Alper Canberk Balcı

Cmpe493 Project1 Report

1. Describe the steps you have performed for data preprocessing.  
   1- The main.py program takes the data and reads it in the main function  
   2- It strips the useless line at the top.  
   3- It finds the id of the document with NEWID.

4- It takes the title and body, and preprocess them with case folding, punctuation removal, and stop word removal. Then, it makes them lists of words  
5- It fills the inverted index

1. Describe the data structures (hash, b-tree, linked list etc.) that you used for representing the inverted index (i.e., the dictionary and the postings lists).  
   I used a dictionary for inverted index: key is word, value is a list containing the document id and indexes.

I used a dictionary for vector space: key is doc id, value is a dictionary, where its key is the word in that document, value is tf-idf score and the vector is normalized.

1. Provide a screenshot of running the indexing module of your system.  
   Text

   Description automatically generated  
   it continues to read the 21 files, I made these prints to console to check whether it runs smoothly.  
     
   It creates the json file of inverted index called sample.json,

A computer screen capture

Description automatically generated with medium confidence

however, the list of indexes of a word in a document is a bit strange. It creates duplicates. It works intendidly in the query processor, that’s why there is no problem. But, I couldn’t resolve the duplication error, my solution only made the main.py work slower, the bug is still there but it is harmless and doesn’t cause any error, or bug in the query search and cosine calculation.

1. Provide four screenshots of running your system for each of the four types of queries.

**Phrase query examples**

**Graphical user interface, text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Free text query examples**

Ranking is outputs in ascending order. The most similar, desired, documents are at the bottom of the output.  
**Query1:** bush oil

Least ranking outputs at the top, best ranking outputs at the bottom Text

Description automatically generatedText

Description automatically generated

**Query2:** cocoa shipment export tonne

Text

Description automatically generated Text

Description automatically generated

**Query3:** perpetual floating rate notes

Text

Description automatically generated Text

Description automatically generated

**Query4:** sugar prices increased

A screenshot of a computer

Description automatically generated with low confidence Text

Description automatically generated