Frequent Itemsets

SON Algorithm

- **First Map Function:** Take the assigned subset of the baskets and find the itemsets frequent in the subset using the simple Apriori Algorithm. As described there, lower the support threshold from s to ps if each Map task gets fraction p of the total input file. The output is a set of key-value pairs (F, 1), where F is a frequent itemset from the sample. The value is always 1 and is irrelevant.
- **First Reduce Function:** Each Reduce task is assigned a set of keys, which are itemsets. The value is ignored, and the Reduce task simply produces those keys (itemsets) that appear one or more times. Thus, the output of the first Reduce function is the candidate itemsets.
- **Second Map Function:** The Map tasks for the second Map function take all the output from the first Reduce Function (the candidate itemsets) and a portion of the input data file. Each Map task counts the number of occurrences of each of the candidate itemsets among the baskets in the portion of the dataset that it was assigned. The output is a set of key-value pairs (C, v), where C is one of the candidate sets and v is the support for that itemset among the baskets that were input to this Map task.
- **Second Reduce Function:** The Reduce tasks take the itemsets they are given as keys and sum the associated values. The result is the total support for each of the itemsets that the Reduce task was assigned to handle. Those itemsets whose sum of values is at least s are frequent in the whole dataset, so the Reduce task outputs these itemsets with their counts. Itemsets that do not have total support at least s are not transmitted to the output of the Reduce task.

Apriori Algorithm

- Create a table containing support count of each item present in the bucket – cancidate set
- compare candidate set item's support count with minimum support count (ps) this gives the itemset
- Eliminate all the elements from the candidate set which are not frequent
- Form combinations of 2 from the updated candidate set and repeat the process as you increase the number of elements in the combinations
- Do the same until the candidate set is empty and no further combination can be produced

Environment requirements- Scala 2.11 and Spark 2.3.1

1. Set $SPARK_HOME to the directory in which spark-2.3.1-bin- hadoop2.7 is stored
   Run source ~/.bashrc

   Where .bashrc file which is in home directory and contains
   export SPARK_HOME=$HOME/Downloads/spark-2.3.1-bin-hadoop2.7
   export PATH=$PATH:~/opt/node/bin

Task 1:

$SPARK_HOME/bin/spark-submit --class NamanaMadanMohanRao_Pawar_SON NamanaMadanMohanRao_Pawar_SON.jar `<input file>` `<support_threshold>` `<output file>`

Eg:

$SPARK_HOME/bin/spark-submit --class NamanaMadanMohanRao_Pawar_SON NamanaMadanMohanRao_Pawar_SON.jar `Data/yelp_reviews_test.txt` `30` `Output/NamanaMadanMohanRao_Pawar_SON_yelp_reviews_test_30.txt`

Output:

```
Time = 91.644466159
```

$SPARK_HOME/bin/spark-submit --class NamanaMadanMohanRao_Pawar_SON NamanaMadanMohanRao_Pawar_SON.jar `Data/yelp_reviews_test.txt` `40` `Output/NamanaMadanMohanRao_Pawar_SON_yelp_reviews_test_40.txt`

Output:

```
Time = 14.780366794
```

Support Threshold	Time
30	91.644466159
40	14.780366794

Task 2:

$SPARK_HOME/bin/spark-submit --class NamanaMadanMohanRao_Pawar_SON NamanaMadanMohanRao_Pawar_SON.jar <input file> <support_threshold> <output file>

Eg:

$SPARK_HOME/bin/spark-submit --class NamanaMadanMohanRao_Pawar_SON NamanaMadanMohanRao_Pawar_SON.jar Data/yelp_reviews_small.txt 500 Output/NamanaMadanMohanRao_Pawar_SON_yelp_reviews_test_500.txt

Output:

```
Time = 16.084930966
```

$SPARK_HOME/bin/spark-submit --class NamanaMadanMohanRao_Pawar_SON NamanaMadanMohanRao_Pawar_SON.jar Data/yelp_reviews_small.txt 1000 Output/NamanaMadanMohanRao_Pawar_SON_yelp_reviews_test_1000.txt

Output:

```
Time = 8.564057861
```

Support Threshold	Time
500	16.084930966
1000	8.564057861

Task 3:

Increase heap size to avoid out of memory error by adding --driver-memory 10g in the spark submit command

$SPARK_HOME/bin/spark-submit --driver-memory 10g --class NamanaMadanMohanRao_Pawar_SON NamanaMadanMohanRao_Pawar_SON.jar <input file> <support_threshold> <output file>

Eg:

```
$SPARK_HOME/bin/spark-submit --driver-memory 10g --class
NamanaMadanMohanRao_Pawar_SON NamanaMadanMohanRao_Pawar_SON.jar
Data/yelp_reviews_large.txt 100000
Output/NamanaMadanMohanRao_Pawar_SON_yelp_reviews_test_100000.txt
```

Output:

```
Time = 324.550892537
```

```
$SPARK_HOME/bin/spark-submit --driver-memory 10g --class
NamanaMadanMohanRao_Pawar_SON NamanaMadanMohanRao_Pawar_SON.jar
Data/yelp_reviews_large.txt 120000
Output/NamanaMadanMohanRao_Pawar_SON_yelp_reviews_test_120000.txt
```

Output:

```
Time = 198.442431457
```

Support Threshold	Time
100000	324.550892537
120000	198.442431457