Assignment 2

Information Retrieval and Text Mining 20/21

Publication: 2020-11-24
Submission Deadline: 2020-12-08
Discussion Video Online: 2020-12-15
Live Q&A Session: 2020-12-17

Roman Klinger
Valentino Sabbatino, Tordis Daum, Tobias Schmid

- **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 received some submissions which were handwritten. That is fine, but they were sometimes barely readable. If you submit handwritten solutions, make sure that they are well organized, easy to read and understand, and that there is not doubt about the interpretation of letters. If you think that this might be hard, please typeset the solutions. We might reduce points if it's really tough for us.

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?

When you set up distributed indexing, one question you need to answer is how you distribute the available computers to jobs.

Please discuss:

- How would you specify the number of parsers? Can you estimate this somehow (helpful corner case scenario to think about: is one parser for each term a good choice? Is just one parser overall a good choice?)
- How would you specify the number of partitions from which the inverters put the index together? (corner cases: only one partition? Or one partition per term?)

Task 2 (4 points)

Let us assume we have a large index up-and-running which needs to receive large updates in a batch (adding documents, no deletions or changes to existing documents) every week. The update does not fit into an auxiliary index, and indexing the batch of new documents with one computer would take approximately two weeks. Hence we need to update the existing index in a distributed setting.

How can you do that? How can you use distributed indexing to update an existing index without slowing it down too much?

Task 3 (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 3.1

Compute the coefficients k and b.

Subtask 3.2

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

Task 4 (4 points)

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

Task 5 (4 points)

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

1111011000100110000

Programming Task 2 (10 points)

Implement the possibility 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

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 implement the postings lists or intersection algorithm, only the method that is required to find candidate entries in the term dictionary.

The goal is to evaluate the runtime of both index types empirically.

Implement a method which takes a word from the dictionary randomly and generates a wild card query automatically, using all of the four cases that we discussed in class ((1) wildcard on the right, (2) on the left, (3) in the middle or (4) two wildcards at the beginning and the end). You could do that by removing a couple of letters randomly and replace them by a wild card; the concrete approach is up to you.

Now implement a method which takes such a randomly generated wild card query and runs it against the bigram index or the permuterm index. This method returns the terms that are candidates to answer the query (but only prints them optionally, there should be switch for that).

Now, measure the runtime of your program for a large number of wild card queries (e.g. 1 million), once on the permuterm index and once on the bigram index.

Please explain your approach in text (in addition to the commented code) and report the runtimes. Further, show the output of the wildcard query for three of the randomly generated wildcards.

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.