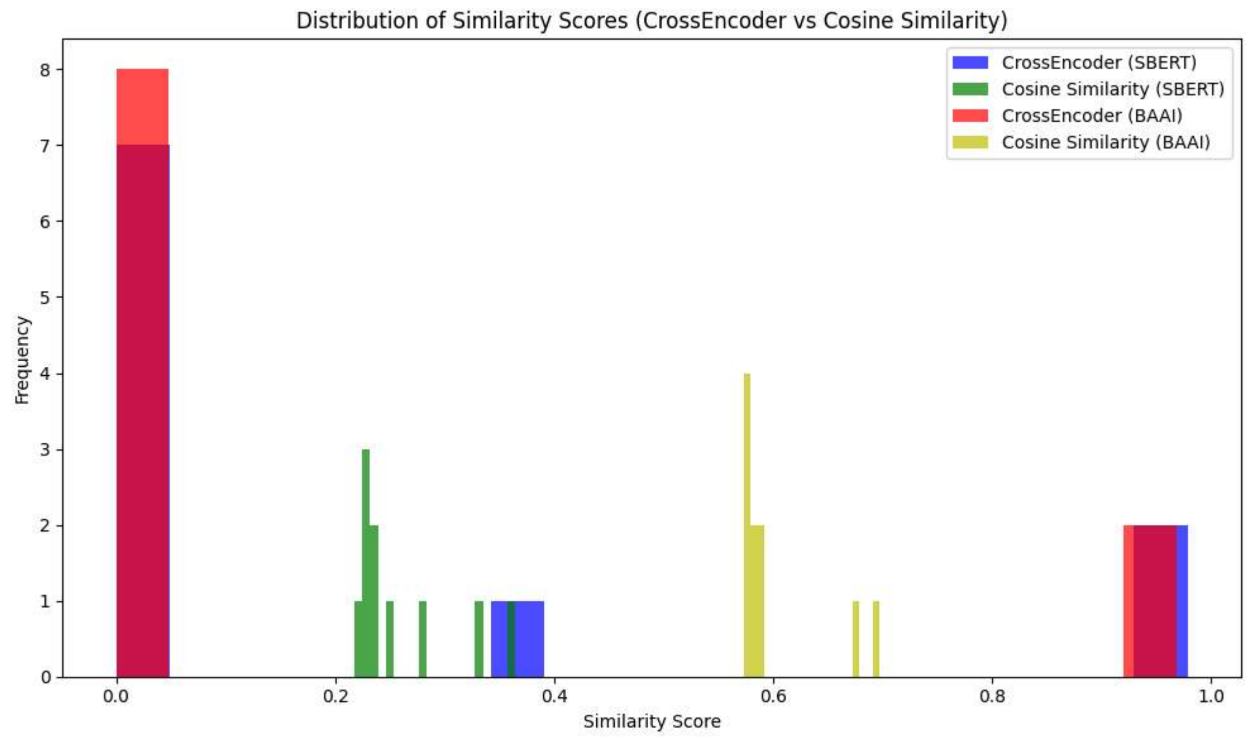


For ChromsDB:



ReRanking:





Result:

Key Observations:

Cosine Similarity (BAAI) has scores distributed primarily between 0.2 and 0.6.

Cosine Similarity (SBERT) has scores mainly concentrated between 0.2 and 0.4.

CrossEncoder (BAAI) shows a strong concentration around 0, suggesting lower or potentially less varied scores.
CrossEncoder (SBERT) also has scores clustered close to 0 but shows a few higher values near 0.4 and 1.

Interpretation:
Cosine Similarity generally produces higher scores and a broader range for both models, indicating a stronger match between queries and relevant items in the reranking task.
CrossEncoder, particularly for BAAI, tends to produce scores close to zero, suggesting weaker relevance or lower similarity scores.

Conclusion:
Based on both the average scores (from the previous bar chart) and the distributions here:

Cosine Similarity appears to be the better-performing reranking method for both models, especially given its more distributed and higher scores.
CrossEncoder, especially with BAAI, may not be as effective in this context, as shown by the concentration of scores near zero.

Therefore, Cosine Similarity is likely the better choice for reranking with these models.

Summary of Findings:

- Chunking: SemanticChunker offered better context retention, leading to improved retrieval quality.
- Embedding: BAAI/bge-base-en-v1.5 outperformed SBERT in both speed and accuracy, making it more suitable for our RAG pipeline.
- Retrieval: ChromaDB showed better retrieval time, while memory usage and similarity scores were almost the same as FAISS. For tasks requiring faster retrieval, ChromaDB is recommended.
- ReRanking: Cosine Similarity performs better than CrossEncoder, producing more distributed and higher scores, making it the preferred choice for reranking with these models.