Assignment 03

Ye Joo Park (USC# 1128-6851-51)

Disclaimer - Issue with my pyspark

Before going forward, I need to mention that runnig pyspark in my PC hasn't been working. I am getting a "java.net.SocketException". I have tried reinstalling Spark, reinstalling IntelliJ, install winutils.exe (as multiple StackOverflow posts suggested), googled the issue for a long long time, but couldn't resolve this issue. I only had a couple of hours to test the Python scripts with my friend's laptop. Please use the Scala files/jars as the main scripts for grading (although I believe the Python scripts should run fine).

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
18/06/28 15:36:21 WARN NativeCodeLoader: Unable to load native-hadoop library for your platfor
18/06/28 15:36:23 ERROR PythonRunner: Python worker exited unexpectedly (crashed)
java.net.SocketException: Connection reset by peer: socket write error
    at java.net.SocketOutputStream.socketWrite0(Native Method)
    at java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:111)
    at java.net.SocketOutputStream.write(SocketOutputStream.java:155)
    at java.io.BufferedOutputStream.flushBuffer(BufferedOutputStream.java:82)
    at java.io.BufferedOutputStream.write(BufferedOutputStream.java:126)
    at java.io.DataOutputStream.write(DataOutputStream.java:107)
    at java.io.FilterOutputStream.write(FilterOutputStream.java:97)
    at org.apache.spark.api.python.PythonRDD$.writeUTF(PythonRDD.scala:688)
```

Part 1 – Jaccard-based LSH

Command Line for Scala & Python

```
spark-submit –class JaccardLSH YeJoo_Park_hw3.jar <rating>
```

```
spark-submit YeJoo_Park_task1_Jaccard.py <rating>
```

For hashing row indices, I've used a function in the form below:

```
def hashRow(rowIndex: Int, hashFuncIndex: Int): Int = {
   (3 * rowIndex + hashFuncIndex) % 671
}
```

For hashing rows in a band, I've multiplied each element by a different prime number and split them to ~9000 buckets. There are 400 hash functions and 40 bands (10 rows per band).

```
val primeArr: Array[Int] = Array[Int](1, 227, 671, 2663, 3547,
6949, 10657, 17389, 32609, 59023)

def hashSetInBand(l: Array[Int], fromIndex: Int, untilIndex:
Int): Int = {
   var hashVal = 0

   for (index <- fromIndex until untilIndex) {
      hashVal += l(index) * primeArr(index - fromIndex)
   }

   hashVal = hashVal % 9066

   return hashVal
}
```

Precision & Recall

Total # of movieId pairs with similarity >= 0.5	370334
Precision	1.0
Recall	0.964

Part 2 – Collaborative Filtering

CF Command Line Format for Scala/Python

Model-based CF

```
spark-submit –class ModelBasedCF YeJoo_Park_hw3.jar <rating> <testing>
```

```
spark-submit YeJoo_Park_task2_ModelBasedCF.py <rating> <testing>
```

User-based CF

```
spark-submit –class UserBasedCF YeJoo_Park_hw3.jar <rating> <testing>
```

```
spark-submit YeJoo_Park_task2_UserBasedCF.py <rating> <testing>
```

Item-based CF

```
spark-submit –class ItemBasedCF YeJoo_Park_hw3.jar <rating> <testing>
<similarityFilePath>
```

```
spark-submit YeJoo_Park_task2_ItemBasedCF.py <rating> <testing>
<similarityFilePath>
```

Model-based Collaborative Filtering

	Model-based CF	
	Small	Large
>=0 and <1	13783	3223997
>=1 and <2	4113	733114
>=2 and <3	730	81689
>=3 and <4	101	7334
>=4	6	197
RMSE	0.951	0.819

User-based Collaborative Filtering

	User-based CF
>=0 and <1	13571
>=1 and <2	4120
>=2 and <3	834
>=3 and <4	186
>=4	40
RMSE	1.003

Item-based Collaborative Filtering

	User-based CF
>=0 and <1	12880
>=1 and <2	4762
>=2 and <3	888
>=3 and <4	195
>=4	8
RMSE	1.013