Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
Patrick Brosi

Information Retrieval
WS 2019/2020

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

UNI FREIBURG

Exercise Sheet 2

Submit until Tuesday, November 5 at **12:00 noon.**

This exercise sheet works with a new dataset *movies2.txt*, which contains more text and is hence more interesting for this sheet. You find it on the Wiki.

Exercise 1 (8 points)

Extend *inverted_index.py* from the Wiki to incorporate BM25 scores, as explained in the lecture and as detailed in the following. You can also continue from your code for ES1; in that case make sure to follow the structure from the code template and include the new unit tests from there.

1. In the method *build_from_file*, add BM25 scores to the inverted lists. Pay attention to the implementation advice given in the lecture, and avoid unnecessary complexity. (4 points)

2. Change your method *intersect* into a method *merge* for merging two lists. That is, the resulting list should contain all elements from the input lists in sorted order. Note that this can be done with a relatively minor change. (2 points)

3. In your method *process_query*, sort the results by the aggregated BM25 scores. You don't have to implement the sorting algorithm yourself, you can use one of the built-in sorting functions. Your *main* function should still ask the user for keyword queries and output the title and description of up to three matching records, in the order returned by *process_query*. (2 points)

Exercise 2 (8 points)

Extend *evaluate.py* from the Wiki to automatically evaluates the quality of your system on a given benchmark as detailed in the following:

1. Write a function *read_benchmark* that reads each query with the associated set of relevant document ids from a file; see the code template regarding the format. (2 points)

2. Write two functions *precision_at_k* and *average_precision* that compute the measures $P@k$ and AP for a given list of result ids as it was returned by your inverted index for a single query, and a given set of ids of all documents relevant for the query. (2 points)

[the best is yet to come]

3. Write a function *evaluate* that evaluates a given inverted index against a given benchmark and computes the measures *MP@3*, *MP@R*, and *MAP* by aggregating *P@3*, *P@R* and *AP* for each query in the benchmark. (2 points)

4. Write a *main* function that takes the paths to a dataset (like *movies2.txt*) and a benchmark file (like *movies.training-benchmark.txt*) as command line arguments, constructs an inverted index from the dataset and does the evaluation of the inverted index against the benchmark using the methods above. Optionally, you can allow more arguments for the value of b and k or other parameters that influence the ranking. (2 points)

Exercise 3 (4 points)

Use the queries from *movies.training-benchmark.tsv* to improve search result quality in the following three ways. First, play around with the BM25 parameters b and k. Second, add or subtract keywords from the queries. Third, anything else to improve the ranking (we discussed various possibilities in the lecture). Do not look at *movies.test-benchmark.tsv* at this point, that is, while you are still tuning your system.

Write a short paragraph about your main insights from this step (successful improvements as well as unsuccessful ideas) in your *experiences.txt*.

Run the final version of your code on *movies.test-benchmark.tsv*. Feel free to add or subtract keywords from the queries, but you should do it only once. You should not tune your system to the test benchmark. Enter your results in a row in the table linked from the Wiki. Concerning the format, just follow the format of the rows which are already there.

Commit your code to our SVN, in a new subfolder *sheet-02*. The dataset and the benchmark file should not be committed.

As usual, in your *experiences.txt*, provide a brief account of your experience with this sheet and the corresponding lecture. As a minimum, say how much time you invested and if you had major problems, and if yes, where.o

What's the line number of your favorite movie in the file *movies2.txt*? If the line number has three digits or more, please explain.