Data Mining in Maintenance of Electronic Component Libraries

Esa Alhoniemi*, Timo Knuutila*, Mika Johnsson†, Juha Röyhkiö† and Olli S. Nevalainen*
*University of Turku
Department of Information Technology
Lemminkäisenkatu 14 A
FI-20520 Turku, Finland
†Valor Computerized Systems (Finland) Oy
Ruukinkatu 2
FI-20540 Turku, Finland

***Abstract*— In this study[1], adding data of new components to an existing electronic component library in considered. The suggested approach uses a particular data mining algorithm to support interactive input of the data. The basic idea is to compute association rules between the attributes of the existing components in the library. The rules can then be used to ease the input of the attributes of a new component. The scheme is general in the sense that the same approach can be easily used in other similar applications as well. We first introduce the necessary basic concepts of the association rules and then illustrate the application of the suggested approach using a fraction of a real component library.**

I. Introduction

Successful operation of a printed circuit board (PCB) assembly robot requires three things: a numerical control (NC) program, an electronic component library, and the configuration data of the machine. In the assembly of a new product using a PCB robot, generation of a new NC program is usually quite straightforward. The machine configuration data needs to be changed seldom and is therefore often not a problem. A laborious task in the assembly of a new product is the maintenance of the electronic component library, which is considered in this article.

In the library, each component is characterized by dozens or even hundreds of attributes, such as the dimensions of the component, nozzles, vision data, handling speeds, polarity, and feeders. There exist both machine independent and dependent attributes. The machine independent attributes can be directly obtained from some external source like a CAD library or Valor parts library[2]. In the machine dependent data, there are several attributes the values of which depend on the type of a particular machine and some even on a specific machine. The reason for this is that the values of some attributes are may depend e.g. on the physical environment, like the lighting conditions of the machine. Generation of the machine-specific data turns out to be the most laborious task when the assembly of a new product is initiated on a certain machine. Traditionally, experience of the human operators, manual browsing through the specification documents, and testing by the machine using trial and error is required. The novel approach suggested in this paper – which utilizes the information in the existing component libraries – does not eliminate all the manual work, but provides a faster semiautomated procedure for the generation of the data.

Even though the component library data has a very complex logical structure, it is possible to ignore without loss of generality the details concerning the structure of the data. From now on, the data of the component library is seen as a table, the rows of which correspond to the components and the columns to the different component attributes. Each time a previously unused component type is included in the assembly by a machine, its attributes have to be fed into the library which requires a large amount of manual checking. This means adding a new row in the data table, which may amount up to 100 rows.

The main contribution of this study is a novel data mining [1] approach to support a human operator to fill in or check the correctness of the attributes for a new component attributed to some component placement machine. The basic idea is to use the existing component libraries to construct a set of so-called association rules, which describe dependencies between values of different attributes. The rules can then be used either to predict the value of an unknown attribute based on the so far recorded ones, or to detect potentially erroneous user input. This is possible due to the fact that the component attributes are not completely independent of each other, but the library contains much redundant information. The redundancy could be removed by clever data structures and by adding dependencies to the library. However, these kind of solutions do not work due to the highly dynamic nature and complex rules of the component attributes.

The goal of this study is to *demonstrate the use of the suggested data mining approach and to preliminarily evaluate its feasibility using real data*. Determination of the so-called large (or frequent) itemsets (see for example [1, pp. 429–433]) – which is an essential part of the computation of the association rules – is carried out using the well-known data

[1]This work was partially supported by the Academy of Finland, Grant 104795.

[2]The library contains data of about 30–40 millions of components, see `www.valor.com` for more details.

mining algorithm Apriori [2]. In the experiments, a small fraction of a component library with 317 components, each characterized by 174 attributes, was used. When analyzing the data, we observed that a large proportion of the attributes were constant (or almost constant) and some of them had a one-to-one correspondence with some other attribute. Further, the values of many other attributes could be predicted by some other attributes using rules with good support and high confidence. To summarize, in the light of the obtained results, the approach suggested in this paper seems to be promising and deserves to be studied in the future in more depth using full-scale component libraries.

This article has been organized as follows. The content of the component library is described in Section II. In order to keep the study self-contained, the basic ideas of the data mining techniques used are briefly discussed in Section III. Experimental results using the association rule algorithm Apriori are shown in Section IV. Section V contains conclusions and lists some topics for the future work.

II. Component library

A component library is a 2-dimensional data table, the columns of which are component attributes and the rows correspond to the components, correspondingly. In the general case, the scale of an attribute may be nominal, ordinal, interval, or ratio [3, pp. 12–14]. For simplicity, in this work we assume that the attributes are nominal (even though this is not actually true for all attributes), which means that all one can say about two values of an attribute whether they are equal or not. However, in the data mining literature there exist approaches to deal with interval scale attributes as well. In practice, such approaches often discretize the data and use clustering algorithms or some other underlying inference mechanisms for the generation of rules [4]–[8]. Use of such an algorithm is straightforward in our application.

Table I contains a simple synthetic example of a component library which is used through the representation for illustration purposes.

TABLE I

A synthetic component library with five components and five attributes.

component	attribute				
	1	**2**	**3**	**4**	**5**
1	A	A	A	A	A
2	A	A	B	A	B
3	B	A	B	B	C
4	B	A	B	C	D
5	B	A	B	B	E

A. Data transformation

Before the component library can be used in the mining of association rules, the data table needs to be transformed into a set of *transactions* each of which consists of a set of *items*. In our application an item is a specific *value*, say A, of an *attribute* i ($i = 1, \ldots, 5$). Such an item, denoted by A_i, can thus be seen as an attribute-value pair. Hence, a transaction in this context stands for the set of all the attribute-value pairs of a component. Such a coding is required, because a single value (like A) may occur as a value of more than one attribute, but the two items (A_j and A_i, $j \neq i$) are different and they must thus be distinguished from each other in the coding.

Also note that the data mining algorithms used in this study allow the number of items to vary in the transactions (i.e., the rows of the table may have a different number of columns), but every row of our data contains a constant number of columns. Table II shows the data of the Table I converted into five transactions, each corresponding to a component of the library.

TABLE II

Data of Table I converted into transactions.

transaction	items				
1	A_1	A_2	A_3	A_4	A_5
2	A_1	A_2	B_3	A_4	B_5
3	B_1	A_2	B_3	B_4	C_5
4	B_1	A_2	B_3	C_4	D_5
5	B_1	A_2	B_3	B_4	E_5

B. Removal of certain attributes

Before the computation of the association rules, it is sensible to make computation of the association rules lighter by removing some attributes from the database. Roughly speaking, such attributes include the ones which would either be part of every rule or would never be part of any rule.

- Attributes which have a *constant value* for every component may be removed. The best prediction for such an attribute is naturally the constant value which is independent of the values of any other attributes and thus needs not to be predicted.
- Attributes which have *different values for (almost) every component* may be removed, because computation of the association rules requires that two attributes have multiple instances of one value. In other words, no rules for attributes with a large number of different values can ever be found.
- Attributes which have a *one-to-one dependency* with each other can all be removed except for one such an attribute. A value of an attribute with one-to-one dependency with some other attribute can easily be predicted once the dependency is known.

One possible heuristic criterion for removal of the attributes of the first two types in the list above is the use of *entropy*.[3] The entropy is close to zero for attributes with almost constant values whereas for attributes with a large number of different values it is high. This suggests that there should be a lower as well as an upper limit for the entropy of the attributes which are considered further.

[3]Use of entropy is only one possible heuristics among many that can be used for this purpose.

The entropy of the ith attribute is computed using the formula

$$H_i = -\sum_{j=1}^{M_i} P_j \log P_j, \quad (1)$$

where M_i is the number of values of attribute i, and P_j is the probability of the jth value to be present in the data; it can be estimated from the database as the inverse of the number of occurrences of the jth value.

For example, consider the first column (attribute) of Table II. It contains two different values (A_1 and B_1), and hence $M_1 = 2$. The probability estimate of the first value is $P_1 = P(A_1) = 2/5 = 0.4$, and the probability estimate of the second value is $P_2 = P(B_1) = 3/5 = 0.6$. Hence, the entropy of that attribute is $H_1 \approx 0.673$. Correspondingly, the entropies of the other three attributes are $0, 0.500, 1.055$, and 1.609. The maximum entropy $H \approx 1.609$ is obtained when all the values are different as is the case for attribute 5.

Hence, it might be sensible to remove the second and the fifth attribute from all the transactions in the case of the synthetic data.

TABLE III
DATA OF TABLE II WITH TWO ATTRIBUTES REMOVED.

transaction	items		
1	A_1	A_3	A_4
2	A_1	B_3	A_4
3	B_1	B_3	B_4
4	B_1	B_3	C_4
5	B_1	B_3	B_4

The one-to-one dependencies are simply computed by passing through the whole database and checking whether certain items always occur in pairs or not. In the example data there are no such dependencies. However, if the fourth attribute of the component 4 was B_4, attributes 1 and 4 would have a one-to-one dependency.

III. MINING OF ASSOCIATION RULES

In this section, we recall the basic terminology of the mining of association rules. A reader who is familiar with the subject may skip to Section IV, where the application itself is described.

Data mining is defined as "analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner" [1, p. 1]. Here, the data consists of the attribute values of components which are stored in a component library. The relationships to be found are dependencies between different combinations of the attribute values. These dependencies are called *association rules*. Using the synthetic data of Table III, one possible association rule would be $B_1 \wedge B_3 \Rightarrow B_4$, which states that if the value of the first attribute is B_1 and the value of the third attribute is B_3, then the value of the fourth attribute is B_4. In order to construct this kind of rules, we first briefly consider two essential concepts related to the association rules: support and confidence.

Support. Support of an itemset X indicates the number of transactions which contain all the items in X. For example, support of itemset $\{B_1, B_3, B_4\}$, denoted by $support(B_1B_3B_4)$, is 2. It is clear from the definition, that the support of a subset of an itemset is always greater or equal to support of the original set. For example $support(B_1B_3) = 3 \geq support(B_1B_3B_4)$. In the computation of the association rules, itemsets with support greater than or equal to some predefined limit are computed. These itemsets are called *large itemsets*. If the limit is set to 2, the large itemsets of our example are $\{B_1, B_3\}$, $\{B_1, B_3, B_4\}$ and $\{A_1, A_4\}$. The supports of these large sets are 3, 2, and 2, correspondingly.

Confidence. Confidence of a rule indicates how often the rule holds. It is defined as the ratio between the support of all items in the rule and support of all items in the left-hand side of the rule. For example, the confidence of our example rule $B_1 \wedge B_3 \Rightarrow B_4$ is given by

$$conf = \frac{support(B_1B_3B_4)}{support(B_1B_3)} = \frac{2}{3} \approx 67\%.$$

A. Computation of association rules

The computation of the association rules consists of two separate subtasks. They are the computation of the large itemsets, and finding the rules using the large itemsets. Before the computation, two parameters have to be fixed: *the minimal support (minsup)* and *the minimal confidence (minconf)* of the rules. The number of rules strongly depends on these two parameters.

Large itemsets. As stated earlier, a large itemset is an itemset with support greater than or equal to a given minimum support. In this work, the large itemsets are computed using the well-known Apriori algorithm [2], which is described below in Algorithm 1. A large itemset with k items (k-itemset) is denoted by L_k. A potentially large (candidate) itemset with k items is C_k, respectively. The data collection (component library) is denoted by D.

Algorithm 1 Algorithm Apriori.

```
1:  L_1= {large 1-itemsets };
2:  for (k = 2; L_{k-1} ≠ 0; k++) do
3:     C_k = apriori_gen(L_{k-1});
4:     for all candidates c ∈ C_k do
5:        c.count = 0;
6:     end for
7:     for all transactions t ∈ D do
8:        C_t = {C_k | C_k ⊂ t};
9:        for all candidates c ∈ C_t do
10:          c.count++;
11:       end for
12:       L_k = {c ∈ C_k | c.count ≥ minsup};
13:    end for
14: end for
15: return ⋃_k L_k;
```

Set L_1 contains the large itemsets of one attribute. These can be trivially found by starting with the first possible value of the first attribute and by counting the number of rows in which it appears. If this number is greater than *minsup* the value is accepted and included in L_1. Then the same is repeated for the second possible value of the first attribute and so on. This counting operation is repeated for all the attributes.

The Apriori algorithm uses function `apriori_gen` which takes all the large $(k-1)$-itemsets as argument and computes out of them the potentially large itemsets with k items. The function has two steps: *join* and *prune*. The join step (Algorithm 2) first generates a set of candidate k-itemsets.

Algorithm 2 Join step of `apriori_gen`.

1: $C_k = \emptyset$;
2: **for all** large $(k-1)$-itemsets $p,\ q \in L_{k-1}$ **do**
3: $C_k = C_k \bigcup \{p.\text{item}_1, p.\text{item}_2, \ldots, p.\text{item}_{k-1}, q.\text{item}_{k-1} \mid p \neq q, p.\text{item}_{k-1} < q.\text{item}_{k-1}\}$;
4: **end for**
5: return C_k;

The prune step (Algorithm 3) removes some of the candidate itemsets. The removal is based on the monotonicity property of large sets, which states that a k-itemset cannot be large if all of its $(k-1)$-subsets are not large.

Algorithm 3 Prune step of apriori_gen.

1: **for all** itemsets $c \in C_k$ **do**
2: **for all** $(k-1)$-subsets s of c **do**
3: **if** $(s \notin L_{k-1})$ **then**
4: $C_k = C_k \setminus c$;
5: **end if**
6: **end for**
7: **end for**
8: return C_k;

To summarize, the join step produces the candidate itemsets, that is, the set of potentially large k-itemsets using the known large $(k-1)$-itemsets. The prune step removes some of them based on the monotonicity of large sets without counting any itemsets from the data. For the remaining candidates, the supports of the remaining candidates need to be computed from the data in order to determine the actual large k-itemsets.

Association rules. When the large sets have been computed using the Apriori algorithm, they are used in the computation of rules. From each large set, all the possible rules are formed so that all possible variable combinations are present on the left-hand and right-hand side of the rule. The rules which have smaller confidence than the predefined minimum (*minconf*) are deleted whereas all the other rules are retained, see Algorithm 4 for pseudo code.

Algorithm 4 describes generation of the rules in the general case. In this work we are looking for rules with a single item on the right hand side of the rule and ignoring all the others, because we are interested in prediction of one attribute at a time, only. This restriction is made in order to keep the update operations simple enough to be managed by a human operator.

Algorithm 4 Procedure `genrules`.

for all large itemsets $l_k, k \leq 2$ **do**
$A = \{(m-1)\text{-itemsets } a_{m-1} | a_{m-1} \subset a_m\}$;
for all $a_{m-1} \in A$ **do**
$conf = support(l_k)/support(a_{m-1})$;
if $(conf \geq minconf)$ **then**
Output the rule $a_{m-1} \Rightarrow (l_k - a_{m-1})$ with confidence = $conf$ and $support = support(l_k)$;
end if
if $(m-1 > 1)$ **then**
`genrules`(l_k, a_{m-1})
end if
end for
end for

B. An illustration on the use of the suggested scheme

Before we consider the experimental results, let us first look at a concrete illustration to see how to use the predictive scheme using another synthetic example shown in Figure 1. It shows how adding 8 attributes of a component to a component library can be supported by the scheme suggested in this article.

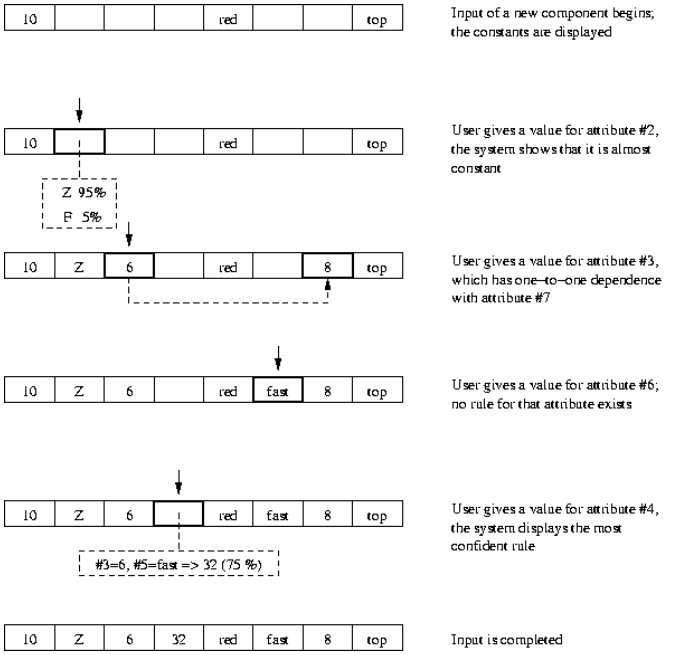

Fig. 1. A fictional example of typing in attributes for one component. If there exist some recommended values for some attributes, they can be also shown to the user who then either accepts or rejects them.

IV. Experiments

A fraction of a real-world component library with 317 components and 174 attributes was used in this experiment. In a full-scale library the number of components is naturally

much larger, but the number of attributes is realistic in our example.

A. Preprocessing of the data

The data table was transformed into transactions as described above and the entropy was computed for every attribute using Equation 1. The attributes, entropy of which was below 10% or above 80% of maximum entropy, were discarded. The entropies of the 174 attributes are shown in Figure 2. The two dashed lines depict the lower and the upper entropy bounds for the attributes which were used in the computation of the association rules. This action decreased the number of attributes to 48. Next, among the remaining attributes the ones with one-to-one correspondence with some other attribute were to be removed. However, such attributes were not found in the remaining data. (Originally there were 37 such attributes in the data, but they were already discarded based on their entropies.)

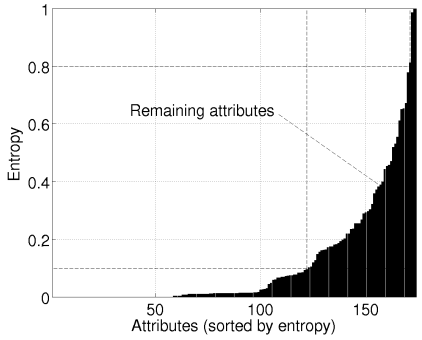

Fig. 2. The attributes of the sample data set ordered in an ascending order according to the entropy.

B. Computation of the rules

Large itemsets. After the preprocessing, there were 48 attributes left which were used to compute the association rules. These attributes were used to compute the large sets for different supports (159, 174, 190, 206, 222, 238, 254, 269, 285), which correspond to 50, 55, ..., 90 % proportion of 317, the total number of components. The number of resulting large sets for each support is shown in Figure 3. Also, the size of the largest itemset is shown for each support level. For example, there are 16 large itemsets with minimum support of 206 (65 % of maximum). This means that there are 16 different sets of attribute-value pairs such that the pairs of every set are present at least for 206 components, and the size (i.e., the number of pairs) of the largest set is 4.

As can be observed, the number of large sets grows rapidly as the minimum support decreases. Also note that if the minimum support is set to a too small value, the computation of large sets may become computationally infeasible.

Association rules. Next, rules with only one item on the right-hand side of the rule were computed for each large itemset using minimum confidences of 50, 55, ..., 95 %. The number of obtained rules is shown for each confidence

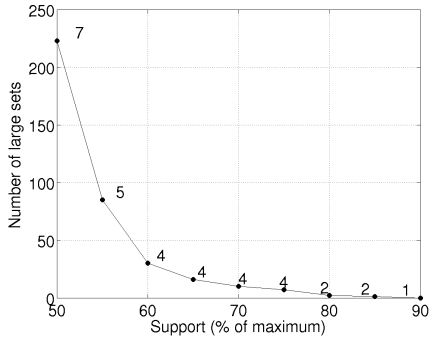

Fig. 3. The number of large sets as a function of minimum support. The number above every point denotes the number of items (i.e., attribute-value pairs) in the largest itemset for the corresponding support.

level and support in Figure 4. The number of rules for each support level is naturally greater than the number of large sets, because it is possible to obtain multiple rules from the items of a large set. For example, if a large itemset is $\{B_1, B_3, B_4\}$, it is possible to obtain nine rules: $B_1 \Rightarrow B_3$, $B_1 \Rightarrow B_4$, $B_3 \Rightarrow B_1$, $B_3 \Rightarrow B_4$, $B_4 \Rightarrow B_1$, $B_4 \Rightarrow B_3$, $B_1 \Rightarrow B_3B_4$, $B_3 \Rightarrow B_1B_4$, and $B_4 \Rightarrow B_1B_3$.

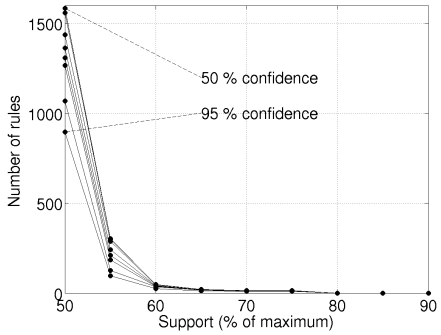

Fig. 4. The number of rules as a function of the support for different confidence levels 50, 55, ..., 95% (curves from top to bottom).

C. Characterization of the obtained rules

The illustration of the number of rules is only one characterization of the obtained rules. An important quantity is also the number of rules with different predictions, that is, different items on the right-hand side of the rule. The number of the different predictions for different support and confidence levels is shown in Figure 5.

Finally, in Figure 6 there are some of the 302 rules which were obtained when the minimum support was set to 55 % of the maximum and minimum confidence to 50 %.

V. Conclusions and Future Work

In this article, we presented a new concept for the maintenance of large electronic component libraries or any other similar database which is based on mining association rules from an existing database. We also demonstrated the use of such an approach and showed that at least the small randomly

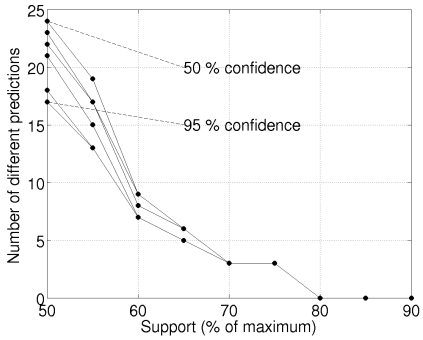

Fig. 5. The number of different predictions (right-hand side items in the rules) as a function of support and confidence.

#167=0 ⇒ #33=Normal (57 %, 100 %)
#170=0 ⇒ #31=0.00 (56 %, 100 %)
#40=NORMAL CHK ⇒ #36=1 (56 %, 77 %)
#38=20 AND #39=60 ⇒ #80=0 (56 %, 89 %)
#59=1 ⇒ #40=NORMAL CHK (56 %, 79 %)
#59=1 AND #104=0 AND #163=100 ⇒ #58=1 (56 %, 99 %)
#58=1 AND #59=1 AND #104=0 ⇒ #163=100 (56 %, 88 %)
#104=0 ⇒ #47=5 (58 %, 65 %)
#38=20 ⇒ #39=60 (63 %, 96 %)

Fig. 6. A small fraction of the obtained rules, when minimum support was 55 % of the maximum and the minimum confidence was 50 %, correspondingly. The numbers in the parentheses after each rule indicate the support (as percentace of the maximum support) and the confidence of the rule.

chosen fraction of the commercial component library did contain enough redundancy to form reliable rules with good support and confidence. In the light of the results obtained it seems that the association rule mining approach is capable of finding useful dependencies in this data. Several issues to be considered in the further research are listed below.

- The tests should be carried out using the entire database to fully determine the usability of the approach. This requires not only similar tests as described above, but also development of a software prototype for the maintenance of the library to verify that it is actually possible for a human to fully benefit from the additional information provided by the rules.
- In this application, data is in a table form and the rules with only one item in the right-hand side of the rule are of interest. Under these assumptions, it is possible to make simplifications in the used algorithms to lighten the computational burden of the computation of the rules. These were not required in the experiments carried out in this study, but this is necessary for full-scale real-world applications.
- It is very common that a large database contains entries which are significantly different from the majority of the data. Such entries, outliers, in this particular application are components which have an unusual combination of attribute values. This may be due to the fact that there has occurred a typing error during the input of the attributes – or the component may actually be somehow different from the others. Identification and removal of outliers would lighten computation of the rules as well.
- In this study, the association rules were computed using a batch run over the whole library. In reality, components are gradually added into the component libraries. If an incremental version of the rule mining algorithm existed, it would naturally dramatically reduce the amount of required computational resources when the rules are updated. This is especially true for the item frequency counters.
- In the computation of the association rules, it was assumed that all the attributes were categorical. In reality, in our data a small fraction of the attributes were numerical. When large databases with numerical attributes are considered, some strategy to deal with such an attributes must be adopted, see [4]–[8].
- In addition to Apriori [2] algorithm used in this work, there exist many other algorithms for mining association rules, see for example [9], [10]. One important issue in the future work is also to consider which algorithm performs best for our data.

Acknowledgments

We wish to thank Dr. Bart Goethals, Basic Research Unit at Helsinki University, Finland for the software[4] used in the experiments.

REFERENCES

[1] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*. MIT Press, 2001.
[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference on Very Large Databases*, 1994.
[3] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice-Hall, 1988.
[4] Y. Aumann and Y. Lindell, "A statistical theory for quantitative association rules," in *Proceedings of the 5th ACM SIGKDD International Conference on Data Mining (KDD '99)*. ACM Press, 1999, pp. 261–270.
[5] S. Brin, R. Ratogi, and K. Shim, "Mining optimized gain rules for numeric attributes," in *Proceedings of the 5th ACM SIGKDD International Conference on Data Mining (KDD '99)*. ACM Press, 1999, pp. 135–144.
[6] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Mining optimized association rules for numeric attributes," in *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database and Knowledgebase Systems (PODS '96)*. ACM Press, 1996, pp. 182–191.
[7] R. J. Miller and Y. Yang, "Association rules over interval data," in *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*. ACM Press, 1997, pp. 452–461.
[8] R. Srikant and R. Agrawal, "Mining quantitative association rules in large relational tables," in *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD '96)*. ACM Press, 1996, pp. 1–12.
[9] J. Hipp, U. Güntzer, and G. Nakhaeizadeh, "Algorithms for association rule mining – a general survey and comparison," *SIGKDD Explorations Newsletter*, vol. 2, no. 1, pp. 58–64, 2000.
[10] G. I. Webb, "Efficient search for association rules," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, 2000, pp. 99–107.

[4] `http://www.cs.helsinki.fi/u/goethals/software/`