INF 553 – Fall 2018 Assignment 1

Overview of the assignment

In this assignment, students will complete three tasks. The goal of these 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 three tasks in details, and the third part is about the files the students should submit and the grading criteria.

Spark Installation

Spark can be downloaded from the official website (refer to: link)

Please use Spark 2.3.1 with Hadoop 2.7 for this assignment. The interface of Spark official website is shown in the following figure.

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".

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.

Environment Requirements

Python: 2.7, Scala: 2.11, Spark: 2.3.1

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.

Data

Please download the "Stack Overflow 2018 Developer Survey" data from this link. Detailed introduction of the data can also be found through the link.

You are required to download the dataset that contains two files: `survey_results_public.csv` and `survey_results_schema.csv`. The first file contains the survey responses and will be required for this homework. The second file describes the 129 columns of the dataset. In this assignment only 3 columns of the dataset will be used: `Country`, `Salary`, and `SalaryType`.

Task1: (20%)

Students are required to compute the total number of survey responses per country that have provided a salary value – i.e., response entries containing 'NA' or '0' salary values are considered non-useful responses and should be discarded.

Result format:

1. Save the result as one csv file;
2. The first line in the file should contain the keyword 'Total' and the total number of survey responses containing a salary;
3. The result is ordering by *country* in ascending order.

The following snapshot is an example of result for Task 1:

```
Total,49457
Albania,41
Algeria,33
Andorra,5
Angola,3
Argentina,329
Australia,1171
Austria,408
Bahrain,6
Barbados,3
Belgium,392
Bhutan,3
Botswana,2
```

Task2: (30%)

Since processing large volumes of data requires performance decisions, properly partitioning the data for processing is imperative. In this task, students are required to show the number of partitions for the RDD built in Task 1 and show the number of items per partition. Then, students have to use the partition function (using the *country* value as driver) to improve the performance of map and reduce tasks. A time span comparison between the *standard* (RDD used in Task 1) and *partition* (RDD built using the partition function) should also be shown.

Hints for Task 2:

1. When initializing the SparkContext, limit the number of processors to 2;
2. Only 2 partitions should be used.

Result format:

1. Save the result as one csv file.
2. The file should have two lines (one for *standard* and another for *partition*). The second and third columns should list the number of items per partition.
3. A separate file should show the total time spent to perform a simple reduce operation for both *standard* and *partition*:

```
rdd.reduceByKey((a, b) => a+b).collect()
```

The following snapshot is an example of result for Task 2:

```
standard,28539,20918
partition,32087,17370
```

Task3: (50%)

Students are required to compute *annual* salary averages per *country* and show min and max salaries.

Hints for Task 3:

1. Some salary values represent *weekly* or *monthly* payments. Recall performing the appropriate transformations to compute the *annual* salary. The value in the column `SalaryType` informs whether the salary amount is annual, weekly, or monthly.

Result format:

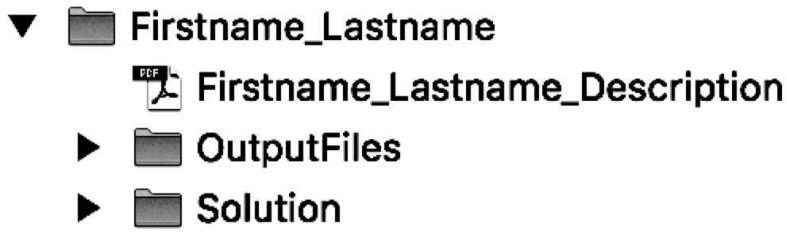
1. Save the result as one csv file.
2. The result is ordering by *country* in ascending order.
3. Columns should present: *country*, *number of salaries*, *min salary*, *max salary*, and *average salary*.

The following snapshot is an example of result for Task 3:

```
Australia,1171,3,8320000,220349.25
Austria,408,12,2400000,61807.93
Azerbaijan,29,144,70000,18748.41
Bahamas,2,57000,120000,88500.00
Bahrain,6,400,24000,9766.67
Bangladesh,240,1,15600000,664997.89
Barbados,3,15600,65000,33853.79
Belarus,132,1,2400000,65490.54
Belgium,392,1,3180000,66195.29
Benin,2,4200000,36000000,20100000.00
Bhutan,3,3000,300000,109666.67
Bolivia,16,400,180000,32185.62
```

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 (including .scala files)



What you need to turn in:

1. Result files of Task1, Task2 (The performance output), Task3.
   Firstname_Lastname_task1.csv
   Firstname_Lastname_task2.csv
   Firstname_Lastname_task3.csv

2. Readme document: Please describe how to run your program in this document (Firstname_Lastname_Description.pdf)

3. Please submit a single scala jar package and name it as: *Firstname_Lastname.jar*
   You will have to submit the .scala source codes as well.

We will run the scala code as follows:

```
$SPARK_HOME/bin/spark-submit --class Task1 Firstname_Lastname.jar
<input_path> <output_path>

$SPARK_HOME/bin/spark-submit --class Task2 Firstname_Lastname.jar
<input_path> <output_path>

$SPARK_HOME/bin/spark-submit --class Task3 Firstname_Lastname.jar
<input_path> <output_path>
```

4. Zip the above files and name it as *Firstname_Lastname_HW1.zip*

Grading Criteria:

1. Your codes will be run according to your Description file. If your programs cannot be run with the commands you provided, your submission will be graded based on the result files you submit and **20%** 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 (except when required), there will be **20%** penalty.

5. If the CSV file generated in task 3 has more than five columns, there will be **10%** penalty.
6. The deadline for assignment 1 is 09/21 midnight. There will be **20%** penalty for late submission within a week and 0 grade after a week.
7. You can use your free 5-day extension.