Assignment 2: Studying Sparse-Dense Retrieval: Part I

RunJie Xia - 879779

CM0622-1 (CM90) 2022-2023

1 Introduction

This assignment consists of an empirical study of the behaviour of the Maximum Inner Product Search (MIPS) over both dense and sparse vectors. The goal is to retrieve the top-k documents with respect to a given query by maximal inner product.

The approach is to take advantage of the linearity of the inner product operator and to break the problem into two smaller MIPS problems:

- Retrieve the top-k' documents from a dense retrieval system defined over the dense portion of the vectors.
- Retrieve the top-k' documents from a sparse retrieval system defined over the sparse portion of the vectors.

The two sets are then merged, and the top-k documents are retrieved from the combined (much smaller) set. As k' grows, the final top-k' set should converge to the exact top-k documents.

The task is to examine the correctness of the algorithm above and study the effect of k' on the quality of the final top-k set empirically.

2 Setup

We have opted to use a Jupyter notebook within DataSpell as our code editor and environment. The assignment was done on a MacBook Pro 2017 with the following specifications:
• Processor: 2,3 GHz Dual-Core Intel Core i5
• RAM: 8 GB
• SSD: 256 GB
• macOS Monterey 12.0.1

2.1 Dataset

We chose to use the Beir package as it offers a wide range of available datasets. For this experiment, we selected the TREC-COVID dataset from the BeIR repository on GitHub at https://github.com/beir-cellar/beir as it is more lightweight than the other options. The dataset consists of 50 queries and a corpus of 171,000 documents.

2.2 Pre-Processing

To begin our information retrieval assignment, we downloaded the dataset and organized the queries and documents as a dictionary, with the document or query ID as the key and the corresponding text and title as the value. The documents in the dataset were structured as a single string keyed by some ID, while the queries were represented as a JSON object with two fields: text and title. To handle the document structure, we converted them into a "compact form" by concatenating the title and text fields into a single string.

Next, we performed tokenization on the sparse data using the BertTokenizer to perform subword-tokenization, which helps to keep the vocabulary size relatively small. Stop-word and punctuation removal were also applied, as they are unlikely to be relevant to the queries in the TREC-COVID

dataset. Lemmatization was not performed separately, as it is included in the BM25 algorithm. The dense system, which uses a pre-trained transformer, did not require any preprocessing.

2.3 Sparse Embedding Computation

A sparse retrieval system is a document retrieval system that uses sparse vector representations of query-document (q, d) pairs for scoring. The BM25 metric was used in our implementation, which can be thought of as an improvement over the TF-IDF metric, which only takes into account the frequency of a term (TF) in a document and the inverse document frequency (IDF) of the term in the corpus. The BM25 metric incorporates additional factors such as the length of the document and the average length of documents in the corpus.

To use the BM25 metric for scoring query-document pairs in a sparse retrieval system, each query and document is represented as a sparse vector. The vector for a query contains non-zero values only for the terms that are present in the query, while the vector for a document contains non-zero values only for the terms that are present in the document. The values in the vectors represent the TFBM25 (an adjusted term frequency used in BM25) of the corresponding term.

The BM25 metric is then used to calculate a score for each query-document pair by taking the inner product of the query vector and the document vector. The inner product measures the similarity between the two vectors, with higher values indicating greater similarity and therefore greater relevance. The scores for all query-document pairs are then ranked in descending order to give the most relevant documents for the given query.

2.4 Dense Embedding Computation

A dense retrieval system is a document retrieval system that uses dense vector representations of query-document (q, d) pairs for scoring. We used the lightweight model all-MiniLM-L6-v2 model from the Sentence Transformers library, which computes 384-dimensional embedding vectors that aim to capture the meaning of the input text.
Similar to the sparse retrieval system, we obtained query-document scores by computing the inner product between the query and document embeddings. For a given query, we applied this procedure to each document in the corpus, resulting in a final document scores vector in R^|corpus|.
It's worth noting that the all-MiniLM-L6-v2 model has a maximum input length of 256 tokens, meaning any tokens beyond this limit are automatically truncated. This has important implications for the types of documents that the system considers to be more relevant.

2.5 Score Normalization

The main idea of this assignment is to combine the results of dense and sparse retrieval systems to capture different information from the queries and documents. However, the two systems assign scores on different scales, which may cause one system to dominate the other due to higher absolute values. To address this issue, a normalization technique was applied to the scores. For each document, its score was divided by the maximum score in the vector of document scores, resulting in a normalized score vector where the largest value is 1. This normalization technique was applied to both the sparse and dense system scores, effectively making them equally weighted.

2.6 Ground Truth (Top K) Computation

To obtain the ground truth for this experiment, we computed the sum of the scores obtained from the sparse and dense retrieval systems for a given value of k. We then selected the k documents with the highest scores to form the ground truth set S. This set will serve as a benchmark to evaluate the performance of the approximate set.

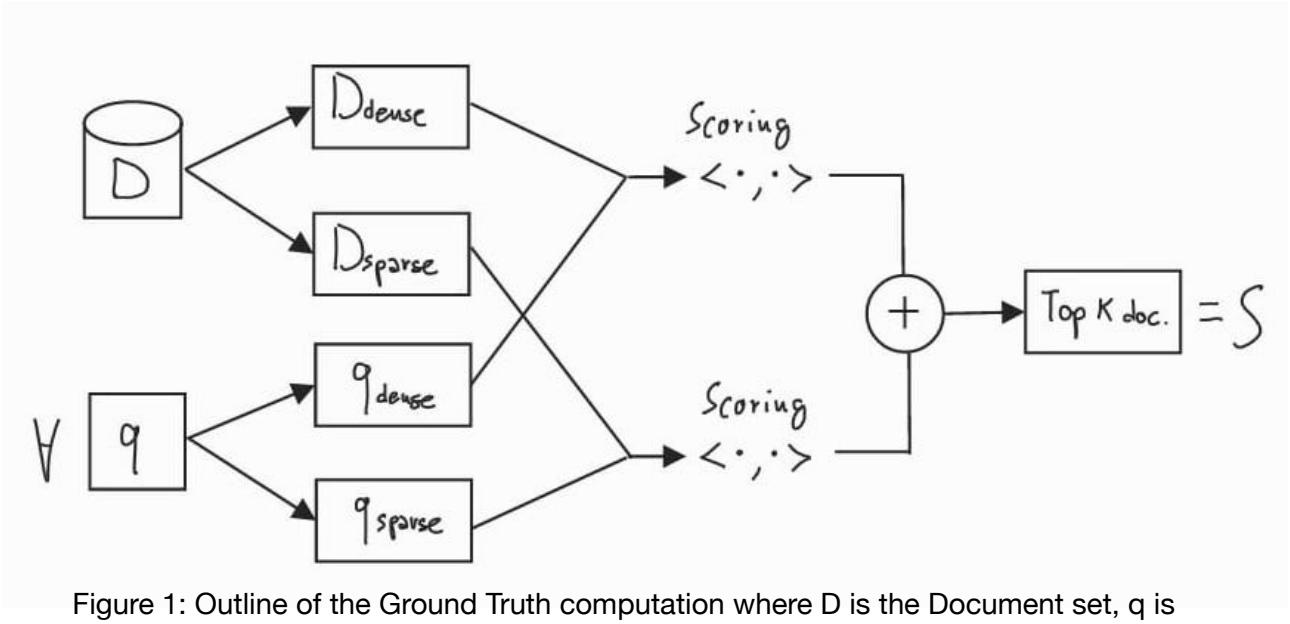

Figure 1: Outline of the Ground Truth computation where D is the Document set, q is the given query, and S is the final top k document set.

2.7 Approximate set (Top K') Computation

To obtain an approximate representation of the top k documents (i.e., the ground truth) for the experiment, we first separately retrieved the top k' documents with the highest scores from the sparse and dense retrieval systems. We then combined these sets by computing their union to form a merged set S'. To obtain the final set of the top k documents from the merged set that is now smaller than the original document set, we used a process similar to the ground truth computation. Specifically, we ranked the documents in the merged set by their score and selected the top k documents to form the final set.

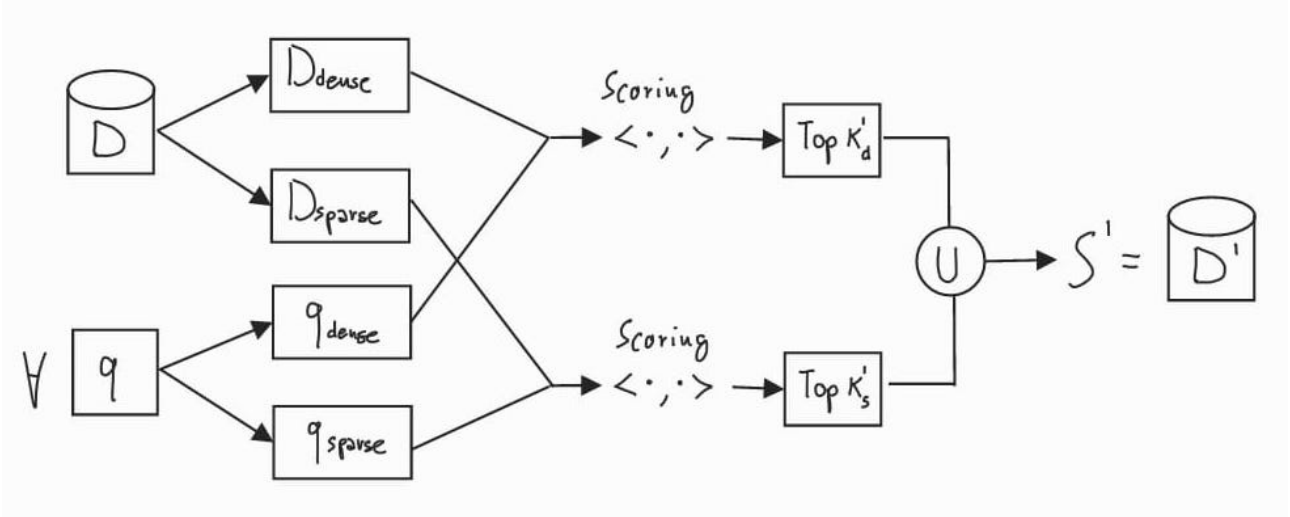

Figure 2: Outline of the Approximate Set computation where D is the Document set, q is the given query, and S' is the union of the two top k' document sets (D') that will be used to select the final top k documents.

2.8 Evaluation

The goal of the evaluation process is to measure how closely the approximate set approaches the ground truth as the value of k' increases. The evaluation metric used in this study is recall, computed with respect to the ground truth. Recall measures the proportion of relevant documents in the ground truth that are also present in the approximate set.

3 Experimental Results

This section presents the results of the experiment on the dataset. We used line plots to illustrate the relationship between the ground truth at k and the approximate top k'.
Specifically, the line plots show the mean recall values for the queries at different values of k' for a given value of k.

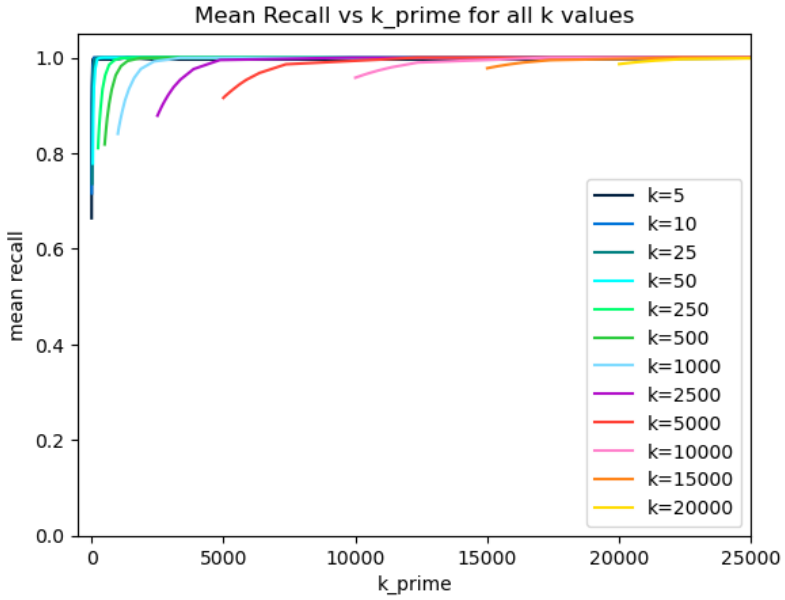

Figure 3.1: Graphical representation of the mean recall values for each value of k, plotted against the value of k'.

By examining the plot, we can observe how the mean recall values change as the value of k' increases for different values of k. This information can help us determine the optimal value of k' to use for a given value of k in order to maximize the recall.

The plot shows that the recall behavior is hyperbolic, with the recall metric increasing as k' increases, and eventually plateauing and slowly approaching a value of 1 as k' approaches infinity.

As expected, when we choose a smaller value of k (e.g., k=5), the relationship between the mean recall and the value of k' converges more quickly to have all recall values equal to 1. In other words, the more documents we retrieve to compute the approximate set (i.e., the larger the value of k'), the more accurate our retrieval becomes. This is because when we want fewer top documents, it is easier to accurately retrieve them within a smaller union set of top k' documents from the sparse and dense system. However, as the value of k increases, it becomes harder to retrieve all of the relevant documents accurately. This may be due to the fact that some documents are very similar to each other or that some documents do not stand out in both retrieval systems.

For example (see table below), when considering the top 3 documents, the actual top 3 includes documents d, e, and f, while the approximate top 3 includes documents b, d, and e. Document f is not included in the approximate top 3 set because, when considering the sparse and dense vectors separately, it is not as relevant compared to the other documents.

	a	b	c	d	e	f
Sparse	30	80	20	90	100	50
Dense	55	10	60	10	60	50

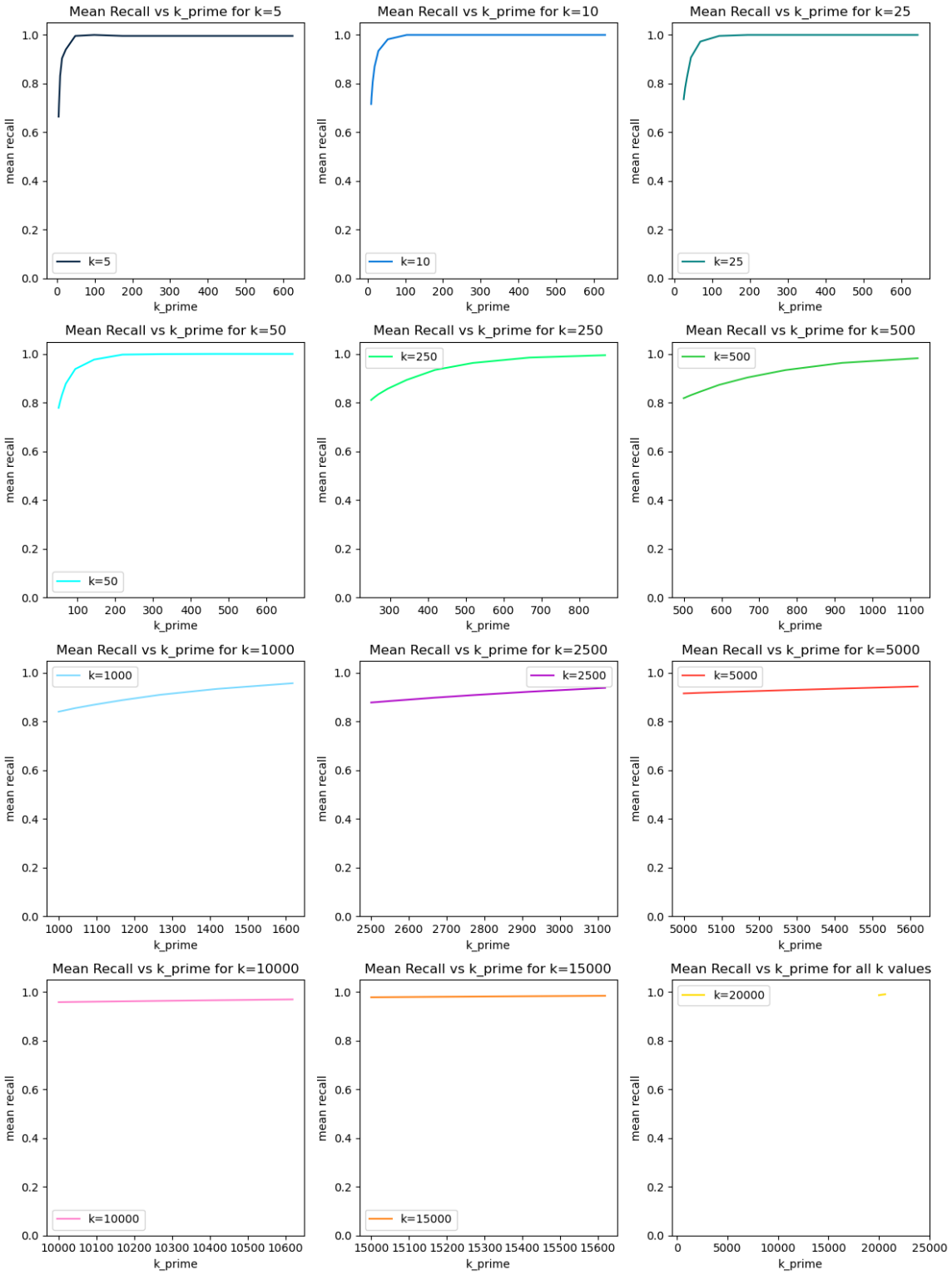

Figure 3.2: Separate plots for each value of k.

4 Conclusion

The two retrieval systems value different characteristics of the texts they score. The sparse system captures lexical signals of relevance, while the dense system captures semantic signals of relevance. Additionally, due to the maximum input length of 256 tokens and the structure of the document text data, the dense system mainly focuses on the first part of each document, assigning the highest scores to documents with a semantically-relevant title and an empty text field. In contrast, the best documents identified by the sparse system were those most lexically similar to the content of the query, and none consisted only of a title.

These differences suggest that the two systems capture different information from a given document. In a more complex retrieval system, a hyperparameter could be defined to weigh dense and sparse scores differently and tuned according to the goals of the system.