Information Retrieval Project

Anonymous ACL submission

Abstract

1 Introduction

Questions:

Information retrieval is the process of retrieving relevant documents from a large collection as a result of a specific query. The goal of information retrieval is to be able to rank documents by relevance to a user query so that the most relevant ones are at the top of the ranking. The goal of this project in specific was to and gain basic understanding of information retrieval by implementing multiple different ranking and re-ranking techniques on multiple datasets, and then comparing accuracy and speed based on model, model size, and the ranking techniques used.

2 Methods

Questions:

The primary retrieval method that was used was BM25 while the primary re-ranking method in use was BERT.

BERT is a deep neural network based on transformers that is pre-trained ona large corpora (Lin et al., 2022). This allows it to perform semantic matching using contextual embedding, and the model learns dense, contextualized embeddings of queries and documents (Lin et al., 2022). In this context, it is not a retrieval method, but rather a re-ranking method for the results initially retried by BM25. It is able to capture context, meaning, and synonyms naturally making it much more accurate for relevance classification. BERT is computationally expensive and is much slower than BM25 which also makes it unsuitable for large-scale/first-stage retrieval due to its slow speed as opposed to BM25.

BM25 (Best Matching 25) is a classic and strong unsupervised ranking function used in traditional IR systems. It computes a relevance score between a query and a document based on term frequency and inverse document frequency, accounting for document length. (Lin et al., 2022) BM25 often serves as a baseline or first-stage retriever in modern IR pipelines such as Google, Yahoo, and Bing amongst other search engines. Its limitations come into play when it comes to understanding semantic similarity and vocabulary mismatch. BM25 utilizes the following scoring function:

$$\text{BM25(q, d)} = \sum_{t\in q\cap d} \log\left(\frac{N-df(t)+0.5}{df(t)+0.5}\right) \cdot \frac{tf(t,d)\cdot(k_1+1)}{tf(t,d)+k_1\cdot\left(1-b+b\cdot\frac{l_d}{L}\right)}$$ (Lin et al., 2022)

- q: the query
- d: a document in the collection
- t: a term that appears in both q and d
- $tf(t, d)$: term frequency of t in document d
- $df(t)$: number of documents containing term t (document frequency)
- N: total number of documents in the corpus
- l_d: length of document d
- L: average document length across the corpus
- k_1: term frequency scaling parameter (commonly set between 1.2 and 2.0)
- b: length normalization parameter (commonly set to 0.75)

Re-ranking is a 2 stage retrieval process involving a first-stage retriever (BM25 in this case), and a second-stage model (BERT-based cross-encoder) in this case. The first-stage retriever efficiently and effectively can return a short list of candidate documents for a given query, and the second-stage model is able to re-score and re-order the list of candidate documents based on semantic relevance and context. This allows this Information Retrieval method to combine the efficiency of traditional retrieval methods with the accuracy of deep transofrmer-based models. (Lin et al., 2022)

Cross-Encoder re-ranking involves concatenating queries and documents into a single input sequence as follows:

`[CLS] query tokens [SEP] document tokens [SEP]`

This input format is then fed into BERT which then produces a single relevance score, often derived from the [CLS] token's final-layer representation and a classifier head (Lin et al., 2022). The cross-encoder is powerful because it allows deep interaction between every token in the query and every token in the document. It is computationally expensive, though, because BERT must be run once per query-document pair (Lin et al., 2022)

Training a cross-encoder for re-ranking starts with input pairs of (query,document) where each pair has binary labels (relevant = 1, not relevant = 0), or graded labels (relevance judgments from datasets like MS Marco or TREC). Next, a pre-trained model BERT model, in this context, can be used as well as adding a classification layer on top of the [CLS] token output to predict a relevance score or class (Lin et al., 2022). A cross-entropy loss should be used for binary classification. Training data is usually human-annotated or distantly supervised data, but for MS Marco relevant documents are marked per query, with many negatives. The training objective is for the model to learn to assign higher scores to more relevant documents (Lin et al., 2022).

To do inference ranking, first-stage candidates must be retrieved using BM25 (in this context) to get the top documents for a query. From there, each (query, document) pair can be formatted to be scored with cross-encoder through BERT. The [CLS] token output should be extracted and the trained classification layer should be applied to get a relevance score (Lin et al., 2022). The documents retrieved from first-stage retrieval should be re-ordered based on the BERT scores outputted, and the top-ranked documents should be the final output. Since this is comutationally expensive, it should only be done on a small batch of candidate documents.

The two metrics used are Precision and Recall, each defined below.

Precision measures the fraction of retrieved documents that are relevant to the query. It is defined as:

$$Precision(R, q) = \frac{\sum_{(i,d)\in R} rel(q, d)}{|R|} \tag{1}$$

- R: The ranked list of documents retrieved for query q.
- $(i, d) \in R$: A document d at rank i in the retrieved set R.
- $rel(q, d)$: A binary relevance function; returns 1 if document d is relevant to query q, and 0 otherwise.
- $|R|$: The number of documents in the retrieved set R.

(Lin et al., 2022). Precision is often evaluated as cutoff k, denoted as Precision@k or P@k, which considers only the top k documents in the ranked list. Precision is easy to interpret and shows the proportion of relevant documents in the retrieved set (Lin et al., 2022). It also does not account for graded relevance judgments; it treats all relevant documents equally, and ignores the rank positions of relevant documents beyond the cutoff k. For instance, having relevant documents at ranks 1 and 2 yields the same P@10 as having them at ranks 9 and 10, even though the former is more desirable (Lin et al., 2022).

Recall measures the fraction of all relevant documents that are retrieved. It is defined as:

$$Recall(R, q) = \frac{\sum_{(i,d)\in R} rel(q, d)}{\sum_{d\in C} rel(q, d)} \tag{2}$$

- R: The ranked list of documents retrieved for query q.
- $(i, d) \in R$: A document d at rank i in the retrieved set R.
- $rel(q, d)$: A binary relevance function; returns 1 if document d is relevant to query q, and 0 otherwise.
- C: The entire document collection or corpus.
- $\sum_{d\in C} rel(q, d)$: The total number of documents in the collection C that are relevant to query q.

(Lin et al., 2022). Recall is also often evaluated at a cutoff k, denoted as Recall@k or R@k. It indicates the system's ability to retrieve all relevant documents, but like precision, it does not consider graded relevance or the rank positions of relevant documents (Lin et al., 2022).

Precision is important when a researcher is interested in the quality of the top results, such as in a scenario where users only exmaine the first few results of a search, therefore the cost of presenting non-relevant documents is high, and so it's important to ensure that retrieved documents are relevant. Recall is important when it's crucial to retrieve as many relevant documents as possible, such as in a scenario where missing any relevant documents could have significant consequences.

3 Experiments

Multiple types of models were utilized to compile results in order to be able to compare. For first-stage retrieval, BM25 and BM25 with RM3 were used and compared. For re-ranking, a BERT-based cross-encoder with sentence transformers were utilized and compared alongside the same models of different sizes denoted by L6, L4, L2, and L12.

3.1 Datasets

The Datasets used were the MS Marco and BEIR datasets.

3.2 Results

		High Resource		**Low Resource (BEIR)**									
Model	**Init. Retrieval**	DL19	DL20	Covid	Touche	News	SciFact	FiQA	SciDocs	DBPedia	Robust04	Signal	**BEIR (Avg.)**
BM25	N/A	49.73	48.76	59.63	36.11	39.1	68.3	24.87	15.13	31.33	41.3	32.87	38.74
BM25 + RM3	RM3	49.73	48.76	59.63	36.11	39.1	68.3	24.87	15.13	31.33	41.3	32.87	38.74
Re-ranking													
BM25 + sentence_transformers (L6)	ST-L6	73.13	67.93	72.91	26.15	42.64	68.57	34.72	16.35	44.03	47.83	34.06	43.03
BM25 + sentence_transformers (L4)	ST-L4	71.99	66.8	72.87	26.75	43.2	68.7	33.84	15.5	43.07	46.56	33.47	42.66
BM25 + sentence_transformers (L2)	ST-L2	70.46	64.56	69.71	26.87	42.3	60.59	27.75	14.21	38.93	41.51	31.97	39.32
BM25 + sentence_transformers (L12)	ST-L12	73.35	67.64	74.54	26.76	45.58	68.3	35.41	16.35	43.8	47.4	34.23	48.49

Table 1: Performance comparison (nDCG@10) of different retrieval models across high-resource and low-resource datasets. BEIR (Avg.) denotes the average over the 10 BEIR datasets. The re-rank section compares performance amongst models of different sizes.

Model	**L6**	**L12**	**L2**	**L4**
Speed on DL19 (MM:SS)	0:03	0:05	0:02	0:02

Table 2: Re-ranking speed comparison (in minutes and seconds) of different sentence transformer models on the DL19 dataset.

References

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2022. *Pretrained transformers for text ranking: Bert and beyond*. Springer Nature.