Trier University
Department IV – Computer Sciences
Prof. Dr.-Ing. Ralf Schenkel (H 539, schenkel@uni-trier.de)

Trier, May 19, 2021

Assignment 5 of the course Big Data Analytics

Summer semester 2021

Deadline 10:15h on Wednesday, June 9, 2021

Task 1: Joins 25(+20) points

In this task we will again work with data that represents postings on StackOverflow. We will use again the (very small) excerpt of the data representing posts, available in Moodle as `post10000.xml`. In addition, we will use a second excerpt representing users, available in Moodle as `user10000.xml`, where one line represents one user. You can again use the Java class `MRDPUtils` for extracting information from one line of any of the two files.

In this task, our goal is to collect statistics about the users and their postings, so we need to compute the join of the two files. In `post10000.xml`, the id of the user who created the posting is available from the attribute `OwnerUserId`. In `user10000.xml`, the attribute storing the id of a user is `Id`.

(a) Write a MapReduce application that computes, for each user, the overall number of posts and the average score of the posts; the score of a post is available from the attribute `Score`. In addition, the name of the user (available from the attribute `DisplayName`) and the number of views of his or her profile (available from the attribute `Views`) should be reported; both are available in `user10000.xml`.
To solve this task, implement a reduce-side join for computing the join of `user10000.xml` and `post10000.xml` on `Id=OwnerUserId`.
Hints:

- Only lines from `post10000.xml` contain the attribute `OwnerUserId`. The mapper and the reducer can exploit this to identify tuples from the different input files.
- In the mapper, emit the value of the join attribute (`Id` for tuples from `user10000.xml`, `OwnerUserId` for tuples from `post10000.xml`) as the key and the complete line as value.
- In the reducer, read all values and, for each value, identify if the value corresponds to a user or a posting. Aggregate scores of postings in a variable of suitable type, and store information on the user in other variables.
- If postings do not have a corresponding user, they can be ignored. In this case, the reducer does not need to output anything.
- Otherwise, the reducer should output the user name as key and a String that encodes the requested information in some reasonable way as value.

Hand in your code and the output of your program for the given input files. To reduce file size, include only output for users with an id **between 20 and 40 inclusive**; this should yield 16 results.

(b) Implement a map-side join that uses the set of user ids that exist in `user10000.xml` as auxiliary information. For simplicity, create this set in the constructor of the Mapper class by reading and processing the file `user10000.xml` and storing the values of the `Id` attributes occurring in that file in an in-memory set data structure. The map function then operates only on lines from `post10000.xml`. For simplicity, the map function should only output the value of the `Id` attribute of a posting for which a corresponding user exists.

Hand in your code and the output of your program for the given input files. To reduce file size, include only output for users with an id **between 10 and 15 inclusive**. **This is an optional task.**

Task 2: n-grams 20 points

Write a MapReduce application that computes all word-level n-grams (with $2 \leq n \leq 4$) that appear **at least 1000 times** in the input text, using the naive approach discussed in the lecture. Represent an n-gram in a reasonable way. Similar to the `WordCount` example, use a `StringTokenizer` to split up a line into tokens, after removing punctuation and converting the text to lowercase.

Hand in your code and the output of the reducer when run on the input file `corpus.txt` which is available in Moodle.

These tasks will be discussed in the tutorial on **June 16**, 2021.

Note that there will be no meetings on Wednesday, May 26 (as there are no teaching events in that week) and on Wednesday, June 9 (due to a parallel project meeting). The deadline for submitting solutions to this assignment was therefore extended by another week to June 9 (with discussion of the solutions on June 16).

General remarks:

- The tutorial group takes place on Wednesdays in the regular Zoom meeting at 10:15, see StudIP (slides with general information on the lecture) for the registration link.
- To be admitted to the final exam, you need to acquire at least 50% of the points in the assignments.
- It is preferred to submit in groups of size 2 (but not larger); in that case, only one submission is sufficient for the whole group. Groups must be chosen in Moodle (see link on the course page in Moodle). Write the names of all group members on your solutions.
- Solutions must be handed in before the deadline in Moodle (`https://moodle.uni-trier.de/`, course `BDA-21`) as as a PDF or, if submitting multiple files, as an archive (.zip or comparable). Submissions that arrive after the deadline will not be considered.
- Graded versions of your submissions will be returned in Moodle until the following tutorial.
- Announcements regarding the lecture and the tutorial group will be done in the area of the lecture in StudIP.