COMP38120 Workshop 4: Exploring tf-idf

You're going to explore term weighting based on term frequency and inverse document frequency. The main idea in this approach to indexing is that different terms differ in their effectiveness for predicting the relevance of a document given a query. Thus, a term that occurs frequently in a document will be a good predictor of what the document is about. Also, a document that occurs infrequently across the document database will be a good predictor of the usefulness of the documents in which it appears. By assigning weights to terms based on these two ideas, we can vary the importance we attach to a term as an index of a document.

1. Setting up

1.1 Linux

Ensure the PC is running linux, not Windows, for these tasks, as you'll be using a shell script that calls unix utilities. You'll need the following:

- At least one command or terminal window
- A calculator application (optional, as only addition of reals involved)

1.2 Download from Blackboard

From COMP38120 BB space, Workshop 4 folder, download the file under the link "tf-idf materials". This is a zip file that will uncompress into a directory called "tf-idf".

The uncompressed directory contains:

- tf-idfcalc: a shell script that calculates tf-idf weights for a number of files. It takes its input from a file passed in argument that contains the list of filenames to process. Tokenisation is basic (only alphabetic sequences are considered tokens).

  ```
  Usage: tf-idfcalc file_of_filenames
  ```

- unstemmed: a directory containing a number of raw text input files: amnesty, archeol, jenna, radon, tanks, trams, smoking, japan, chemical, ozone, asylum. The names are self-explanatory apart from 'jenna': fictional work. The files are somewhat 'noisy', that's life.

- lancaster: a directory containing stemmed versions of the above input files, run against the Lancaster Husk/Paice stemmer

- porter: a directory containing stemmed versions of the above input files, run against the Porter stemmer

- flist2, flist3, flist4: files of input file names from the directories 'unstemmed', 'lancaster' and 'porter', respectively, used as argument to tf-idfcalc.

- fpick: a script that allows you to re-sort the file output by tf-idfcalc, to show in first position the scores for an individual file of interest compared to all others. The first argument is the weights file, second argument is the sort key. You don't have to use this, but it may help.

  ```
  Sample usages:
    fpick /tmp/flist2.weights tanks
    fpick /tmp/flist3.weights tanksl       (suffix 'l' for files in lancaster directory)
    fpick /tmp/flist4.weights tanksp       (suffix 'p' for files in porter directory)
  ```

- myStemmer.class: An implementation of the Porter stemmer. This takes one argument, a text file containing words to stem.

- paicedemo.class: An implementation of the Lancaster Husk/Paice stemmer, menu driven. Press return when asked for rulefile and prefixing to use the defaults.

For the following tasks, note your group observations and be prepared to report back to the class to discuss what you have found.

2. Group task A

   Use tf-idfcalc with the file 'flist2' (so, the raw input files from the directory named 'unstemmed').

   Your task is to calculate the relevance score of different documents given the following queries, and decide whether the score is a good indicator of how well the document satisfies the query. I.e., you produce a ranked list of documents for the query (say, the top 3 or 4 to save time and effort).

   To calculate the relevance of a document given a query, you simply add the weights of the query terms which occur in the document. Remember the terms have different weights for different documents.

   Use the tf-idfcalc script to calculate the tf-idf weights for each term in each document. This writes a file to /tmp called "<filename>.weights" where <filename> is the name of the file given in argument, i.e., the file containing the list of filenames to process. This script is suitable for processing small amounts of texts and we use it here for pedagogic purposes as it's simple to use. The output in the weights file looks like this:

   ```
   access:  amnesty:1.450935 archeol:0.743855 jenna:0.439333 trams:1.048378
   ```

   which you will recognize as an inverted index entry, with the indexed word followed by a postings list of document IDs with the tf-idf score for that index word in that document. I've just used the filename as the doc ID for ease of consultation. The sample entry shows that 'access' is more relevant to the 'amnesty' document than to the others (documents not mentioned have zero occurrences of the term).

   The queries (Take e.g. one query each, work in turn and switch query for tasks B and C):

   1. system including
   2. cancer food
   3. automatic arms decision
   4. department director
   5. nation officials people
   6. problem affections
   7. food government health industry
   8. asylum political proposal immigration
   9. accident findings
   10. better english campaign

3. Group task B

Same task, but use now 'flist3' with tf-idfcalc to process the files in the 'lancaster' directory. Observe what changes if any occur to your relevance scoring/ranking calculations due to stemming with the Lancaster algorithm.

4. Group task C

Same task, but use now 'flist4' with tf-idfcalc to process the files in the 'porter' directory. Again, observe what changes if any occur due to stemming with the Porter algorithm.

5. Tidying up: delete the weights files from /tmp.