Set Spark home environment variable.

Scala – 2.11 Spark – 2.3.1

Increase the memory so that you won't get memory error.

Approach to implement the algorithm:

1) Initially baskets are formed from the input by grouping over key.
2) Number of partitions are found, and threshold is calculated accordingly.
3) SON Algorithms runs in 2 phases (2 map-reduce).
4) In first phase, baskets are split into chunks and apriori algorithm is run on each chunk.
5) Once you get results from Apriori, then a map and reduceByKey are performed on output of Apriori
6) After the first MapReduce we get a list of candidate item sets which is broadcasted.
7) Now you perform second phase(MapReduce) to find the true frequent itemsets by following monotonicity idea i.e itemset cannot be frequent in the entire set of baskets unless it is frequent in at least one subset.

Command to run:

$SPARK_HOME/bin/spark-submit --class SaiTeja_Chava_SON path_to_jar path_to_data_txt_file support_value path_to_output

Example for support 30:

$SPARK_HOME/bin/spark-submit --class SaiTeja_Chava_SON ~/Documents/Data_Mining_INF_553/Assignments/hw3/SaiTejaChava/OutputFiles/SaiTeja_Chava_SON.jar ~/Documents/Data_Mining_INF_553/Assignments/hw3/Data/yelp_reviews_test.txt 30 ~/Documents/Data_Mining_INF_553/Assignments/hw3/yelp_reviews_test_30.txt

Example for support 40:

$SPARK_HOME/bin/spark-submit --class SaiTeja_Chava_SON ~/Documents/Data_Mining_INF_553/Assignments/hw3/SaiTejaChava/OutputFiles/SaiTeja_Chava_SON.jar ~/Documents/Data_Mining_INF_553/Assignments/hw3/Data/yelp_reviews_test.txt 40 ~/Documents/Data_Mining_INF_553/Assignments/hw3/yelp_reviews_test_40.txt

Example for support 500 and small dataset:

$SPARK_HOME/bin/spark-submit --class SaiTeja_Chava_SON ~/Documents/Data_Mining_INF_553/Assignments/hw3/SaiTejaChava/OutputFiles/SaiTeja_Chava_SON.jar ~/Documents/Data_Mining_INF_553/Assignments/hw3/Data/yelp_reviews_test.txt 500 ~/Documents/Data_Mining_INF_553/Assignments/hw3/yelp_reviews_small_500.txt

Example for support 1000 and small dataset:

$SPARK_HOME/bin/spark-submit --class SaiTeja_Chava_SON ~/Documents/Data_Mining_INF_553/Assignments/hw3/SaiTejaChava/OutputFiles/SaiTeja_Chava_SON.jar ~/Documents/Data_Mining_INF_553/Assignments/hw3/Data/yelp_reviews_test.txt 1000 ~/Documents/Data_Mining_INF_553/Assignments/hw3/yelp_reviews_small_1000.txt

Example for support 100000 and large dataset:

$SPARK_HOME/bin/spark-submit --class SaiTeja_Chava_SON ~/Documents/Data_Mining_INF_553/Assignments/hw3/SaiTejaChava/OutputFiles/SaiTeja_Chava_SON.jar ~/Documents/Data_Mining_INF_553/Assignments/hw3/Data/yelp_reviews_test.txt 100000 ~/Documents/Data_Mining_INF_553/Assignments/hw3/yelp_reviews_large_100000.txt

Example for support 120000 and large dataset:

$SPARK_HOME/bin/spark-submit --class SaiTeja_Chava_SON ~/Documents/Data_Mining_INF_553/Assignments/hw3/SaiTejaChava/OutputFiles/SaiTeja_Chava_SON.jar ~/Documents/Data_Mining_INF_553/Assignments/hw3/Data/yelp_reviews_test.txt 120000 ~/Documents/Data_Mining_INF_553/Assignments/hw3/yelp_reviews_large_120000.txt

Problem 2 Execution Table:

Support Threshold	Execution Time
500	8
1000	5

Problem 3 Execution Table:

Support Threshold	Execution Time
100000	197
120000	138

Bonus Question:

We need such a large threshold because since the dataset is very large and contains many baskets, there will be many freq item sets that occur many times. So to avoid these and get only item sets that occur often when compared to no of baskets, we need a high threshold. Most of computation time of bottleneck comes from Shuffle reads and Shuffle writes which is evident from visualizing from Spark UI.