Files

run.py
integrated-dataset.py
README.doc

Dataset:

We used the NYC Open Dataset which contains the details of all the 311-Requests in the year 2013. The dataset itself was of 800MB containing close to a million rows.

The raw dataset is available here https://data.cityofnewyork.us/Social-Services/311-Requests-2013-No-Timestamp/ve2c-ib6g

For our integrated dataset, we wrote a python script which sorts the rows according to the 'Months' field and then took the top 1000 rows for each month. We also took only six rows from the raw dataset, vis-à-vis- Month, Department, Complaint Type, Location Type, Zipcode, Borough.

The resulting dataset set therefore has 12000 rows (1000 rows for each month of the year) and 6 rows.

We chose this dataset because we wanted to analyse the trends in the domestic complaints that New Yorkers file. We wanted to check if there are relationships between the borough and the type of complaints, or between the type of complaint and the department handling the complaint, etc.

We have attached the create_dataset.py script which we used to create out integrated dataset. When this script is run on the raw dataset defined about, our integrated dataset is generated.

Description

We used the apriori algorithm described in section 2.1 of the Agrawal and Srikant paper in VLDB 1994 and the one discussed in class. We calculate the frequent itemsets as freq_itemset from scratch each time the program is run.

The *tran_list[]* contains each row of elements in the integrated dataset as an element.
The *item_list[]* contains every unique item from the entire dataset as its elements.
The *tran_list[]* and *item_list* are returned to *itemlist* and *tranlist* respectively. **(line 61)**
The *min_support()* function calculates and returns all those items that satisfy the minimum support condition. **(line 34)**
The *min_support()* function takes each element in *itemlist* and traverses the *tranlist* while keeping a count of the number of occurences of the *itemlist* element. If the count is greater than the support, the *itemlist* element is returned. **(lines 40-49)**
Freq_itemlist is the dictionary that holds the *<item:frequency>* pairs as its elements. **(line 40-44)**

For the association, first a dictionary *final_itemset* is created that that holds the *<item:frequency>* pairs as its elements. **(lines 65-67)**

The *print_results()* function prints the items satisfying the support and computes the association rules. **(line 79)**

How to Run

The run.py is the python file that has our implementation of the Apriori algorithm. It is to be run as;

$ python run.py integrated-dataset.csv <min_sup> <min_conf>

Interesting Sample Run

```
python run.py integrated-dataset.csv 0.3 0.5

python run.py integrated-dataset.csv 0.1 0.5
```