MSCI 541 – Search Engines

Homework 5

Professor Mark Smucker

Matthew Erxleben

ID: 20889980

Date: December 7th, 2023

Problem 1)

If I were to have a method of finding the terms most similar to term t, I would use the dot product in a few ways. I would take the dot product between the current doc row for the term t and compute it with every other doc row. The highest dot product will be the most similar to the current term, as that is how TF IDF and Cosine similarity are also built on.

Problem 2)

I would use a different scale then the 1,2,3,4,5 scale. Instead, I would use a scale of 0,1,2. I would do this because there is a need to have a 0 option for the user. This is important so the user can say this result is not good, and therefore can't be added to the gain function. For example, something inappropriate that the user does not want to see at all. With the previous scale, even the worst option is still a 1/5 therefore including it in the gain function (nDCG).

Problem 3)

Installation Requirements:

1. Please make sure Python is installed on your computer before running the program.

2. Clone the repository on your device by entering this into your terminal: git clone https://github.com/UWaterloo-MSCI-541/msci-541-f23-hw5-matterxleben.git

Running the Programs:

To run these programs, please navigate to where you cloned the repository and open the working directory .../msci-541-f23-hw5-matterxleben

IndexEngine:

This program accepts two command line arguments:

1. a path to the latimes.gz file

2. a path to a directory where the documents, metadata, and term files will be stored.

For example, you would run IndexEngine from the command prompt / terminal / shell as:

python IndexEngine.py C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW5/raw-data/latimes.gz C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW5/store

BM25SearchEngine:

The program does not expect command line arguments to run. The program can be run by simply running the program (through and IDE or through the terminal). The program will prompt a user input through the terminal command prompt / terminal / shell, which will be the first query for the user to enter.

For this program, the location of the inverted index and other term files folder is in the code for the program, the path is hardcoded as:

term_files_path = "C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW5/store/term_files"

store_path = "C:/Users/matth/OneDrive/Desktop/University/3B/MSCI541-Search-Engines/HW5/store"

Please change this to your own computers path folder to obtain term files!

Query-biased summaries:

I used a scoring for the snippets of the document like this:

h +(0.1)*l+c+(1.5)*d+(1.5)*streak

The reason for this scoring depending on a few ideas. The idea was that the first or second line of the document were appearing too often as the snippet, and I felt that the score from the “l” parameter was too high. Therefore, it was scaled down drastically using a scaler multiple of 0.1. Furthermore, I felt the most important metrics for relevance to the user were the query terms being in the snippet, and them being in a streak. Therefore, I added a scalar multiple of 1.5 to the “d” and “streak” parameters to increase their weighting. This resulted in the query biased summaries to be more heavily weighting on the query terms being in the summary and them being in a row.