COMP7306, Hong Kong University

Lab Assignment: Search Engine

Pierre Senellart (pierre.senellart@telecom-paristech.fr)

10 April 2013

Handing out the assignment

The assignment must be uploaded on the Moodle Web site as a .zip archive containing all source code files, in a way that facilitates as much as possible their assessment. Further instructions might be given by the teaching assistants.

Development environment

This assignment is written with a Java implementation in mind; a working JDK is the only required software (apart from libraries directly referenced in the assignment). Students are allowed to use a different programming language, but the task will likely be more difficult. Check with the teaching assistants whether the technology you want to use is acceptable.

Objective

The objective of this lab assignment is to create an inverted index from a corpus of text documents, so as to allow answering (conjunctive) keyword queries,and to design a Web interface to query this index.

Corpus

The dataset we use in this assignment is the content of the *Simple English* Wikipedia http://simple.wikipedia.org/ (an encyclopedia simpler and smaller than the English Wikipedia). The whole content of this encyclopedia can be downloaded from http://dumps.wikimedia.org/ but a preprocessed version is provided for this assignment on the course Web site. The format of this file is as follows: The first line is the title of the first article, while the following lines (up to the first blank line) form the content of this article, in plain text format. The second article comes after the next blank line, and so on. There are 50,441 articles in total.

1 Inverted Index (12 points)

1. Create a class `InvertedIndex` that will be used to store an in-memory version of an inverted index (that is, for each token occurring in the collection, this token and the set of all documents that contain this token, along with the weight of the token in this document, ordered by decreasing weight). You can use the collections from the Java API, especially `java.util.TreeMap`

and `java.util.TreeSet`. To define a `TreeSet<A>` or a `TreeMap<A,B>`, class `A` must implements the `Comparable` interface.

2. Add a constructor to this class that takes for argument a filename (containing the collection, in the format described above) and a list of stop words (one per line) and creates the in-memory index from the collection. The following rules can be used to preprocess the text:

   - Tokens are defined as successions of either alphabetical characters or digits (e.g., "Hello", "123", "a22"; "vice-versa" or "it's" are decomposed into two tokens). One can use the following line to compile a `java.util.regex.Pattern` that will match these tokens:

     ```
     Pattern alphaDigit = Pattern.compile("[\\p{L}\\p{N}]+");
     ```

     All tokens should be made lower case.

   - The Porter stemming can be used. An implementation of the Porter stemmer in various programming languages is available at `http://tartarus.org/~martin/PorterStemmer/`. The Java implementation is used as follows:

     ```
     Stemmer stemmer=new Stemmer();
     stemmer.add(word.toCharArray(),word.length());
     stemmer.stem();
     String token=stemmer.toString();
     ```

   - Stop words will be removed from the index if they are contained in the provided file (found on the Web site).

   A file can be read line by line in Java using a `java.io.BufferedReader` constructed as follows:

   ```
   BufferedReader r=new BufferedReader(new FileReader(filename));
   ```

   Use standard tf-idf weighting.

3. Test your index construction. In case you encounter memory limitation issues, just consider only the n first articles from the dataset (say, $n = 15{,}000$). Use debugging facilities or logging to have a look at the tokens stored in your inverted index and check if everything goes right.

4. Add a method `conjunctiveQuery` that takes as argument an array of String representing a conjunctive query, and run this query with the help of the inverted index to get the list of all matching results, along with their weight. Addition can be used to combine weights. `conjunctiveQuery` returns the whole set of results (ordered by decreasing total weight), so it is not necessary to use advanced top-k evaluation methods.

5. Test these methods. You can use for instance the query "president france 2007". Are the results relevant?

2 Web Interface (8 points)

1. Follow the tutorial at `http://wiki.eclipse.org/Jetty/Tutorial/Jetty_HelloWorld` to implement a minimal Web server in Java.

2. Use this Web server to add a simple Web interface to the inverted index. It should be possible to input a query in a form and, when clicking on "Ok", to display the list of top-k results.

3. For extra credit, you can add CSS styling, auto-completion of queries in AJAX (using past queries, or using existing words in the index), pagination features to display the set of resuts page per page, or any other feature that you think would be useful.