Programming Assignment 1

Assigned: Jan. 25
Due: Feb. 22

In this assignment, you will build programs to do basic indexing and querying.

As in all the programming assignments and the course project, you should use the Apache Lucene library. Most of what you need for this assignment is built as basic functions in that library. Pretty much all of the programming you need to do for this assignment is to build two drivers; and even most of that is laid out in the Gospodnetić and Hatcher textbook. This assignment is basically a "hello world" assignment for Lucene. That is, the point of this assignment is to get familiar with the basics of getting Lucene functions and an HTML parser running; it is not supposed to be a challenging programming assignment.

The assignment involves writing two separate programs: an indexer and a retriever.

The indexer creates an index for all the words (including stop words) in all the HTML files in a particular directory. HTML markup should *not* be included as search terms in the index.

The retriever should be implemented as a Google-like web page. That is, there should be a web page where an external user submits a query. Once the query is submitted, the retriever consults the index and generates a results page.

Retriever

Queries are simple single- or multi-word queries with no punctuation.

The results page is an HTML page with the following parts.

- A header: **Results for query** [echo query] **in directory** [directory name].
- Top-ranked 10 documents that match the query, or all the matching documents if there are fewer than 10. For each document,
  - Create a separate paragraph
  - List the file name for the document, with extension.
  - If the file has an HTML TITLE list the title. If the file has no title but the body begins with an HTML H1, H2, or H3, list that.

That's all.

For the query engine, use the default Lucene `search` and `score`, described in chap 3 of Gospodnetić and Hatcher.

Indexer

In Java, the name of the directory of files to be indexed should be a command line argument. In Python, it should be an argument to an indexer function called in the interpreter. The indexer function should be called "indexer".

You may assume that HTML files all have the file extension .html or .htm, and conversely you may assume that any file with those extensions is legitimate HTML.

For tokenizing, use the Lucene `StandardAnalyzer`. For indexing use the `Indexer` . You do not have to worry about optimizing the indexer for performance or parallism, so you can skip G&H section 2.7 - 2.9.

You will need to use an HTML parser. This does not come with Lucene; however there are plenty of fine, free, parsers available. G&H (pp. 242-245) recommend and discuss HTML Tidy; that should be fine. If you want to use another parser, that's allowed. It does not have to be very sophisticated, but it should be fairly forgiving (e.g. it should not demand things like < /p > close tags.)

Directory structure

The code should be written so that:

- The index, your code for the indexer, and your code for the retriever are all in the same directory.
- The Lucene code is in a sibling directory, with the name "Lucene". Your code can therefore access Lucene functions using the relative path (in Unix) "../Lucene".
- Similarly the HTML parser code is in another sibling directory, with the name "HTMLParser" and can be accessed using the pathname "HTMLParser".

Examples

An example zipped directory (just the HTML files in this course directory at some stage) is here.

Sample Outputs
Output for query "machine"
Output for query "machine learning"

Programming Language

Lucene is available in multiple programming languages. Java or Python are certainly OK for this assignment. If you want to use any other languge, consult with me.

Web Hosting

Information about hosting web pages on the CIMS server can be found at The Web at Courant

Submission

Upload to the NYU Classes site:

- Your source code. Please do not upload Lucene or the HTML parser.
- An identification of the HTML parser. If this is something other than JTidy, give a URL where it can be found.
- The URL of the web-based interface.
- A README file with instructions for compiling and running. (These should be extremely simple.)