KTH - Kungliga Tekniska högskolan - DATA MINING 2017

Assignment 2: Discovery of Frequent Itemsets and Association Rules

Filippo Boiani <boiani@kth.se>
Riccardo Sibani <rsibani@kth.se>

19 Nov 2017

NOTE
The code can be found at the following link: https://github.com/filippoboiani/data-mining

Introduction

Implementing the two subproblems as defined in [R. Agrawal and R. Srikant, VLDB '94]
- Finding frequent itemsets with support at least ***s;***
- Generating association rules with confidence at least ***c*** from the itemsets found in the first step.

Our implementation is in pure Java and tries to follow as much as possible the algorithms proposed in the paper. However, some operation between sets have been replaced with more efficient method (such as matrixes of booleans).

How to Run

- via console, open the assignment directory
- run the following set of commands

```
cd frequent_itemsets_jar
java -jar frequent_itemsets.jar <filename> <threshold> <confidence>
```

Command Line Parameters

The .jar file accepts either 3 or no parameter.

NO PARAMETER:

```
java -jar frequent-itemsets.jar
```

Defaults:
- file: T10I4D100K.dat
- threshold: 0.01 (1%)
- confidence: 0.5

WITH PARAMETERS:

```
java -jar frequent-itemsets.jar T10I4D100K.dat 0.01 0.5
java -jar frequent-itemsets.jar <filename> <threshold> <confidence>
```

Approach

The project has the following classes:
- **Main**: the main class reads the input parameters (or the default ones), instantiate the **AprioriAlgorithm** class with these parameters and run first the **findFrequentItemsets** method, then the **findAssociations** within the frequent itemises found.
- **AprioriAlgorithm**: exposes 2 methods the **findFrequentItemsets** and the **findAssociations**, the implementation follow the algorithm proposed in the paper. When the class is instantiated, the input file is prepared and converted to a List of integer arrays. After that the constructor filters the frequent singletons which will be the starting set for the apriori algorithm.
- **Combination**: this utility class is in charge of computing combination (n over k), cartesian products between sets and operations on arrays.

Results

Description of the file and the parameters:

Parameter	Value
File Name	T10I4D100K.dat
Items	1000
Baskets	100000
Threshold (baskets)	**1000 (1%)**
Min Confidence	**0.5**

FREQUENT ITEMSETS

```
Frequent itemsets of size 3 (1):
[39, 704, 825]

Frequent itemsets of size 2 (9):
[39, 704][39, 825][217, 346][227, 390][368, 682][368, 829][390, 722][704,
825][789, 829]

Frequent singletons (375):
[1][4][5][6][8][10][12][17][21][25][27][28][31][32][33][35][37][38][39]
[41][43][45][48][51][52][54][55][57][58][68][69][70][71][72][73][75][78]
[85][90][93][94][97][100][104][105][110][111][112][115][116][120][122]
[125][126][129][130][132][140][143][145][147][151][154][157][161][162]
[163][168][170][171][173][175][177][181][183][185][192][196][197][198]
[201][204][205][207][208][210][214][217][227][229][234][236][239][240]
[242][258][259][265][266][274][275][276][279][280][283][285][290][294]
[296][308][309][310][319][322][325][326][332][334][335][336][343][346]
[348][349][350][351][354][357][361][362][366][368][373][377][378][381]
[385][387][390][392][394][401][403][405][411][413][414][419][422][423]
[424][427][428][429][438][440][448][449][450][458][460][461][468][469]
[470][471][472][477][480][486][487][489][490][494][496][500][504][509]
[510][511][513][515][516][517][521][522][523][526][527][529][530][534]
[538][540][541][546][548][550][554][561][563][567][569][571][572][573]
[574][576][577][578][579][580][581][583][591][593][594][597][598][600]
[605][606][611][614][617][618][620][623][628][630][631][632][634][638]
```

```
[639][641][649][651][653][658][661][663][665][673][674][675][676][678]
[682][684][686][687][688][692][694][701][704][706][707][708][710][716]
[718][720][722][733][735][736][738][740][744][746][749][752][758][763]
[765][766][769][774][775][778][780][782][784][788][789][790][792][793]
[795][797][798][800][803][804][805][809][810][812][814][815][819][820]
[823][825][826][829][832][834][841][843][844][846][853][854][857][859]
[860][862][867][871][874][878][883][884][885][886][887][888][890][893]
[895][897][899][900][906][910][912][913][914][918][919][921][923][928]
[932][935][937][941][944][946][947][948][949][950][952][956][960][963]
[964][966][967][970][975][978][981][982][984][988][989][991][992][995]
[998]
```

ASSOCIATION RULES

```
RULE: [704] => [39]      with conf = 0.617056856187291 and support= 1107
RULE: [227] => [390]     with conf = 0.577007700770077 and support= 1049
RULE: [704] => [825]     with conf = 0.6142697881828316 and support= 1102
RULE: [704, 825] => [39] with conf = 0.9392014519056261 and support= 1035
RULE: [39, 825] => [704] with conf = 0.8719460825610783 and support= 1035
RULE: [39, 704] => [825] with conf = 0.9349593495934959 and support= 1035
RULE: [704] => [39, 825] with conf = 0.5769230769230769 and support= 1035
```

RELATIONSHIP BETWEEN COMPUTATION TIME AND SUPPORT THRESHOLD