Information Retrieval Report (Assignment 2)

Part 2 (Worked on 2A)

Name: Abhinandan De
Roll no: 19CS10069

Key Features:

Transpose the inverted index to decrease space complexity

It was a challenge to construct the term frequency matrix. Since the size of our corpus was around **37k** and there were **140k** terms in the vocabulary, it wasn't possible for us to directly construct the term frequency matrix. It was amounting to **38.6 GB** and allocation wasn't possible. Hence, we compromised on time and constructed a transposed version of the inverted index. Essentially, it is a map from cord-ids of documents to terms and frequencies.

Using a dictionary instead of NumPy arrays to decrease time complexity

We use normal dictionaries and harness a simple merge function to compute cosine similarity. This is because both our query and document vectors are sparse and |V| has a size of around 140k. So storing two terms in such a large array didn't make sense. Our runtimes were reduced from **90 mins** to **3 mins** with this simple change.

Tf-idf implementation

We compute the tf-idf representation of each document and query in the preprocessing phase. After this, we reuse the values by fetching them from a dictionary. This also reduces the time wasted in repeated computation. This halved our original runtimes.

General Procedure

All in all, we just compute the transpose of the document vectors and obtain a similar representation for the query vectors.

The tf-idf implementation is varied as per the methods which were required. All the configurations viz "**lnc.ltc**", "**lnc.lpc**" and "**anc.apc**" were handled

Then we just compute the similarity using a custom similarity function to match dictionaries

Finally, we sort the documents in the reverse order of scores and retrieve the top 50 documents.