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 5

Submit until Wednesday, November 28 at 4:00pm

This exercise sheet is about implementing approximate word matching using a k-gram index. Put all your methods in a class *ApproximateMatching*. As usual, write a unit test for each non-trivial method you implement. Consider the explanations given in the lecture and the code design suggestions (*ApproximateMatching.H*) linked on the Wiki.

Exercise 1 (5 points)

Implement a method that builds a k-gram index for a given vocabulary and given k.

Exercise 2 (5 points)

Implement a method for computing the edit distance between two given strings x and y in time and space $O(|x| \cdot |y|)$.

Exercise 3 (5 points)

Implement a method that computes the union (not intersection, as it wrongly said in a previous version of this sheet) of a given (arbitrary) number of inverted lists of word ids (in time $O(N \log k)$, where N is the sum of the list sizes and k is the number of lists). If you have already implemented a k-way intersect for Exercise Sheet 1 or 2, you can easily adapt that code (just make sure you output each element from each list exactly once).

Exercise 4 (5 points)

Implement a method that, based on a your k-gram index from Exercise 1, and using the methods from Exercises 2 and 3, computes all words from the vocabulary that are within a given edit distance of a given query word.

Write a program that applies this method to the 1.000 query words linked on the Wiki, for $k = 3$ and a max edit distance of $\lceil |w|/5 \rceil$, where $|w|$ is the length of the query word. Report the index construction time, average query time, average edit distance computation time, and average number of matches in the table linked on the Wiki, following the instructions given there.

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.