Introduction to Text Analytics - HW4

Laiba Zubair Ellahi (24472), Saad Lakhani (24471), Talal Khan (25253)

Problem

While studying complex courses (in this example, we use Design and Analysis of algorithms), we found that answers to difficult questions could be found from the book. LLMs themselves were not good at explaining the concepts like the professor taught in class. Therefore, we attempt to see if Retrieval Based Augmentation over the textbook used in the course can help us solve this problem.

In this assignment we use Retrieval Augmented Generation over the book “Introduction to the Design and Analysis of Algorithms” by Anany Levitin.

Platform details

To carry out this project, we used collab with T4-GPU

Data Details

We split the selected book by page and treated each page as a document, this gave us a corpus size of 593.

Algorithms, Models, and Retrieval Methods

Retrieval Methods

We experimented with two primary retrieval strategies:

1. Keyword-based Retrieval using BM25
- We used rank_bm25 to build a traditional keyword-based index over the tokenized text. BM25 computes the term frequency–inverse document frequency (TF-IDF) score of documents with respect to the query and ranks them accordingly.
- This method is effective for exact term matching but less robust for semantic similarity.

2. Semantic Retrieval using Dense Embeddings
- We employed the BAAI/bge-small-en embedding model via HuggingFaceEmbeddings.
- All document chunks were embedded into vector space using this model.
- FAISS was used to index these vectors and perform fast similarity search based on cosine similarity.
- Dense retrieval excels at semantically matching user queries with relevant content, even if exact terms are absent.

3. Hybrid Retrieval using Reciprocal Rank Fusion (RRF)

- We combined both BM25 and dense retrieval using Reciprocal Rank Fusion.
- Each document's rank in both retrieval outputs was fused to compute a unified relevance score.
- This approach balanced lexical and semantic similarity, and consistently improved retrieval quality.

Justification:
Hybrid retrieval helped retrieve more diverse and relevant content across various question types. While BM25 performed better on formula-heavy or structured content, dense retrieval captured more nuanced semantic intent. Fusing the two maximized answer coverage.

LLMs and Answer Generation

We tested the following language models from Hugging Face:

- Qwen/Qwen2.5-3B-Instruct
- microsoft/phi-2
- meta-llama/Llama-3.2-3B-Instruct

Justification:
We used small instruction-tuned models to balance quality with inference efficiency on limited GPU memory (Google Colab). Qwen and Llama-3.2 consistently produced the most coherent and factual answers in our tests.

Chunking Strategy

To prepare the corpus for retrieval:

We used RecursiveCharacterTextSplitter from LangChain to split the textbook into overlapping chunks.
Parameters used:

- chunk_size = 500 and 1000
- chunk_overlap = 50

We chose **recursive splitting** to preserve semantic continuity around section boundaries.

Other Techniques Used

Long-context structuring:
Retrieved context was clearly enumerated in the prompt (1., 2., 3.) with explicit instructions for the LLM to **merge**, **compare**, or **reject** based on relevance.

This prompt engineering reduced hallucination and increased factual alignment. The RAG prompt instructed the LLM to:

- Provide a coherent answer.
- Mention contradictions if present.
- Output “no info found” when the retrieval was irrelevant.

This explicit control made the system robust in edge cases.

Evaluation:

We used LLM-based evaluation to evaluate both the retrieved responses and the LLM generated response. The evaluation criteria was as follows.

Part 1: Relevance of Retrieval
Each retrieved chunk was assessed for relevance to the question. For every chunk, the following was recorded:

- Whether the chunk was relevant (Yes/No)
- A brief justification for the assessment

Part 2: Faithfulness of Retrieval
The generated answer was broken down into individual factual claims. For each claim, the following was evaluated:

- The exact claim text
- Whether the claim was supported by the retrieved content (Yes/No)
- Which specific chunk(s), if any, supported the claim

A faithfulness score was then calculated as: **Faithfulness Score = (Number of Supported Claims) / (Total Number of Claims)**

Part 3: LLM Response Evaluation
The overall answer was rated on a scale of 1 to 5 across the following five dimensions:

- **Correctness**: Accuracy of the answer compared to the ground truth
- **Relevance**: Focus on the core aspects of the question
- **Coherence**: Clarity and logical flow of the response
- **Completeness**: Coverage of the full scope expected by the ground truth
- **Faithfulness**: Degree to which the response stayed grounded in the retrieved content and avoided hallucination

Experimentation

The exact outputs are in the code file. The scores are given here.

Question	Chunking size	Qwen/Qwen2.5-3B-Instruct"
What is the Master Theorem?"	500	Faithfulness Score: 6/8 = 75.0% Correctness: 4/5 Relevance: 4/5 Coherence: 4/5 Completeness: 4/5 Faithfulness: 4/5
Explain what is meant by Divide and Conquer algorithms.		Faithfulness Score: 6/6 = 1.00 Correctness: 5/5 Relevance: 4/5 Coherence: 4/5 Completeness: 3/5 Faithfulness: 4/5
What is the Knapsack Problem?	500	Faithfulness Score: 6/7 = 85.71 Correctness: 5/5 Relevance: 4/5 Coherence: 5/5 Completeness: 4/5 Faithfulness: 4/5

In our experimentation, QWEN produced the best results, which are given above.Hence this is the best model.