CSCI 6350 Spring 2006
project 1 : Experimental Study of the Association Rule Generation Algorithms
Due: beginning of class, Thursday, February 21st

For association rule discovery with large transaction data, scalability is crucial. This assignment studies the scalability of the Apriori and the FP-Growth algorithm, as well as other characteristics of the algorithms, including their performance with respect to the chosen minimum support and minimum confidence values.

We will use an implementation of the Apriori algorithm by Christian Borgelt and the FP-Growth implementation by Bart Goethals. All program mentioned below can be downloaded from the course web page.

Apriori Algorithm:

After downloading the program, do:

```
%gunzip apriori.tar.gz
%tar -xvf aprior.tar
```

This creates a directory named "apriori".
Change directory to the source directory and compile the program by:

```
%cd apriori/src
%make
```

This generates the executable program "apriori"

Run the program by:

```
%./apriori  -o –s0.1 -c70 -l data1.dat -
```

data1.dat is the file containing all transactions.
The option "-o" specify the use of support definition as used in Agrawal paper
The option "-s" specify the minimum support to be used (-s0.1 means 0.1%)
The option "-c" specify the minimum confidence to be used, -c70 means 70%.
(These two numbers can be changed for different runs of the program on the same data to study the characteristics of the algorithm)
The option "-l" specify the location of the data file
The last "-" specify the output to be displayed on screen, rather than to another file.

Different options of the program can be viewed by type "./apriori" by itself at the Unix prompt.
Read the "readme" files in both the doc/ and the ex/ directories.

FP-Growth Algorithm:

After downloading the program, do

```
%gunzip fpgrowth.tar.gz
% tar –xvf fpgrowth.tar
%cd fpgrowth
% make
```

This generates the executable program “fpgrowth”

Run the program by:

```
% ./fpgrowth DatafileName 3 MinSupportCount OutputfileName
```

Option 3 means the data type is “flat”, all items per transaction on a single line.
Minimum support count, instead of the min support percentage is used in this implementation.
In this fpgrowth implementation, all items have to be in the form of integer values, i.e., no character or string allowed as item name.

Synthetic data generation:

The transaction data generation procedure proposed in [Agrawal, et. al. 94] takes several parameters and generates a set of transactions, each of which consists of a set of items. In order to mimic the transactions in the retailing environment, it is assumed that people tend to buy sets of items together. In [Agrawal et. al 94], such a set is called a "potentially large itemset". The size of a potentially large itemset is chosen from a Poisson distribution with mean equal to a parameter I. Given another parameter L, totally L potentially large itemsets are generated. For each such large itemsets, half of its items are generated randomly, and the other half are picked from the previous itemset. This captures the phenomenon that potentially large itemsets often have common items. Each such itemset is also assigned a weight that is drawn from an exponential distribution with unit mean. The potentially large itemsets are then used to generate D transactions, where D is another parameter. The size of each transaction is picked from a Poisson distribution with mean equal to a parameter T. Each transaction is a union of a set of potentially large itemsets that are chosen one by one according to their weights. To capture the phenomenon that the items in a potentially large itemset are not always bought together by customers, a noise level is set from a normal distribution with mean 0.5 and variance 0.1. While adding a potentially large itemset to a transaction, we keep generating a uniformly distributed random real number between 0 and 1 and dropping an item from the itemset as long as the random number is less than the noise level.

We use, an implementation of this data generation proedure, the Quest Synthetic Data Generation program, from IBM Quest data mining group to generate all synthetic data.

After downloading the program, uncompress the program, and extract the files using:

```
%gunzip assoc-gen.tar.gz
%tar -xvf assoc-gen.tar
```

To compile the program, type

```
%make
```

Run the program with :

```
%gen lit [options]
```

The available options can be viewed by typing:

```
%gen lit -help
```

Here are a description of the options:

-ntrans number_of_transactions_in_000s (default: 1000)
-tlen avg_items_per_transaction (default: 10)
-nitems number_of_different_items_in_000s) (default: 100)
-npats number_of_patterns (default: 10000)
-patlen avg_length_of_maximal_pattern (default: 4)
-corr correlation_between_patterns (default: 0.25)
-conf avg_confidence_in_a_rule (default: 0.75)

-fname <filename> (write to filename.data and filename.pat)
-ascii (default: False)
-randseed # (reset seed used generate to x-acts; must be negative)
-version (to print out version info)

For our experiments:

1. set the sizes of the transaction data (e.g., the number transactions in the data) to 10k, 50k, 100k, 150k, 200k, 300k.
2. set the average size of the transactions to 10;
3. set the number of different items to 1, (that means 1000 different items);
4. set the number of large item sets to 1000;
5. set the average size of the individual large item sets (maximal patterns) to the default value, 10

select -ascii option.

Real world data – msnbc anomymous web data

The data comes from Internet Information Server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of September, 28, 1999 (Pacific Standard Time). Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. Requests are not recorded at the finest level of detail---that is, at the level of URL, but rather, they are recorded at the level of page category (as determined by a site administrator). The categories are "frontpage", "news", "tech", "local", "opinion", "on-air", "misc", "weather", "health", "living", "business", "sports", "summary", "bbs" (bulletin board service), "travel", "msn-news", and "msn-sports". Any page requests served via a caching mechanism were not recorded in the server logs and, hence, not present in the data.

Experiments

(1) Experiment One: Scalability of the algorithms in frequent item generation

Apply assoc-gen program to create six artificial data sets of sizes (transaction size), 10k, 50k, 100k, 150k, 200k, 300k. Run Apriori and FP-Growth algorithms on data, record the CPU time it takes for each data.The min_sup and min_conf values for each data should be kept the same. Min_sup should be set to a value around 1%. Record the CPU times in tabular form, generate the CPU vs. data size plots, and analyze the result. For Apriori algorithm, record CPU time by adding up time used in : (1) reading data, (2) sorting items, (3) recoding loaded item sets, (4) checking subsets of increasing sizes. For FPgrowth, the CPU time at the last line of output should be recorded.

(2) Experiment Two: Characteristics of the Apriori algorithm

Select one data sets from Experiment One, (say, 150k). Apply Apriori on this data > 5 times, each time with a different min_sup value. The min_sup values should be systematically increased, e.g., of the same increment. Keep the min_conf value fixed.

Record :

- the CPU times used to generate the frequent itemsets, and
- the number of rules generated,

in tabular form, and generate the CPU vs. min_sup value plot and number of rules vs. min_sup plot. Analyze the result.

Apply Apriori on the data set > 5 times, each time with a different min_conf value. The min_conf values should be systematically increased.

Record:

- the CPU times used for rule generation, and
- the number of rules generated,

in tabular form, and generate the CPU vs. min_conf plot and number of rules vs. min_conf plot. Analyze the results.

How do the plots differ from those obtained with changing min_sup values?

(3) Experiment Three: Analyze the real world data

Analyze the real world data using Apriori, and try to answer the questions:

- What are the most frequent itemsets?
- According to you, what are the most interesting rules discovered?
- What pages tend to be visited together?
- Which rules do you find interesting? Why?
- What is the min_support, min_confidence values you use to find the most interesting rules?
- When you are determining interesting patterns, what factors did you use, length of itemset? Consequence of a rule? Pages in the itemsets? Support? Confidence? Others?
- How do you think the information discovered from the analysis maybe used?.

What to turn in

You need to turn in a typed mini-report, including

(i) A brief description of the Apriori algorithm.
(ii) A brief description of the FP-Growth algorithm
(iii) The plot that demonstrates the scalability of the two algorithms, e.g., CPU time vs. size of transaction data plots, and show your analysis of the plots.
(iv) The plots that demonstrate the characteristics of the algorithm, e.g., CPU time vs. min_support plots, CPU time vs. min_conf plots, number of rules vs. min_sup plots, number of rules vs. min_conf plots, using both artificially generated data sets
(v) Your analysis results of the real world data set.