Programming Design, Spring 2015
Final Project

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

1 The story

You are running a retail store. A customer comes in, selects some items, checks them out, and then the next customer comes in, and so on. You have experienced this for two years (from 2013/5/1 to 2015/4/30). Your point-of-sales (POS) information system records all the sales transactions in the past two years, including some basic information about consumers and what they purchase. The data are there, though you did not touch them in the past two years.

One day you suddenly realize that these historical sales data can help your business. In particular, you plan to use them to help you upsell, i.e., to recommend a consumer one additional item based on what she/he plans to buy. Instead of doing it randomly, you want to utilize your sales data to make better recommendations, i.e., to recommend an item that is more likely to be purchased.

Unfortunately, your POS system only records transactions; it cannot give you suggestions on what to recommend. The good news is that you know computer programming. Now it is the time to implement your own recommendation system to increase your sales volume.

2 Data format

You have two sets of data: sales transaction data and member ID mapping data.

2.1 Sales transaction data

In each transaction, one consumer buys one or multiple items. To make our lives easier, let's assume that no one buys more than one unit of the same item in a transaction. Each item has a unique ID, which is an integer from 1 to 500. A customer may be a member or not. If she/he is a member, her/his member ID is recorded in a transaction; see below for a detailed description on member IDs. The system also allows the clerk to enter the (estimated) gender and age of a consumer. The gender is recorded as a single letter: M for male, F for female, and O for others. The age is recorded as a single integer between 0 and 255. Finally, the date for that transaction is recorded in the YYYYMMDD format.

As you have upgraded your POS system twice, the system records the set of items purchased in three different ways. At beginning, a Boolean vector $x = (x_1, x_2, ..., x_{500})$ is recorded with $x_i = 1$ meaning that item i is purchased; after the first upgrade, a list of item IDs is recorded; after the second upgrade, each transaction is recorded in multiple rows, one for an item. Below are examples for the three formats. In these examples, we assume that there are only 5 items:

- Transactions recorded in the first format look like:

  ```
  1,20130516,78765123,M,30,1,0,0,1,0
  2,20130516,,F,20,0,0,1,1,0
  3,20130624,,,,0,0,0,0,1
  ```

  Each row represents a transaction, where values are separated by commas. The first value is the transaction ID, which is a positive integer. The next four sets of values are the transaction date, member ID (if there is one), gender (if there is one), age (if there is one), and the five Boolean values. Please note that in the second transaction, there is no member ID. This happens when the consumer is not a member. Please note that in the third transaction, the gender and age are also missing. This happens if the clerk does not enter her/his estimation into the POS system.

- Transactions recorded in the second format look like:

  ```
  1,20130516,78765123,M,30,1,4
  2,20130516,,F,20,3,4
  3,20130624,,,,5
  ```

  The first five values are recorded in the same ways. However, now only the IDs of purchased items are recorded.

- Transactions recorded in the last format look like:

  ```
  1,20130516,78765123,M,30,1
  1,20130516,78765123,M,30,4
  2,20130516,,F,20,3
  2,20130516,,F,20,4
  3,20130624,,,,5
  ```

  The first five values are recorded in the same ways as those in the first two formats. In this format, a transaction with multiple items are recorded in multiple rows, one for each item.

These transactions are provided to you in 24 text files, one for each month. You have no idea about the format of recording transactions in each file before you read that file. Fortunately, the format is consistent within each file. Though some values may be missing as described above, you may assume that all the information are recorded correctly. In particular, the gender and age information of a member is consistent among all past transactions.

2.2 Member ID mapping data

At beginning, you used the last eight digits of a member's cellphone number to be her/his member ID. At a certain time, however, you switched to a new way by letting members choose their IDs. Each member can choose a string of English letters (uppercase or lowercase) and numbers of at most ten characters to be her/his member ID.

Starting from the date of change, you allow old members to update their member IDs by creating the strings they want. When that happens, a new recorded is added into the member ID mapping table. The table looks like:

```
78765123,llcckk
12654789,rrrroooo12
```

Each row contains two IDs, the old one and the new one. The two IDs are separated with a comma. If a consumer makes purchases before and after she/he changes her/his ID, the transactions will be recorded with different IDs. However, by looking at the ID mapping table, you are still able to know that these transactions are made by the same person. The ID mapping table is provided to you in a single text file.

3 Your task

3.1 General description

Given the transaction data and member ID mapping data, your task is to build a recommendation system for future sales. While a true recommendation system is very complicated, in this project we will focus on the following idea: the association among items. As an example, consider cereals and milk. If most people who buy cereals also buy milk, you may recommend milk to a consumer who puts cereals in her/his basket. In general, once you see several items in one's basket, you want to find the item (outside the basket) that is most likely to be purchased *given that* this consumer has decided to purchase these items.

To formalize this idea, let's define some terms. For a set of items S, the *support* is the percentage of historical transactions that contain S (and maybe some other items). For a set of items S and an item $k \notin S$, the *confidence* from S to k is the percentage of historical transactions containing S that also contains k. In other words, the confidence from S to k is

$$\Pr(\text{one purchases item } k | \text{one purchases all items in } S).$$

We say that a confidence is defined with respected to a *source set* S and a target item k. Note that S may contain only one item.

As an example, suppose that there are four transactions for five items. The items included in these transactions are $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{2, 3, 4, 5\}$, and $\{1, 3, 5\}$. Then for items 1, 2, 3, 4, and 5, their supports are 75%, 75%, 75%, 50%, and 50%; for the set of items 1 and 5, the support is 25%. The confidence from item 1 to item 4 is 33% and that from item 4 to item 1 is 50%. Note that the confidences from i to j and from j to i may be different! Finally, the confidence from the set of items 1 and 2 to item 5 is 0: Among all consumer who purchased items 1 and 2 together, no one has purchased item 5.

Now, if you see one consumer puts a set of items T in the basket, it should be a good idea to recommend k to her/him if there is a subset of items $S \subseteq T$ such that

1. The confidence from S to k is high.
2. The support of the set $S \cup \{k\}$ is high.

The first condition tells you that, among those people who bought S in the past, many of them also bought k. Moreover, you know it is not a coincidence if the second condition is also true. In practice, people typically sets a minimum level for supports and look for an association rule $S \to k$ with the highest confidence.

In this project, you will be given hypothetical transactions containing a set of items T. You need to find a source set of items $S \subseteq T$ and a target item $k \notin T$ so that the association rule $S \to k$ has (1) a support above a given level and (2) the highest confidence among all association rules that meets the minimum support requirement. Consider the previous example with four transactions and five items. Now, if a consumer comes to you with a set of items $T = \{2, 5\}$ in her/his basket, you will consider all possible association rules in the following way:

- For the subset of item $S_1 = \{2\}$: The confidences from S_1 to item k, $k = 1, 3, 4$, are 67%, 67%, and 67%, respectively.
- For the subset of item $S_2 = \{5\}$: The confidences from S_2 to item k, $k = 1, 3, 4$, are 50%, 100%, and 50%, respectively.
- For the subset of items $S_3 = \{2, 5\}$: The confidences from S_3 to item k, $k = 1, 3, 4$, are 0%, 100%, and 100%, respectively.

There are three association rules that have the highest confidence (100%): $\{5\} \to 3$, $\{2, 5\} \to 3$, and $\{2, 5\} \to 4$. However, their supports, which are 50%, 25%, and 25%, are different. If we set the minimum support to be 40% or 50%, then $\{5\} \to 3$ would be the most significant association rule obtained for the basket $T = \{2, 5\}$. You will recommend item 3 to this consumer. If we set the minimum support to be 20%, then you may recommend either item 3 or item 4. Of course people typically recommend item 3 because when two confidences are identical, the one associated with a higher support is preferred. Finally, if we set the minimum support to be 60%, then all the three rules do not meet this requirement. In fact, you may see that no association for the basket $T = \{2, 5\}$ meets this requirement. We will have no conclusion (and thus maybe randomly recommend one item) in this case.

To implement your recommendation system, you decide to start by mining the association rules among all you items. To test your system, you will be given hypothetical transactions for you to identify an association from a subset of items inside the basket to an item outside the basket. The association you identify should have the highest confidence among all those meeting the given minimum support requirement. The correctness of your program will be graded based on the association you find. However, do not forget that there is always a time limit; make your program as efficient as possible so that you do not exceed the time limit.

3.2 Technical requirements

Your program should read inputs from PDOGS. There will be 30 input data files, each containing 26 lines, one for a file name. The first 24 lines contain the 24 files of past transactions, where the first is for May 2013, the second is for June 2013, ..., and the last is for April 2015. The 25th line contains the file name of the member ID mapping table. The 26th line contains the file name of the hypothetical transactions. Each line is a sequence of English characters and numbers with no white space, a dot, "txt" as the extension name, and a newline character.

All the 30 input files refer to the same 24 past sales data and the same member ID data. However, they refer to different hypothetical transaction data. In each hypothetical transaction file, there will be multiple lines of transactions recorded in the format after the first update but before the second update. Nevertheless, there is no date for hypothetical transactions. Moreover, at the end of each line, one additional positive integer s will be given to you after a semicolon. This means that $s\%$ is the minimum support for this hypothetical transaction. Naturally, you have $1 \leq s \leq 100$.

For each hypothetical transaction of the set of items T, you need to search for an association rule $S \to k$, $S \subseteq T$, $k \notin T$, which has the highest confidence among all those meeting the $s\%$ minimum support requirement. Your search depends on whether the member or gender information is available:[1]

- If there is a member ID in the hypothetical transaction, you should consider those past transactions made by consumers of the same gender as this member.
- If there is a gender that is M or F, you should consider those past transactions made by consumers of the same gender as this consumer.
- If there is no member and gender information in the hypothetical transaction, you should consider all past transactions.

When your winning association is unique, print out the target item of the association rule, support of this association rule, and confidence of the rule. Separate them with two white spaces. The support and confidence should be naturally rounded to the second digit after the decimal point;[2] the two resulting digits after the decimal point should then be printed out. If there are multiple winning associations, print out the one with the highest support. If there is still a tie, print out the one whose target item has the smallest ID. If no association meets the minimum support requirement, print out `-1`.

As an example, suppose there are four past transactions $\{1,2,3\}$, $\{1,2,4\}$, $\{2,3,4,5\}$, and $\{1,3,5\}$. If a hypothetical transaction is

```
1,,,3,2,5;50
```

which means in the first hypothetical transaction, a 3-year-old non-member with uncertain gender puts items 2 and 5 in her/his basket; the minimum support required is 50%. Please note that there is no field for dates! For this hypothetical transaction, the output should be

```
3 50 100
```

[1]The date and age information is just to complicate your data processing works. They will not be used in mining the association rules.

[2]For example, 0.334 rounds to 0.33 and 0.125 rounds to 0.13.

which means that we will recommend item 3 to this hypothetical consumer with support 50% and confidence 100%.

For an input file, you are considered correct if you find the correct winning association rules for all given hypothetical transactions. The key challenge of this task is the efficiency of your program. While some input testing files will be given with loose time limits, some will be with pretty strict ones. Naive algorithm and implementation gives you some points, but getting full points really requires you to design a good algorithm and create an efficient implementation.

4 Rules

4.1 Team formation and tasks

Please form a group of ***at most four*** people for this project. One student cannot participate in two teams. In each team, at most one student can be an "excellent start" student (i.e., her/his average grades for the first two lab exams ranked in the top 20% in the class; see the instructor's e-mail for the list). Please indicate the team name and each team member's name and student ID on the sign up sheet (see the instructor's e-mail for the address) and your written report.

For this project, each team needs to do the following tasks:

1. Write a program, upload it to PDOGS, complete with other teams, and earn points based on the relative performance. PDOGS will be closed at ***23:59:59*** on ***June 21, 2015***.
2. Demonstrate your program and explain your design in class. For any feature or functionality not required in the problem, you may demonstrate them in class. Demonstrations will be held on ***June 22, 2015***. On the demonstration day, half of the teams will be randomly chosen to be "presenters" while the others will be "criticizers." Presenters and criticizers will be randomly paired up. After a presenter's presentation, the corresponding criticizer asks questions and makes comments.
3. Write a written report (I mean, type it) to summarize all your works. Your report is due at your demonstration. Please submit a PDF file on PDOGS by ***23:59:59*** on ***June 21, 2015 AND*** a hard copy in class on ***June 22, 2015***. In the report, please describe:
   (a) The design of your algorithm. Please describe how it works and why it is designed in this way in words with no code.
   (b) The design of your system. Describe the meaning of those classes or functions in words.
   (c) The work division of your team. E.g., who wrote the part of reading input data, who implemented the main algorithm, who did the logistics by buying dinner for teammates, etc.
   (d) Any other feelings you have about this project.

   Your report cannot be longer than six pages (i.e., three double-sided sheets). It can be written in English or Chinese.

4.2 Grading

- 60 points will be given based on the performance of your system. On PDOGS, you may see all teams' grades. Of course, you have no idea who they are.
- 10 points will be given based on the structure of your program and the comments therein.
- 20 points will be given based on the design of the algorithm and system and the report quality.
- 10 points will be given based on the performance of demonstrations.

5 One final remark

You probably wonder what is a good algorithm to make the recommendation. There are of course some classic algorithms while researchers are still trying to invent better ones. To understand those algorithms, or even to design a good one, you probably want to study Data Mining, Statistical Learning, Marketing, Operations Management, Consumer Behaviors, Electronic Commerce, etc. To be able to make good implementations, you probably want to study Data Structures, Algorithms, Database Management, Software Development Methods, etc. Some foundations for these courses are Management Mathematics, Discrete Mathematics, Statistics, Operations Research, and, as always, Calculus.

Let's why I always believe that studying Information Management is useful!