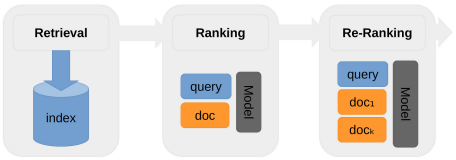

INTERNSHIP TASK REPORT

Presented

on Implementation of

Enhancing Q&A Text Retrieval using Reranking

Presented By:

Manasvi Shekhar

Indian Institute of Technology Kharagpur

Date of Submission: October 18, 2024

Introduction

The task of efficiently retrieving relevant information for question-answering (QA) systems has become increasingly important due to the vast amount of data available. This project focuses on implementing a **multi-stage text retrieval pipeline** designed to enhance retrieval accuracy by using both embedding and ranking models. The pipeline consists of two stages:

1. Candidate Retrieval

In this stage, embedding models are used to retrieve a set of potentially relevant passages from a large dataset based on a given query. These models convert both the query and the passages into dense vectors in a shared embedding space, allowing for the quick retrieval of the top-k passages with high semantic similarity. While efficient, this step focuses on broader relevance, which might not always prioritize the most contextually appropriate passages.

2. Reranking

After the initial retrieval, the top-k candidate passages are reranked using more sophisticated ranking models. These models assess the deeper contextual relevance of the query and passages, often incorporating cross-attention mechanisms that consider the query-passage pairs together. This fine-tuning step ensures that the most relevant results are prioritized, significantly improving accuracy compared to the initial retrieval stage, though at a higher computational cost.

Using datasets from the **BEIR benchmark** (such as NQ, HotpotQA, and FiQA), the pipeline's performance will be evaluated with metrics like **NDCG@10** to measure retrieval accuracy. The goal is to compare the effectiveness of different combinations of embedding and ranking models, while balancing trade-offs related to model size, accuracy, and computational efficiency.

Literature Review

1. Multi-Stage Retrieval

Text retrieval in information retrieval systems involves retrieving relevant documents or passages in response to a query. A multi-stage retrieval pipeline enhances this process by splitting it into two steps: candidate retrieval and reranking. In the first stage, embedding-based models quickly retrieve a set of potentially relevant documents from a large corpus, prioritizing efficiency. The reranking stage then refines these results using more complex models, such as cross-encoders, to capture deeper semantic relationships, improving overall accuracy while maintaining scalability.

2. Embedding Models for Candidate Retrieval

Embedding models form the backbone of the initial retrieval step in a multi-stage pipeline. These models represent both queries and documents as dense vectors in a shared semantic space, enabling efficient similarity calculations. Early approaches to embedding-based retrieval include static word embeddings such as Word2Vec and GloVe, which provide vector representations for individual words. However, more recent approaches have shifted towards **contextualized embeddings**, which take into account the surrounding context of words in a passage, thanks to transformer-based models like BERT and its variants.

One advantage of embedding models is their efficiency. By precomputing embeddings for a large corpus of documents, the retrieval process is reduced to a simple similarity search, typically using methods like *Maximum Inner Product Search (MIPS)* or *Approximate Nearest Neighbors (ANN)*. While these models excel at retrieving passages that are broadly relevant, their primary limitation lies in their inability to consider complex query-passage interactions, which may result in suboptimal ranking for very specific queries.

3. Ranking Models for Reranking

Following the candidate retrieval phase, the returned passages are rearranged according to query relevance using ranking models. While embedding models examine queries and documents independently, ranking models assess their relationship in tandem, frequently through the use of **cross-attention techniques**. These models result in much improved result precision, although being more computationally costly.

Reranking techniques that take the query and passage as a single input and use attention mechanisms to better capture their interplay are called **cross-encoders**. Although this nuanced method decreases computational efficiency, it enhances relevance comprehension. As a result, reranking models are typically limited to the **top-k outcomes** of the candidate retrieval stage in order to balance efficiency and accuracy without overwhelming the corpus.

4. Evaluating Retrieval Pipeline

The performance of text retrieval systems is typically assessed using metrics like **NDCG (Normalized Discounted Cumulative Gain)**, which evaluates the quality of ranked results based on document relevance. Specifically, **NDCG@k** measures how effectively the system retrieves and ranks the top-k relevant documents. A **high NDCG score** indicates that relevant passages are positioned near the top, which is vital for effective question-answering systems.

Other metrics, such as **Precision@k** and **Recall@k**, also gauge the accuracy of the top-k results. **Precision** reflects the proportion of relevant documents within the retrieved set, while **recall** indicates the proportion of relevant documents retrieved from the entire corpus. Analyzing these metrics helps researchers understand the **trade-offs between various model combinations** regarding accuracy and computational efficiency.

Implementation Details

Candidate Retrieval

Implementation Overview

Candidate retrieval involves selecting relevant passages from a large corpus based on a given query. This can be accomplished using an embedding model and a vector database. The embedding model embeds the text into vectors and then they are indexed using the **Faiss** database and retrieval is done using the ANN algorithm internally.

- **Model Selection and Embedding the corpus:**

  The model selected for embedding was `sentence-transformers/all-MiniLM-L12-v2`. It creates embeddings for all the context in the dataset and the questions (used only top 100 questions and contexts because of computational constraints). The data was already chunked and tokenised.

- **Query Processing:** When a query is received, it is processed through both embedding models to generate a vector representation of the query. It is then searched in the indexed Faiss dataset.

- **Similarity Search:**

  - **Cosine Similarity:** To retrieve relevant passages, cosine similarity between the query vector and the passage vectors is computed. This can be optimized using libraries like FAISS (Facebook AI Similarity Search) for efficient similarity searches, especially in high-dimensional spaces.
  - **Top-k Retrieval:** For each model, retrieve the top-k passages based on similarity scores, keeping track of the k highest scores and their corresponding passages.

Reranking

Implementation Overview

After candidate retrieval, the next step is to rerank the top-k passages based on relevance scores to provide the best results to the user.

- **Model Selection for Reranking:** Choose a reranking model that might be larger and more complex, such as `cross-encoder/ms-marco-MiniLM-L-6-v2`, which can better capture the nuanced relationships between the query and the retrieved passages.
- **Ranking Process:** Pass the top-k passages along with the query into the reranking model. The model processes the input to calculate relevance scores for each passage based on the model's understanding of the query-context relationship.
- **Reordering:** The passages are then reordered based on their relevance scores. This can be done by sorting the passages in descending order of their scores. The final output is the reordered list of passages that represent the most relevant to least relevant based on the query.

Evaluation

The output was evaluated for HotPot Q/A Dataset for the values of NDCG@10 (Normalized Discounted Cumulative Gain), Precision@10 and Recall@10.

(**Note**: Only one of the mentioned models were implemented as `nvidia/nv-embedqa-e5-v5` was deprecated from HuggingFace.)

Average NDCG@10 for 100 samples: 0.8407

Average Precision@10 for 100 samples: 0.29

Average Recall@10 for 100 samples: 0.77