**FINDING ASSOCIATIONS**

**IN COMPUTER SCIENCE COURSE DATASET**

**PROJECT 3**

Vasanthi Vuppuluri

**CONTENTS:**

I. INTRODUCTION

II. DATASET AND PREPROCESSING

1. The Raw Dataset
2. The Refined Dataset
3. The input for the algorithms

* R-Code for importing Datasets
* Frequency plots

III. IMPLEMENTING ALGORITHMS

1. On Course Dataset
   * Results – Rules and Interpretations
   * Rule Visualization
2. On Instructor Dataset
   * Results – Rules and Interpretations
   * Rule Visualization

IV. SUMMARY

V. REFERENCES

**APPENDIX:**

APPENDIX – A

* Association Analysis

APPENDIX – B

1. Apriori Algorithm
2. Eclat Algorithm

APPENDIX – C

1. Interestingness measure
2. Remove redundant rules

APPENDIX – D

* Gephi

**I. INTRODUCTION**

In a nutshell, we used the dataset that comprised of the courses taken by the students graduated from the Computer Science Department of University of Houston to generate interesting rules on how and students chose courses in our department and to find out the courses that were frequently opted by the students. This is a novel idea and to our knowledge this is the very first time Data Mining is performed on the UH CS dataset.

We hope that using this information we can help students decide their courses in their future semesters depending upon the courses they already completed. Also, we can find out the areas that students find more interesting hoping that can help us concentrate more on the research in those areas.

Since our project’s goal is to implement an association rule mining algorithm, we tried not focusing too much on analyzing the details of the algorithms. Instead, we are concentrating on interpreting the findings and hope this could help students handle their course selection.

**II. DATASET AND PREPROCESSING**

i. The raw dataset:

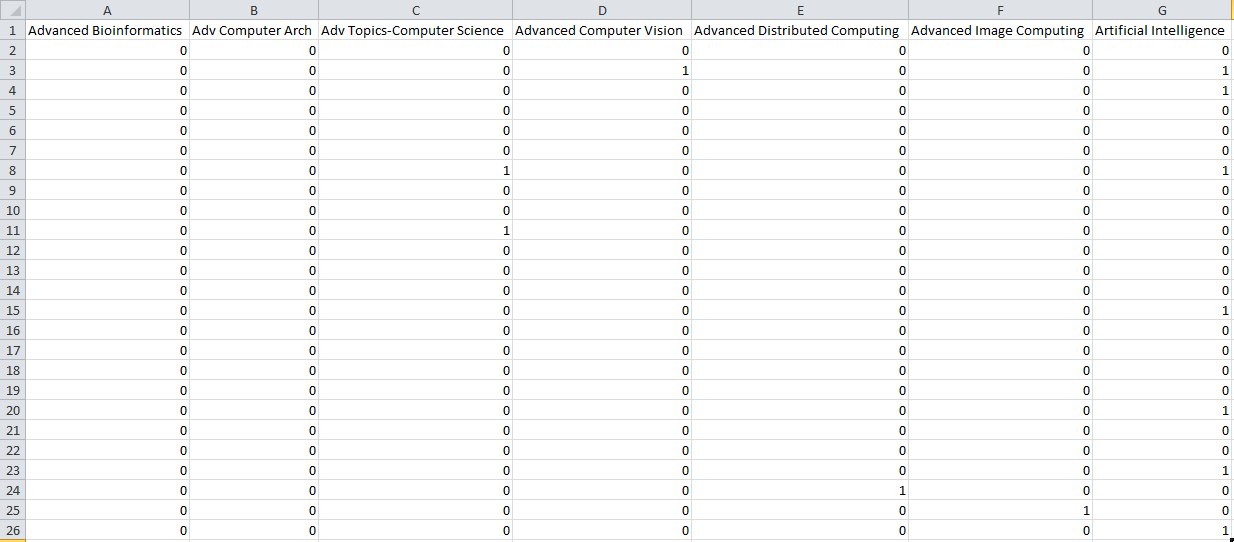
The original file provides by Jackie composes 1213 rows and 8 columns. We only used the attributes (columns) that we are interest in, named by ‘ID’, ‘course description’ and ‘Instructor’. The reason why we select course name instead of the number of course is because we found that some of the course number has been changed in different semesters. We used the course information of 92 graduated students from spring-2010 to fall-2012.

This is a screenshot of the raw data we got from the Department:

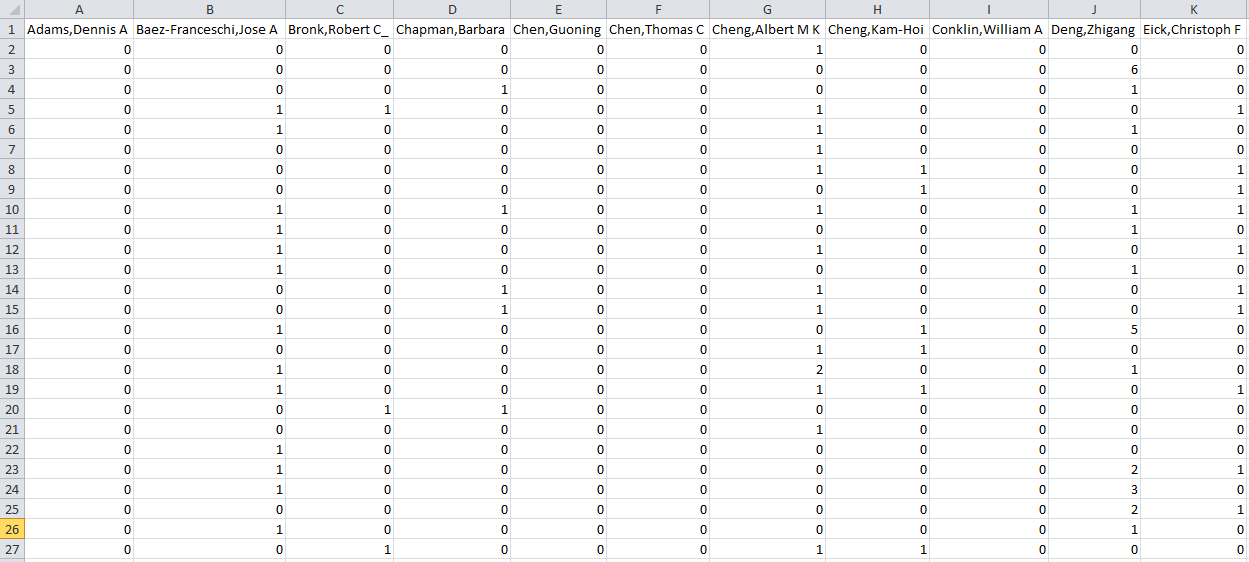
# Not permitted to disclose

ii. The refined dataset:

We concatenated the course description to one line per person and formed a long vector:

Then we constructed a table with 1 and 0 in it and used it as transactions. The column headings represent the Subject names and rows represent each of the student. If a student took a course, then it is represented as ‘1’ in the dataset or else as ‘0’. Screenshot of the .csv file looks as follows:

Also, we have a file with Instructor names as column headings and number of students and each of the rows represent the students and the number of times a student took a course with that Instructor is represented by the cell value in the excel file. Sample looks as follows:



iii. The input for the algorithms

After we imported the \*.csv format data files into R studio, we transformed them into ‘transactions’ structure which could be imported by Apriori algorithm.

We have two input files:

* Instructors.csv: Student and Instructor data
* Grad-Data-CourseNames.csv: Student and Course data

Reading the input files:

1. From ‘Grad-Data-CourseNames.csv’

# Reading input from .csv file

Grad.Data.CourseNames <- read.csv("Grad-Data-CourseNames.csv")

course\_data<-Grad.Data.CourseNames

# Transforming course\_data into Matrix

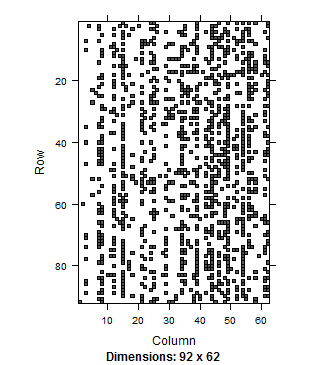
Course\_Data\_Matrix <- as(course\_data, "matrix")

# Converting Course\_Data\_Matrix to transactions which will be used by Apriori and Eclat

Course\_Data\_table <- as(Course\_Data\_Matrix, "transactions")

#Pictorial (Matrix) representation of the dataset with 62 courses and 92 Graduated students

image(Course\_Data\_table)



# Frequency plot

itemFrequencyPlot(Course\_Data\_table, col=1:8)

The frequency plot shows us that the course ‘Computer Architecture’ has the highest frequency amongst all other courses. The second most frequent course is ‘Machine Learning’ followed by ‘Operating Systems’, ‘Real Time Systems’ and ‘Inter. Game Art and Animation’.

Amongst the least frequent courses opted by students are: ‘Advanced Bio-Informatics’, ‘Advanced Computer Vision’, ‘Business Applications of Database Management Systems’, ‘Computer Organization Programming’ and ‘Ethics in Science’.

b. From ‘Instructors.csv’

# Reading input from the .csv file

Instructors <- read.csv("H:/Professional/FALL 2013/DM/Project 2/Project-3/Instructors.csv")

Instr <- Instructors

# Transforming Instructors into Matrix

Instr\_Matrix <- as(Instr, "matrix")

# Converting Instr\_Matrix to transactions

Instr\_Data\_table <- as(Instr\_Matrix, "transactions")

# Frequency plot

itemFrequencyPlot(Instr\_Data\_table, col=1:8)

The frequency plot shows that the Instructor ‘Dr. Paris Jehan Francois’ taught more number of students in the past three years compared to other instructors, followed by, ‘Dr. Ricardo Vilalta’, ‘Dr. Pavlidis Ioannis T’, ‘Dr. Albert M K Cheng’ and ‘Dr. Yuriy Fofanov’.

**III. IMPLEMENTING ALGORITHMS**

 # Importing libraries

library(arulesViz) – The *arulesViz* add additional features for graphing and plotting the rules.

library(arules) – Association rules use the R *arules* library.

1. On Course Dataset

Generating rules:

# 'arules' – APRIORI

# The rules can then be created using the apriori function on the transaction dataset “Course\_Data\_table” that was imported earlier

# ‘rules’ has the set of association rules generated by the apriori algorithm

# After performing a lot of trial and error on Support and Confidence values, we got the optimum rules when support was 0.2 and confidence was 0.5

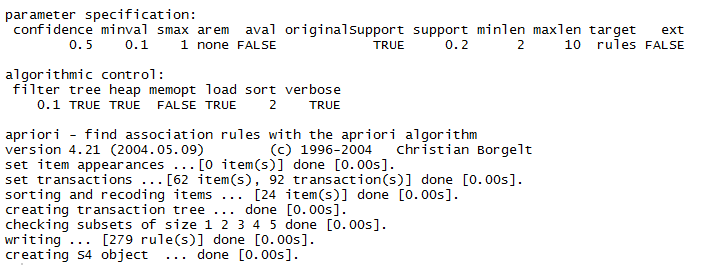
# minlen=2 states that there antecedent should have atleast one course

rules <- apriori(Course\_Data\_table, parameter=list(support=0.2, confidence=0.5, minlen=2, target = "rules"))

279 rules were generated.

Of the rules generated there are redundant rules which are unnecessary and reduce the quality to the set of rules that were generated.

Output looks as follows:



Our next step is to find the redundant rules that were generated.

# We sort the rules by the ‘lift’ parameter

rules.sorted <- sort(rules, by="lift")

# find redundant rules

# Matrix of the sorted rules is created

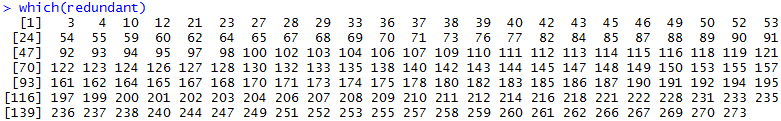
subset.matrix <- is.subset(rules.sorted, rules.sorted)

subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA

redundant <- colSums(subset.matrix, na.rm=T) >= 1 # The redundant rules are stored into ‘redundant’

which(redundant) # Displays redundant rules as output

Screen shot of the output is as follows:



Each of the number represents one of the 279 rules generated and that are redundant in nature.

Now we need to remove the redundant rules from the rule set we have.

# Remove redundant rules

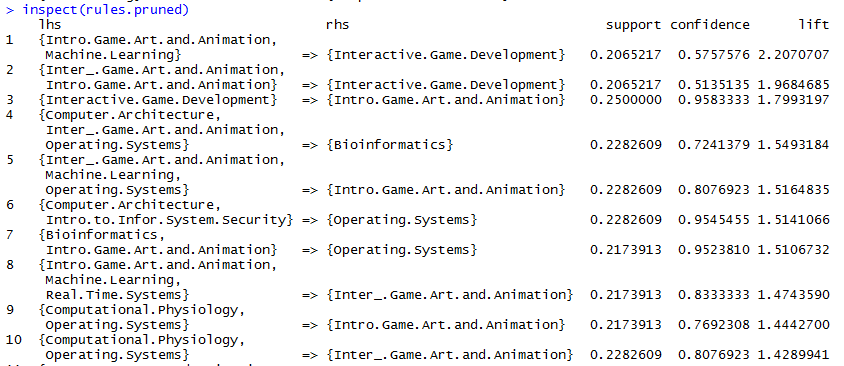
# The rules that are remaining after remaining the redundant rules are stored into rules.pruned

rules.pruned <- rules.sorted[!redundant]

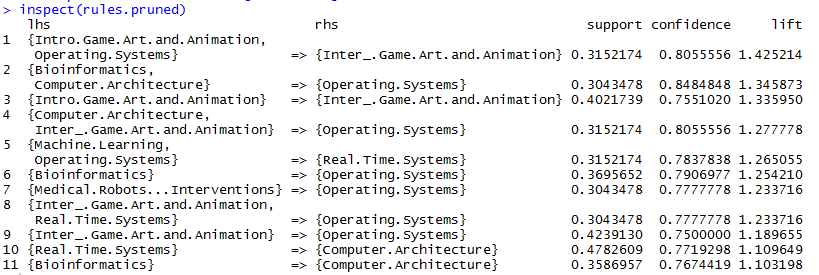
inspect(rules.pruned) # Displays the set of non-redundant rules

After removing the redundant rules we are left with 119 rules. The rules are sorted by the ‘lift’ parameter.

Screenshot of some of the rules is as follows:

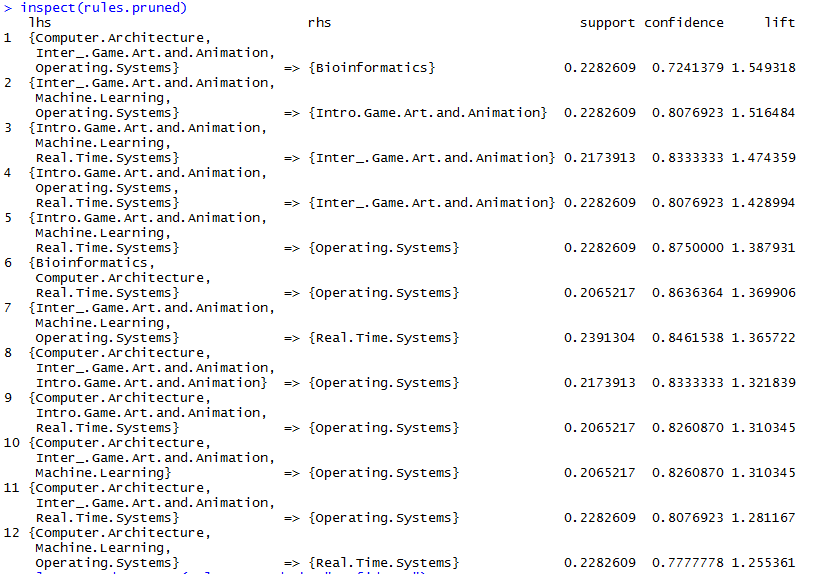


Because we cannot discuss all the rules generated in this report (keeping in mind the space constraint), we increased the support to 0.3 and confidence to 0.75 and obtained a set of 11 pruned rules. They are:



These rules state that if a student took ‘Introduction to Game Art and Animation’ and ‘Operating Systems’ then the student also took ‘Intermediate Game Art and Animation’. Using this rule, we can estimate the number of students who will take ‘Intermediate Game Art and Animation’ in the next semester depending upon the strength of students who took ‘Intro to Game Art’ and ‘Operating Systems’ in their earlier semesters. Also, we can help the students decide which course they can opt for if they completed the course on the LHS side.

When the support and confidence were changed back to 0.2 and 0.5, but minlen was changed to 4, we got 12 pruned rules as follows:



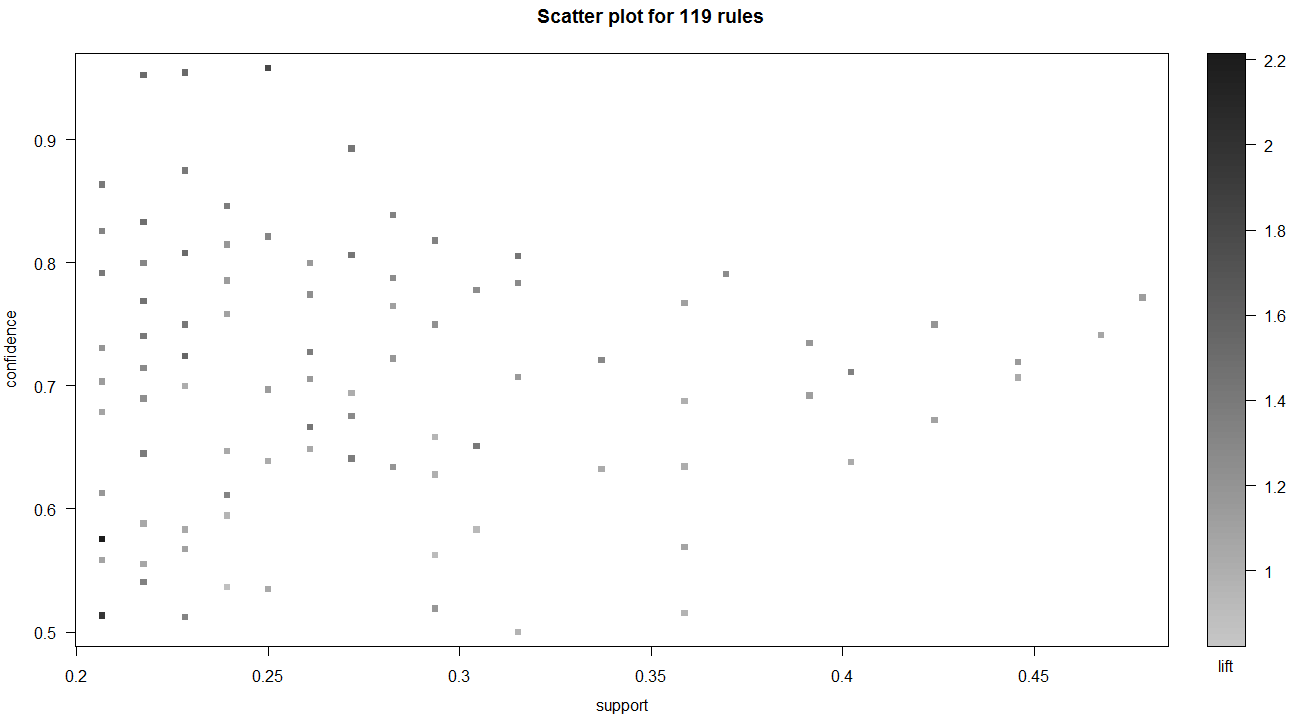
Visualizing the rules:

# Visualizing the rules - 'arulesviz'

plot(rules.pruned, col=1:8)

We are plotting the 119 pruned rules that were generated when support was 0.2, confidence was 0.5 and minlen was 2.

The scatter plot is dense when support was low, from 0.2 to 0.3 and for the confidence values 0.6 and 0.8 which imply that more number of rules exist in this region of the support and confidence values.



We now generated a graph of 119 pruned rules and saved the graph in ‘graphml’ format which can be accessed using ‘Gephi’.

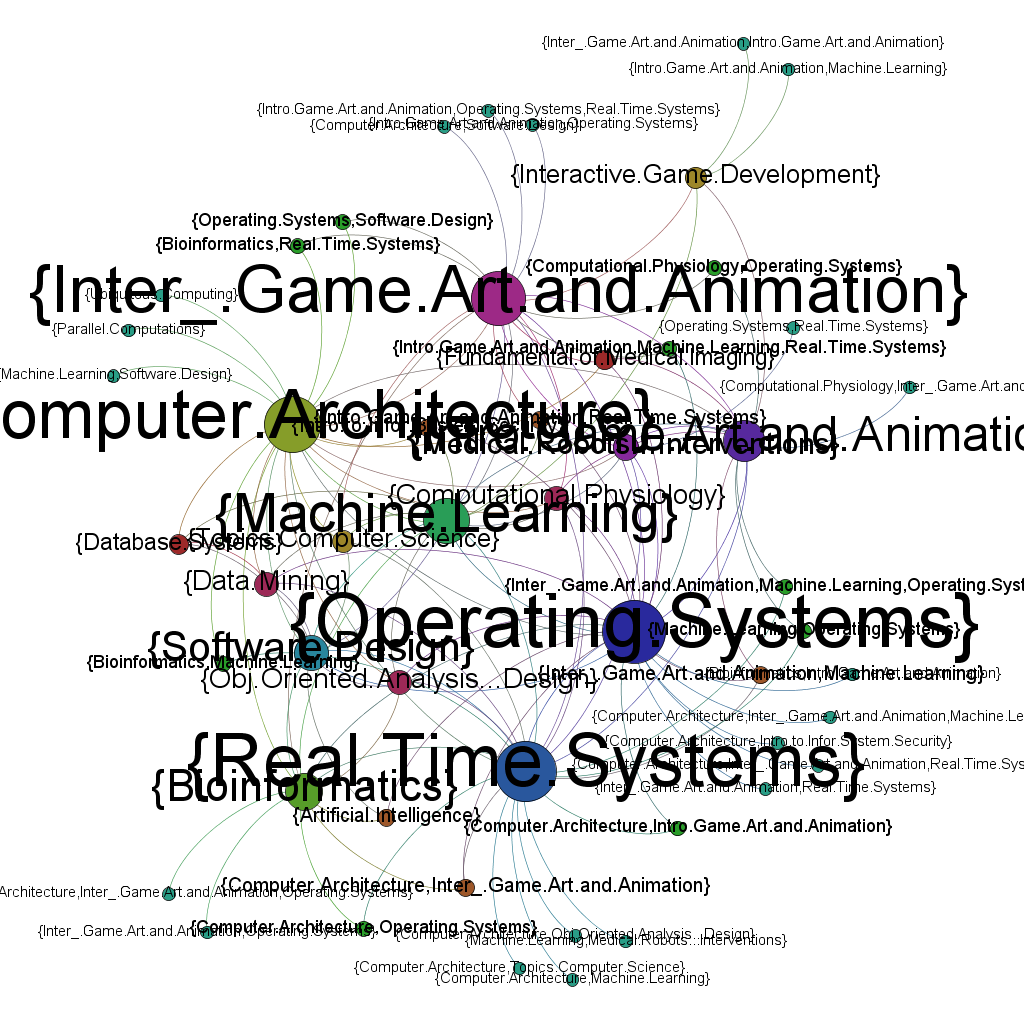
# Generates a graph of the rules

saveAsGraph(head(sort(rules.pruned, by="lift"), 1000), file="H:/Professional/FALL 2013/DM/Project 2/Project-3/rules.graphml")

The graphical representation of all the rules is as follows:

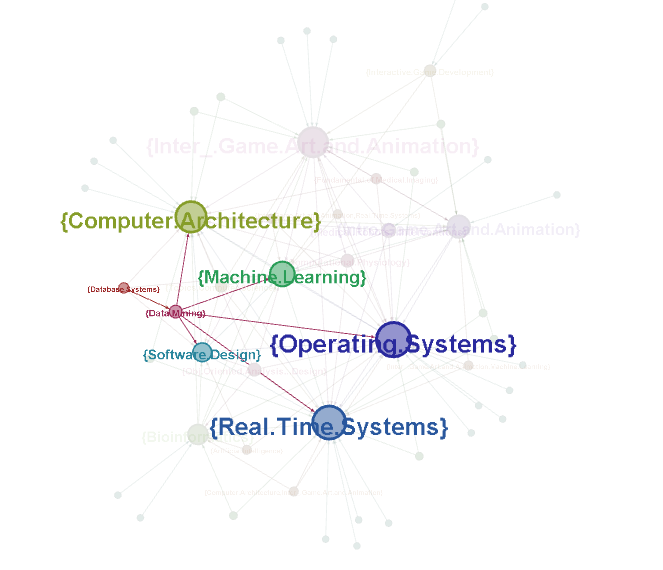
The bigger font represents that those courses were more frequent and more rules are involved with those courses. The graph below states that the courses ‘Intermediate Game Art and Animation’, ‘Computer Architecture’, ‘Operating Systems’, ‘Machine Learning’, and ‘Real Time Systems’ were chosen by more students and more number of rules have these courses in their RHS part.

The edges represent the association each node has with other nodes, i.e. the association of each of the course with other course. The smaller the node size, the less frequently opted that course is.



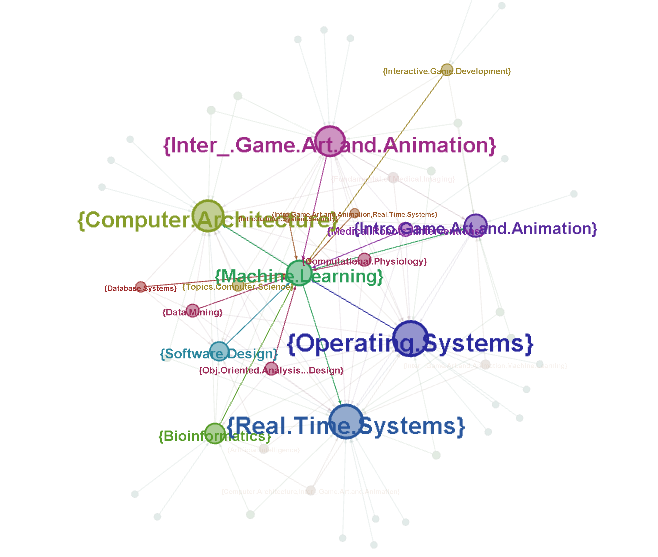
To facilitate representation, we dimmed the unnecessary informations and trying focus on what we want.

For example when focusing on the course ‘Data Mining’, the graph is as follows:



The graph states that ‘Data Mining’ has strong associations with Computer Architecture, Database Systems, Machine Learning, Operating Systems, Software Design and Real Time Systems; meaning if a student took any of these courses then that student also took Data Mining before graduating.

Similarly, when focusing on ‘Machine Learning’ the graph is as follows:



The graph states that when a student takes any of the courses linked to Machine Learning, that student also takes Machine Learning before graduating.

We can also generate rules corresponding to a particular course. For example, we can generate rules that have ‘Data Mining’ as RHS. This will help us understand when a student will take Data Mining and also help us estimate the number of students who will take Data Mining in the next semester depending upon the courses they took in their previous semesters.

#APRIORI FOR DATA-MINING

DataMining <- apriori(Course\_Data\_table, parameter=list(supp=0.1, conf=0.5),

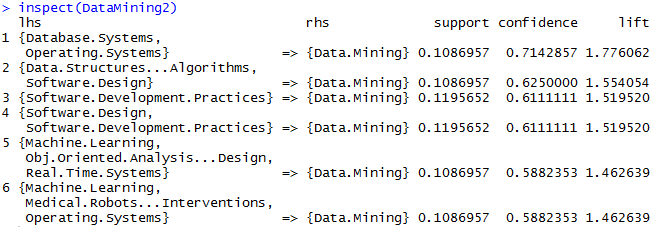
appearance=list(rhs="Data.Mining",

default="lhs"))

DataMining2 <- head(sort(DataMining, by="lift")) # We are sorting rules by ‘lift’

inspect(DataMining2)

The output is as follows:



The output above states that a student will take Data Mining if he already took the courses in the LHS.

Rules for each of the courses can be generated as above.

Finding the most famous courses in the Department:

# ECLAT

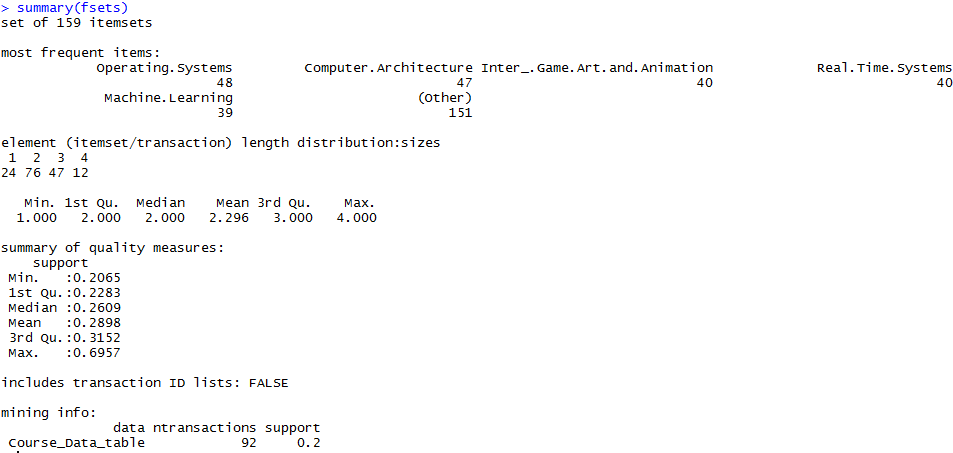
# We took support as 0.2 and maxlen as 15

fsets <- eclat(Course\_Data\_table, parameter<-list(support = 0.2, maxlen=15))

singleItems <- fsets[size(items(fsets)) == 1]

summary(fsets) # Gives the summary of frequently taken courses

The output is as follows:



The output states that the most famous courses are:

* Operating Systems
* Computer Architecture
* Intermediate Game Art and Animation
* Real Time Systems
* Machine Learning

1. On Instructor data

# How the dataset was imported is shown in section II of this report

# We are now using that dataset to find out the Instructors who taught more number of students in the past three years

# We used ECLAT for this purpose

fsets <- eclat(Instr\_Data\_table, parameter<-list(support = 0.2, maxlen=15))

singleItems <- fsets[size(items(fsets)) == 1]

summary(fsets)

The output is as follows:

The output states that the Instructors who taught more number of students in the past three years (Spring 2010 – Fall 2012) are:

* # Not permitted to disclose

**IV. SUMMARY**

We are using Apriori and Eclat algorithms to analyze the association rules in ‘UH CS graduate students’ courses dataset’. Jackie Baum from the Department helped us collect the dataset and this is the first time this dataset being used.

We found the most popular classes among students and also found the connection between several core courses and their extended classes. These rules generated help us predict which courses will be taken by majority of students in the next semester depending upon the courses chosen in the current semester. Also, they help the student decide which courses they can take depending upon the courses they already completed. Also, the pattern a pattern among the courses can help us understand the student interests in an efficient way.

Graph representation is a good way to visualize multiple rules, and the combination of Arule+ arulesViz+Gephi would be a powerful way to analyze and represent the rules. Especially with Gephi generating interactive graphs which help us understand the rules and relationship among courses in a very efficient way.

**V. REFERENCES**

1. <http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_Apriori_Algorithm>
2. <http://www.r-bloggers.com/association-rule-learning-and-the-apriori-algorithm/>
3. <http://cran.r-project.org/web/packages/arules/arules.pdf>
4. <http://en.wikipedia.org/wiki/Gephi>
5. [https://gephi.org](https://gephi.org/)

**APPENDIX:**

**APPENDIX – A**

Association Analysis:

Association analysis, according to the definition in ‘Introduction to data mining(Pang-Ning-Tan, etc.)’, is useful for discovering interesting relationships hidden in large datasets.

Following the original definition by Agrawal et al. the problem of association rule mining in our program is defined as :

Let *I = {i1,i2,…,in}* be a set of n binary attributes called items.

Let *D = {t1,t2,…,tn}* be a set of transactions.

Each transaction in D has a unique transaction ID and contains a subset of the items in I.

A rule is defined as the form X=>Y, where X, Y belongs to I and X AND Y = empty set.

The sets of items (for short item-sets) X and Y are called antecedent (left-hand-side or LHS) and consequent (right-hand-side or RHS) of the rule respectively. [Wikipedia]

To illustrate the concepts, we use a toy example from our dataset. For example, the set of items is I= {machine learning, computer vision, data mining, operating system}, a rule generated could be {machine learning, computer vision} => {operating system}, which means if a student takes machine learning and computer vision, s/he will take operating system.

Firstly, we collected the data from our Program Coordinator Jackie Baum. She passed us an MS excel file with all of the information we need. We preprocessed it in Excel and saved the output as a \*.csv file which could be read by R Studio. After the file is imported into the R environment, we converted them to transactions. Apriori and Eclat algorithms from the *‘arules’* package of ‘R’ toolbox are run on the imported datasets in-order to find out interesting rules and frequent item-sets. The parameters of the algorithm were assigned empirically and the results were sorted based on ‘Lift’ and ‘confidence’. Several graphs were presented at last to help visualize the rules using *‘arulesviz’* package of ‘R’ and at last we came up with some interesting findings.

The most interesting parts of association rule analysis algorithms are about rule generations. Different algorithms use different ways to decrease the number of item sets based on heuristic information and so that cut down the time and cost of going through the redundant rules. We first illustrate the characteristics of the two algorithms used by us and then show the code we run in R script along with the output.

**APPENDIX – B**

Apriori Algorithm:

Apriori is a classic algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. It is based on an assumption that a subset of a frequent itemset must also be a frequent item set. The Apriori algorithm employs level-wise search for frequent itemsets. The algorithm uses the frequent itemsets to generate association rules.

Usage: apriori(data, parameter = NULL, appearance = NULL, control = NULL)

Arguments:

data - object of class transactions or any data structure which can be coerced into

transactions - (e.g., a binary matrix or data.frame)

parameter - object of class APparameter or named list. The default behavior is to mine rules

with support 0.1, conﬁdence 0.8, and maxlen 10.

appearance - object of class APappearance or named list. With this argument item appearance can be restricted. By default all items can appear unrestricted

control - object of class APcontrol or named list. Controls the performance of the mining

algorithm (item sorting, etc.)

Value: Returns an object of class rules or itemsets.

Eclat Algorithm:

This algorithm is also used to perform item set mining. It uses Tid set intersection that is transaction id intersection to compute the support of a candidate item set for avoiding the generation of subsets that does not exist in the prefix tree. For each item store a list of transaction id. It uses vertical data layout. This algorithm scans the data base only once and creates the vertical data base, which identifies each item in the list of transactions that supports the items.

Description:

We can mine frequent itemsets with the Eclat algorithm. This algorithm uses simple intersection operations for equivalence class clustering along with bottom-up lattice traversal.

Usage: eclat(data, parameter = NULL, control = NULL)

Arguments:

data - object of class transactions or any data structure which can be coerced into

transactions - (e.g., binary matrix, data.frame).

parameter - object of class ECparameter or named list (default values are: support 0.1 and

maxlen 5)

control - object of class ECcontrol or named list for algorithmic controls.

Value: Returns an object of class itemsets.

**APPENDIX –C**

Interestingness measure:

We use the following three interesting measurement for rule sorting and selection.

**support (*XY*) = support (*XY*)**

**confidence (*XY)* = support (*XY*)/support (*X*)**

**lift (X ** Y)=conf(X ** Y )/support(Y)**

The support of an itemset is defined as the proportion of transactions in the data set which contain the itemset.

rules <- apriori(Course\_Data\_table, parameter=list(support=0.2, confidence=0.5, minlen=2, target = "rules"))

inspect(rules[1:20])

rules.sorted <- sort(rules, by="lift")

inspect(rules.sorted)

The lowest support and confidence were assigned as 0.5 and 2. Hope to show 20 rules in console window. The minLen seted as 2, which means there should be 2 items included in one rule equation. We sort the rules based on ‘lift’, which is a better criteria to describe the correlation between rules than support or confidence.

Remove redundant rules:

We found that there is some redundant rules such as X=>Y and Y=>X both appears in rule mining. So we only keep the upper triangular matrix of the whole relationship matrix between transactions.

# find redundant rules

subset.matrix <- is.subset(rules.sorted, rules.sorted)

subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA

redundant <- colSums(subset.matrix, na.rm=T) >= 1

which(redundant)

# remove redundant rules

rules.pruned <- rules.sorted[!redundant]

inspect(rules.pruned)

**APPENDIX – D**

Gephi:

Gephi is an open-source network analysis, interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs written in Java on the NetBeans platform. Gephi has been selected for the Google Summer of Code in 2009, 2010, 2011, 2012, and 2013.

Gephi has been used in a number of research projects in the university, journalism and elsewhere, for instance in visualizing the global connectivity of New York Times content and examining Twitter network traffic during social unrest along with more traditional network analysis topics.

The Gephi Consortium is a French non-profit corporation which supports development of future releases of Gephi. Members include SciencesPo, Linkfluence, WebAtlas, and Quid.

Gephi inspired the LinkedIn InMaps and was used for the network visualizations for Truthy.

We used ‘Gephi’ to visualize the association rules that were generated using Apriori as Graphs. Gephi graphs are interactive in a way that they allow us to concentrate on each of the graphs node separately to visualize the edges and other nodes (in our case, Courses) of each of the node.

## APPLICATIONS:

**Exploratory Data Analysis**: intuition-oriented analysis by networks manipulations in real time.

**Link Analysis**: revealing the underlying structures of associations between objects, in particular in scale-free networks.

**Social Network Analysis**: easy creation of social data connectors to map community organizations and small-world networks.

**Biological Network analysis**: representing patterns of biological data.

**Poster creation**: scientific work promotion with hi-quality printable maps

METRICS READY:

**Centrality**: used in sociology to indicate how well a node is connected. Available: degree (power-law), betweenness, closeness.

**And more**: density, path length, diameter, HITS, modularity, clustering coefficient.

TECHNOLOGY:

**Ergonomic interface**: based on NetBeans UI

**High-performance**: built-in 3D rendering engine.

**Native file formats**: GDF (GUESS), GraphML (NodeXL), GML, NET (Pajek), GEXF and more.

**Customizable by plugins**: layouts, metrics, data sources, manipulation tools, rendering presets and more.

R-SCRIPT:

# Importing libraries

library(arulesViz)

library(arules)

# Importing the .csv file as dataset

course\_data<-Grad.Data.CourseNames

length(course\_data)

# Transforming course\_data into Matrix

Course\_Data\_Matrix <- as(course\_data, "matrix")

# Converting Course\_Data\_Matrix to transactions

Course\_Data\_table <- as(Course\_Data\_Matrix, "transactions")

image(Course\_Data\_table)

# Frequency plot

palette()

itemFrequencyPlot(Course\_Data\_table, col=1:8)

# 'arules' - APRIORI

rules <- apriori(Course\_Data\_table, parameter=list(support=0.2, confidence=0.5, minlen=2, target = "rules"))

inspect(rules[1:20])

rules.sorted <- sort(rules, by="lift")

inspect(rules.sorted)

# find redundant rules

subset.matrix <- is.subset(rules.sorted, rules.sorted)

subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA

redundant <- colSums(subset.matrix, na.rm=T) >= 1

which(redundant)

# remove redundant rules

rules.pruned <- rules.sorted[!redundant]

inspect(rules.pruned)

rules.sorted <- sort(rules.pruned, by="confidence")

inspect(rules.sorted)

# Visualizing the rules - 'arulesviz'

plot(rules.pruned, col=1:8)

plot(rules.pruned, method="graph", control=list(type="items"))

plot(rules.pruned, measure=c("support","lift"), shading="confidence");

plot(rules, shading="order", control=list(main ="Two-key plot"));

# Generates a graph of the rules

saveAsGraph(head(sort(rules.pruned, by="lift"), 1000), file="rules.graphml")

#Trimming down the rules

subrules = rules[quality(rules)$confidence > 0.8];

subrules

inspect(subrules)

plot(subrules, method="matrix", measure="lift");

#APRIORI FOR DATA-MINING

DataMining <- apriori(Course\_Data\_table, parameter=list(supp=0.1, conf=0.5),

appearance=list(rhs="Data.Mining",

default="lhs"))

DataMining2 <- head(sort(DataMining, by="lift"))

inspect(DataMining2)

# ECLAT

fsets <- eclat(Course\_Data\_table, parameter<-list(support = 0.2, maxlen=15))

singleItems <- fsets[size(items(fsets)) == 1]

summary(fsets)

# ECLAT # Finding famous instructors

Instr <- Instructors

length(Instr)

# Transforming Instructors into Matrix

Instr\_Matrix <- as(Instr, "matrix")

# Converting Instr\_Matrix to transactions

Instr\_Data\_table <- as(Instr\_Matrix, "transactions")

# Frequency plot

itemFrequencyPlot(Instr\_Data\_table, col=1:8)

fsets <- eclat(Instr\_Data\_table, parameter<-list(support = 0.2, maxlen=15))

singleItems <- fsets[size(items(fsets)) == 1]

summary(fsets)

## Get the col numbers we have support for

singleSupport <- quality(singleItems)$support

names(singleSupport) <- unlist(LIST(items(singleItems),decode=FALSE))

head(singleSupport, n = 5)

itemsetList <- LIST(items(fsets), decode=FALSE)

allConfidence <- quality(fsets)$support / sapply(itemsetList, function(x)

+ max(singleSupport[as.character(x)]))

quality(fsets) <- cbind(quality(fsets), allConfidence)

summary(fsets)

fsetsDataMining<- subset(fsets, subset = items %pin% "Data.Mining")

inspect(sort(fsetsDataMining[size(fsetsDataMining)>1],by="allConfidence"))