PSG COLLEGE OF TECHNOLOGY

Department of Applied Mathematics and Computational Sciences

IX Semester M.Sc TCS - 2019 - 2020

15XT96 Information Retrieval Lab

Exercise 1

---

1. The aim of this programming assignment is to create a simple index and design a retrieval system for a search engine. To create a search engine for your desk top do the following:

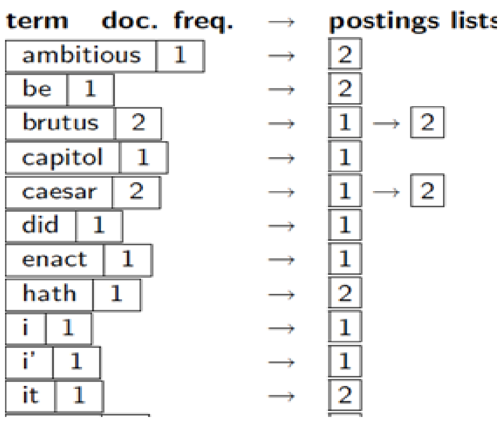
A) You should tokenize the text by any characters that are neither alphabetic characters (a-z and A-Z) nor digits (0-9). You should normalize the tokenized words into lowercase characters. (Do stemming for the words and remove stop words). Count and report the following statistics of the corpus:

- The total number of documents in the corpus
- The average length of each document
- The number of unique words appeared in the whole corpus
- Find the longest document(s) Report the docno(s) of the longest document(s) and the length
- The document frequency of words “information” and “retrieval”. (No. Of documents in which they appear)
- Compute and report the IDF of the two words as well IDF(w) = log(N/nw): N - the total number of documents in the corpus. nw: the number of documents containing the word w. The length of a document field refers to the total number of words

B). Build an uncompressed index over files from the desk top. The index is an inverted index which takes the following form:

The index has two components: Dictionary and Postings list.

Dictionary is an array of words and postings list is a list of documents that contains the word. Implement retrieval of Boolean conjunctive queries.



term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2

Your program should print to stdout the list of documents containing the query terms, one document file name on each line. The document file name should include its absolute path. The files should be arranged in lexicographic order

2. To maintain the integrity of your data, it's a good idea to reduce duplicate documents in the system. Desktop Plagiarism Checker helps us to detect documents. The database has N documents. In each document, the title is separated from the rest of the text by: (colon). A Plagiarism Checker reads a new document and adds it to the database if it is not a duplicate of any document in the database. Design a Plagiarism Checker with the following features.

A) Verify if the titles are exactly same (Apply BinaryDistance(u,v), which gives the binary distance between vectors u and v, equal to 0 if they are identical and 1 otherwise.). If same, label the document as duplicate and discard it else proceed to second part of the Checker.

B) Represent documents (excluding the title) as term document vectors with weight of a term in a document computed as

$$w_{ik} = \frac{tf_{ik}}{len_i} \log \frac{N+1}{0.5+df_k}$$

C) Identify a document as duplicate if the similarity of the document is more than the threshold α. (α=0.85). Apply cosine similarity for similarity computations.

D1: information requirement: query considers the user feedback as information requirement to search.

D2: information retrieval: query depends on the model of information retrieval used.

D3: prediction problem: Many problems in information retrieval can be viewed as prediction problems

D4: search: A search engine is one of applications of information retrieval models.

New documents

D5: Feedback: feedback is typically used by the system to modify the query and improve prediction
D6: information retrieval: ranking in information retrieval algorithms depends on user query

3) Build a compressed index over the same corpus using variable length encoding/Gamma encoding and implement Boolean conjunctive queries. Create a write up which includes timing statistics on how long the indexing takes, the size of the generated index, and the amount of memory used when building the index, total number of documents you output matches the expected count.