Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Florian Bäurle

Information Retrieval
WS 2012/2013

http://ad-wiki.informatik.uni-freiburg.de/teaching

UNI FREIBURG

Exercise Sheet 6

Submit until Wednesday, December 5 at 4:00pm

This exercise sheet is about implementing a web application that provides error-tolerant search as you type. Consider the explanations given in the lecture and the code design suggestions linked on the Wiki.

Exercise 1 (5 points)

Modify or extend your class *ApproximateMatching* from Exercise Sheet 5 to compute approximate prefix matches. That is, for a given query word q and a given δ, compute all words w in the vocabulary with $PED(q, w) \leq \delta$, where PED is the prefix edit distance explained in the lecture.

Also, don't read the vocabulary from a separate file anymore, like in Exercise Sheet 5, but take it from the inverted index for the full words, which you need for Exercise 4 below.

Exercise 2 (5 points)

Write a class *SearchServer* that provides the functionality for listening to HTTP requests on a given port, and for a given request, containing a query string q, returns words w with $PED(q, w) \leq \lfloor(|q| - 1)/3\rfloor$ as a JSONP object.

The words should be sorted by their document frequency. If more than 10 words match, return only those 10 with the highest document frequency (ties broken arbitrarily).

Exercise 3 (5 points)

Write a web application (with an HTML file, a CSS file, and a JavaScript file, all in a separate subfolder *exercise-sheet-06/www*) that lets the user type a query in a search field, and after each keystroke displays the error-tolerant prefix matches as computed by your server.

Exercise 4 (5 points)

Extend your server and web application such that, upon selection of one of the suggested matches (that is, as soon as a suggestion is highlighted), the top-10 hits for that query word are displayed (below the suggestion box). Re-use your code from Exercise Sheet 2 for that.

Optionally make your UI look nice and extend it in any way you want, for example to support error-tolerant prefix queries with two or more keywords. There is ample opportunity for playing around and adding cool features here.

[please turn over]

Your server should take two mandatory arguments: the name of a CSV file (of the same kind as in Exercise Sheet 2), and the port on which the server should listen. Any other arguments should be optional.

The executable should be called *SearchServerMain* (C++) resp. *SearchServerMain.jar* and should be automatically created after a *make all* resp. *ant all* in your subfolder *exercise-sheet-06*, and not in a folder further down in the hierarchy. As said above already, the web app files should be put in a subfolder *exercise-sheet-06/www*.

Commit your code to our SVN, in a new sub-directory *exercise-sheet-05*, and make sure that everything (including checkstyle) runs through without errors on Jenkins. Also commit a text file *experiences.txt* with your feedback. As a minimum, say how much time you invested and if you had major problems, and if yes, where.