Name:	Matrikelnummer:

Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Claudius Korzen

Information Retrieval
WS 2017/2018

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

Exam

Monday, February 19, 2018, 14:00 - 16:30, Lecture Halls 101-036 and 082-006

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. You have 150 minutes of time overall. If you do five tasks, this is 30 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 or parallel universes.

You may write down your solutions in either English or German.

Please write your solutions on this hand-out, below the description of the tasks! You can also use the back side of the pages. Please write your name and Matrikelnummer on the top of this cover sheet in the framed box. If you need additional pages, please write your name and Matrikelnummer on each of them, too.

Important:

For the programming tasks, you can use Python, Java, or C++. None of your functions should be longer than TEN lines, otherwise you risk point reduction.

For all other tasks: do not simply write down the final result; it should also be clear how you derived it.

Good luck!

T1	T2	T3	T4	T5	T6	Points	Grade

Task 1 (Ranking and Evaluation, 20 points)

Assume that we distinguish between four grades of relevance (0, 1, 2, 3), where 0 means not relevant, 1 means slightly relevant, 2 means somewhat relevant, and 3 means very relevant. Consider a query with the following result list and relevances:

Result #1:	not relevant	(0)
Result #2:	very relevant	(3)
Result #3:	not relevant	(0)
Result #4:	somewhat relevant	(2)
Result #5:	slightly relevant	(1)

1.1 (5 points) Consider binary relevance, where the grades 0 and 1 count as not relevant and 2 and 3 count as relevant. Assume that there are three relevant results overall for the query above. Then compute the following measures: P@2, P@R, AP (average precision).

1.2 (5 points) Consider the original four grades of relevance and compute the following measures: DCG@2, iDCG@2, nDCG@2.

1.3 (5 points) Write a function *dcg* that computes the DCG@k for a given k and a given array *rels* that provides the relevances of all documents, in the order in which they appear in the result list. For the example above, the first five entries of that array would be $[0, 3, 0, 2, 1]$. You can assume that the array is not empty.

1.4 (5 points) Write a function *ndcg* that computes the nDCG@k for a given k and a given array *rels* of relevances like for 1.3. You can assume that there is a function *sort*, which sorts the elements of a given array and which has a second argument that specifies whether the elements are sorted in ascending or descending order.

Task 2 (Encodings, 20 points)

Consider the following *unary* encoding scheme: encode integer i by $0^{i-1}1$ (that is, $i-1$ times 0 followed by 1), for $i = 1, 2, 3, \ldots$. For example, the number 5 is encoded by 00001.

2.1 (5 points) Write a function *unary_encode* that encodes a given sequence of integers (given as an array) according to this encoding. The function should return an array of elements which are 0 or 1 each.

2.2 (5 points) Consider a sequence of n integers, where each integer is chosen uniformly at random from $\{1, \ldots n\}$, independently from the other integers. What is the expected number of bits of the encoded sequence? With explanation, of course.

2.3 (5 points) Is this encoding entropy-optimal for this distribution? With explanation, of course.

2.4 (5 points) Name a prefix-free entropy-optimal encoding for this distribution and prove that it is entropy-optimal.

Task 3 (Fuzzy search and UTF-8, 20 points)

3.1 (5 points) Let $x = \textit{cute}$ and $y = \textit{computer}$. Compute $\mathrm{ED}(x, y)$, $\mathrm{ED}(y, x)$, $\mathrm{PED}(x, y)$, and $\mathrm{PED}(y, x)$. It is sufficient if you provide plausible arguments for your numbers, you don't have to fill out the complete edit-distance table (it would take way too much time).

3.2 (5 points) Find two strings x and y with edit distance 2 such that the number of 3-grams that x' and y' have in common is exactly $\max\{|x|, |y|\} - 4$, where $x' = \$\$x\$\$$ and $y' = \$\$y\$\$$.

3.3 (10 points) Write a function *valid_utf8_char* that checks if a given array of exactly three bytes is a valid UTF-8 encoding of a single character. You can assume that you can compute with the individual bytes like with unsigned integers.

Task 4 (Naive Bayes, 20 points)

In the following, we consider Naive Bayes applied to text documents, and we assume that classes and words are represented by their ids, starting from 0. We also assume that if in prediction two classes get the same probability, the class with the larger id is predicted.

4.1 (10 points) Write a function *naive_bayes_predict* that predicts the class of a given document. The function takes three arguments: the training probabilities *pc* (given as a 1D array), the training probabilities *pwc* (given as a 2D array), and the ids of the words in the document (given as a 1D array, with ids repeated as many times as that word occurs in the document). Don't use any special linear-algebra functions, just simple for-loops. Don't use smoothing. You can disregard rounding problems (that is, you don't have to work with log probabilities).

4.2 (5 points) Let d_1 and d_2 be two documents, where d_2 simply consists of two copies of d_1. That is, if $d_1 = (w_1, \ldots, w_k)$, then $d_2 = (w_1, \ldots, w_k, w_1, \ldots, w_k)$. Prove that if the *pc* are all equal, then Naive Bayes will, for any *pwc* and any $w_1, \ldots, w_k$, make the same prediction for d_1 as for d_2.

4.3 (5 points) Is the statement from 4.2 still true if the *pc* are not all equal? Prove it or provide a counterexample.

Task 5 (Latent Semantic Indexing, 20 points)

Consider the following matrix:

$$A = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 2 & 1 & 3 & -1 \end{pmatrix}$$

5.1 (10 points) Determine the matrices U and S of the singular value decomposition of A.

5.2 (10 points) Write a function *compute_V* that computes the V of the singular value decomposition, given A, U, S and the dimensions $m \times n$ of A. Assume that A and U are given as two-dimensional arrays and that S is given as a one-dimensional array with just the diagonal entries of S. Also assume that V is already given in the right shape with all entries zero, so that you only have to set the entries to the right values. Your function should not use any special linear-algebra functions, just simple for-loops.

Task 6 (Hidden Markov Model, 20 points)

Consider a HMM with three observables *good*, *great*, *bad* and two hidden states *plus*, *minus*, and the following initial probabilities (left), transition probabilities (middle), and emission probabilities (right). There is no special END state for this task.

plus	1/2
minus	1/2

	plus	*minus*
plus	1/3	2/3
minus	2/3	1/3

	good	*great*	*bad*
plus	1/3	2/3	0
minus	2/3	0	1/3

6.1 (5 points) Given the sequence *good good good*, what is the probability that the second state is *plus*?

6.2 (5 points) Given the sequence *great bad great*, what is the most likely sequence of hidden states? Don't execute the complete scheme, it takes too long. However, you must provide a proper argument for your answer.

6.3 (5 points) Write a function *compute_likelihood* that computes the likelihood of a given sequence of observations and a given sequence of hidden states. You can assume that the initial, transition, and emission probabilities are available as global variables *iprob*, *tprob*, *eprob* (as arrays or dictionaries, whatever you prefer).

6.4 (5 points) Write a function *brute_force_hmm* that computes the most likely sequence of hidden states for a given sequence of *exactly three* observations by simply trying out all possible sequences of hidden states. You can use the function from 6.3 and assume the same global variables.