CPS842 Fall 2018

Assignment 1 report

Group members:

Baheer Kamal - 500574632

Mikhail Korchevskiy – 500652136

For this assignment, we decided to implement the algorithm using Go programming language, since it is a great combination of C-like performance and structural simplicity.

In the core of it, the algorithm is as follows:

* Read the entire ***cacm.all*** file into an array of maps. Each entry in that array represents a single document expressed as a map in the form of *<string> -> <string>*. The key is one of “I”, “W”, “T”, etc. and the value is the corresponding string value for a read identifier (“W” -> “Actual abstract text for a Nth document…”, “T” -> “Title of the document…”)
* Then for each document entry in the list, spawn a thread, which will parse through its contents, tokenize and index the terms (for context display) as well as ignore stop words and stem terms, if desired by user.(This is why Go was a great choice for this) The result is a structure, containing the following fields:
  + ID – the document ID
  + Tokens ( map<**string**, **int**> ) – map containing each term in the document with its corresponding term frequency within the document
  + Info ( map<**string, string**> ) – map containing the contents of corresponding tags in the document (like .T and .W), used for context display.
  + Occurences ( map<**string, int[]**> ) – map containing all the terms and their corresponding array of positions within the document
* The structures produced by each thread are formatted and pushed into a final map, which represents the postings file. The structure of each element is as follows: <**string**> -> <**structure**>, where the string key is the term itself and the structure is the data about the term described in following fields:
  + TotalFrequency – total frequency of the term across all documents
  + DocumentFrequency – frequency across documents (maximum 1 per document)
  + DocumentInfo – a map in the form of <*Doc ID* **string**> -> <structure { Frequency: **int**; Location: **int[]**}>, which shows frequency and locations of the term within a particular document

In the end, there are 3 files created: ***dictionary*** (as required), ***postings*** (as required) and ***docinfo*** to look up for context for the user search.

**Please look at the README.md to build/run the program.**