



The University of Sydney

**MINING OPEN SOURCE SOFTWARE(OSS)
DATA USING ASSOCIATION RULES NETWORK
TECHNICAL REPORT NUMBER 535**

FEBRUARY 2003

Sanjay Chawla, Bavani Arunasalam and Joseph Davis

ISBN 1 86487 538 0

**School of Information Technologies
University of Sydney NSW 2006**

Mining Open Source Software(OSS) Data using Association Rules Network

Sanjay Chawla¹, Bavani Arunasalam¹, and Joseph Davis¹

Knowledge Management Research Group
School of Information Technologies
University of Sydney, NSW, Australia
{chawla,bavani,jdavis}@it.usyd.edu.au

Abstract. The Open Source Software(OSS) movement has attracted considerable attention in the last few years. In this paper we report our results of *mining* data acquired from SourceForge.net, the largest open source software hosting website. In the process we introduce Association Rules Network(ARN), a (hyper)graphical model to represent a special class of association rules. Using ARNs we discover important relationships between the attributes of successful OSS projects. We verify and validate these relationships using Factor Analysis, a classical statistical technique related to Singular Value Decomposition(SVD).

Keywords: Open Source Software, Association Rule, Networks, Hypergraph clustering, Factor Analysis.

1 Introduction and Motivation

In this paper we detail our exploration through Open Source Software(OSS) data in search for *patterns* with the goal of modeling the nature of OSS development. In the process we will use and modify several standard data mining techniques in conjunction with classical statistical techniques. In particular we will introduce the concept of an Association Rules Network(ARN)¹ which provide insight into the interrelationships between relational data attributes in general and OSS data in particular. The relationships discovered using ARNs will be validated using Factor Analysis, a classical statistical method. Our work will be a step towards validating an often cited maxim in the data mining community: *Data mining is for hypothesis generation and statistics for hypothesis verification* [8, 7].

The OSS movement has attracted considerable attention in recent years, primarily because it is perceived to offer a non-proprietary and socially beneficial model of software development. The source code for this software is freely available and not subject to limitations on possible modifications or distribution. The main idea behind open source is that good software evolves when a dedicated community of programmers and developers can read, redistribute, and modify the source code for a piece of software. Members of this community improve and adapt the software and fix bugs. It is claimed that this rapid, evolutionary process produces better quality software than the traditional hierarchical and closed model [5, 13, 14].

Several researchers have attempted to “explain” the relative success of the open source model of software development. However, many of these endeavours were constrained by ad hoc and premature theorising and the anecdotal or limited case study approach to analysing the OSS development process. This was partly due to the lack of availability of systematic data on OSS projects. The initiation of the SourceForge project in November 1999(itself and OS project) with the goal of hosting OSS projects and gathering data about them offers an exciting opportunity to synthesise and test models about OSS development.

SourceForge.net is currently the world’s largest open source development website with the largest repository of Open Source code and applications available on the Internet. SourceForge.net provides free services to OS developers, including project hosting, version control, bug and issue tracking, project management, backups and archives and communication and collaboration resources. As of March 2002 SourceForge.net had 40,002 registered project and 424,862 registered users.

Our use of data mining methods to analyze OSS data led us to some unique observations about the methods themselves. For example, given a set of association rules R and a specific rule $A \rightarrow B$, where B is a singleton, we were able to recursively build a network of rules which flowed into B . When this network was visualized certain relationships between variables (*items*) stood out which which were not apparent in a flat representation of the rules. This in turn prompted the use of Factor Analysis which

¹ This term has been used before on <http://www.statsoftinc.com>, where it used for a graphical representation for generic association rules. Our terminology is, as we will explain, for representing a subclass of association rules as directed hypergraphs.

helped to validate these relationships. This was almost like “*getting a second opinion*” about patterns that were inferred using association rules.

The main contributions of this paper are as follows. Using several variants of the association rule and clustering paradigm we were able to discover crucial relationships which characterize successful OSS projects. Similar results can be derived using Factor Analysis, a classical, albeit exploratory, statistical method. Both these approaches reinforced the findings of one another. At a more abstract level we were able to represent a set of rules discovered using association rule algorithms as a family of *flow* networks. We call these networks Association Rules Network(ARN). ARNs provide a robust and systematic way to visualize certain special classes of association rules, namely rules whose consequents are singletons.

The rest of the paper is as follows. In Section 2 we formally define the problem of pattern discovery in the context of OSS data. Section 3 is devoted to an overview of related work in OSS analysis as well as relevant data mining approaches. In Section 4 we introduce the concept of ARNs and provide an algorithm to construct them. Results from extensive experimentation on OSS data are the focus of Section 5. We conclude in Section 6 with a summary and directions for future work.

2 Problem Definition and Objectives

Open Source Software(OSS) is emerging as a serious competitor to commercial or closed software. The development of OSS which is largely based on volunteer effort necessarily evolves along a different path as opposed to proprietary software. Is there a way to characterize OSS development and determine why certain OSS application are more successful than others? Formally,

Given OSS data downloaded from the SourceForge.net website.

Find Patterns which characterize the success or failure of an OSS application.

Constraints Success is determined by the number of downloads of the software and pageviews on the webpage related to the application.

Most of the projects hosted on the SourceForge receive very few hits. Infact like much of the traffic on the web, the project downloads are governed by a Zipf like distribution. This has been confirmed in a recent study [9] where it is noted that the software downloads from SourceForge.net follow a Pareto(Zipf like) distribution. This means that most of the downloads are for a few projects and the tail of the distribution is very long. For example, for a one month time period they note that the median of the downloads was 70 but downloads extended upto 600,000. Thus a direct analysis based on regression or classification will be problematic because of the skewness in the distribution of positive and negative samples(with respect to downloads) in the data. We decided to use association rules as a first step with the hope of discovering underlying trends.

3 Association Rules Network

In this section we briefly overview association rules and also introduce Association Rules Network(ARN). An ARN is a (hyper)graphical model to represent certain classes

of rules, namely rules whose consequents(right-hand sides) are singletons. We found that that in the case of OSS data, ARNs provide good insights into the inter-relationships between the attributes of the software projects. But ARNs can be viewed as an abstract and general concept that may be useful in other domains as well.

3.1 Association Rules

Association rules are considered one of the major success stories of data mining research [1, 4]. Association Rules are traditionally described in the framework of market basket analysis. Given a set of items I and a set of transactions T consisting of subsets of I , an Association Rule is a relationship of the form $A \xrightarrow{s,c} B$ where A and B are subsets of I while s and c are the minimum support and confidence of the rule. A is called the *antecedent* and B the *consequent* of the rule. The support $\sigma(A)$ of a subset A of I is defined as the percentage of transactions which contain A and the confidence of a rule $A \rightarrow B$ is $\frac{\sigma(A \cup B)}{\sigma(A)}$. Most algorithms for association rule discovery take advantage of the anti-monotonicity property exhibited by the *support* level: If $A \subset B$ then $\sigma(A) \geq \sigma(B)$.

Our focus is to discover association rules in a more structured and dense relational table. For example suppose we are given a relation $R(A_1, A_2, \dots, A_n)$ where the domain of A_i , $dom(A_i) = \{a_1, \dots, a_{n_i}\}$, is discrete-valued. Then an *item* is an attribute-value pair $\{A_i = a\}$. We are interested in rules of the form $\{A_{m_1} = a_{m_1}, \dots, A_{m_k} = a_{m_k}\} \rightarrow \{A_j = a_j\}$ where $j \notin \{m_1, \dots, m_k\}$

The reason we are interested in these specific rules is because we want to characterize projects based on their download activity. Thus our problem is, in the first order, a supervised learning problem where the download activity serves as the dependent variable(class label). But as we have noted in Section 2, the distribution of download activity follows a Zipf-like power law and a direct application of supervised learning techniques will smooth over the skewness. We get around this problem by recursively constructing a family of interrelated rules. The link between the rules is through their consequent. We begin by fixing an item c and finding all rules whose consequent is c . Then the antecedents of the rules discovered play the role of consequents in the next stage. This way we build a network of association rules. We call this network an Association Rule Network(ARN). The integration of supervised learning and association rules has been the subject of the previous study by Liu, Hsu and Ma [12]. We build upon their work by creating an ARN which flows into an instance of the class label and choosing a different class label in each stage.

Example: Consider a relation $R(A, B, C, D, E, F)$ where the attributes are binary-valued. Assume the following association rules were derived from the relation R using an association rule algorithm

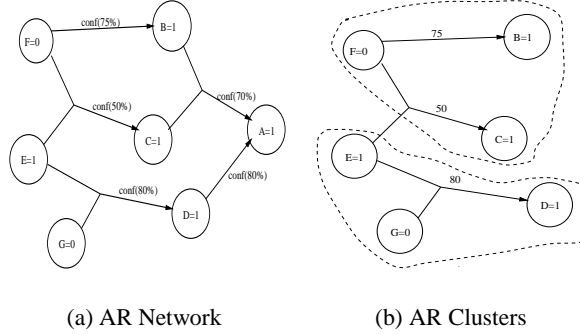


Fig. 1. (a) Association Rules Network(ARN)(b) Items clustered using mincut hypergraph partitioning.

$$\begin{aligned}
 \{B = 1, C = 1\} &\rightarrow \{A = 1\} \\
 \{D = 1\} &\rightarrow \{A = 1\} \\
 \{F = 0, E = 1\} &\rightarrow \{C = 1\} \\
 \{E = 1, G = 0\} &\rightarrow \{D = 1\} \\
 \{A = 1, G = 1\} &\rightarrow \{E = 1\}
 \end{aligned}$$

Suppose we fix the consequent $\{A = 1\}$. Then we can recursively build a network of association rules which flow into $\{A = 1\}$. This is shown in Figure 1(a). Notice the edges of this network are hyperedges. A hyperedge is a generalization of an edge which can span more than two nodes of a graph. As has been noted before [6] a natural way to model itemset transactions is to use hypergraphs. This is because a transaction can consist of more than two items. Also notice that the last rule is not part of the ARN. This is because once we fix the first consequent $\{A = 1\}$ we want to exclude rules in which this consequent appears as an antecedent.

Definition 1. A Hypergraph $G = (V, E)$ is a pair where V is the set of nodes and $E \subset 2^V$. Each element of E is called a hyperedge. A directed hyperedge $e = \{v_1, \dots, v_{k-1}; v_k\}$ is a hyperedge with a distinguished node v_k . A directed Hypergraph is a hypergraph with directed hyperedges.

Definition 2. Given a set of association rules R which satisfy the minimum support and confidence threshold. An association rule network, $ARN(R, z)$, is a weighted directed hypergraph $G = (V \cup z, E)$, where z is a distinguished sink item(node) such that either G is an \emptyset OR

1. Each hyperedge E corresponds to a rule in R whose consequent is a singleton.
2. There is a hyperedge which corresponds to a rule r_o whose consequent is the singleton item z .
3. The distinguished vertex z is reachable from any other vertex in G

4. Any vertex $p \neq z$ is not reachable from z .
5. The weight on the edges correspond to the confidence of the rule that they encapsulate.

Lemma 1. Given a rule set R and a distinguished item z , Algorithm 1 will build an $ARN(R, z)$.

Proof: The algorithm builds the $ARN(R, z)$ using a breadth-first strategy. $Rules.getRules(R, u, z)$ returns all the rules whose consequent is u . $G.addHyperEdge(r, u)$ adds a hyperedge corresponding to a rule r in R . (1), (2) (4) and (5) follow directly from the construction of the graph. We prove (3) as follows: Let p be a vertex in G . If $p = z$ then we are done. Therefore without loss of generality assume $p \neq z$. We will construct a set which consists of all vertices which are reachable from p . Let

$$\begin{aligned} C_0 &= \{p\} \\ C_k &= \{u \in G \mid v \in Rules.getAntecedents(u), v \in C_{k-1}\} \\ C &= \cup_k C_k \end{aligned}$$

Then C is the set of all vertices reachable by p . We claim that C contains z . If not, then by construction of C none of the vertices which are immediately adjacent to z are in C . Following this argument we conclude that there does not exist any w such that $p \in Rules.getAntecedents(w)$. But the only vertex which does not have an antecedent is z . This contradicts $p \neq z$.

Note: Algorithm 1 will not exclude cycles. That that can be guaranteed by enforcing the constraint: *A node which has served as a consequent cannot be an antecedent*. This constraint will destroy the uniqueness of the network. A rule set R can also be mapped to a directed acyclic graph but the rules represented by the edges, will not necessarily satisfy the minimum confidence though they will satisfy the minimum support. In case of a DAG, and because the edges represent conditional probabilities(confidence) association rules may be used to learn special classes of Bayesian networks [10].

3.2 Clustering

An Association Rule Network(ARN) provide information about the hierarchical relationships between the variables(attributes) of the system under investigation. A natural thing to follow-up is to cluster the items that are elements of the ARN. Hypergraph clustering based on min-cut hypergraph partitioning is a well researched topic and has been used for VLSI design. Han et. al. [6] have used it for itemset clustering.

One issue that arises when the clustering problem is being setup is to decide on which similarity measure to use. For example in [6] weighted confidence rule is used: if a hyperedge spans the items $\{A, B, C\}$ then weight of the hypergraph is the average of the confidence of the rules $\{A, B\} \rightarrow \{C\}$, $\{A, C\} \rightarrow \{B\}$ and $\{B, C\} \rightarrow \{A\}$. Another similarity measure than can be used is related to the *lift* of an itemset. Again consider three items $\{A, B, C\}$. Then the *lift* of $\{A, B, C\}$ is $\frac{\sigma(A, B, C)}{\sigma(A)\sigma(B)\sigma(C)}$.

```

Data      : Rules  $R$ , Consequent  $c$ 
Result   : A Directed hypergraph  $G$  representing an AR Network which flows
              into  $c$ 
 $visited[i] = 1$  if  $i$  has been visited as a consequent;
 $u \leftarrow c$ ;
 $s \leftarrow c$ ;
Add  $u$  to queue  $q$ ;
 $visited[u] \leftarrow 1$ ;
repeat
   $RuleSubset \leftarrow Rules.getRules(R, u, s)$  /* Get all rules whose consequent
  is  $u$  but antecedent does not contain  $s$  */;
  for all rules  $r \in RuleSubset$  do
     $G.addHyperEdge(r, u)$  /* directed hyperedge flowing into  $u$  */;
     $a \leftarrow Rules.getAntecedents(u)$ ;
    for all elements  $w \in a$  do
      if  $visited[w] = 0$  then
        Add  $w$  to  $q$ ;
         $visited[w] = 1$ ;
      end
    end
  end
  if  $q$  is empty and  $G$  is singleton then
    return  $G = \emptyset$ ;
  else if  $q$  is empty and  $G$  is not singleton then
    return  $G$ ;
  end
  Delete  $u$  from  $q$ ;
until false;

```

Algorithm 1: Association Rule Network(ARN) constructed from a rule set R and flowing into consequent c using a breadth-first like strategy.

In our case we just use the original confidence as the measure of similarity. We used the hMetis package [11] to cluster the items. The result of clustering items in Figure 1(a) are shown in Figure 1(b). We looked for two clusters and dropped the sink consequent $\{A = 1\}$ before clustering. In the next section we will show that the attributes whose instances(items) end up in the same cluster are also grouped together using Factor Analysis. We argue that our approach is a testament to the inductive power of data mining techniques.

4 Related Work

Association rules and fast algorithms to mine them were introduced by [1]. Our work is more closely aligned with [12] and [6]. Liu, Hsu and Ma [12] have integrated classification with association rules into *class association rules*(CARs). The key operation in their approach finds all ruleitems of the $\langle \text{condset}, y \rangle$ where *condset* is the set of items and *y* is a class-label. We extend this by recursively constructing a network of ruleitems where the antecedents in one stage play the role of consequents in the next stage. Han, Karypis and Kumar [6] have used hypergraphs to represent itemsets. They have also used min-cut hypergraph partitioning to cluster itemsets. We apply their techniques on ARNs which are a special class of directed hypergraphs.

Factor Analysis is a standard technique in statistics and is closely related to Principal Component Analysis. Our treatment follows [8].

Analysis of OSS development has been the subject of lively discussion. Raymond [13] proposed the well-know bazaar metaphor to explain the complex workings and the perceived effectiveness of OSS project groups as opposed to the “cathedral” analogy for traditional software development. It has been observed that most OSS projects start with initial designs and important design decisions made by project leader(s). Torwalds [14] has convincingly argued that the quality of such designs and of initial decisions have had a major effect on the degree of success of the Linux project.

5 Experiments and Analysis

We have carried out extensive experimentation on the OSS data using the techniques described in the previous section. In this section we describe some of the main results. However, we briefly digress to provide a summary of the data extraction, data cleaning and data engineering phases that were carried out [2].

Data Extraction: As mentioned earlier we downloaded OSS data from the SourceForge website, www.sourceforge.net. As of March 2002 the SourceForge website had 40002 projects hosted(including SourceForge!) and 424,862 registered users. Only projects which had attracted a minimum threshold of activity were investigated. The dataset was obtained by a program which crawls through the project summary page for each of these projects and stores the necessary information in a text file. The dataset holds features of each project and there activity information. The following are the facts about the dataset:

Total number of records	24002
Total number of variables	46
Number of categorical variables	15
Number of numeric variables	31

Data Cleaning: After the data was extracted we noticed that many entries related to several attributes were mislabeled. For example the attribute *audience* which captured the end-user for the application was populated by the entries for the *environment* variable. Similarly the *time* attribute was either missing or lacked a consistent format.

Data Engineering: An important transformation we carried out was to convert the numeric value attribute data to ordinal data by using equal-sized binning. The number of bins used was upto four representing missing, low, medium and high values. This was necessary since most off-the-shelf association rule programs accept only categorical data. However as we will show this did not result in a substantial loss of information as similar results were obtained using Factor Analysis which operated on the original numerical values.

After carrying out a rather detailed preliminary analysis of the data we decided to focus on twelve attributes which together indicate project *activity* along different dimensions. These attributes are shown in Table 1.

Attribute Number	Attribute Name
1	Number of Administrators
2	Number of Developers
3	Number of CVS commits
4	Number of Bugs found
5	The percentage of bugs fixed
6	Number of support requests
7	Percentage of support requests completed
8	Number of patches started
9	Number of patches completed
10	Number of public forums
11	Number of mailing lists
12	Number of forum messages

Table 1. Twelve variables were used in our study. The variables were discretized to discover association rules but the original numerical values were retained for factor analysis.

5.1 Experiment 1: Constructing Association Rules Network(ARN)

As described in Section 3 we constructed ARNs using the algorithm described in Algorithm 1: Given a set of rules R and an item(attribute-value pair) z we find all rules in R in which z appears as a consequent. We recursively extract more rules where the antecedents in the previous step now play the role of consequents. This results in a directed hypergraph which we refer to as a ARN. Using a support level of 6% and a confidence level of 50%. the resulting ARN is shown in Figure 2.

Analysis: Figure 2 shows the ARN extracted from the OSS data. The ARN flows into *high pageviews and downloads*. The ARN clearly shows that associaton rules naturally discover attributes which work in concert. For example, the three variables: high activity in public forums, morum messages and mailing list encapsulate communication activity related to the project and have a first-order effect on the success of the software project. This validates the importance of effective communication which has often been cited as a predictor for successful software engineering projects [3].

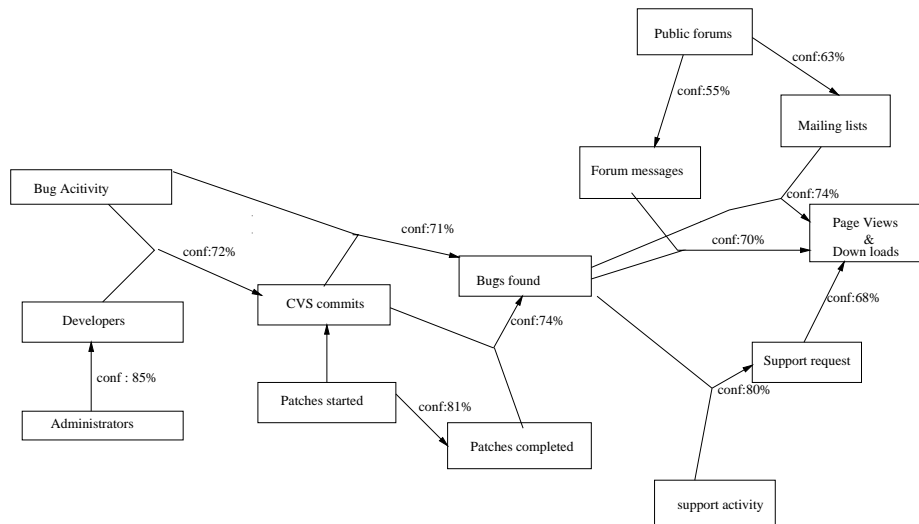


Fig. 2. Association Rule Network(ARN) constructed using Algorithm 1. The ARN is a directed hypergraph. Each node corresponds to the item(attribute=high). The distinguished sink node is *Page Views and Downloads*. Each edge corresponds to an association rule. The minimum support level was 6% and the minimum confidnce was 50%

5.2 Experiment 2: Clustering and Factor Analysis

As described in Section 3.2 we partitioned the ARN into four components using mincut hypergraph partitioning. We also applied factor analysis as described in Section 3.3 on the original numerical valued dataset. The result of clustering are shown in Figure 3. The items naturally fall in clusters which we have labeled Communication, Support, Improvement and Organization/Commitment. The result of factor analysis is shown in Table 2.

Analysis: Factor Analysis is a statistical technique which tries to find latent variables which best describe the data. It is quite similar to Principal Component Analysis(PCA) except that it works on the correlation matrix as opposed to the covariance matrix. The twelve variables in Table 2 whose description is given in Table 1 are represented by four factors C1,C2,C3 and C4. Each of these factors tries to captures the essence of the

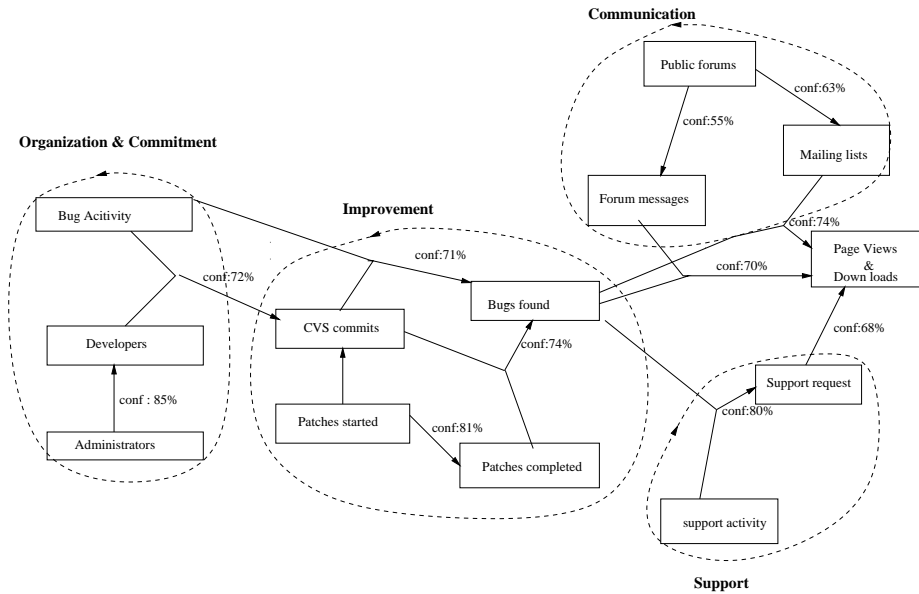


Fig. 3. The four clusters derived using min-cut hypergraph partitioning on the ARN. The four clusters have a natural meaning as indicated in the figure.

original variables. For example the factor C1 captures the effect of variable 9,8,4 and 11. Like in PCA a threshold which can be used for truncation. In our case the threshold was 0.5. Table 2 shows the combined result of clustering on ARN and factor analysis. For example row 1 of the table is read as follows. Variable number 9 which stands for “Number of patches completed”, was mapped to factor C1 and cluster 1(Improvement). The actual number .968 represents the loading of the variable on the factor C1. It seems remarkable that two completely distinct techniques result in very similar results.

6 Summary, Conclusion and Future Work

We have used several data mining techniques to analyze OSS data acquired from SourceForge.net. We have discovered important inter-relationships between variables which characterize successful OSS projects. For example we have discovered that variables can be clustered into four meaningful groups which we have labeled *Communication*, *Support*, *Improvement* and *Organization/Commitment*. In the process we have introduced the concept of an Association Rules Network(ARN). An ARN is a directed hypergraph which can be used to represent and integrate special kinds of association rules. We used ARNs to discover *patterns* in the OSS data and verified the resulting patterns using Factor Analysis, a classical statistical technique. Thus our work is also a step towards validating an often cited maxim in the data mining community: *Data Mining is for hypothesis generation and statistics for hypothesis verification*.

Variable	C1	C2	C3	C4
9	.968(1)			
8	.967(1)			
4	.791(1)			
2		.800(2)		
1		.756(2)		
3		.557(1)		
5		.449(2)		
12			.711(4)	
6			.686(3)	
7			.521(3)	
10				.903(4)
11	.517(4)			.561(4)

Table 2. C1 to C4 represent the factors(latent variables) into which the original 12 variables are mapped. For example the first line should be read as variable 9 which represent ‘number of patches completed’ was mapped to factor C1 using factor analysis and cluster 1 using min-cut hypergraph partitioning. The actual number, 0.968 represents the loading of variable 9 on factor C1. The table shows that all but two variables are mapped identically by two very different techniques.

There are several issues that we plan to investigate in the future including temporal analysis of the OSS data and extending the applicability of Association Rules Network.

References

1. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.
2. R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, (1):5–32, 1999.
3. A. Dutoit and B. Bruegge. Communication metrics for software development. *IEEE Transactions On Software Engineering*, 24(8):615–628, 1998.
4. L. Feng, J. Yu, H. Lu, and J. Han. A template model for multi-dimensional, inter-transactional association rules. *VLDB Journal*, 11(2):153–175, 2002.
5. R.L Glass. The sociology of open source: of cults and cultures. *IEEE Software*, 17(3):104–105, 2000.
6. Eui-Hong Han, George Karypis, Vipin Kumar, and Bamshad Mobasher. Clustering based on association rule hypergraphs. In *Proceedings SIGMOD Workshop Research Issues on Data Mining and Knowledge Discovery(DMKD '97)*, 1997.
7. Han, J., Kamber, M., 2001. *Data Mining, Concepts and Trends*. Morgan Kaufmann.
8. Hand, D., Mannila, H., Smyth, P., 2001. *Principles of Data Mining*. M.I.T Press.
9. F. Hunt and P. Johnson. On the pareto distribution of sourceforge projects. In *Open Source Software Development Workshop, University of Newcastle, UK*, February, 2002.
10. F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.
11. George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Applications in VLSI domain. In *Proceedings of the 34th Conference on Design Automation, Anaheim California, June*, 1997.
12. Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Knowledge Discovery and Data Mining*, pages 80–86, 1998.
13. E.S. Raymond. *The Cathedral and Bazaar: Musings on Open Source and Linux by an Accidental Revolutionary*. O'Reilly, 2001.
14. L. Torwalds. The linux edge. *Communications of the ACM*, 42(4):38–39, 1999.