Spring 2018 INF553 - HW2

Tong Lyu

This assignment aims to use SON algorithm with Spark to find the frequent itemsets of baskets. There are three different datasets with different size to test the scalability of the program. The programs will find all frequent itemsets from input data.

1. Environment Requirements

1) Java: 1.8
2) Scala: 2.11.12
3) Spark: 2.2.1

2. Datasets

1) Dataset1: /small2.csv
2) Dataset2: /beauty.csv
3) Dataset3: /books.csv

3. Implementation of algorithm

- SON algorithm

  There are total 2 phases of SON algorithm.

  1) In phase1, the program divide the data into several chunks and run Apriori in each chunk.

     Map task: Generate candidate itemsets in each chunk with the proportionate min support. (Actually in the program, to save time, we only save the list of candidate sets rather than the list of tuples.)

     ```
     for each chunk:
           run Apriori(min_support/number_of_chunks)
           emit (candidate itemsets, 1)
     ```

     Reduce task: Eliminates duplicates (across chunks)

     ```
     emit (candidate itemsets, 1)
     ```

  2) In phase2, the program will count the real occurrence of candidate itemsets.

     Map task: count the occurrence of candidate itemsets

     ```
     for each chunk:
           emit(candidate itemsets, number of occurrence)
     ```

     Reduce task: Filter the candidate itemsets whose occurrence is less than min support.

     ```
     emit (true frequent itemsets, number of occurrence)
     ```

- Apriori algorithm

  In each chunk, we applied Apriori algorithm to get all k-size frequent itemsets. For each itemsets with k size, we construct two sets of k-tuples (sets of size k):

  C(k) = candidate k-tuples

  L(k) = the set of truly frequent k-tuples

  First, we generate the single items from the baskets and count the occurrence to get C(1), and filter to get L(1).

  1) Get candidate itemsets: When k > 2, we use the L(k-1) to generate C(k).

   a) Each time we choose one itemsets in the L(k-1) and add another single item occurred in L(k-1), then generate a new itemsets with k size.
   b) For each new itemset, check if all its subsets of k-1 size are in L(k-1), if so, add it into C(k).

2) Get frequent itemsets:

```
for each basket in baskets:
      for each itemsets i in C(k):
            if i isSubSetOf  basket: (the occurrence of i)++
```

Repeat the above two steps until the L(k) is empty, then we get all frequent itemsets.

```
while (L(k-1) is not empty) {
      C(k) = getCandidate(L(k-1))
      L(k) = getFrequent(C(k), baskets)
      freq_sets += L(k)
}
```

4. Usage Example

Open your terminal, using `` `$SPARK_HOME/bin/spark-submit --class <main-class> <application-jar> args(0) args(1) args(2)` `` in the top-level Spark directory to launch the applications.
args(0): case number; args(1):input data; args(2):min support

For example,

```
`.bin/spark-submit --class son Tong_Lyu_SON.jar 1 /small2.csv 3`
`.bin/spark-submit --class son Tong_Lyu_SON.jar 2 /small2.csv 5`
`.bin/spark-submit --class son Tong_Lyu_SON.jar 1 /beauty.csv 50`
`.bin/spark-submit --class son Tong_Lyu_SON.jar 2 /beauty.csv 40`
`.bin/spark-submit --class son Tong_Lyu_SON.jar 1 /books.csv 1200`
`.bin/spark-submit --class son Tong_Lyu_SON.jar 2 /books.csv 1500`
```

Tips:

1) The command line should be executed under the directory of `$SPARK_HOME/bin`
2) The path of jar file and input data is relative, please make sure the jar file and input data are also under the current directory where you run the command line.
3) The program will automatically generate an output file for each execution under the same current folder. When the program runs successfully, the console will show:
   "There are total 'number' frequent itemsets."
   "The execution time is: 'time' s".

5. Execution result

Table1. Execution time for larger datasets

File Name	Case Number	Support	Runtime(sec)
beauty.csv	1	50	569
beauty.csv	2	40	58
books.csv	1	1200	254
books.csv	2	1500	45