Medical Paper Recommendation System via Re-ranker Retrieval Augmented Generation

Jiahao Cheng
School of Literature, Science, and Arts
University of Michigan
Ann Arbor, US
jiahaoch@umich.edu

***Abstract*—Retrieval-Augmented Generation models improve response accuracy by combining external document retrieval with generative models. However, balancing retrieval speed with accuracy remains a challenge. This paper proposes a hybrid re-ranker model that integrates speed and accuracy, addressing the trade-off between retrieval efficiency and semantic precision. The re-ranker outperforms Bi-encoder and Cross-encoder models, showing a 7% accuracy improvement and a 95% reduction in retrieval time. It is also applied to a medical paper recommendation system, using RAG and prompt engineering for concise, well-formatted recommendations.**

***Index Terms*—Retrieval-Augmented Generation (RAG), Large Language Models (LLM), Information retrieval**

I. Introduction

A. Background and Motivation

With the rapid development of Large Language Models (LLMs), they have become valuable assistant of human. However, LLMs can be inconsistent, occasionally hallucinate facts, and rely on outdated knowledge.

Retrieval-Augmented Generation (RAG) addresses these issues by combining an external document retriever with a generator, which improves accuracy and reduces hallucinations. Although RAG improves accuracy, there is often a trade-off between speed and accuracy. Models like Cross-encoders offer high accuracy but are computationally expensive, while Bi-encoders are faster but less accurate. A hybrid approach that combines the speed of the Bi-encoder and the accuracy of the Cross-encoder is needed to overcome this trade-off.

B. Goal

The goal of this work is to propose the re-ranker model, which combines the Bi-encoder and Cross-encoder. The model will be compared with the two single models to evaluate whether it overcomes the trade-off between accuracy and speed. Finally, the model will be applied to build a medical paper recommendation system.

C. Literature Review

a) Benefits of Retrieval-Augmented Generation(RAG):

- Access to real-time, reliable sources, ensuring responses are grounded and checkable
- Reduction in hallucination and sensitive data leakage, by limiting reliance on internal memory
- Cost-effective updates, as there's no need to continuously retrain the model for knowledge refresh
- Transparent sourcing, allowing users to trace generated content back to specific documents

b) Bi-encoder: The Bi-encoder independently encodes the query and document into vector representations, using a distance metric (e.g., cosine similarity) to compute relevance. It is highly scalable and efficient for real-time search, but less accurate in tasks requiring fine semantic matching.

c) Cross-encoder: The Cross-encoder jointly encodes the query and document, producing a single score based on their interaction. While more accurate, especially for tasks with subtle distinctions, it is computationally expensive and inefficient for large-scale retrieval.

II. Method

A. Problem Formulation

a) Input and Output Definition: The input is a user's query, where they ask about topics of interest. The output is a recommendation feedback generated by a language model, which includes the title of a relevant academic paper, a direct link to the paper's entry on PubMed, a concise summary of the paper, a brief explanation of why the paper is relevant to the user's query.

b) Dataset Description: The dataset used in this study originates from the MedRAG/PubMed dataset available on Hugging Face. This dataset is a collection of PubMed papers. PubMed is one of the most widely used literature resources in the biomedical domain, containing over 36 million articles. The model uses a reduced version of MedRAG, selecting the first 10,000 articles from MedRAG.

c) Model Formulation: This study employs both a Bi-encoder and a Cross-encoder to realize the Re-ranker RAG model. The details of the model components are as follows:

- **Test Data Generation:** `llama3.2 3B`
- **Retrieval:** `all-MiniLM-L6-v2`
- **Augmentation:** `ms-marco-MiniLM-L6-v2`
- **Result Generation:** `DeepSeek-R1-Distill-Llama-8B`

B. Methodologies

a) Data Loading and Cleaning: Loading the dataset from HuggingFace. The cleaning process involves tokenizing the text, removing stop words and punctuation, and lemmatizing the content to standardize words to their root forms. This procedure helps reduce subsequent computational workloads by simplifying the text.

b) Generate Test Data: Randomly select 100 papers from the dataset, using their content to generate simulated user queries with the `Llama3.2 3B` model.

c) Single Bi-encoder and Cross-encoder Deployment and Evaluation: The `all-MiniLM-L6-v2` model and the `ms-marco-MiniLM-L6-v2` model are solely deployed, and their performance are evaluated by the test data.

d) Re-ranker Deployment and Evaluation: The re-ranking is realized by firstly using Bi-encoder for quickly retrieving a group of relevant papers, then using Cross-encoder for precisely re-ranking within the group. Its performance is evaluated on the test data.

e) Generate Recommendation: The `DeepSeek-R1-8B` model is called to generate recommendations based on the previously retrieved and augmented results. Prompt engineering techniques are employed to fine-tune the prompts, ensuring better and more accurate outcomes.

f) Command Line User Interface: Provides an easy-to-use interface for interacting with the system.

III. Result

A. Data and Model Set Up

a) Data Cleaning Outcome: The original contents have lengths ranging from 22 to 675 characters, with a mean of 161 characters. After cleaning, the lengths range from 14 to 303 characters, with a mean of 90 characters.

b) Models performance comparison: By evaluating on 100 papers test data, each model has performance below:

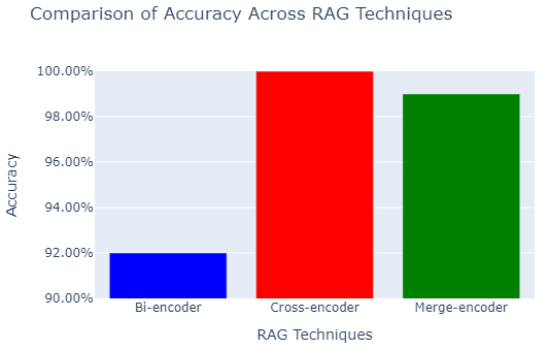

Fig. 1. Accuracy comparison

B. Interpretation

a) Data Cleaning: The data cleaning process reduced the average content length by 44.38%, from 161 to 90 characters.

TABLE I
MODELS PERFORMANCE

Table Head	Models		
	Single Bi-encoder	***Single Cross-encoder***	***Re-ranker***
Accuracy	92%	100%	99%
Time Mean(s)	0.0167	21.6735	0.0425
Time Std(s)	0.0018	0.4688	0.0071
Time Max(s)	0.0247	22.9641	0.0774
Time Min(s)	0.0137	20.3698	0.0329

This reflects effective lemmatization and removal of stop words, punctuation, and irrelevant content, allowing the data to focus on essential information. Also, it reduces future computational workload.

b) Models Performance: The single Bi-encoder model runs the fastest since the embeddings of papers can be precomputed and stored. However, its accuracy is the lowest. The single Cross-encoder model offers the highest accuracy but is slow during inference, as it needs to jointly process the query with each document. The re-ranker model achieves 7% higher accuracy than the single Bi-encoder model while being 95% faster than the single Cross-encoder model, making it an effective and practical RAG technique.

c) Time Complexity of Each Model: Assume the time complexity of encoding one query or document is $O(f(n))$, where n is the input length. Let there be N papers, each of length n, and one query of length m.

- **Single Bi-encoder:** Encoding N papers takes $O(N \cdot f(n))$. However, since the document embeddings can be precomputed and stored, the query process at inference time only involves encoding the query:

$$O(f(m)) \tag{1}$$

- **Single Cross-encoder:** At inference, the model must process each of the N query-document pairs jointly, with no precomputation possible. The total time complexity is:

$$O(N \cdot f(m+n)) \tag{2}$$

- **Re-ranker:** The Bi-encoder is first used to retrieve the top k most relevant documents (with precomputed embeddings), followed by Cross-encoder re-ranking on these k documents. The total time complexity is:

$$O(k \cdot f(m+n))(1 \leq k \leq N) \tag{3}$$

IV. Conclusion

In conclusion, the re-ranker model successfully balances the trade-off between accuracy and speed, making it a superior RAG technique. Moreover, using RAG and prompt engineering, the LLM-based recommendation system generates well-formatted and concise results.

However, there are still challenges: while using DeepSeek as the target LLM improves the outcome through reasoning (Chain of Thought), it incurs higher computational costs. The average reasoning time for a query is around 2 minutes, which is relatively slow compared to other practical applications.