Description on the source code

In this assignment, I used Scala version 2.10.6, spark version 1.6.1 and sbt version 0.13.6. Within the main object “frequentItemsets”, I created three helper functions:

1. findCandidates
   This function takes the frequent itemsets of size n, and generate the candidate itemsets of size n+1.
2. findSet
   This function takes the candidate itemsets, and check how many times they appeared in a certain chunk, and output the frequent itemsets among them.
3. apriori
   This function calls “findCandidates” and “findSet” functions, and integrate them to find all the frequent itemsets in a chunk using A-priori algorithm. More specifically, it first checks which of the singletons are frequent, then constructs candidate pairs using Monotonicity principle out of those singletons; then it checks which of the candidate pairs are frequent, and construct candidate triples based on them; … It will follow this process, and loop until it cannot construct any candidate itemsets of size n+1 from the given frequent itemsets of size n.

The basic logic of this solution is to follow the SON algorithm. Firstly, it joins the ratings and users datasets, and groups the records by userid in case 1 or movieid in case 2, to get the market baskets. Then in SON phase 1, it first uses the “apriori” function in mapPartitions() method to find out the frequent itemsets in each chunk, then takes all the unique itemsets from all the chunks as the candidate itemsets. In phase 2, it will count all these candidate itemsets in the whole dataset to find overall frequent itemsets.

Instruction on running the Jar file

To run the Jar file, please first go to the directory (“Yu_Dong”), then run the commands in terminal. The commands include exactly following five arguments:

arg0 - scala jar path
arg1 - case number (1/2)
arg2 - path for ratings.dat
arg3 - path for users.dat
arg4 - support value

And the output file will be saved in the directory with the name like “Yu_Dong_SON.case1_1200.txt”.

Detailed commands are as following:

1. Case1, support threshold 1200:

```
userName$ ./bin/spark-submit   --class frequentItemsets --master local[4]
Solutions/Yu_Dong_SON.jar 1 "ml-1m/ratings.dat" "ml-1m/users.dat" 1200
```

output file name: Yu_Dong_SON.case1_1200.txt

This should take around 40 seconds.

2. Case1, support threshold 1300:

```
userName$ ./bin/spark-submit   --class frequentItemsets --master local[4]
Solutions/Yu_Dong_SON.jar 1 "ml-1m/ratings.dat" "ml-1m/users.dat" 1300
```

output file name: Yu_Dong_SON.case1_1300.txt

This should take around 30 seconds.

3. Case2, support threshold 500:

```
userName$ ./bin/spark-submit   --class frequentItemsets --master local[4]
Solutions/Yu_Dong_SON.jar 2 "ml-1m/ratings.dat" "ml-1m/users.dat" 500
```

output file name: Yu_Dong_SON.case2_500.txt

This should take around 20 seconds.

4. Case2, support threshold 600:

```
userName$ ./bin/spark-submit   --class frequentItemsets --master local[4]
Solutions/Yu_Dong_SON.jar 2 "ml-1m/ratings.dat" "ml-1m/users.dat" 600
```

output file name: Yu_Dong_SON.case2_600.txt

This should take around 20 seconds.