Programming Assignment 7: Web Crawler and Search Engine

Dustan Levenstein

Overview

For this project, I was to write an web crawler and a search engine which would use the index built by the crawler. My goal was to build a memory efficient indexing method that would be comprehensive enough to handle the various queries that it was supposed to handle.

WebIndex

I used two structures in my WebIndex. The first is a mapping from words to search for to the files containing those words, plus I added an inner mapping from the files containing a specific word to the indecies at which that word is present. The second structure is a mapping from files to the number of words in the file; this structure helps to maintain the number of words as the index is built and afterward serves as a set of all files, useful for making complements.

I then implemented several convenient methods: getFiles, complement, getNegatedFiles (those not containing a given word) and hasPhrase.

WebCrawler

The web crawler modifications were pretty straightforward. One thing I did that I'm not sure is always right is to avoid browsing any pages that don't start with “file”, to avoid browsing the actual web. If that's not right on the computer you're testing on, you can change it to similar check to make sure it really is a file on the computer.

WebQueryEngine

I defined 3 structures for tokens: Token, QueryToken, and Tree. The token object holds a string, an isOperator flag, and no other information. QueryToken represents a specific instance of a token, and holds a reference to the index of the query it was pulled from. In the case of a phrase or implicit and search, it may hold extra Tokens in the otherTokens list. Tree is, naturally, a tree of tokens, as described in the packet.

All the algorithms were pretty straightforward. When parsing a query, I didn't take substrings until I needed to, electing to maintain a reference to the current index instead. When performing a query search, I simply had to implement some intersection and union algorithms and recurse down the tree.

Testing

My testing environment was adapted from the TreapTest class that Aidan and I wrote together for the last project. I made sure the tree parsed correctly, and the WebIndex, then the crawler, and finally the search engine. I haven't included this class in my submission, since it is was written with system dependencies, and won't work on a different machine.

Memory Concerns

I was never able to crawl more than about 1700 pages in the rhf folder with the default amount of memory allocated to java. I did try to make my program more memory efficient, but it didn't help much. However, the crawler works just fine if it is allowed more memory, by calling the program on the terminal with “java -Xmx256m” (for example). Unfortunately, I haven't had enough time to improve the memory efficiency of my algorithms.

Overall, I had a fair amount of success in getting the basic algorithms, but I wasn't able to get much done in the way of optimization. If I had another chance, I would probably have gotten started sooner, so I would be able to get more done, but unfortunately, this is what I have.