Apriori Algorithm Implementation Design Details

1. Namespaces and source folder organization:

Application has 2 namespaces:

- *edu.datamining.apriori.common :* All the entities used in the implementation are in this namespace
- *edu.datamining.apriori* : The actual implementation (APrioriImplementation) and the client class consuming this class (APrioriClient) are in this namespace

All source is under **src** source folder. Unit tests are under ***test*** folder

2. Class structure :

The classes used in the application are as follows :

- *ApplicationException:* Whenever a application specific exception is needed to be raised, this exception is thrown.
- *CombinationGenerator:* This is a general purpose class that generates specific combinations for a given number. Original source code can be found at : http://www.merriampark.com/comb.htm
- *Item:* This is a fundamental entity as it represents the items purchased in a transaction. Since out implementation works on integers, it is basically a wrapper-class for integer.
- *ItemSet:* A set of items. It is a wrapper-class for an ArrayList.
- *ItemSetCollection:* A collection of itemsets. It ısed to hold lists of itemsets, such as candidates. It is a wrapper-class for hashtable. It keeps track of the support counts of its itemsets.
- *SystemDefinitions:* This is a class just to hold system-wide constant values. Currently it only holds the value for the separator in the input files. If for any reason, the separator changes in the input file, the application can be modified easily by only changing this value.
- *Transaction*: It represents the items purchased in a row in the database. Every row read from the file is initially used to construct a transaction object. After that, the items can be reached via this object.

3. Important Methods:

First of all we need the frequent 1-itemsets. This list is kept in a ItemSetCollection object.

To get this list we call ***findFrequentOneItemSets()*** method. This method, scans the database. When it reads a row it creates a Transaction object by calling static fromRow method of the Transaction class. This method parses the string and creates the object. Then, the items can be get by calling getPurchasedItems() method.

Every items ,in the row are counted and the results are stored in a temporary ItemSetCollection object. After the counting process, we loop through this temporary collection and check if the items' support count is equal or greater than the minimum support count. If so they are added to the frequent 1-itemset collection.

After obtaining this collection, we generate canditates by calling ***generateCandidates()*** method. Candidate generation is a crucial step in this algortihm. This method requires the previous frequent itemsets. It takes two itemsets from this collection and checks if they can be joined. They can be joined if all the items in the list expcet for the last one are equal and the second set's last item is ggreater than the first set's last item. If these conditions are met a new itemset is generated. This new *joined* itemset is 1 item greater than the previous ones.

If the joining is process is unsuccessful, then null value is returned and no action is taken. If joining is succesful, then another process called ***pruning*** is utilized. Pruning is the process of eliminating itemsets which are imposible to be frequent. We can understand this, if the new itemset has subsets which are not in the previous frequent itemset collection. If there's even one subset that's not frequent then the new joined set can not be frequent either. So counting the database for this set is redundant and it is removed from the candidate list.

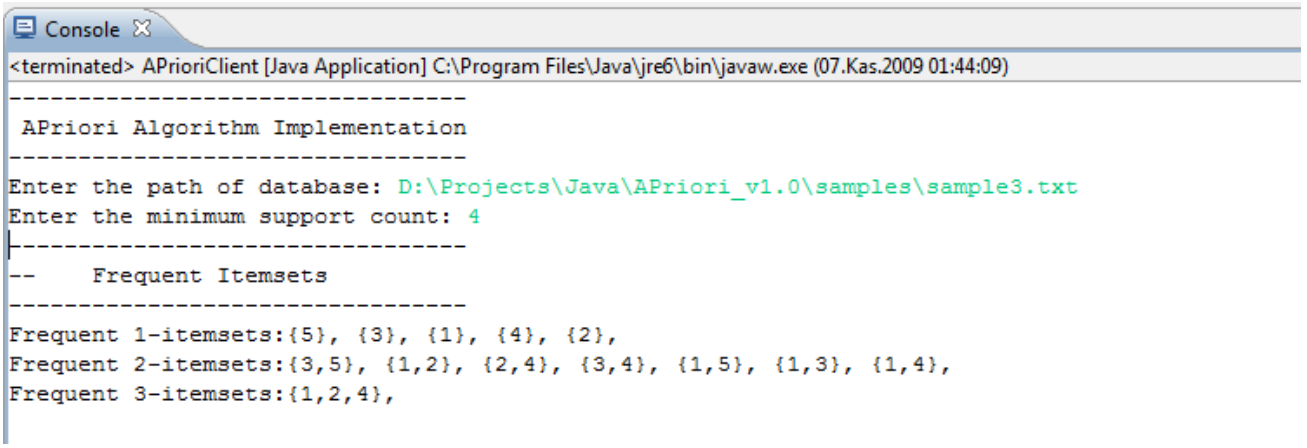
After candidates are generated, we head back to the database to count for the new candidates. And this process is repated until there aren't any more candidates.

4. Implementation Flaws :

First and foremost, the implementation cannot run from command prompt. When it is run by calling java.exe it throws a NoClassDefFoundError exception.

```
D:\Projects\Java\APriori_v1.0\bin\edu\datamining\apriori>C:\Sun\SDK\jdk\bin\java
 APrioriClient.class
Exception in thread "main" java.lang.NoClassDefFoundError: APrioriClient/class
Caused by: java.lang.ClassNotFoundException: APrioriClient.class
        at java.net.URLClassLoader$1.run(URLClassLoader.java:200)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:188)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:307)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:252)
        at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:320)
Could not find the main class: APrioriClient.class.  Program will exit.
```

While implemeting this project, i ran the application from the eclipse by Run command (Ctrl + F11) as shown in the screenshot below :



I noticed this defect very late so i could not have the time to fix it yet.

Another drawback is ItemSet.sort() method's implementation. Bubble Sort is used for sorting the items, a more effective algorithm should be used instead of this.

Also there are some code repetitions in the constructors which should be removed.

And there a few minor tasks that needs to be fixed. These are marked as TODO and can be seen in the Tasks window.

5. Test runs :

The application is test with three different input files. First the expected values are calculated by hand. These runs can be found in the accompanying excel file Test_Runs.xls under doc folder. In the manual calculations, pruning process is omitted.

6. File generator :

When i first started implementing this project, i developed a tool to generate input files.