Name:	Matrikelnummer:

Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Natalie Prange

Information Retrieval
WS 2022/2023

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

UNI FREIBURG

Exam

Tuesday, 28th of February 2023, 11:00 - 13:15 h, in building 101 SR 00-010/14, SR 01-009/13 and SR 02-016/18

General instructions: There are five tasks. Each task is worth 20 points, 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. This is 20 minutes per task plus 35 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 handwritten 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 handout, 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 should use Python. None of your functions must be longer than *fifteen* 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 (Exponential Search, Elias-Gamma, Lagrangian multipliers, 20 points)

1.1 (5 points) Implement a function *exponential_search(A, x)*, which takes an array A of integers sorted in ascending order and an integer x. You can assume that x is contained in A. Let j be the smallest index for which $A[j] = x$. The function should compute an index i with $j \leq i \leq 2j$ and the function should run in $O(\log(1 + j))$ time.

```
def exponential_search(A, x):
    i = 0
    while A[i] < x:
        i = min(max(2 * i, 1), len(A) - 1)
    return i
```

1.2 (5 points) Write down the Elias-Gamma codes for $x = 3, 4, 7, 8, 15, 16$ with the leading zeroes underlined. Then write down the code length (in number of bits) for each of these numbers. Then write down the general formula for the code length for an abritrary number $x \in \mathbb{N}$ and briefly explain why it's correct.

1. The Elias-Gamma codes are as follows, with the leading zeroes underlined:
   $3 = \underline{0}11$, $4 = \underline{00}100$, $7 = \underline{00}111$, $8 = \underline{000}1000$, $15 = \underline{000}1111$, $16 = \underline{0000}10000$
2. The code lengths in bits for these numbers are: $3, 5, 5, 7, 7, 9$.
3. The general formula for the code length for number x is $2 \cdot \lfloor \log_2 x \rfloor + 1$
4. It's $\lfloor \log_2 x \rfloor$ for the leading zeroes and $\lfloor \log_2 x \rfloor + 1$ for the binary representation of x

1.3 (10 points) Let $p_1, \ldots, p_n$ be a probability distribution over n symbols. Determine the probability distribution for which the entropy is maximized, using the method of Langrange multipliers. You can assume without proof that the assumptions for applying the method are satisfied and that the point where all partial derivatives of the Lagrangian function are zero is a global maximum.

1. The entropy is defined as $H = -\sum_{i=1}^{n} p_i \cdot \log_2 p_i$
2. We want to maximize H under the constraint that $\sum_{i=1}^{n} p_i = 1$
3. The corresponding Lagrangian is $L = -\sum_{i=1}^{n} p_i \cdot \log_2 p_i + \lambda \cdot (\sum_{i=1}^{n} p_i - 1)$
4. The partial derivatives with respect to a p_i is $\partial L / \partial p_i = -\log_2 p_i - 1/\ln 2 + \lambda$
5. These n partial derivatives are all zero if and only if the p_i are all equal
6. The partial derivative with respect to λ is $\partial L / \partial \lambda = \sum_{i=1}^{n} p_i - 1$
7. So all partial derivatives are zero if and only if $p_i = 1/n$ for all i.

Task 2 (Q-Grams and Edit Distance, 20 points)

2.1 (5 points) Write down the edit distance between the strings $x = alfa$ and $y = alpha$ and the sequence of operations needed to get from x to y (you don't have to write down the table). Then pad the words with dollar signs on both ends, as appropriate for a 3-gram index, and write down the set of 3-grams that the two padded strings have in common.

1. The edit distance is 2.
2. A possible sequence of operation is: replace f by p, then insert the missing h.
3. The set of common q-grams is $\{\$\$a, \$al, a\$\$\}$.

2.2 (5 points) Calculate the minimal possible edit distance between two strings of length 5 that have three 3-grams in common when using padding on both sides. Write down the intermediate steps for the calculation.

1. Let x and y denote the two strings without padding, and x' and y' the versions with padding.
2. Then $\mathrm{ED}(x, y) = \mathrm{ED}(x', y')$ and the number of 3-grams of x' and y' is $|Q_3(x')| = |Q_3(y')| = 7$.
2. From the lecture we know that $|Q_3(x') \cap Q_3(y')| \geq \max\{|Q_3(x')|, |Q_3(y')|\} - 3 \cdot \mathrm{ED}(x, y)$.
3. With $|Q_3(x') \cap Q_3(y')| = 3$ and $\max\{|Q_3(x')|, |Q_3(y')|\}$, we have $3 \geq 7 - 3 \cdot \mathrm{ED}(x, y)$.
4. The smallest edit distance, for which is inequality is not violated is $\mathrm{ED}(x, y) = 2$.

2.3 (10 points) The general algorithm for computing the edit distance between two strings of length n and m runs in $O(n \cdot m)$. Write a function *one_delete(x, y)* that runs in time $O(m + n)$ and returns *True* if the string x can be transformed into the string y with exactly one *delete* operation, and *False* otherwise.

```
def one_delete(x, y):
    if len(x) 1 != len(y) + 1:
        return False
    numDel = 0
    for i in range(len(y)):
        if x[i + numDel] != y[i]:
            if numDel >= 1:
                return False
            numDel += 1
    return numDel == 1
```

Task 3 (Ranking and Linear Classifiers, 20 points)

3.1 (8 points) You are given the following term-document matrix (3 terms, 5 documents):

$$A = \begin{bmatrix} 1 & 2 & 0 & 1 & 5 \\ 4 & 1 & 0 & 2 & 3 \\ 1 & 1 & 2 & 1 & 0 \end{bmatrix}$$

State a query $q \in \mathbb{N}^3$ such that $q^T A$ returns the following document ranking in descending order (ties will be decided in favor of lower document numbers): $1, 3, 4, 2, 5$. Which documents should be relevant for q such that the ranking gets a P@R score of 0.5 and a P@4 score of 0.25? Calculate the average precision of your solution.

1. The query $q^T = (0, 1, 3)$ produces the scores $(7, 4, 6, 5, 3)$ and hence the desired ranking.
2. If documents 3 and 5 are relevant, we have P@R = P@2 = 0.5 and P@4 = 0.25.
3. The average precision then is $(\text{P@2} + \text{P@5})/2 = (0.5 + 0.4)/2 = 0.45$.

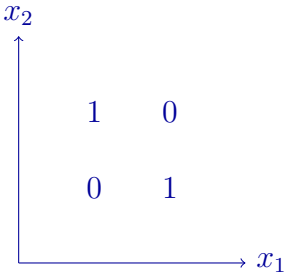
3.2 (5 points) Let $H = \{x : w \bullet x = b\}$ be a hyperplane in $\mathbb{R}^d$ with $w \in \mathbb{R}^d, w \neq \mathbf{0}$ and $b \in \mathbb{R}$. For which b does H contain the origin (with proof)? Now assume that $d = 4$, $w = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, and $b = 0$. Compute the distance of the point $p = (1, 1, 1, 1)$ to H.

1. The origin $\mathbf{0}$ is contained in H if and only if $w \bullet \mathbf{0} = b$.
2. When w is not all zero, this can only happen if $b = 0$.
3. The distance of a point p to H can be computed via $(w \bullet p - b)/|w|$.
4. For the given w, we have $|w| = 1$, and so the distance is $w \bullet p - b = 2$.

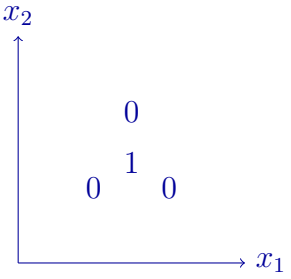
3.3 (7 points) The *F1 score* of the predictions for a class is defined as $2/(1/P + 1/R)$, where P is the precision and R is the recall. Prove that $F1 = 1$ if and only if $P = 1$ and $R = 1$. Draw four two-dimensional points, each with a label from one of two classes, such that no linear classifier can achieve $F1 = 1$ for both classes. Briefly explain why your points have this property.

1. If either $P < 1$ or $R < 1$, we have $1/P + 1/R > 2$ and hence $F1 < 1$.
2. Therefore $F1 = 1$ can only be achieved when $P = R = 1$.
3. To achieve $F1 = 1$ for both classes, all points must be classified correctly. Here are two examples of four points in $\mathbb{R}^2$ with labels, such that no hyperplane (which in $\mathbb{R}^2$ is a line) exists, which separates the labels:

$$D_1 = \left[\begin{array}{cc|c} 1 & 1 & 0 \\ 2 & 2 & 0 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{array}\right]$$



$$D_2 = \left[\begin{array}{cc|c} 1 & 1 & 0 \\ 2 & 1 & 0 \\ \frac{3}{2} & 2 & 0 \\ \frac{3}{2} & \frac{4}{3} & 1 \end{array}\right]$$



Task 4 (Web Apps and Naive Bayes, 20 points)

4.1 (10 points) Write a function *predict(doc, wordIds, pwc, pc)* that expects a document (as an array of strings), the word IDs (as a map where *wordIds[<word>]* gives the ID), the p_{wc} learned during training of Naive Bayes (as an array of k arrays, one for each of the k classes, and all entries are non-zero) and the p_c (as a one-dimensional array). The function should return the most likely class and its probability. Do not use smoothing. You can ignore rounding problems.

```
def predict(doc, wordIds, pwc, pc):
  c_best, p_best = -1, -1
  for c in range(0, len(pc)):
    p = pc[c]
    for w in doc:
      wid = wordIds[w]
      p = p * pwc[c][wid]
    if p > p_best:
      c_best, p_best = c, p
  return c_best, p_best
```

4.2 (5 points) Write an HTML page with heading *Enter Query*, an input field, a paragraph, and a button labeled *Predict*. The input field and the paragraph should have an ID. The HTML should include a JavaScript file *script.js*, to be written for Task 4.3.

```
<html>
  <body><h1>Enter Query</h1>
  <input type="text" id="query"/><p id="result"></p>
  <button>Predict</button>
  <script src="script.js"></script></body>
</html>
```

4.3 (5 points) Write *script.js* with the following functionality: If the user presses the button, send the content of the input field to a server using a GET request to a URL of your choice. Assume that the backend implements the method from Task 4.1 and returns a single JSON object with fields *class* and *probability*. Replace the content of the paragraph with the predicted class and its probability, separated by a space.

```
document.querySelector("button").onclick = async function() {
  let q = document.querySelector("#query").value;
  let ob = await fetch("/backend?q=" + q).then(res => res.json());
  document.querySelector("#result").innerHTML = ob.class + " " + ob.probability;
};
```

Task 5 (Latent Semantic Indexing and SPARQL, 20 points)

5.1 (10 points) Write a function *top(q, A, U, k)* that returns the index of the top-ranked document using the document expansion method (the third variant discussed in the lecture) and takes the following parameters, each as a *numpy.array*: the query vector q (as one column), the original term-document matrix A, the U from the SVD of A, and the dimension k of the approximation. Use the function *transpose* for transposition and *dot* for a matrix-matrix or matrix-vector product.

```
def top(q, A, U, k):
  Uk = U[:, :k]
  Tk = Uk.dot(Uk.transpose())
  scores = q.transpose().dot(Tk).dot(A)
  return scores.argmax()
```

5.2 (5 points) Let A be an $m \times n$ matrix with rank r, and let $A = U \cdot S \cdot V$ be the singular value decomposition, where U is an $m \times r$ column-orthonormal matrix, S is a diagonal $r \times r$ matrix, and V is an $r \times n$ row-orthonormal matrix. Prove that $V = S^{-1} \cdot U^T \cdot A$.

$$
\begin{aligned}
& & U \cdot S \cdot V &= A & \\
\Rightarrow & & U^T \cdot U \cdot S \cdot V &= U^T \cdot A & U^T \cdot U = I_r \\
\Rightarrow & & S \cdot V &= U^T \cdot A & \\
\Rightarrow & & S^{-1} \cdot S \cdot V &= S^{-1} \cdot U^T \cdot A & S^{-1} \cdot S = I_r \\
\Rightarrow & & V &= S^{-1} \cdot U^T \cdot A &
\end{aligned}
$$

Remark concerning S^{-1}: The entries $s_1, \ldots, s_r$ in the diagonal matrix S are all non-zero. Then the inverse S^{-1} exists and is also a diagonal matrix, with values $1/s_1, \ldots, 1/s_r$.

5.3 (5 points) Assume that we have a knowledge graph which includes relations for *occupation* (relating persons to their occupation), *place_of_birth* (relating persons to their place of birth, which you can assume to be unique) and *educated_at* (relating persons to the university they studied at). Express the following question as a SPARQL query: YouTubers who studied at the same university as MrBeast, each with their place of birth.

```
SELECT ?p ?b WHERE {
  <MrBeast> <educated_at> ?u .
  ?p <educated_at> ?u .
  ?p <occupation> <YouTuber> .
  ?p <place_of_birth> ?b .
}
```