Technische Universität Berlin

Database Systems and Information Management Group

AIM-3 Scalable Data Science | WS 2019/20

Assignment 3 (Worth: 100 points) | Due 31.01.2020 at 14.15

Instructions.

1. For all exercises, **show all work to obtain full credit**, where appropriate follow the additional instructions specific to the individual exercise.
2. Upload a zip file including your last name (**HW3_lastname.zip**) to the course portal no later than the announced deadline.
3. The **zip file** must contain the following: your *written solutions*, *source code* corresponding to the implementation of key algorithms, *screenshots highlighting error-free compilation* and the *initial frame from the results under execution* **in a single *.pdf* file** as well as the *execution results (e.g.,* either as a *.txt* or *.csv* file). Our intent is to ease grading. When information is scattered across numerous files, it creates a great deal of overhead.
4. **Optional**. Students who want feedback must drop off printed/stapled copies of their *.pdf* file.
5. *Note: You may be asked to meet with an instructor to run your codes at a later date.*
6. Lastly, work individually. You may drop hints, but your solutions must be your own.

A. Collaborative Filtering (20 points)

1. Solve Exercise 9.3.1 as described in the MMDS book. **(12 points)**

2. Solve Exercise 9.3.2 as described in the MMDS book. **(8 points)**

B. Computing Moments in Data Streams (10 points)

1. Solve Exercise 4.5.1 as described in the MMDS book. **(3 points)**

2. Solve Exercise 4.5.2 as described in the MMDS book. **(7 points)**

C. Stream Processing in Apache Flink (30 points)

In this exercise, you will analyze the *DEBS 2013 Grand Challenge*[1] dataset for real-time, event-based sports analytics using Apache Flink. *Below are some facts to keep in mind:*

- timestamps are expressed in *picoseconds* (ps), i.e., one-trillionth of a second (10^{-12}),
- the timestamp at **10,753,295,594,424,116** designates the *start of the match*,
- the timestamp at **879,639,146,403,495** marks the *end of the match*,
- the timestamp at **12,557,295,594,424,116** indicates that the *first half of the match has ended*,
- the timestamp at **13,086,639,146,403,495** signals the *end of the second half*.

[1] Find more details at https://www2.informatik.uni-erlangen.de/publication/download/DEBS2013b.pdf.

Additional information:

- *Code stubs* are here: *https://gitlab.tubit.tu-berlin.de/AIM3-SDS/ComputationalExercises*.
- The multi-GB dataset is stored as a compressed file in Google Drive.
- To access the data, download and extract the compressed file from: https://drive.google.com/file/d/1G65ZzH9j5XVtpjni95aZpIRGhthrB8n9/view?usp=sharing, which will yield a large CSV file.

Below are a few questions that you will need to solve. Be sure to adhere to the **Instructions**, specified way above, in particular, *3.*

1. *Compute the average distance* that Player A1 runs in a five-minute window, where the duration between consecutive windows is one minute. Also, report the *highest (distance) average among all windows for Player A1*. To answer these questions, update the *writeHighestAvgDistanceCovered()* method in the *FootballStatisticsImpl* class. The output of your program must be stored in a .csv file (*highestAvg.csv*) and contain three columns: the *startTimestamp, endTimestamp,* and *highestAverage*. **Hint 1**: Use a moving average over sliding windows. **Hint 2**: *Sensors attached to Player A1 have the following IDs:* the *left leg is 47 and the right leg is 16*. **(15 points)**

2. For each team, compute the *total number of events* (TNE) over the entire game for which the team almost scored a goal. This means that the ball was inside the penalty area box, but was pushed out by a player. To answer this question, you will need to update the *writeAvertedGoalEvents()* method in the *FootballStatisticsImpl* class. The output of your program must be stored in a .csv file (avertedGoals.csv) and contain two columns: *teamName* (i.e., A or B) and the *TNE-count*. The boundary of the penalty area is approximated as follows: for **Team A**, it is: *6300 < x < 46183* and *15940 < y < 33940* and for **Team B**, it is: *6300 < x < 46183* and *-33968 < y < -15965*. **Hint**: Use a moving sum over a tumbling window of one-minute duration. **(15 points)**

D. PageRank in GraphLab Create (20 points)

In this exercise, you will gain familiarity with GraphLab Create.

- First, you will need to register for academic use: https://turi.com/download/academic.html.
- Second, you will need to install *GraphLab Create V2.1*: https://turi.com/download/install-graphlab-create.html.
- Third, you should familiarize yourself with the *GraphLab Create API* documentation pages*:* https://turi.com/products/create/docs/index.html.

1. Use GraphLab Create to compute the PageRank score of every node in the Stanford Web Graph: https://snap.stanford.edu/data/web-Stanford.html. You will need to submit your GraphLab program, a screenshot of the execution of the program, and a histogram of the PageRank scores. You are free to use any visualization tool. **(10 points)**

2. Repeat the GraphLab experiment using the latest version of Apache Flink[2]. Submit your Apache Flink program and a screenshot of the execution results. Are the results consistent between the two systems? If not, can you explain why this is so? **(10 points)**

[2] PageRank - https://ci.apache.org/projects/flink/flink-docs-release-1.9/dev/batch/examples.html#page-rank.

E. Network Statistics in Apache Flink (20 points)

Use *Gelly*, Flink's Graph API: https://ci.apache.org/projects/flink/flink-docs-release-1.8/dev/libs/gelly/ to compute the *distributions* and *summary statistics* requested below for the **Slashdot-Zoo** dataset, available on the course portal as a compressed tar file. For this exercise, you will need to write several Apache Flink programs to compute the corresponding distributions and statistics.

1. Compute both the ***in-degree*** and ***out-degree*** distributions. **(4 points)**

2. Compute the ***out-degree distributions*** for both the friend & foe nodes, individually. **(4 points)**

3. Calculate both the **average degree** and the **maximum degree** of all vertices. **(4 points)**

4. Identify the **vertex** with the **most friends** and the **most foes.** **(4 points)**

5. Calculate the **average friend to foe ratio.** **(4 points)**

For each exercise, *report the requested statistic* or *plot the distribution*, correspondingly. In addition, you must also interpret your results. As a starting point, see this code stub*:* *https://gitlab.tu-berlin.de/AIM-3/ComputationalExercises/tree/master/NetworkStatisticsTask*.

Below are some additional guidelines/recommendations.

- Adjust the method *de.tuberlin.dima.aim3.exercise6.Config#pathToSlashdotZoo* to point to the location of the file *out.matrix* that you unpacked from the dataset.

- Class *de.tuberlin.dima.aim3.exercise6.DegreeDistribution* calculates the degree distribution. Note: The signs of edges are ignored.

- For Ex. 1, write a new Flink program InDegreeDistribution to compute the *in-degree* distribution. Save the output as *in-degree_dist.csv*. In addition, write a new Flink program OutDegreeDistribution to compute the *out-degree* distribution. Save the output as *out-degree_dist.csv***.**

- For Ex. 2, write a new Flink program SignedOutDegreeDistribution to compute two *out-degree* distributions, one solely for *friend* edges and one solely for *foe* edges. Save the output as: *out-degree_friend_dist.csv* and *out-degree_foe_dist.csv*, respectively.

- For Ex. 3, update the *de.tuberlin.dima.aim3.exercise6.DegreeDistribution* method to calculate the *average degree* and *maximum degree*. Save the result as *avg_degree.txt* and *max_degree.txt*, respectively.

- For Ex. 4, write new Flink program *VertexQuery* to identify the vertex with the most friends and the most foes. Save the result as *vertex_max_friend.txt* and *vertex_max_foe.txt*, respectively.

- For Ex. 5, write a new Flink program *AverageFriendFoeRatio* to compute the average ratio of friends to foes per vertex in the network. Save the result as *avg_ratio.txt*. Note: Ignore vertices that only have friends or foes.

You must submit your source codes, .csv & .txt files, as well as address the questions raised above.
Note*:* You are free to use your preferred plotting software.