Project 1 Report

Eddie Tribaldos, et7226

In this report I will discuss the approach taken to complete the assignment, the general algorithm employed, instructions on running the system, and insightful discussion on experimental results obtained.

Approach and Algorithm

For this assignment, we were tasked with improving the performance of a simple Information Retrieval (IR) system by adding common phrases (bigrams). For this, I had to modify the code to "find common two-word phrases that appear in a document corpus, recognize occurrences of these phrases in documents, index these documents using the entire phrase, and finally recognize these phrases in queries and use them in retrieval". This was accomplished with a HashSet object that stored the top 1000 bigrams found in the documents. I decided to use this since the search time needed to be constant (O(1)) for fast system retrievals. The process of generating this set was O(n) where n is the number of unique words in the corpus, but we don't care about this since it's a one time cost. The program first goes through every two word pair (after removing stopwords) and increments the occurrence count on a HashMap. Then every entry on the HashMap is converted to an object and added to a List for sorting. After sorting, the top 1000 objects are converted to Strings and added to the HashSet for our final set of most common phrases. Once we have this, we have to make sure that these are recognized during indexing, so a function was added to ir/vsr/Document.java to replace hashMapVector that took in the bigrams set as a parameter. This function acts the same as the previous method, but checked if the pair of words was in the set. If it was it would index the bigram and not index the individual words. Otherwise it would just index the word and keep going. Since queries use the same function, this new function could be reused and the queries now recognized and retrieved with the bigrams.

Running

This can be run by using the command:

```
java ir.vsr.InvertedPhraseIndex [-html | -stem | -v] corpus/
```

The optional flags here are -html, -stem, and -v. I added the -v flag to make testing more readable, and to make it easier for the user to use the system without a wall of text. The parameter "corpus/" is a directory containing the files that are indexed. The -html flag removes html characters and the -stem flag runs a stemmer on all the words.

Results

The result is a significant result of the relevance of the retrieved documents. Previously, the query "file systems" returned documents that had several occurrences of the word "file", even though these files had nothing to do with file systems. The query "logic programming" returned a document containing the word "logic" several times without mention of "logic programming". With the added bigrams, these documents were buried under relevant results containing "file systems" and "logic programming". Implementing and calculating a quantified performance increase is beyond the scope of this assignment, but from this and similar cases it is evident that the new system has increased the relevance of these retrievals, without hindering other queries in obvious ways. However, without running tests and collecting empirical data, there is no way to be sure.