UCT 2004 CSC303 Information Management

Practical Assignment 4 : Information Retrieval

Due: 8 September 2004

Test Data:
1200 XML files. Contained in testdata4.zip.

Submit:
A single ZIP archive file containing all source files (including project, workspace, makefile, etc.) and your report, named REPORT.DOC or REPORT.PDF.

Problem

Build an information retrieval system conforming to an extended Boolean model – documents should be filtered on the basis of a Boolean "or" expression and ranked according to an appropriate ranking formula.

Test data is provided in the form of XML files. Your indexer must extract all text nodes from all files in a given directory and create inverted files (stored in files). Do case folding and stopping, but do not do stemming. Your query module must then perform a query based on a space-separated list of terms and list the matching document IDs (filenames) along with their titles, in standard ranked order. Name your indexer "index" and your query module "query" – both must take command-line arguments. The following sequence of commands should then produce output (on a BSD/*Nix system):

```
./index ../testdata
./query 'computer science'
```

Method

You can follow the following indexing algorithm:

- For each XML file:
  - Increment the DocId counter
  - Read in the file
  - Extract the title
  - Store (DocId, Filename, Title) in a separate index file (FileTitleIndex)
  - Remove all tags, punctuation, spaces, etc.
  - Fold case and remove stopwords
  - Count number of occurrences of each word (TF)
  - Add (DocId, TF) to a Hashtable keyed on words (the I.F.)
- Save the Hashtable to disk
- (You can store all inverted files into a single file, separate files, or you can save words starting with "aa" in a file called "aa", etc.)

Your FileTitleIndex could look like:

```
1:etd-123-456.xml:Some sample title
2:etd-123-789.xml:Another sample title
…
```

Your inverted file(s) could look like:
abacus:1:3:5:1:45:1:78:1:345:1
anthill:7:1:67:1:234:1:678:1
…
where each word is followed by a set of pairs of DocId/Weight values.

You can follow the following querying algorithm:

```
Separate query into list of terms
Create a vector W of DocId/Weight pairs for each document
Initialise all weights in W to 0
For each term
        Load (DocId, TF) list from inverted file
        Calculate DF (= number or sum of TF values)
        For each TF
                Use the ranking formula to combine TF and IDF, and add to
                appropriate weight in W
Sort vector W on weights
Use DocIds to extract filenames and titles from FileTitleIndex and list
```

Write a short report on the algorithms and design decisions you made, as well as the different modules/programs of your solution and how to use them. Include some sample output for 3 typical queries that demonstrate reasonably successful retrieval. Your report should be at most 5 pages long (12pt Times New Roman body text, single line spacing, 2.5cm margins all around).

General Notes

You may use Java, C++ and/or Perl, and associated XML processing tools if you wish. (Talk to the TA first if you would like to use a different language/set of tools)

Your assignment will be marked according to the following criteria:

- Correctness – indexing and querying (40%)
- Output – presence in report and reasonableness (10%)
- Documentation – comments within programs, descriptive report (20%)
- Efficiency (10%)
- Stress – tutors testing your code with different test data/queries (10%)
- Creativity – extra effort (10%)

Final Note: Please check your results to make sure they make sense!