Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Claudius Korzen

Information Retrieval
WS 2017/2018

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

Exercise Sheet 6

Submit until Tuesday, December 5 at **12:00pm (noon)**

This exercise sheet and the next are about implementing a fully functional web application for fuzzy prefix search. That is, by entering the right URL in a browser, you get a web page with an input field, and when you type something in the input field, you get a list of fuzzy completions for what was typed. This exercise sheet is about the static version of this application: you type something, you press the search button, you get the results. The next exercise sheet will be about the dynamic version, were you get results after each keystroke.

The dataset for this sheet and the next is the same as for ES5 (entities from Wikidata), except that we added an additional column containing the URL of an image depicting the entity (empty, if no such image exists). The use of this additional column is optional.

Exercise 1 (20 points)

Build a web application with the following components and functionality:

1. (10 points) Extend the server code provided on the Wiki (you can choose between Java and C++) to serve HTTP requests for HTML, CSS, and plain text, with the correct response headers and MIME types. When a requested file is not found, return a 404 header. When a file outside of the server directory is requested, return a 403 header. Make sure that you deal with all possible client behaviors, in particular: connecting and then aborting, connecting and not sending anything, sending the request data in several packets (some of which maybe empty).

Extend your build file (*Makefile* in C++, *build.xml* in Java) by a target *start* and two variables for the port and the file, such that *make PORT=<port> FILE=<file> start* in C++ (resp. *ant -Dport=<port> -Dfile=<file> start* in Java) starts your server and computes fuzzy completions from <*file*> and listens to requests on port <*port*>. See the Makefile (resp. build.xml) provided on the Wiki for an example.

Note: This was done live in the lecture, however, only a part of that code is provided on the Wiki. The reason is that you only learn this stuff by doing it yourself. Also note that Python is again not an option for this sheet, because the next item requires that you integrate your code from ES5, for which Python was not an option (because of efficiency reasons).

GET /next+page HTTP/1.1 Host: us User-Agent: you Accept: finger

2. (5 points) Implement a web page *search.html* containing an input field and a search button. The URL of the web page should be *http://<host>:<port>/search.html*, where *<host>* is the name of the machine where the server is running and *<port>* is the port on which the server is listening. A click on the search button should load a page with the URL *http://<host>:<port>/search.html ?q=<query>*, where *<query>* is the content of the input field at the time of the click. You can achieve this via a form, as explained in the lecture. Extend your server such that the web page returned for this URL looks like the original page, but containing the query in the search field and also displaying the top-5 fuzzy prefix matches (or less if there are less than 5 matches) of *<query>*, using your code from ES5. You can achieve this via template variables, as explained in the lecture.

Note 1: You can either use your own code from ES5 (or an improved or repaired version of it), or our master solution for ES5. In either case, please copy the code to your folder *sheet-06*, so that everything need to make your web application work is in this folder (and, of course, in our SVN).

Note 2: Check your implementation by starting the server and trying out the URLs from the TIP file on the Wiki, and verify that the results displayed are indeed those from the TIP file (as usual, don't worry about the exact syntax of the test cases in the TIP file, it's only the contents which is important). Your tutor will do the same when correcting your submission.

3. (5 points) Add a CSS stylesheet to your webpage and work on your design such that both the original page and the result page look reasonably nice. As a minimum, add some colors, some background, and make sure that things are well-aligned with reasonable and regular spacing in between. Beyond that give free reign to your creativity. Make the matching entities in the result page clickable, such that a click leads to a page related to the corresponding entity.

Important notice: if you are the first to find an error in the TIP file, you will be awarded a pack of Bahlsen ABC with which you can explore a variant of the problem from ES5 called tasty search.

Add your code to a new sub-directory *sheet-06* of your folder in the course SVN, and commit it. Make sure that *compile*, *test*, and *checkstyle* run through without errors on Jenkins. Also commit the usual *experiences.txt* with your much appreciated brief and concise feedback.

What is your favorite Wikidata entity and why?