CS380 Information Retrieval and Web Search

Homework 2

Vector Space Retrieval

Tuesday Feb 18 before Noon

This assignment is simple to state: build, use, then report on a vector space Information Retrieval system.

Your system should use a basic inverted index (by basic I mean no positional information, no phrases). Further your system should use TF-IDF weights to compare your documents and queries.

I will not be reviewing your code; so your coding practices are your own concern. That said, I recommend using a language with at least a good hashtable implementation.

Do not use the Python NLTK package in doing this assignment.

What to turn in:

A paper copy of a report, containing at least the 5 topics listed below. The report should be turned in at my office. If I am not in, slide it under the door. The script, part 3 below, should be printed and submitted with your report.

1. A table showing, for one document in your collection, the calculation of TF-IDF weights for the 50 highest ranking tokens in one document in your collection (ranked by TD-IDF weight). The columns of your table should be:
    1. The token
    2. Number of documents in collection
    3. Number of documents containing that token
    4. The IDF weight
    5. The frequency of the token in the document

6. The frequency of the most common token in the document
7. The TF weight
8. The TF-IDF value

2. A textual description of how your IR engine works. This description might mention the programming language (but there is no need to do so). It probably should have pseudocode. It should not include actual code. (1-2 paragraphs should be sufficient.)

3. A "script" showing a recording of your program working on queries (you write the queries). (It is acceptable to hard code your queries into your program. In this case there is no user interface, but there should still be a bunch of reporting to the screen.) By script, I refer here to the UNIX script command. The recording must be sufficient to demonstrate that the program works correctly. The recording could be quite long depending on how much information you show for each query and how you define "sufficient".

4. Annotations for the script. These annotations should explain to an uninformed user (perhaps me) exactly what the script shows and why it is interesting or significant.

5. A table, showing for 5 of your documents, the 3 closest documents to those 5. The 5 should be chosen to have a range on some identifiable property (for instance 5 different authors.) For each of the 5 documents, feed it into your IR system as a query. Then, in the table indicate the the best matches to those 5 queries. Accompany this table with a paragraph analyzing the best matches. For instance "For document A, the best matches were Q, X and R. Those documents were written by a different author in a different century. But they are good matches because … The collection does contain several other works by the same author as A. However, those other articles are on a completely different topic…" Your analysis may be considerably more exciting.