Assignment 2

Information Retrieval and Text Mining 22/23

Publication: 2022-11-10
Submission Deadline: 2022-11-24
Discussion Session: 2022-12-01

Roman Klinger
Yarik Menchaca, Tobias Schmid

irtm-teachers@ims.uni-stuttgart.de

- **Groups:** Working in groups of up to three people is encouraged, up to four people is allowed. More people are not allowed. Copying results from one group to another (or from elsewhere) is not allowed. Changing groups during the term is allowed.
- **Grading:** Passing the assignments is a requirement for participation in the exam in all modules IRTM can be part of. Altogether 80 points need to be reached. There are five assignments with 20 pen & paper points and 10 programming points each. That means, altogether, 150 points can be reached.
- **Submission:** First make a group in Ilias, then submit the PDF. Write all group members on the first page of the PDF. Only submit *one* PDF file. If you are technically not able to make a group (it seems that happens on Ilias from time to time), do not submit a PDF multiple times by multiple people – only submit it once. Submission for the programming tasks should also be in the same PDF.
- **Make it understandable:** Do the best you can such that we can understand what you mean. Explain your solutions, comment your code. Print the code in a readable format, write your solutions in a way we can read them.
- **Handwriting:** We typically receive some submissions which are handwritten. That is fine, but if you submit handwritten solutions, make sure that they are well organized, easy to read and to understand, and that there is not doubt about the interpretation of letters. If you think that this might be hard, please typeset the solutions with a computer. We might reduce points if it's really tough for us and cannot read your submission properly.
- **Language:** Please submit your solutions in English. We have limited capacity of correcting German submissions.

Task 1 (4 points)

Answer the following questions about distributed indexing:

- What information does the task description contain that the master gives to a parser?
- What information does the parser report back to the master upon completion of the task?
- What information does the task description contain that the master gives to an inverter?
- What information does the inverter report back to the master upon completion of the task?

Task 2 (4 points)

Heaps' law is an empirical law.

Assume that you have a collection with the following properties:

dataset	collection size	vocabulary size
subset 1	1,000,000	10,000
subset 2	100,000	3,000

Subtask 2.1

Compute the coefficients k and b.

Subtask 2.2

Compute the expected vocabulary size for a larger collection (100,000,000 tokens).

Task 3 (4 points)

Calculate the variable byte code and the gamma code for 216.

Task 4 (4 points)

¿From the following sequence of γ-coded gaps, reconstruct first the gap sequence and then the postings sequence:

1111011000100110000

Task 5 (4 points)

We have the following dictionary content given:

- A AB ABACUS ABACUSES ABAFT ABAKA ABAKAS ABALONE ABALONES ABAMP ABAMPERE ABAMPERES ABAMPS AC

How much memory do you need if you allocate a fixed amount of memory to each string (i.e., the minimum required given these instances)?

How much do you save if you store all these strings as one string? (called "Dictionary as a string" in the lecture)

How much memory would you save if you do blocking in addition (no front coding)? (set parameters like the memory consumption of a pointer appropriately, if necessary)

Programming Task 2 (10 points)

Implement the functionality to use wild card queries from scratch. Do not use regular expressions which your programming language might support.

Choose between one of the following subtasks:

Subtask 1

a)

This task is more suitable if **you do not have** an implementation you can build on top of from assignment 1. Still, use the data from Assigment 1 for this task.

Implement both a **bigram index** and a **permuterm index**. You do not need to integrate these methods in the postings lists or with the intersection algorithm. Only **implement the method that is required to find candidate entries in the term dictionary**.

Test both your methods (bigram, permuterm) with the vocabulary from the data from Assignment 1, with the following queries:

- `zeit*`
- `*zeit`
- `ze*st`

Please output for each query:

- For the bigram index:
  - Which words you found as **candidates to match** (only show up to 10)
  - Which words you actually **return as a match** (only show up to 10)
- For the permuterm index:

  - The **rotation** of the query
  - Which words you find as a **match** (only show up to 10)
- For both, the **runtime** for each query, without the time to load the index.

Please explain your approach (in addition to the commented code).

b)

Please play around with some other queries with wildcards: can you find a query in which the number of words that you actually return is much smaller than the set of candidates you found?

As usual, submit your commented code in the pdf, explain your solution, and discuss alternatives to the approach you have chosen (or explain why you have chosen the approach that you implemented).

Subtask 2

This task is more suitable **if you do have** an implementation you can build on top of from assignment 1.

Choose bigram index or permuterm index and **implement one of these approaches**. Define **4 queries** which contain **two search terms connected with AND**. At least one of the two search terms should contain a wildcard. (1) One of the four queries should have the wildcard on the left, (2) one should have the wildcard on the right, (3) one should have a wildcard between other characters and (4) one should have one wildcard on the left and one on the right.

Submit the programming code, comment your code, explain your code, list the queries you used in the submission together with (examples for) the documents that you find in the data. Discuss the results.