Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Elmar Haußmann

Information Retrieval
WS 2015/2016

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

UNI FREIBURG

Exercise Sheet 5

Submit until Tuesday, November 24 at **2:00pm**

For this sheet, extend the Python code from the lecture. You are also free to re-implement the code from the lecture in Java or C++ (you can re-use much of your code for Exercise Sheet 1), but it will be more work (recall the comments on the back side of Exercise Sheet 1).

Note: for this sheet, the specifications /hints for Exercises 1 - 3 are contained in the file *qgram-index.py*, not in a separate TIP file.

Exercise 1 (5 points)

Implement a function 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.

Exercise 2 (5 points)

Implement a function that computes the prefix edit-distance (PED) between two given strings.

Exercise 3 (5 points)

Implement a function that finds all records that are within a given PED of a given string. First use the q-gram index to generate a subset of candidate records. Then compute the exact PED for each candidate. Along with each matching record, also return the PED.

Note: before computing q-grams normalize each string by lowercasing and removing whitespace as shown in the lecture. Don't forget to also normalize the input string.

Exercise 4 (5 points)

Implement a main function that builds a q-gram index for a given file of records. Use the file provided on our wiki (one record per line) with $q = 3$. Then, in an infinite loop, answer arbitrary fuzzy prefix queries from the user. As threshold for the PED, take $\lfloor |x|/4 \rfloor$, where x is the (normalized) query word. Output the top-5 results ranked by a combination of PED and score (implicit in the record's line number, as used before). Try to find a combination of PED and score that gives a good ranking. In your final output the records should not be normalized.

[turn over with intensity]

As a naïve baseline, implement a brute-force approach that computes the PED against all records. For the queries in the table on our Wiki, measure the time as well as the number of PED computations for the baseline and your q-gram index. Report your result following the examples of the rows already there.

Exercise 5 (optional)

Briefly(!) explain, in your *experiences.txt*, how a q-gram index can be used for wildcard search with a single $*$ at an arbitrary position. How much padding ($ at the beginning and at the end of each record from the dictionary) do we need and why?

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. An of course, commit the usual *experiences.txt*.

Note: We very much value your feedback. But please try to be brief and concise, then it's more fun for us to read a lot of it.