Inf553 – Foundations and Applications of Data Mining

Summer 2018 Assignment 1

Deadline: 05/28 2018 11:59 PM PST

1 Overview of the Assignment

In this assignment, students will complete two tasks. The goal of these two tasks is to let students get familiar with Spark and perform data analysis using Spark. In the assignment description, the first part is about how to configure the environment and data sets, the second part describes the two tasks in details, and the third part is about the files the students should submit and the grading criteria.

2 Environment Configuration

Please use Spark 2.2.1 with Hadoop 2.7 for this assignment.

2.1 Spark Installation

Spark can be downloaded from the official website (refer to: Spark Page)
The interface of Spark official website is shown in the following figure.

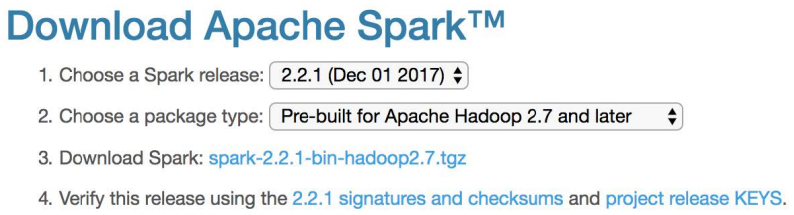

Figure 1: The Interface of Spark Official Website

2.2 Python Configuration

You need to add the paths of your Spark (path/to/your/Spark) and Python (path/to/your/ Spark/python) folders to the interpreter's environment variables named as SPARK_HOME and PYTHONPATH, respectively.

2.3 Scala Installation

You can use Intellij if you prefer IDE for creating and debugging projects. And install Scala/SBT plugins for Intellij. You can refer to the tutorial "Setting UP Spark 2.0 environment on intellij community edition".

2.4 Environment Requirements

Python: 2.7 Scala: 2.11 Spark: 2.2.1

Student must use python to complete both Task1 and Task2.

There will be 10% bonus if you also use Scala for both Task1 and Task2 (i.e. 10 - 11; 9 - 9.9).

IMPORTANT: We will use these versions to compile and test your code. If you use other versions, there will be a 20% penalty since we will not be able to grade it automatically.

2.5 Write your own code!

For this assignment to be an effective learning experience, you must write your own code! I emphasize this point because you will be able to find Python implementations of most or perhaps even all of the required functions on the web. Please do not look for or at any such code! Do not share code with other students in the class!!

TA will combine some python code on Github which can be searched by keyword "INF553" and every students' code, using some software tool for detecting Plagiarism. Write your own code.

2.6 Data

Please download the data from MovieLen over the following link: MovieLen

You are required to download two data sets. The first is ml-20m.zip, which size is 190MB, the second is ml-latest-small.zip, which size is 1MB. Each zip file contains five CSV files. The files tags.csv and ratings.csv are needed for the tasks.

3 Task 1: (40%)

Students are required to calculate each movie's average rating. The ratings.CSV file is needed for this task.

3.1 Result format

1. Save the result as one csv file with header (movieId, rating_avg).
2. The result is ordering by movieId in ascending order.

The following snapshot is an example of result for task 1. It just shows the format of the result.

```
movieId,rating_avg
1,3.8724696356275303
2,3.4018691588785046
3,3.1610169491525424
4,2.3846153846153846
5,3.267857142857143
6,3.8846153846153846
7,3.2830188679245285
8,3.8
9,3.15
10,3.4508196721311477
```

Figure 2: Example of Result for Task 1

3.2 Execution Example

The program that you will implement should take two parameters as input and generate one file as an output.

The first parameter must be the location of the ratings.csv file and the second one must be the path to the output file followed by the name of the output file.

Java/Scala Execution Example

Please use Task1 as class name

```
Python:
bin/spark-submit Firstname_LastName_task1.py <input path> <output path>

Scala:
bin/spark-submit --class Task1 Firstname_LastName_hw1.jar <input path> <output path>
```

Figure 3: Example of Command Line for Task 1

4 Task 2: (60%)

Students are required to calculate the average rating of each tag. Both the rating.csv and tags.csv files are required for this task.

4.1 Result format

1. Save the result as one csv file with header (tag, rating_avg).
2. The result is ordering by tag in descending order.

The following two snapshots is an example of result for task 2. The unreadable codes in the first snapshots are because encoding problem. It just shows the format of the result. In the second picture, the data is sorted by first column in descending order.

```
1  tag,rating_avg
2  é˜®ä¸€é¸£,3.92123956132
3  ç»å…¸,4.02900018135
4  æš´åŠ›,4.17423116922
5  æµ‹è¯•,3.46666666667
6  å¥½äºº,3.58919902913
7  ä½Žä¿–å°è¯´,4.17423116922
8  Ø§ØØ³Ø§Ø³Ø§ØªÙŠ,2.94611375134
9  Özgür Yildirim,3.16666666667
10 Özer Kiziltan,3.82692307692
```

Figure 4: Example of Result for Task 2

```
zeichnungen als übergang,2.81337103615
zegist,3.70224719101
zef,3.34615384615
zebras,3.36527208894
zebra,3.36527208894
zealots,3.37777777778
zany,2.10344827586
yuppies,3.50327998511
yukon,3.61274509804
yuen woo-ping,3.64606741573
yuen chor,4.0
```

Figure 5: Example of Result for Task 2

4.2 Execution Example

The program that you will implement should take three parameters as input and generate one file as an output.

The first two parameter must be the location of the ratings.csv file and tags.csv file and the third one must be the path to the output file followed by the name of the output file.

Java/Scala Execution Example

Please use Task2 as class name

```
Python:
bin/spark-submit Firstname_LastName_task2.py <rating path> <tag path> <output path>

Scala:
bin/spark-submit --class Task2 Firstname_LastName_hw1.jar <rating path> <tag path> <output path>
```

Figure 6: Example of Command Line for Task 2

4.3 Hints for Task2

1. You can create Dataframe objects and save the Dataframe objects as csv file.
2. You can learn more about Dataframe by this link: DataFrames

Submission Details

Your submission must be a .zip file with name: Firstname_Lastname_hw1.zip. The structure of your submission should be identical as shown below. The Firstname_Lastname_Description.pdf file contains helpful instructions on how to run your code and other need informations. The OutputFiles directory contains the deliverable output files for each problem and the Solution directory contains your source code.

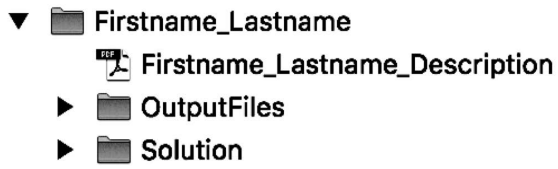

Figure 7: Submission Structure

What you need to turn in

1. Source codes for two tasks (Python or Scala) and name it respectively as
Firstname_Lastname_task1
Firstname_Lastname_task2
(For example, Yuanbin_Cheng_task1.py)
2. Result files of two tasks for large and small data sets and name it as

Firstname_Lastname_result_task1_big.csv
Firstname_Lastname_result_task2_big.csv
Firstname_Lastname_result_task1_small.csv
Firstname_Lastname_result_task2_small.csv

3. Documents: please describe how to run your program in this document.
4. If you use Scala, please submit the jar package as well and name them as
Firstname_Lastname_hw1.jar
5. Zip the above files and name it as Firstname_Lastname_hw1.zip

Grading Criteria

1. Your codes will be run according to your Readme file. If your programs cannot be run with the commands you provide, your submission will be graded based on the result files you submit and 80% penalty for it.
2. If the file generated by your program is unsorted, there will be 20% penalty.
3. If your program does not use the required Scala/Python/Spark versions, there will be 20% penalty.
4. If your program generates more than one file, there will be 20% penalty.
5. If the csv file generated has more than two columns, there will be 20% penalty.
6. If the header of the csv file is missing, there will be 10% penalty.
7. The deadline for assignment 1 is 05/28 midnight. There will be 20% penalty for late submission within a week and 0 grade after a week.
8. You can use your free 5-day extension.
9. There will be 10% bonus if you use both Scala and python for the entire assignment.