Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Patrick Brosi

Information Retrieval
WS 2016/2017

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

Exercise Sheet 5

Submit until Tuesday, November 29 at **2:00pm**

This sheet is again about performance, so Python is not an option; see ES3 for more explanations. Some basic code is provided in Java and C++ on the Wiki.

Exercise 1

Write a program for fuzzy entity search with the following functionality:

1. Build a q-gram index from a given file (one entity-score pair per line). Basic code for this is provided on the Wiki. The scores are needed for the method from item 4 below. The file from the Wiki also provides a geo-location for each entity. You can ignore this for this exercise or use it for any kind of fancy extension that comes to your mind.

Note: before computing the q-grams, normalize each string by lowercasing and removing whitespace as shown in the lecture. Don't forget to also normalize the input string in your main function, as described in item 6 below.

2. Implement a method that merges the inverted lists for a given set of q-grams. Pay attention to either keep duplicates in the result list or keep a count of the number of each id.

Note: it is ok, if you do this merging by simply concatenating the lists and then sort the concatenation (that is, without making use of the fact, that the lists are already sorted.

3. Implement a method that for two given strings x and y and a given integer δ, returns $\mathrm{PED}(x, y)$ if it is $\leq \delta$, and $\delta + 1$ otherwise. The method must run in time $O(|x| \cdot (|x| + \delta))$, as explained in the lecture.

4. Implement a method that for a given string x, finds all entities y with $\mathrm{PED}(x, y) \leq \delta$ for a given integer δ. First use the q-gram index to exclude entities y that do not have a sufficient number of q-grams in common with x, as explained in the lecture. Only for the remaining candidate entities should the PED be computed with the method from item 3. The method should record the number of these PED computations and the correctness of that number should be verified for the given test cases. The method should return: all y with $\mathrm{PED}(x, y) \leq \delta$, and for each such y the value of $\mathrm{PED}(x, y)$ and the score of y (from the file read by the method from item 1).

[please turn over with no errors watsoever]

5. Implement a method for sorting a given set of entities y, each with a score s and a PED value p. The entities should be sorted by (p, s). That is, all entities with PED $= 0$ should come before all entities with PED $= 1$, etc. And the entities with the same PED should be sorted by score (higher score first).

6. Implement a main function that builds a 3-gram index from a given file (given as a command-line argument), and then, in an infinite loop, lets the user type an x and prints the top-5 y according to the method from item 4 and the ranking from item 5. Take $\delta = \lfloor |x|/4 \rfloor$, where x is the normalized version of what the user typed (in particular, with whitespace removed, see above). Take care that you print the original entity names, not the lowercase names without whitespace. If there are more than 5 matches, also print the total number of matches. Also print the time it took to process the query.

7. Add a row to the result table on the Wiki following the examples already given there.

As usual, you must implement the test cases provided in the TIP file on the Wiki and make sure that everything runs though on Jenkins without errors. Please also adhere to the structure and function names suggested in the TIP file. Otherwise, it quickly becomes a nightmare for the tutors to correct your code and provide meaningful feedback. If you want to deviate from the given structure or modify the test cases, please ask on the forum.

Add your code to a new sub-directory *sheet-05* of your folder in the course SVN, and commit it. Make sure that *compile*, *test*, and *checkstyle* run through without errors on Jenkins. And of course, commit the usual *experiences.txt*.

What is the shoe size of a motor protein and what does it walk on?