INF 553 FALL 2018 ASSIGNMENT 3

Name: Aditya Chavan
USC ID: 3416409224

I am Running the Scala Code, using the following IDE:
intellij IDEA

As per instructions, the environment was set as follows:
1) Java – 1.8.0_181
2) sbt – 1.2.3
3) Hadoop – 2.7
4) Scala – 2.11.12
5) Spark – 2.3.1

Description:

The *Aditya_Chavan_SON.jar* file is the SON algorithm, which is used to find frequent itemsets given the required input file. The main scala class is *Aditya_Chavan_Son.scala*

You first have to read-in the input test file and convert to a RDD and then have to form all the required baskets using the groupByKey method.

The baskets contain the words corresponding to each reviewID. Each basket then needs to be divided intp segments and this is done using the mapPartitions function.

Each segment is then run on the Apriori function which finds the candidate frequent itemsets given the threshold value. This threshold value is calculated as given_support / number of partitions.

Then we need to remove duplicate words from different partitions using the reduceByKey method. This is the first MapReduce task.

Another MapReduce Task is used to find the count of the candidate items in each segment, then from all segments filter out the real frequent itemsets.

why did we need to use such a large support threshold ?
Since there are approximately 1,00,000 items in the large dataset, and if the support threshold value is less than 10% of the total items (i.e. <10,000) then there will be many items which may not be frequent.

where do you think there could be a bottleneck that could result in a slow execution for your implementation, if any. ?
If there were a bottleneck situation, it would occur due to a lower value of support_threshhold, which would give a higher number of candidate itemsets from first MapReduce task and would result in slow execution of second MapReduce task.

Steps to Run the code using command line

1. Change to directory from the folder that contains *Aditya_Chavan_SON.jar*

2. To run .jar file, execute the command:

$SPARK_HOME/bin/spark-submit --class Aditya_Chavan_SON Aditya_Chavan _SON.jar <input_file> <Support_Threshold> <output_file>

Example:
"/Users/mahima/Desktop/spark-2.3.1-bin-hadoop2.7/bin/spark-submit --class Aditya_Chavan_SON Aditya_Chavan_SON.jar /Users/mahima/Desktop/yelp_reviews_test.txt 30 /Users/mahima/Desktop/yelp_reviews_test_30.txt"

Problem 1 Execution Table

Support_Threshold	Execution Time
30	~120 sec
40	~30 sec

Problem 2 Execution Table

Support_Threshold	Execution Time
500	~30 sec
1000	~15 sec

Problem 3 Execution Table

Support_Threshold	Execution Time
100000	~350 sec
120000	~280 sec