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

Exam

Friday, March 1, 2013, 14:10 - 15:50, Kinohörsaal in Building 082

General instructions:

There are six tasks, of which you can select *five tasks of your choice*. Each task is worth 20 points. If you do all six tasks, we will only count the best five, that is, you can reach a maximum number of 100 points.

You need 50 points to pass the exam. The exact correspondence between the number of points and the grade will be decided when we correct your exams.

You have 100 minutes of time overall. If you do only five tasks that is 20 minutes per task on average.

You are allowed to use any amount of paper, books, etc. You are not allowed to use any computing devices or mobile phones, in particular nothing with which you can communicate with others or connect to the Internet.

Please write your solutions on this hand-out! If you need additional pages, please write your Matrikelnummer and your name IN PRINTED LETTERS on each of them.

Good luck!

Task 1 (Inverted index + encoding, 20 points)

Consider the following non-sense document collection:

```
Document 1:  bla bla
Document 2:  bla bli bla
Document 3:  blu blu
Document 4:  bla blu bla
Document 5:  bli blu blu
```

1.1 (10 points) Write down the inverted lists for this document collection in gap-encoded form. The first gap in each inverted list is just the gap from 0 to the first document id in that list. For the example, an inverted list $1, 4$ would be gap-encoded as $+1, +3$. Scores are simply tf = term frequency. Each entry in an inverted list should thus be of the form (gap, score).

1.2 (10 points) Design two prefix-free entropy-optimal codes, one for the gaps and one for the scores. You only need codes for those gaps / scores that actually occur (*hint*: if you did task 1.1 correctly, there are just three different gaps and two different scores). Then write down the inverted lists in encoded form (for each inverted list entry, simply concatenate the code for the gap and for the score).

Task 2 (List intersection, 20 points)

Assume inverted lists are represented as bit arrays, where the ith bit is 1 if and only if i is in the list. For example, for a collection with 5 documents, the list $1, 2, 4$ would be represented as 11010. Scores are ignored in this task.

2.1 (10 points) Write the code for a function that intersects two inverted lists given as bit arrays and outputs the result again as a bit arrays. You can write the code in Java or in C++. Syntax details are not important. You can assume that you have a bit array data type with random access to each element.

2.2 (10 points) For comparison, consider the gap-encoded representation again, and assume that the probability distribution for the value X of a fixed gap is $\Pr(X = i) = 2^{-i}$. What then is the expected length of an entropy-optimally encoded list with m elements (and hence m gaps)? When is this encoding more efficient than the bit array representation from task 2.1. You can use without proof that $\sum_{i=1}^{\infty} i/2^i = 2$.

Task 3 (Error-tolerant search, 20 points)

Consider the following three words:

```
bla
bli
blu
```

3.1 (10 points) Write down the 2-gram index for these words (pad each word with one # on the left and on the right). For given words x and y with $\mathrm{ED}(x, y) \leq 1$, state how many 2-grams x' and y' will certainly have in common (where x' and y' are the padded versions of x and y, respectively).

3.2 (10 points) Use the 2-gram index and the lower bound from task 3.1 to determine which of the three words above have an edit distance ≤ 1 from the query word `blaa`. Use as few actual edit distance operations as possible. Explain each of your steps, and don't just write down the final result.

Task 4 (Latent semantic indexing, 20 points)

Consider the following 3×5 term-document matrix A and query Q:

D_1	D_2	D_3	D_4	D_5	Q
1	1	0	1	0.5	1
1	0	1	2	1	1
0	0	1	1	0.5	0

4.1 (10 points) Argue that the matrix A has rank 3. Then change a single entry such that the rank of the resulting matrix A' becomes 2. Prove this by writing A' as a product of a 3×2 matrix with a 2×5 matrix.

4.2 (10 points) Compute the cosine similarity between the query Q above and each of the five documents from A'. Then rank the documents according to these similarity scores. If two documents have the same similarity to the query, rank that one with the smaller index first (e.g. D_1 before D_2). Note that you do not have to compute the exact numerical values of the scores (there are square roots involved) to determine the ranking.

Task 5 (k-means, 20 points)

Assume the elements to be clustered are real numbers, and the distance between two elements x and y is just the absolute of their difference $|x - y|$. The average of a set of elements is just the average of the respective numbers.

5.1 (10 points) Perform 2-means clustering on the 10-element set $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Take 1 and 2 as the initial cluster centroids. In the (re-)assignment step of 2-means, when an element has the same distance from both centroids, assign it to the centroid from the smaller cluster (= the number of elements from which the centroid was computed is smaller).

5.2 (10 points) Prove that for any choice of two different elements from S as initial cluster centroids, 2-means will converge to the same clustering (namely, the one from task 5.1 above).

Task 6 (Naive Bayes, 20 points)

Consider the following string objects, each with one of two class labels (`X` and `Y`):

`xy`	`Y`
`xxxy`	`Y`
`xyy`	`X`
`xxxyy`	`X`
`xxxxxyy`	`X`
`xxxxxxxyy`	`X`

6.1 (10 points) Do the learning step of Naive Bayes on these objects, considering each object as a 2-dimensional feature vector (n_x, n_y), where n_x is the number of occurrences of the letter x and n_y is the number of occurrences of the letter y. Then classify the first two objects from the training set (`xy` and `xxxy`) using the learned classifier.

6.2 (10 points) Prove that the Naive Bayes classifier from task 6.1 will classify *any* object as `X`.

Draw the six objects in a $2D$-coordinate system (n_x on the x-axis, n_y on the y-axis). Also draw (in a different color) the seperating line + margin for an SVM trained on these objects without allowing outliers. You can tell this from the geometry of the points alone, no need to do any calculations.