Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Sebastian Walter

Information Retrieval
WS 2023/2024

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

UNI FREIBURG

Exercise Sheet 8

Submit until Tuesday, December 19 at **12:00pm (noon)**

This exercise sheet and the next are about implementing a fully functional web application combining fuzzy prefix search and querying knowledge graphs (this part is optional now, but mandatory in the next sheet). 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 matching entities 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, where you get results after each keystroke.

Exercise 1 (20 points)

Build a web application with the following components and functionality:

1. Extend the server code provided on the Wiki to serve HTTP requests for HTML, CSS, and plain text, with the correct response headers and media types. When a requested file is not found, return a 404 status code (not found). When a file outside of the server directory is requested, return a 403 status code (forbidden). Make sure that your code handles *all* possible client behaviors correctly, in particular: connecting and then aborting, connecting and not sending anything, sending the request data in several packets (some of which may be empty).

Your program should be callable from the command line as follows: *python3 server.py <entities> <port> [--database <database> --use-synonyms]*, where the last two arguments are optional (both in the sense that you don't have to implement it and that a user does not have to specify it).

Note: Much of this was done live in the lecture, however, only a fraction of that code is provided on the Wiki. The reason is that this is conceptually simple but full of tricky details and pitfalls. You would learn only little if you just copied the code, but you learn a lot if you come up with fully functional code yourself. We recommend that you do not copy the code from the video but that you only consult the video when you get stuck for a longer time. For the same reason, you should not use any libraries providing web-specific functionality (in particular *http.server*).

2. 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 additionally contains the query in the search field and displays the entities for the top-5 fuzzy prefix matches (or less if there are less than 5 matches) of *<query>*. You can achieve this via template variables, as explained in the lecture.

Note: Check your implementation by starting the server and trying out the URLs which are given at the beginning of *server.py* in the code template. Verify that the results displayed are identical to the results specified there. Your tutor will do the same when correcting your submission.

3. 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. Make the matching entities in the result page clickable, such that a click leads to a page related to the corresponding entity. Beyond that give free reign to your creativity. Note that the last column of *wikidata-entities.tsv* provides a URL of an image depicting the entity (empty, if no such image exists). If you feel like it, use this column to display a picture for each match.

4. Optionally (not needed for full points), add the functionality to display the relations from a knowledge graph for an entity using the SPARQL-to-SQL-engine from exercise 6. This will be a mandatory part of the next exercise sheet, so you might want to already look into it now. Again, the page should look like the page returned in item 2, but by clicking on an entity, it will additionally show all triples from a knowledge graph in which the entity is the subject. You get the Wikidata identifier of an entity from one of the additional columns in *wikidata-entities.tsv*. We provide additional triples in *wikidata-properties.tsv* on the Wiki which contains predicate labels and counts. Add these triples to the database built with *wikidata-complex.tsv* for exercise 6 such that you are able to display the names of the predicates and order them by their count. We also provide a *sparql_to_sql.py* in the code template, which you can use to query the updated database.

Commit your code to our SVN, in a new subfolder *sheet-08*. The dataset files should not be committed.

As usual, in your *experiences.md*, provide a brief account of your experience with this sheet and the corresponding lecture. Make sure to add a statement asking for feedback. In this statement specify to which degree and on which parts of the sheet you want feedback. In addition, say how much time you invested and if you had major problems, and if yes, where.