Name: Matrikelnummer:

Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Allthetu Tors

Information Retrieval
WS 2021/2022

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

Exam

Friday, 4th of March 2022, 14:00 - 16:15 h, HS 026

General instructions:

There are five tasks, of which you can select *four tasks of your choice*. Each task is worth 25 points. If you do all five tasks, we will only count the best four. That is, you can reach a maximum number of 100 points.

You need 50 points to pass the exam. You have 2 hours 15 minutes of time overall. If you do four tasks, this is 30 minutes per task on average plus 15 minutes buffer time.

You are allowed to use one DIN A4 sheet of paper with any contents of your choice related to the lectures and the exercises. You can write on the front and on the back and it can be hand-written or a printout, but it must be from yourself, not from someone else. 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 must be longer than TWENTY lines.

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

Good luck!

This is the version of the exam with solution sketches. Do not distribute, it's for your personal use only. Don't use it when you do old exams for training, it's the worst way to learn. Use it only for checking your solutions, after you tried to solve the tasks yourself.

Task 1 (Ranking, Evaluation and List Intersection, 25 points)

1.1 (5 points) Consider the following collection of four documents D_1, ..., D_4. Write down the term-document matrix for this collection with *tf.idf* scores.

D_1: bla bla bla
D_2: bli blu blo
D_3: bla blo
D_4: blu blu

The *idf* for bli is 2, and for all other words 1. Hence the *tf.idf* matrix:

	bla	bli	blu	blo
D_1	3	0	0	0
D_2	0	2	1	1
D_3	1	0	0	1
D_4	0	0	2	0

1.2 (5 points) Consider a query "bla bli". Compute the dot-product similarities of the four documents above (using your *tf.idf* scores from 1.1) to that query.

With A the term-document matrix from above and $q = (1, 1, 0, 0)$, $q \cdot A = (3, 2, 1, 0)$.

1.3 (5 points) Rank the four documents by the scores from 1.2 (highest score first). Assume that documents D_1 and D_3 are relevant. Compute the following metrics: P@2, P@R, AP (average precision).

The ranking is (D_1, D_2, D_3, D_4). P@2 = P@R = 0.5, AP = (1 + 2/3)/2 = 5/6.

1.4 (10 points) Given an array A with integer values in ascending order, we want to locate element x in A with galloping search (an exponential search followed by a binary search). Prove that, if x is contained in A, the algorithm needs $O(\log d)$ time, where d is the position of x in A.

1. Let $j_1, ..., j_k$ be the search positions in the exponential search, where $j_i = 2^i - 1$.
2. Since $j_{k-1} < d$, we have $2^{k-1} - 1 < d$ and hence $2^{k-1} \leq d$ and $k \leq \log_2 d + 1$.
3. It follows that $j_k \leq 2^{\log_2 d+1} - 1 < 2d$. The binary search halves the search interval in every step, so we have to do at most $O(\log(2d)) = O(\log 2 + \log d) = O(\log d)$ steps.
5. Putting both parts together, the algorithm needs $O(\log d)$ time.

Task 2 (Zipf's law, Compression, and UTF-8, 25 points)

2.1 (8 points) Zipf's law states the frequency F of the n-th most frequent term in a text collection as a function of n. State the law and prove that F shows as a straight line with negative slope in a log-log plot.

1. Zipf's law states that $F(n) = c/n^\alpha$, for some (positive) constants c and α.
2. In a normal x-y plot, $x = n$ and $y = c/x^\alpha$.
3. Taking the log on both sides, we get $\log y = -\alpha \cdot \log x + \log c$.
4. That is, with $\log x$ on the x-axis and $\log y$ on the y-axis, we see a line with negative slope.

2.2 (7 points) Write the following numbers in Elias-Delta encoding: 1, 7, 16. For each code, underline the part that comes from the Elias-Gamma encoding.

$1 = \underline{1}$
$7 = \underline{011}\,11$
$16 = \underline{00101}\,0000$

2.3 (5 points) Consider the following encoding: $a = 000$, $b = 001$, $c = 01$, $d = 10$, $e = 11$. Find a distribution over $\{a, b, c, d, e\}$ such that this encoding is entropy-optimal, with a proof that it indeed is.

1. For perfect entropy-optimality, $L_i = \log_2 1/p_i$, where L_i is the length of the code for symbol i.
2. We here have $L_1 = 3$, $L_2 = 3$, $L_3 = 2$, $L_4 = 2$, $L_5 = 2$.
3. We thus get perfect entropy-optimality with $p_1 = 1/8$, $p_2 = 1/8$, $p_3 = 1/4$, $p_4 = 1/4$, $p_5 = 1/4$.

2.4 (5 points) Give an example of a two-byte UTF-8 sequence that is a *valid* encoding of a single Unicode character. Give an example of a two-byte UTF-8 sequence that is an *invalid* encoding of a single Unicode character. With explanation! Specify each sequence as a bit sequence and as a sequence of four hexadecimal digits.

1. Bits: 1100 1111 1011 1111. Hex: $CFBF$. Valid because codepoint doe not fit into 7 bits.
2. Bits: 1100 0001 1011 1111. Hex: $C1BF$. Invalid because codepoint fits into 7 bits.

Task 3 (SPARQL and fuzzy prefix search, 25 points)

3.1 (5 points) Assume we have a knowledge base which includes relations for *nationality* (relating persons to countries), *place_of_birth* (relating persons to their place of birth), *language_spoken* (relating countries to languages), *olympic_discipline* (relating persons to their olympic disciplines). Express the query Olympians from German-speaking countries with their place of birth and their olympic discipline in SPARQL.

```
SELECT ?p ?b ?d WHERE {
    ?p <place_of_birth> ?b .
    ?p <nationality> ?c .
    ?c <language_spoken> <German> .
    ?p <olympic_discipline> ?d .
}
```

3.2 (10 points) Write a function *ped(x, y)* that computes the PED between two strings x and y in $O(|x| \cdot |y|)$ time.

```
def ped(x, y):
    matrix = [[0 for _ in range(len(y)+1)] for _ in range(len(x)+1)]
    for row in range(len(x)+1):
        for col in range(len(y)+1):
            if row == 0:
                matrix[row][col] = col
            elif col == 0:
                matrix[row][col] = row
            else:
                s = 0 if x[row - 1] == y[col - 1] else 1
                rep_costs = matrix[row - 1][col - 1] + s
                add_costs = matrix[row][col - 1] + 1
                del_costs = matrix[row - 1][col] + 1
                matrix[row][col] = min(rep_costs, add_costs, del_costs)
    return min(matrix[len(x)])
```

3.3 (10 points) Let x and y be non-empty strings, and let $\delta \in \mathbb{N}$ such that $|x| + \delta < |y|$ and $\text{PED}(x, y) \leq \delta$. Proof that under these conditions $\text{PED}(x, y) = \text{PED}(x, y')$, where y' is the prefix of length $|x| + \delta$ of y.

1. $\text{ED}(x, y[1 \ldots |x| + \delta + i]) > \delta$ for all $i \in \{1 \ldots |y| - (|x| + \delta)\}$ since at least $\delta + i$ characters have to be deleted to transform x into y.
2. Since $\text{PED}(x, y) \leq \delta$ there is some $j \in \{1 \ldots |x| + \delta\}$ such that $\text{ED}(x, y[1 \ldots j]) \leq \delta$.
3. Since PED is defined as the minimal $\text{ED}(x, y')$ for all prefixes y' of y and since $y[1 \ldots j]$ is a prefix of both $y[1 \ldots |x| + \delta]$ and y it follows that $\text{PED}(x, y) = \text{PED}(x, y')$, where y' is the prefix of length $|x| + \delta$ of y.

Task 4 (Web Apps and Naive Bayes, 25 Punkte)

4.1 (5 Points) Write a valid HTML page with heading *Hardle*, three text input fields with ids *f1*, *f2*, *f3*, and a button labeled *Guess*. The HTML should include a JavaScript file *script.js*, to be written in Task 4.2.

```
<html>
  <head><script src="script.js"></head>
  <body><h1>Hardle</h1>
    <input type="text" id="#f1"/><input type="text" id="#f2"/>
    <input type="text" id="#f3"/><button>Guess</button></body>
</html>
```

4.2 (10 Points) Write *script.js* with the following functionality. Whenever the user presses the *Guess* button, send the content of input field *f1* to a backend at the relative URL */backend*. The backend returns a JSON object with 3 integer attributes *correct_positions*, *correct_letters* and *tries_left*. Show the value of *correct_positions* and *correct_letters* in input fields *f2* and *f3*. If *correct_positions* matches the length of the content of field *f1*, show *You win* in field *f1*. If field *tries_left* is 0, show *You lose*. You may use *jQuery*.

```
$("button").click(function() {
  $.get("/backend?q=" + $("#f1").val(), function(res) {
    if (res["correct_positions"] == $("#f1").text().length) $("#f1").text("You win");
    else if (res["tries_left"] == 0) $("#f1").text("You lose");
    else {
      $("#f2").text(res["correct_positions"]);
      $("#f3").text(res["correct_letters"]);
}}});
```

4.3 (10 Points) Consider each number from 0..15 as a document with four words, where the words are 0 or 1 and the ith word stands for the ith bit in the binary representation of the number. For example, document 1100 is the number 3. Each document is labeled with class A if the number is even, and with class B if the number is odd. Train a Naive Bayes classifier on these 16 labeled documents. Determine all probabilities p_c and p_{wc} (and write down the intermediate steps). Show that using these probabilities, Naive Bayes predicts class A for the number 1.

1. Of the 16 documents/numbers, 8 are even and 8 are odd, hence $p_A = p_B = 1/2$.
2. All documents have the same length, hence $n_A = n_B = 8 \cdot 4 = 32$.
3. For A, the first bit is always 0 and of the other bits half are 0 and half are 1.
4. So $n_{0A} = 8 + 12 = 20$ and $n_{1A} = 12$, and similarly, $n_{0B} = 12$ and $n_{1B} = 20$.
5. Therefore $p_{0A} = p_{1B} = 20/32 = 5/8$ and $p_{1A} = p_{0B} = 12/32 = 3/8$.
6. $\Pr(A|1000)/\Pr(B|1000) = p_{1A} \cdot p_{0A}^3/(p_{1B} \cdot p_{0B}^3) = 5^2/3^2 > 1$, so 1 is classified as A.

Task 5 (Latent Semantic Indexing and Linear Classifiers, 25 points)

5.1 (10 points) Write a function *top(q, U, S, V, k)* that returns the index of the top-ranked document for a given query vector q, the *numpy* matrices U, S, V from the singular value decomposition, and the dimension k of the approximation. Use the function *transpose* for transposition and *dot* for matrix-matrix or matrix-vector products. In your code, indicate (using a different color) the dimensions of the vectors and matrices and make sure that they match.

```
def top(q, U, S, V, k):
  Uk = U[:, 0:k]
  Sk = S[0:k, 0:k]
  Vk = V[0:k, :]
  scores = q.transpose().dot(Uk).dot(Sk).dot(Vk)
  return numpy.argmax(scores)
```

5.2 (5 points) Let $H = \{x : 2 \cdot x_1 - 2 \cdot x_2 + x_3 = 7\}$ be a 2D plane in 3D space. Compute the distance of the point $x = (1, 2, 3)$ to H. If you know the formula for the distance computation, you can just use it. Otherwise, derive the formula by writing x as the sum of a point on H and a multiple of the unit normal vector of H; the multiple is then the distance.

1. Let $w = (2, -2, 1)$ denote the normal vector of H. It's length is $|w| = \sqrt{4+4+1} = 3$.
2. The distance from x to H is then $|w \bullet x - 5|/|w| = |2 - 4 + 3 - 7|/3 = 2$.

5.3 (10 points) Let $\sigma(t) = 1/(1 + e^{-t})$ be the sigmoid function. Prove that $\sigma'(t) = \sigma(t) \cdot \sigma(-t)$.

1. By the chain rule, $\sigma'(t) = -1/(1 + e^{-t})^2 \cdot (-e^{-t})$.
2. This can be equivalently written as $1/(1 + e^{-t}) \cdot e^{-t}/(1 + e^{-t})$.
3. The first factor is just $\sigma(t)$ again.
4. Multiplying both numerator and denominater in the second factor by e^t, we obtain $1/(1 + e^t)$.
5. The second factor is thus $\sigma(-t)$, and so we get $\sigma'(t) = \sigma(t) \cdot \sigma(-t)$.