NYU Tandon School of Engineering
Computer Science and Engineering
CS 6913, Fall 2024

October 25, 2024

Assignment #3 (due November 13)

In this assignment, you will be comparing the BM25 scoring function from assignment #2 with a Dense Vector Retrieval approach. BM25 is effective for exact term matching but struggles with capturing semantic similarity when query terms and relevant documents do not share the same words. Dense vector retrieval, on the other hand, encodes both queries and documents as high-dimensional vectors using pre-trained language models such as BERT or MiniLM. The relevance between a query and a document is determined by the similarity between their vector representations (e.g., using cosine similarity).

You will use three search systems in this assignment: #1 is the BM25 index, #2 is a popular graph-based vector search index (preferably HNSW), and #3 is a re-ranking system where you would generate a list of candidate documents using BM25 as a ranking function, say the top 100 or 1000, and re-rank them by computing the similarity of their vector embeddings.

Since HNSW is an in-memory index, using the complete MS-MARCO passage-ranking dataset with 8.8 million passages might not be feasible. Thus you will be provided with a subset of 1 million passages from the original 8.8 million. This subset will be in h5 format, and python snippets will be provided on how to read this file. The h5 file contains the ids of passages or queries, along with their corresponding 384-dimensional embeddings. You will also be given a tsv file with all the passage ids selected in the subset, and an h5 file with the embeddings of all queries that need to be evaluated. These are the embeddings of all the queries present in the MS-MARCO queries.eval.tsv and queries.dev.tsv files. Note that these embeddings were generated using dot product as similarity measure, so that is what you will use to build the vector index.

The first step will be to re-build your BM25 index using the subset passages chosen for this assignment. The second step will be to build the vector index – it is recommended that you use `faiss` library that has implementations of various ANN vector search algorithms. If you use HNSW, be careful with parameter selection. It has 3 parameters: *m*, *ef_construction*, and *ef_search*. A reasonable choice for them would range from (4,50,50) to (8,200,200), with higher values generally giving better accuracy but with increased search time.

For evaluation, you will be given three files - #1 and #2 are named `qrel.eval.one.tsv` and `qrel.eval.two.tsv`. These file have query_id, document_id and relevance_label from 0 - 3 (3 means perfect match). These files are from TREC Deep Learning passage ranking evaluation from 2019 and 2020 respectively. #3 is `qrels.dev.tsv`, which has the same fields but only has a relevant(1)/not-relevant(0) field, meaning it does not have multiple levels of relevance. The actual queries for query_id's in these files are in `queries.eval.tsv` and `queries.dev.tsv`, respectively. You will need to run your queries on all 3 systems and evaluate MRR@10, Recall@100, NDCG@10, and NDCG@100. You should use MAP instead of NDCG for the queries in `qrels.dev.tsv`, as NDCG only makes sense when you have multiple levels of relevance. To compute these measures, you can use the `trec_eval` github repository. You will find all the data (except `collections.tsv` which you have from assignment #2) on this google drive link.

Make sure to go through the `README` file to understand each item on the drive.

For submission, you need to submit the code and a report with your analysis. Describe the trade-offs in terms of retrieval quality and overall efficiency (time and memory). It is highly recommended that you read the posted research papers on the reading list while doing this homework and creating the report. Python would be the most convenient language for this assignment, but you can choose other languages if you prefer. There will be a final demo with the TAs on campus.