CS300 – Spring 2019-2020 - Sabancı University

Homework #3 – Marketing Strategy

Due: 18/04/2020, 23:55

Brief description

In this homework we, as the manager of a grocery store, we ask you to develop a tool which will help us to improve our marketing strategy. We will provide you the purchased transactions of our customers as an input file and your tool needs to extract all the sales relations between the products. For this purpose, you are required to use a **templated** hashtable with **separate chaining**.

Overview

When we go shopping, we generally have a standard list of things to purchase. This list is prepared based on one's needs and preferences for each customer. A housewife might purchase fresh food like vegetables or milk for the kitchen whereas someone alone might purchase chips and coke. Understanding these buying interests can offer assistance to extend deals in a few ways.

For instance, if X and Y are frequently bought together:

- X and Y can be both placed on the same shelf, so that buyers of one item would be prompted to buy the other.
- Promotional discounts could be applied to just one out of the two items (X or Y) since buying one would encourage the buyer to buy the other one, too.
- Advertisements on X could be targeted at buyers who purchase Y.

The Goal

The goal of this homework, for a given list of transactions, is to extract all possible implications. For instance;

```
chips -> coke
```

This implies that, for the customers who bought chips most likely bought coke, too. Or, as another example, consider the following implication;

```
sugar, milk -> flour
```

Implies that, the customers who bought sugar and milk most likely bought flour, too.

Input File Format

Your program needs to ask two numbers (ranging from 0 and 1) for given support and confidence threshold values (see their definitions in the document for details below) from the user. Then, your program needs to ask the name of the input file. You can decide on the appropriate display messages for them.

The input file contains the list of purchased transactions of our customers. In each line, there are several products (items) that are purchased for that time of shopping. For a transaction, there is no repeated product and the number of products are not fixed. An example of input file is given as below:

inputFile.txt

```
milk bread bananas cake
bread apple onion
coke chips sandwich
bananas yoghurt chocolate grapes orange
milk orange bread
```

In the input file, there are no empty input lines and all of the words are given as in lower cases. The words in the same line are separated with an empty space. An item will be only one word, so, you will not see an item containing multiple words such as “orange juice”.

Output File Format

The output file format needs to be **exactly the same** as described here. You need to write your outputs to a file named as results.txt. In each line there will be only one rule. For each rule, you need to put commas between the products of the same side of the implication and you need to put a semicolon instead of the implication symbol (**->**). So, basically, what we expect from you is to generate the file as given in the second column of the following table if the extracted implications are as given in the first column of the table. There will be no space character in the output file. Moreover, you need to give the confidence values of rules in the output file. Round your confidence values having 2 decimals, i.e., having 2 digits after the dot as below. Continue reading this document for more details on confidence values.

Extracted Rules with Confidences	Output file format
A -> B conf = 0.30123	A;B;0.30
{B, C} -> D conf = 0.12139	B,C;D;0.12
{B, C} -> {A, D} conf = 0.32125	B,C;A,D;0.32

Background Information

Before going into detail with the algorithm, we need to define two concepts: *support* and *confidence*.

Support describes how an item (or set of items) are popular among the whole item set. Hence, the support value of an item (or set of items) is the ratio of the number of times the item appeared among all the transactions. For instance, if there are 10 transactions in the list and milk is purchased 4 times, the support value of milk is 0.4. On the other hand, if you need to calculate the support value of several items together, such as milk and bread, you need to find the number of transactions which contain both milk and bread together to find the support value of them.

Confidence value describes how likely item Y is purchased when item X is purchased, expressed as X->Y. This is calculated as:

```
confidence(X -> Y) = support(X, Y) / support(X)
```

For an implication having multiple items, such as {X, Y} -> {Z, W}, confidence is calculated as:

```
confidence({X, Y} -> {Z, W}) = support(X, Y, Z, W) / support(X, Y)
```

For instance; for the given support values;

```
support(bread) = 0.5
support(milk) = 0.4
support(bread, milk) = 0.2
```

The confidence of {bread -> milk} is calculated as:

```
confidence(bread -> milk) = support(bread, milk) / support(bread)
                          = 0.2 / 0.5 = 0.4
```

Whereas the confidence of {milk -> bread} becomes,

```
confidence(milk->bread) = support(milk, bread) / support(milk)
                        = 0.2 / 0.4 = 0.5
```

The Flow of the Program

The program has 2 main parts: in the first part, you need to find the support value of several item sets such as `support({bread})` and `support({bread, milk})`. Then, by using those support values, you need to calculate the confidence values and extract the rules.

Calculating Support Values

- Find all support values of items in the transaction such as `support({bread})` and `support({milk})`.
- Store all the items whose support value is greater than or equal to the given support threshold value.
- Find all possible item pairs of the stored items (selected in the previous step).
- Find all support values of item pairs in the transaction such as `support({bread, milk})`, `support({apple,orange}).`
- Store all the items pairs whose support value is greater than or equal to the given support threshold value.

Rule Extraction Process

Up to this point, you have discovered all items and item pairs whose support values are greater than or equal to given support threshold value. Assume that all of these items and item pairs are stored in the **same** hash table named as *lookupTable*.

Using *lookupTable* (a hash table) you need to enumerate all possible 2 way permutations of all the *lookupTable* elements. For example, assume that in the hash table, we have the following elements:

`{A}, {B}, {C}, {A, B}` and `{B, C}`

You need to find all possible 2 length permutations of these elements and construct the rules as below:

```
A -> B
A -> C
A -> {B, C}
B -> A
B -> C
C -> A
C -> B
C -> {A, B}
{A, B} -> C
{B, C} -> A
```

However note that, there can not be the same item on both sides of the implication. For example, the following rules are **not valid**.

```
A -> {A, B}
{A, B} -> {B, C}
{A, B} -> A
```

After enumerating all possible rules, you need to calculate the confidence values of them. Note that you may need to calculate the support value of item sets having 3 or 4 elements in that process. Simply, make another input file reading and calculate the support values of only those item sets i.e., not every 3 or 4 length item sets. For example, for the rules given above, you need to read the input file again to find the support value of only item set: {A, B ,C}.

The output of your program will be the rules whose **confidence values are greater than or equal to the given confidence threshold value**.

Sample Runs

in.txt

```
milk bread cake
bread apple onion
coke chips sandwich
bananas yoghurt chocolate grapes orange
milk orange bread
yoghurt bananas juice
apple cucumber garlic
tea egg melon
bread milk juice bananas orange
milk tea bread
bread milk chocolate
milk coffee
broccoli bananas orange bread milk
```

Sample Run 1

For the given input file above, the output will be:

```
Please enter the transaction file name: in.txt
Please enter support and confidence values between 0 and 1: 0.20 0.60
14 rules are written to results.txt
```

results.txt

```
bananas;orange;0.75
orange;bananas;0.75
bread,orange;bananas;0.67
orange,milk;bananas;0.67
orange;bread;0.75
bread;milk;0.86
milk;bread;0.86
bananas,orange;bread;0.67
```

```
orange,milk;bread;1.00
orange;milk;0.75
orange;bread,milk;0.75
bananas,orange;milk;0.67
bread,orange;milk;1.00
bananas,orange;bread,milk;0.67
```

Sample Run 2

For the given same input file above, the output will be:

```
Please enter the transaction file name: in.txt
Please enter support and confidence values between 0 and 1: 0.60 0.45
There is no rule for the given support and confidence values.
```

Sample Run 3

For the given same input file above, the output will be:

```
Please enter the transaction file name: in.txt
Please enter support and confidence values between 0 and 1: 0.10 0.55
24 rules are written to results.txt
```

results.txt

```
bananas;orange;0.75
orange;bananas;0.75
juice;bananas;1.00
yoghurt;bananas;1.00
bread,orange;bananas;0.67
orange,milk;bananas;0.67
orange;bread;0.75
bread;milk;0.86
milk;bread;0.86
bananas,orange;bread;0.67
bananas,milk;bread;1.00
orange,milk;bread;1.00
orange;milk;0.75
bananas,bread;orange;1.00
bananas,milk;orange;1.00
orange;bread,milk;0.75
bananas,bread;milk;1.00
bananas,orange;milk;0.67
bread,orange;milk;1.00
```

```
bananas,bread;orange,milk;1.00
orange,milk;bananas,bread;0.67
bananas,orange;bread,milk;0.67
bananas,milk;bread,orange;1.00
bread,orange;bananas,milk;0.67
```

Frequently Asked Questions

- **How am I going to use the hash table?**
  - Since the search function takes constant time in the hash table, you will use it to find the number of occurrences of items and item pairs.
- **Can I generate all item pairs while I am reading the input file?**
  - No. You need to enumerate all item pairs from the items whose support values are greater than or equal to the given threshold value.
- **How am I going to find the support value of k items?**
  - You need to find the number of transactions which contains all of the k items at the same time. Then, divide this number with the number of total transactions. Normally, what you have in your hash table contains only items and item pairs. Thus, you need to read the input file again to find the support values of item sets when needed. See the updated Rules Extraction section for details.
- **Do I need to rehash?**
  - Yes, you need to rehash when necessary.
- **Do I need to make input validation?**
  - No, you can assume that all of the given input values are valid.
- **Do I need to print the rules in the same order as you did in the sample run?**
  - No, you do not need to do that as long as you find the same rules.

General Rules and Guidelines about Homeworks

The following rules and guidelines will be applicable to all homeworks, unless otherwise noted.

How to get help?

You may ask questions to TAs (Teaching Assistants) of CS300. Office hours of TAs can be found here. Recitations will partially be dedicated to clarify the issues related to homework, so it is to your benefit to attend recitations.

What and Where to Submit

Please see the detailed instructions below/in the next page. The submission steps will get natural/easy for later homeworks.

Grading

Careful about the semi-automatic grading: Your programs will be graded using a semi-automated system. Therefore, you should follow the guidelines about input and output order; moreover, you should also use the exact same prompts as given in the Sample Runs. Otherwise semi-automated grading process will fail for your homework, and you may get a zero, or in the best scenario you will lose points.

Grading:

- **We will hold a demo session for your homework grading. Each one of you must show that your code is running as expected and explain several parts of your code if necessary. Wait for an announcement for the demo scheduling.**
- Late penalty is 10% off the full grade and only one late day is allowed.
- **Having a correct program is necessary, but not sufficient to get the full grade. Comments, indentation, meaningful and understandable identifier names, informative introduction and prompts, and especially proper use of required functions, unnecessarily long program (which is bad) and unnecessary code duplications will also affect your grade.**
- Please submit your own work only (even if it is not working). It is really easy to find out "similar" programs!
- For detailed rules and course policy on plagiarism, please check out http://myweb.sabanciuniv.edu/gulsend/courses/cs201/plagiarism/

Grade announcements: Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

Plagiarism will not be tolerated!

Grade objections: Since we will grade your homeworks with a demo session, there will be very likely no further objection to your grade once determined during the demo.

What and where to submit (IMPORTANT)

Submission guidelines are below. Most parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Add your name to the program: It is a good practice to write your name and last name somewhere in the beginning program (as a comment line of course).

Name your submission file:

- Use only English alphabet letters, digits or underscore in the file names. Do not use blank, Turkish characters or any other special symbols or characters.
- Name your cpp file that contains your program as follows.

  "SUCourseUserName_yourLastname_yourName_HWnumber.cpp"

- Your SUCourse user name is actually your SUNet username which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

  cago_ozbugsizkodyazaroglu_caglayan_hw3.cpp
- Do not add any other character or phrase to the file name.
- Make sure that this file is the latest version of your homework program.
- You need to submit ALL .cpp and .h files including the data structure files in addition to your main.cpp in your VS solution.
- The name of the main cpp file should be as follows.

  "SUCourseUserName_yourLastname_yourName_HWnumber.cpp"

  For example zubosman_Osmanoglu_Zubeyir_hw3.cpp is a valid name, but hw3_hoz_HasanOz.cpp, HasanOzHoz.cpp are NOT valid names.

Submission:

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Hanefi Mercan and Gülşen Demiröz