Combined association rules for dealing with missing values

Jau-Ji Shen

Department of Management Information Systems, National Chung Hsing University, Taichung 402, Taiwan, R.O.C.

Chin-Chen Chang

Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan, R.O.C.

Yu-Chiang Li

Department of Computer Science and Information Engineering, National Chung Cheng University, ChiaYi 62102, Taiwan, R.O.C.

Correspondence to: Chin-Chen Chang, Department of Information Engineering and Computer Science, Feng Chia University, 100 Wenhwa, Seatwen, Taichung 40724, Taiwan, R.O.C. E-mail: ccc@cs.ccu.edu.tw

Abstract

With the rapid increase in the use of databases, the problem of missing values inevitably arises. The techniques developed to effectively recover these missing values should be highly precise in order to estimate the missing values completely. The mining of association rules can effectively establish the relationship among items in databases. Therefore, discovered association rules are usually applied to recover the missing values in databases. This study presents a Fast Recycle Combined Association Rules (FRCAR) method to fill in the missing values. FRCAR applies a technique to recycle sub-frequent itemsets and bit-arrays to discover more association rules than the Missing Value Completion (MVC) approach.

The experimental result demonstrates that FRCAR results in a higher recovery rate and higher recovery accuracy for missing values.

Keywords: Association rule; data mining; missing value; relational database

1. Introduction

Knowledge management (KM) is one of the most important issues in computer science and is a key to obtaining competitive advantages in the modern enterprises. Successful KM requires useful tools to extracting valuable information from large databases [1-3]. The techniques developed for data mining or Knowledge Discovery in Databases (KDD) attempt to mine hidden and useful information from large databases [4]. Typically, an enterprise collects and stores important business information, resulting in a surprising growth of data. Although current computer technologies can handle massive amounts of data, the rapid growth of databases causes some attribute values to be missed or causes inconsistencies in the data gathering process. Before data analysis begins, the data cleaning step deals with errors and inconsistencies from raw data to improve the quality of the following discovered information. Therefore, the problem of recovering missing values has become a top priority and has played an important role in the data mining field [5-7].

One simple method of dealing with missing values is to delete all tuples with missing values. The resulting truncated databases often include too little data to analyze effectively. As an alternative, users can apply some simple statistical methods such as using mean or median [4, 8] to predict missing values. However, the predicted values, which are still inaccurate, become noise and influence the quality of the information. Consequently, to effectively deal with missing values, several researches have been proposed [4, 9-15].

Agrawal, et al. has proposed an efficient algorithm to discover association rules from large databases for data mining [16, 17]. An association rule reveals relationships among items. For example, an item, *A*, which appears in a transaction, implies that another item, *B*, has a high possibility of appearing in the same transaction. Association rules are widely used to find associative patterns among products in an electronic commerce environment [7]. Therefore, in dealing with missing values, Ragel and Cremilleux have presented the Robust Association Rule (RAR) approach based on the association rules method to partially discard the missing values in relational databases [14]. Ragel and Cremilleux have also proposed the Missing Value Completion (MVC) approach, which is based on RAR, to resolve value completion issues [15]. MVC also supports recovering a single tuple with multiple missing values. However, with regard to the effectiveness of value completion techniques [4, 13, 15], including MVC, they emphasize the guessing precision (the number of right guesses/the number of guesses) rather than considering the recovery rate (the number of guesses/the number of missing values) and recovery precision (the number of right guesses/the number of missing values) of the missing values.

Therefore, this study focuses on a method that considers the recovery accuracy and the recovery rate of inferring and recovering multiple missing values to effectively improve prediction quality.

The MVC method applies association rules to fill in missing values and involves a trade-off between accuracy and the recovery rate of the missing values. In other words, to achieve higher accuracy, the recovery rate must be decreased, and vice versa, because the recovery accuracy of MVC depends on the minimum confidence threshold of the mining process. A high threshold value of the mining process results in a lower number of association rules being generated. Therefore, in many situations, no association rule can be used to recover the missing values. To solve the problem, this study presents a Fast Recycle Combined Association Rules (FRCAR) method to discover extra combined association rules from recyclable itemsets. Moreover, FRCAR uses bit-arrays and simple Boolean AND/OR operations to efficiently discover association rules. The whole process requires scanning an entire relational database only once without establishing a candidate trie structure (a hash tree structure) [17].

For the same minimum thresholds, FRCAR generates more association rules to recover the missing values than MVC does. Thus, the proposed method can enhance the MVC method to achieve a higher recovery rate and higher accuracy.

The rest of this paper is organized as follows: Section 2 introduces some related researches including the Boolean association rules, the RAR, and the MVC methods. Section 3 proposes a novel rules generation method to obtain more association rules than MVC does. Section 4 proposes an array-based method to efficiently identify rules by bit-wise AND/OR operations. Section 5 provides experiment results and evaluates the performance of FRCAR. Finally, conclusions are drawn in Section 6.

2. Related Work

2.1. Association rules

Agrawal, et al. has proposed and defined the mining of association rules [17]. Each item has a binary attribute that presents whether an item has appeared in a transaction. These association rules are also called Boolean association rules [4]. The formal statement is as follows [16, 17]: Let $I = \{P_1, P_2, \ldots, P_n\}$ be a set of n distinct literals, called items. Let DB be a transaction database, where each transaction $T_i \in I$ is a set of items. A k-itemset, X, is the set of items with length k, containing in a transaction. An association rule, denoted as $X \Rightarrow Y$, has two attributes, *support* and *confidence*, where $X \cap Y = \phi$. The value of the two attributes of each rule must be no less than the minimum support (*minSup*) and the minimum confidence (*minConf*) values, respectively. The support value, s%, of the rule is the percentage of transactions containing $X \cup Y$ in the transaction database, denoted as $Sup(X \Rightarrow Y) = |db_{X \cup Y}|/|DB|$, where $db_{X \cup Y}$ is the transaction set containing $X \cup Y$; and the confidence value, c%, of

the rule is the conditional probability, $Pr(Y|X) = Pr(X \cup Y)/Pr(X)$, denoted as $Conf(X \Rightarrow Y)$. If the support value of an itemset is not less than *minSup*, the itemset is called the frequent itemset; otherwise, the itemset is an infrequent one. After the frequent itemsets are discovered, users can straightforwardly derive the corresponding confidence value of each rule. Therefore, discovering all frequent itemsets dominates the performance of the mining association rules process.

The Apriori algorithm, a multiple-pass algorithm, is a typical and famous technique to identify frequent itemsets [17]. In the *k*-th pass, Apriori generates all candidate *k*-itemsets (the candidate itemsets with length *k*) and scans the database once to determine whether a candidate *k*-itemset is frequent. In each pass, two arbitrary frequent (*k*-1)-itemsets join to form a *k*-itemset while their first *k*-2 items are identical. Then, Apriori applies an efficient prune strategy, the Apriori property (the downward closure property) [4], to reduce the number of candidates. The Apriori property indicates that no superset of an infrequent itemset can be frequent. Therefore, Apriori can delete the useless candidates, each containing at least one infrequent itemset, to speed up the performance of the mining process. To efficiently discover frequent itemsets, several researchers have proposed certain fast algorithms [18-22].

2.2. Robust Association Rules (RAR) method

For a relational database, a tuple usually contains multiple attributes; a quantitative attribute is often partitioned into multiple intervals and each interval is treated as an item. Therefore, researchers have presented the mining methods of Quantitative Association Rules (QARs), which are based on traditional association rules, to discover quantitative association rules in relational database tables [23, 24].

Let $I = \{P_1, P_2, \ldots, P_n\}$ be a set of n distinct items in a relational database. An item includes a pair (AT^i, V_i), which is its attribute, AT^i, with the associated quantitative value V_i. That is $P_i = (AT^i, V_i)$. For convenience, an item (AT^i, V_i) is written as $AT^i{}_{Vi}$. For example, the symbol, C_2, denotes the item with the attribute C and the quantitative value 2.

In a relational database with some missing values, users must discard the victim tuples (a tuple containing at least one missing value) to discover QARs. Deleting tuples with missing values often generates too few useful rules to be applied effectively [14]. Ragel and Cremilleux have proposed the Robust Association Rules (RAR) approach to reduce the impact of missing values in a database [14]. Instead of deleting each victim tuple, RAR partially disables the victim tuples to solve the lost rules issue. The RAR approach cuts a database into several valid databases (*vDB*), which contain no missing values, in order to discover rules. The definitions in RAR are as follows [14]:

Let *DB* be a relational database, T_i be a tuple in *DB*, and *R* be a rule $X \Rightarrow Y$.

Definition 2.1 (Disabled data) Consider an itemset X. T_i is disabled in DB if T_i has at least one item containing a missing value for X.

Let $Dis(X)$ denote the tuples of all disabled T_i for X in DB.

Definition 2.2 (Valid database) Let vDB_X be the valid database for X in DB. Then, $vDB_X = DB - Dis(X)$.

RAR redefines the support and confidence values as follows.

$Sup(R) = \frac{|db_{X\cup Y}|}{|vDB_{X\cup Y}|}$, and $Conf(R) = \frac{|db_{X\cup Y}|}{|db_X| - |Dis(Y)\cap db_X|}$.

Definition 2.3 (Representative) Let $Rep(X)$ be the representative value of X in DB. The definition of $Rep(X)$ is $Rep(X) = \frac{|vDB_X|}{|DB|}$. A large representative value of $X \cup Y$, which is not less than the threshold, can avoid rules for discovery in a small $vDB_{X\cup Y}$. The user-specified minimum threshold of the representative value is denoted as *minRep*.

Example 2.1 Consider the sample relational database in Table 1 with *minSup* = 60%, *minConf* = 60%, and *minRep* = 70%. The table has ten tuples and four attributes. The four attributes, *A*, *B*, *C*, and *D*, separately include one, one, two, and three quantitative values. The symbol "?" denotes the missing value of the corresponding item in a tuple. In this example, $B_1 \Rightarrow C_1$ is the rule, but $A_1 \Rightarrow C_1$ is not. Because the representative value, $Rep(A_1 \Rightarrow C_1)$ = 6/10 = 60%, is below *minRep*. The support, confidence, and representative values of the rule $B_1 \Rightarrow C_1$ are as follows:

$Sup(B_1)$ = 8/8 = 100%, $Sup(C_1)$ = 6/8 = 75%, $Sup(B_1 \Rightarrow C_1)$ = 6/7 = 85.7%

$Conf(B_1 \Rightarrow C_1)$ = 6/(8-1) = 85.7%, and $Rep(B_1 \Rightarrow C_1)$ = 7/10 = 70%.

Table 1. An example of a relational database

TID \ Attribute	A	B	C	D
T01	1	1	1	1
T02	1	1	1	1
T03	1	?	?	1
T04	1	1	1	2
T05	1	1	1	2
T06	1	1	1	2
T07	1	1	?	3
T08	1	1	1	3
T09	?	1	2	3
T10	?	?	2	2

2.3. Missing Value Completion (MVC) method

RAR only partially disables the victim tuples to passively discover the association rules. For active discovery, Ragel and Cremilleux have also proposed the Missing Values Completion (MVC) approach, which is based on the RAR method, to recover multiple missing values in a database [15]. First, MVC applies RAR to discover all association rules. Then, MVC applies the most appropriate rule to fill in a single missing value in a tuple. If a tuple has multiple missing values, MVC runs the process repeatedly. To avoid the propagating of the wrong value, MVC uses the rules, which have a high confidence value (more than 95%), to recover a tuple with multiple missing values.

Example 2.2 Consider the sample database in Table 1 again. The three minimum thresholds, *minSup*, *minConf*, and *minRep*, are identical with those in Example 2.1. MVC executes the RAR process to generate the four rules, $A_1 \Rightarrow B_1$, $B_1 \Rightarrow C_1$, $B_1 \Rightarrow A_1$, and $C_1 \Rightarrow B_1$, with 100%, 85.7%, 100%, and 85.7% confidence values, respectively. Except for tuple *T10*, MVC recovers each missing value by filling "1" in *T03*, *T07*, and *T09*.

3. Combined Association Rules Using Recycle Technique

Definition 3.1 (Sub-frequent itemset) Give two pre-defined thresholds *minSup* and *minRSup*. If the support value of an itemset, X, is less than *minSup* and not less than *minRSup*, the itemset is called a sub-frequent itemset. That is $minRSup \le Sup(X) < minSup$.

The discovered association rules can be used to recover corresponding missing values. Therefore, increasing the discovered rules can increase the recovery rate for missing values in a relational database. To increase the identified rules, this study combines at least two sub-frequent itemsets to become a frequent combined itemset. The support value of each combinable sub-frequent itemset requires archiving the pre-defined recycle support threshold value, *minRSup,* where *minRSup* < *minSup*. The method of recycling some sub-frequent itemsets is called the Recycle Combined Association Rules (RCAR) method. The definitions are as follows:

Definition 3.2 (Combinable attribute) Let a database include m attributes. Consider n sub-frequent k-itemsets $\{X_1, X_2, \ldots, X_n\}$, if $|\bigcap_{i=1}^{n} X_i| = k\text{-}1 > 0$ and AT^j is the only attribute in $\bigcup_{i=1}^{n} X_i - \bigcap_{i=1}^{n} X_i$, then the attribute, AT^j, is called the combinable attribute of these itemsets, where $1 \le j \le m$.

Definition 3.3 (Combined itemset and combinable itemset) Consider n sub-frequent k-itemsets $\{X_1, X_2, \ldots, X_n\}$ having a common combinable attribute AT^i. The combined itemset $CX = \{X_1 \cup X_2 \cup \ldots \cup X_n\} = \{P_1, P_2, \ldots P_{i\text{-}1}, P_{i+1}, \ldots, P_k\} \cup \{AT^i_{V1}, AT^i_{V2}, \ldots, AT^i_{Vn}\}$. For convenience, CX is denoted as $\{P_1, P_2, \ldots, P_{i\text{-}1}, AT^i_{(V1, V2, \ldots, Vn)}, P_{i+1}, \ldots, P_k\}$. Each sub-frequent k-itemset of CX is called the combinable itemset. For a certain attribute in a sub-frequent itemset, if a combined itemset involves the maximum distinct number of combinable itemsets, it is called the Longest Combined Itemset (LCI).

Definition 3.4 (Frequent combined itemset) Consider n sub-frequent k-itemsets $\{X_1, X_2, \ldots, X_n\}$, according to Definitions 3.2 and 3.3, the support value of CX becomes $sup(CX) = sup(X_1) + sup(X_2) + \ldots + sup(X_n)$. If $Sup(CX) \ge minSup$, then CX is called a frequent combined itemset.

Example 3.1 Consider the relational table in Table 1 with *minSup* = 60%, *minConf* = 60%, *minRep* = 70%, and *minRSup* = 35%. The dataset includes four attributes A, B, C and D. The two sub-frequent itemsets, $\{A_1, D_1\}$ and $\{A_1, D_2\}$, are combinable. The attribute D is the combinable attribute. Therefore, the longest combined itemset on the attribute D is $\{A_1, D_{(1, 2)}\}$.

To further recycle these combinable sub-frequent itemsets to become useful association rules, this study defines the recycled combined association rules as follows:

Definition 3.5 (Recycle combined association rule) An association rule, $X \Rightarrow Y$, is called the Recycle Combined Association Rule (RCAR) if X is a frequent combined itemset and Y is a frequent itemset.

For example, if the frequent combined itemset $\{A_1, D_{(1, 2)}\}$ is generated from Example 3.1, the two rules, $A_1 \Rightarrow D_{(1, 2)}$ and $D_{(1, 2)} \Rightarrow A_1$, can be derived. The former rule inadequately recovers the missing value at attribute D. Therefore, the succedent of a recycle combined association rule does not involve a frequent combined itemset. Example 2.2 shows that MVC can recover four missing values in three tuples. Given the same threshold as Example 2.2 with *minRSup* = 35%, RCAR recovers six missing values in four tuples. Table 2 lists the recovery result of Table 1 by RCAR.

Table 2. The recovered relational database using RCAR

Attribute / TID	A	B	C	D
T01	1	1	1	1
T02	1	1	1	1
T03	1	1	1	1
T04	1	1	1	2
T05	1	1	1	2
T06	1	1	1	2
T07	1	1	1	3
T08	1	1	1	3
T09	1	1	2	3
T10	1	1	2	2

4. An Array-Based Algorithm for RCAR

To speed up the mining process of RCAR, this study establishes the bit-arrays of a relational database table using simple Boolean AND/OR operations, called the Fast Recycle Combined Association Rules (FRCAR), on the arrays. Given a relational database table, performing the array-based algorithm involves three matrices, the Itemset Matrix (IM), the Missing Value Matrix (MVM), and the Recycling Itemset Matrix (RIM). IM and MVM are used to calculate the representative and support value of each corresponding itemset. RIM stores the sub-frequent itemsets, which are prepared to be recycled. The three matrices are described as follows:

Itemset Matrix (IM) and Missing Value Matrix (MVM)

Since each item includes a binary attribute, a relational table can be transformed simply into a bit-array. Table 3 lists the transformed results, in which the missing values from Table 1 are set to "0". Similarly, each missing value also has a binary attribute as listed in Table 4. Performing a bit-wise "AND" operation between columns of IM obtains the support count of the corresponding itemsets by counting the number of "1"s. Similarly, performing a bit-wise "OR" operation between columns of the MVM obtains the value, $|DB| - |vDB|$, of the corresponding itemsets. These two values can be applied to easily compute the representative and support values. For example, in Table 3, the bit-wise "AND" operation between the two columns A_1 and B_1 obtains the occurrence count of the itemset $\{A_1, B_1\}$, which is seven. Similarly, in Table 4, the bit-wise "OR" operation between the two columns A and B obtains the value three. Since the total number of tuples is ten, the representative value of $\{A, B\}$ is (10 - 3)/10 = 70%. The support value of $\{A_1, B_1\}$ is (7/10)/70% = 100%.

Recycling Itemset Matrix (RIM)

The FRCAR method requires that sub-frequent itemsets be recycled. Therefore, in each pass of the mining process, FRCAR creates an itemset matrix, called the recycling itemset matrix (RIM), to store the sub-frequent itemset. For example, except for the last two rows, Table 5 lists the RIM of the data in Table 1, with *minSup* = 60%, *minRSup* = 35%, and *minRep* = 70%. The bit-wise "OR" operation between the two columns $\{A_1, D_1\}$ and $\{A_1, D_2\}$ obtains the combined itemset $\{A_1, D_{(1, 2)}\}$, which has the frequent support value 75%.

Table 3. The itemset matrix of Table 1

Attribute	A	B	C		D		
Item / TID	A_1	B_1	C_1	C_2	D_1	D_2	D_3
T01	1	1	1	0	1	0	0
T02	1	1	1	0	1	0	0
T03	1	0	0	0	1	0	0
T04	1	1	1	0	0	1	0
T05	1	1	1	0	0	1	0
T06	1	1	1	0	0	1	0
T07	1	1	0	0	0	0	1
T08	1	1	1	0	0	0	1
T09	0	1	0	1	0	0	1
T10	0	0	0	1	0	1	0

Table 4. The missing value matrix of Table 1

Attribute / TID	A	B	C	D
T01	0	0	0	0
T02	0	0	0	0
T03	0	1	1	0
T04	0	0	0	0
T05	0	0	0	0
T06	0	0	0	0
T07	0	0	1	0
T08	0	0	0	0
T09	1	0	0	0
T10	1	1	0	0

Table 5. The recycling itemset matrix of Table 1 with minSup = 60%, minRSup = 35% and minRep = 70%

Itemset *X* / TID	A_1, D_1	A_1, D_2
T01	1	0
T02	1	0
T03	1	0
T04	0	1
T05	0	1
T06	0	1
T07	0	0
T08	0	0
T09	0	0
T10	0	0
Rep(*X*)	80.0%	80.0%
Sup(*X*)	37.5%	37.5%

The FRCAR algorithm for discovering all frequent itemsets is as follows:

Algorithm 1 (Discovering frequent itemsets)

Input: Relational database *DB* (including missing values), *minRep*, *minSup*, and *minRSup*

Output: All frequent itemsets (including frequent combined itemsets)

Procedure:

```
1.   k:=1; F1:=ϕ; C1:=I; // Fk: the set of frequent k-itemsets
2.   Scan DB to Construct MVM and IM1; // IMk: IM with k-itemsets
3.   foreach Pi∈C1  {
4.        if Sup(Pi) ≥ minSup && Rep(Pi) ≥ minRep {
5.             F1:= F1+Pi; }
6.        else if Sup(Pi) < minRSup×minRep || Rep(Pi) < minRep {
7.             IM1:=IM1 - Pi_array; } } // Pi_array: the bit array of the item Pi
8.   for (k:=2; |IMk-1|≥k; k++) {
9.        Fk:=ϕ; CFk:=ϕ; RIMk:=ϕ;
10.       (IMk, RIMk):= Array-operation(IMk-1, MVM);
11.       delete IMk-1;
12.       foreach Xi_array∈IMk  { // Xi_array: the bit array of the itemset Xi
13.            if Sup(Xi)≥ && Rep(Xi) ≥ minRep {
14.                 Fk:=Fk+Xi; }    }
15.       CFk:= Combine(RIMk); }    }    //CFk: the set of frequent combined
  k-itemsets
16.  return F:=⋃k Fk+⋃k CFk;
```

The first pass of the process, which scans the database only once, is described in Lines 1-7. In Lines 6 and 7, FRCAR eliminates the bit array of P_i from IM_1 when $Sup(P_i)<minRSup\times minRep$. The reason is that a superset of the item P_i may be frequent or sub-frequent when $Sup(P_i)\geq minRSup\times minRep$. In Line 10, the process creates the two matrices, IM and RIM, and recycles the sub-frequent itemsets. The frequent combined itemsets generated are in Line 14. The two sub-procedures, Array-operation() and Combine(), are as follows:

Sub-procedure Array-operation (IM_{k-1}, MVM)

```
1.  IMk:=ϕ; RIMk:=ϕ;
2.  foreach Xi_array∈IMk-1 {
3.   foreach Xj_array∈IMk-1 {   // j>i
4.        if true==Apiroi-join(Xi, Xj) {
5.             Xi'_array:= Xi_array∧Xj_array;
6.             Rep(Xi'):= Representative(MVM, Xi, Xj);
7.             Count Sup(Xi');
8.             if Sup(Xi')≥minRSup×minRep && Rep(Xi')≥minRep {
9.                  IMk:= IMk + Xi'_array; }
10.            if Sup(Xi')≥minRSup && Sup(Xi')<minSup && Rep(Xi')≥minRep {
11.                 RIMk:= RIMk + Xi'_array;     }    }
12.       else { break;  }    }    }
13. Return IMk and RIMk;
```

In Line 4, the sub-procedure Apriori-join() determines whether two itemsets can be joined. In Line 6, the sub-procedure Representative() calculates the representative value of the corresponding itemset. In Lines 10 and 11, FRCAR recycles the sub-frequent itemsets.

Sub-procedure Combine (RIM_k)

Procedure:

```
1.  CFk:=ϕ;
2.  foreach Xi_array∈RIMk   {
3.   foreach attribute∈Xi {
4.        discover LCI;  // LCI: the set of Longest Combined Itemsets
5.        if Sup(LCIi)≥minSup && Rep(LCIi)≥minRep {
6.             CFk:=CFk+LCIi;  }
7.   foreach Xi∈LCI {
8.        remove Xi_array from RIMk;   }    }
9.  Return CFk;
```

Since all frequent itemsets can be used to straightforwardly generate the corresponding recycled combined association rules, this study omits the algorithm of the rule generation.

5. Experimental Results

The performance of FRCAR was compared with that of MVC using an AMD Barton ES 2900+ (2000MHz) PC with 1 GB of main memory, running Windows XP Professional. All algorithms were coded in Visual C++ 6.0, and applied to two real datasets, Breast-cancer and Tic-tac-toe, which were taken from the UCI Machine Learning Repository [25]. Table 6 lists a summary of the two datasets and the used thresholds in this experiment. Each attribute of Breast-cancer and Tic-tac-toe randomly introduces 25 and 50 missing values, respectively.

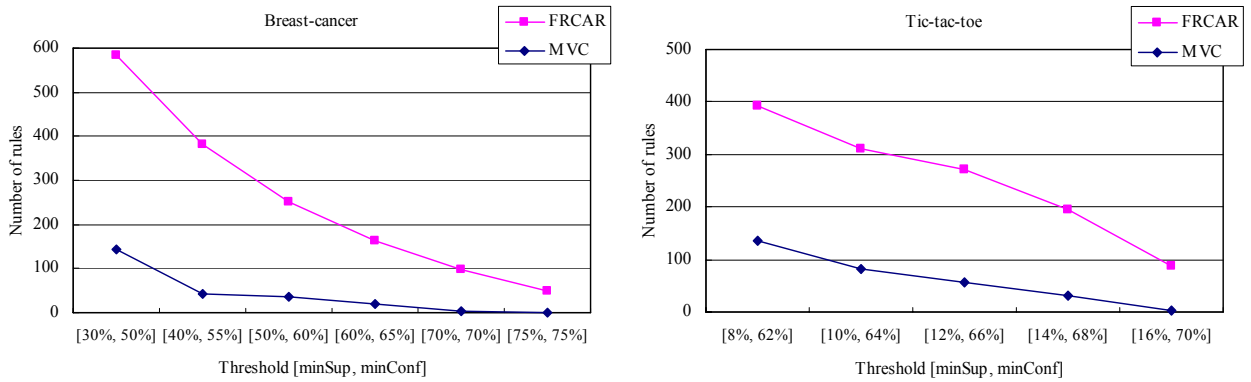
Table 6. Characteristics of the two experimental datasets and the used threshold values

Dataset	Tuple number	Attribute number	Generated number of missing values	minRSup	minRep
Breast-cancer	277	10	250	5%	70%
Tic-tac-toe	958	10	500	4%	75%

Figure 1 shows the comparison of the number of rules generated by MVC and FRCAR over various sets of *minSup* and *minConf* thresholds. In the figure, FRCAR generated more rules than MVC, particularly in a low *minSup* value scenario. The reason was that FRCAR generated not only association rules, but combined association rules. By using these generated rules, Figure 2 demonstrates the power of FRCAR to recover the missing values. For example, MVC had no rule resulting in the zero recovery rate in the scenario *minSup* = 75% and *minConf* = 75%, while FRCAR generated 66 rules and had a recovery rate of 26.4% as shown in Figures 1(a) and 2(a). FRCAR generated more rules than MVC did. Therefore, FRCAR had the higher recovery rate than that of MVC as shown in Figure 2. Figure 3 shows that FRCAR also demonstrated a better recovery precision than MVC. In other words, the number of right guesses of FRCAR was greater than that of MVC.

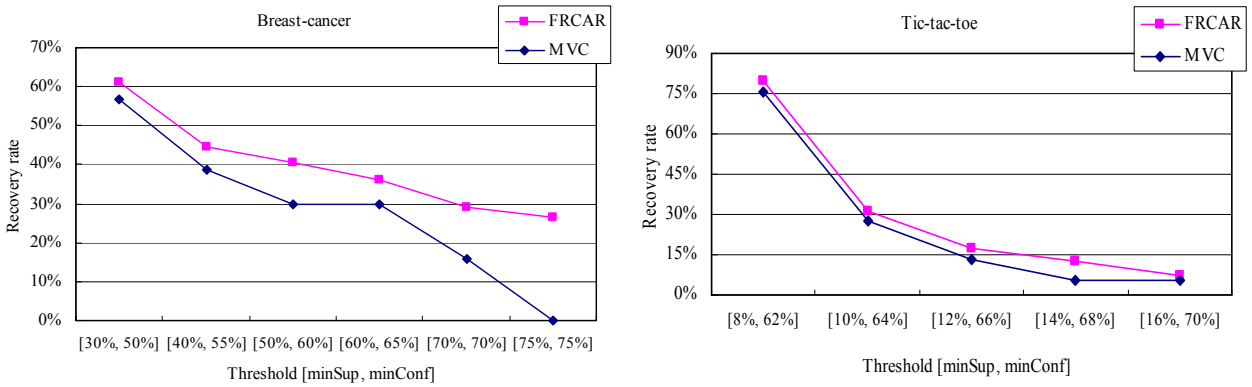
Tables 7 and 8 compare the hit rates (the total right guessing number/total guessing number) for MVC and FRCAR. In the middle four columns of Table 7, although FRCAR had a lower hit rate than MVC, the four right guessing numbers of FRCAR were higher than those of MVC. In Table 8, FRCAR had a higher hit rate than MVC except in the scenario, [14%, 68%].

This experiment also compared the running time of the two algorithms for generating frequent itemsets and frequent combined itemsets. Figures 4(a) and 4(b) present the running time curves of the two algorithms on Breast-cancer and Tic-tac-toe, respectively. FRCAR generated more candidate itemsets than MVC. Therefore, FRCAR had a higher running time. Although FRCAR had a longer running time, it ran quickly on the two datasets. For example, as shown in Figure 4, FRCAR only spent less than 0.08 seconds and 0.4 seconds on Breast-cancer and Tic-tac-toe, respectively.



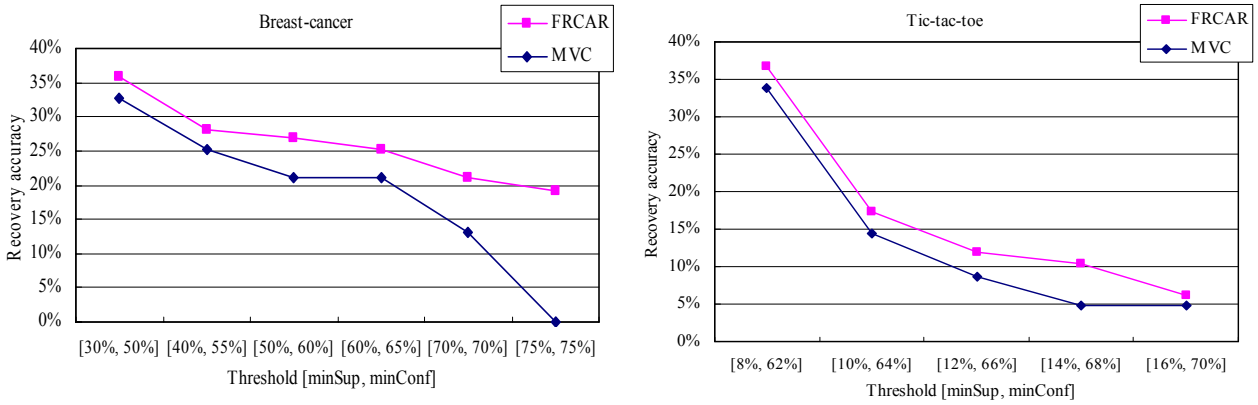
(a) (b)

Figure 1. The comparison of the association rules number for MVC and FRCAR on (a) Breast-cancer and (b) Tic-tac-toe



(a) (b)

Figure 2. The comparison of recovery rates for MVC and FRCAR on (a) Breast-cancer and (b) Tic-tac-toe



(a) (b)

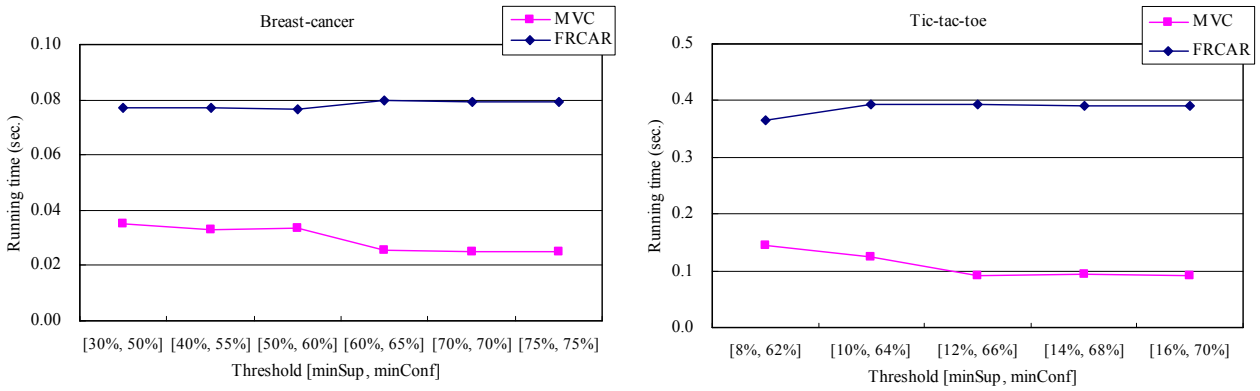
Figure 3. The comparison of recovery precision for MVC and FRCAR on (a) Breast-cancer and (b) Tic-tac-toe

Table 7. The hit rate comparison for MVC and FRCAR on Breast-cancer

[minSup, minConf]		[30%, 50%]	[40%, 55%]	[50%, 60%]	[60%, 65%]	[70%, 70%]	[75%, 75%]
MVC	Hit rate	57.7%	64.9%	70.7%	70.7%	82.5%	0.0%
	Hit number	82	63	53	53	33	0
FRCAR	Hit rate	58.8%	63.1%	66.3%	70.0%	72.6%	72.7%
	Hit number	90	70	67	63	53	48

Table 8. The hit rate comparison for MVC and FRCAR on Tic-tac-toe

[minSup, minConf]		[8%, 62%]	[10%, 64%]	[12%, 66%]	[14%, 68%]	[16%, 70%]
MVC	Hit rate	44.8%	51.8%	64.2%	85.7%	85.7%
	Hit number	169	72	43	24	24
FRCAR	Hit rate	46.0%	55.8%	68.2%	83.9%	86.1%
	Hit number	184	87	60	52	31



(a) (b)

Figure 4. The comparison of running times for MVC and FRCAR on (a) Breast-cancer and (b) Tic-tac-toe

6. Conclusions

Data preparation is an essential phase of data mining, and the recovery of missing values is an important issue in data preparation. The association rules have the advantage of effectively pointing out the relationship between items in databases. Therefore, this study presents a Fast Recycle Combined Association Rules (FRCAR) method to recover missing values in data. FRCAR allows increasing the recovery rate and recovery precision of the missing values. In comparing FRCAR with the MVC method, this study found that by applying a technique to recycle sub-frequent itemsets, FRCAR discovers more association rules than MVC. Moreover, FRCAR uses bit-wise operations on three created bit-arrays to efficiently discover frequent combined itemsets. In the same threshold values scenario, the experimental results show that FRCAR has a better recovery rate and higher recovery accuracy for recovering missing values than MVC.

Acknowledgements

The authors would like to acknowledge M. Zwitter and M. Soklic for kindly providing the Breast-cancer dataset and thank David W. Aha for kindly providing the Tic-tac-toe dataset.

References

[1] M.Y. Chen and A.P. Chen, Knowledge management performance evaluation: a decade review from 1995 to 2004, *Journal of Information Science* 32(1) (2006) 17-38.

[2] M.B. Nunes, F. Annansingh, and B. Eaglestone, Knowledge management issues in knowledge-intensive SMEs, *Journal of Documentation* 62(1) (2006) 101-19.

[3] Y.B. Wu, Q. Li, R. S. Bot, and X. Chen, Finding nuggets in documents: a machine learning approach, to appear in *Journal of the American Society for Information Science and Technology*.

[4] J. Han and M. Kamber (eds), Data Mining: Concepts and Techniques (Morgan Kaufmann, San Francisco, CA, 2000).

[5] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, The KDD process for extracting useful knowledge from volumes of data, *Communications of the ACM* 39(11) (1996) 27-34.

[6] S. Zhang, C. Zhang, and Q. Yang, Data preparation for data mining, *Applied Artificial Intelligence* 17(5-6) (2003) 375-81.

[7] S.Y. Chen and X. Liu, The contribution of data mining to information science, *Journal of Information Science* 30(6) (2004) 550-8.

[8] J.F. Hair, R.E. Anderson, R.L. Tatham, and W. Black (eds), Multivariate Data Analysis (Prentice Hall, London, 5th Ed., 1998)

[9] J.R. Quinlan, Unknown attribute values in induction, In: A.M. Segre (ed.), *Proceedings of the 6th International Workshop on Machine Learning*, Ithaca, New York, 1989, (Morgan Kaufmann, San Francisco, CA, 1989) 164-8.

[10] J.R. Quinlan (ed.), C4.5: Programs for Machine Learning (Morgan Kaufmann, San Francisco, CA, 1993).

[11] K. Lakshminarayan, S.A. Harp, R, Goldman and T. Samad, Imputation of missing data using machine learning techniques, In E. Simoudis, J. Han, and U.M. Fayyad (eds), *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining,* Portland, OR, August 1996 (AAAI Press, Menlo Park, CA,1996) 140-6.

[12] W.Z. Liu, A.P. White, S.G. Thompson, and M.A. Bramer, Techniques for dealing with missing values in classification, In: X. Liu, P.R. Cohen, and M.R. Berthold (eds), *Lecture Notes in Computer Science* 1280 (Springer, Berlin, 1997) 527-36.

[13] P. Wright, Knowledge discovery preprocessing: determining record usability, In: *Proceedings of the 36th Annual Conference on Southeast Regional Conference,* Marietta, GA, April 1998, (ACM Press, New York, 1998) 283-8.

[14] A. Ragel and B. Cremilleux, Treatment of missing values for association rules, In: X. Wu, K. Ramamohanarao, and K.B. Korb (eds), *Lecture Notes in Computer Science* 1394 (Springer, Berlin, 1998) 258-70.

[15] A. Ragel and B. Cremilleux, MVC - a preprocessing method to deal with missing values, *Knowledge-Based Systems*, 12(5-6) (1999) 285-91.

[16] R. Agrawal, T. Imielinski, and A. Swami, Mining association rules between sets of items in large database, In: P. Buneman and S. Jajodia (eds), *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, D.C., May 1993, (ACM Press, New York, 1993) 207-16.

[17] R. Agrawal and R. Srikant, Fast algorithms for mining association rules in large databases, In: J.B. Bocca, M. Jarke, and C. Zaniolo (eds), *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, September 1994, (Morgan Kaufmann, San Francisco, CA, 1994) 487-99.

[18] J.S. Park, M.-S. Chen, and P.S. Yu, An effect hash-based algorithm for mining association rules, In: M.J. Carey and D.A. Schneider (eds), *Proceedings of the ACM-SIGMOD International Conference on Management of Data,* San Jose, CA, May 1995, (ACM Press, New York, 1995) 175-86.

[19] A. Savasere, E. Omiecinski, and S. Navathe, An efficient algorithm for mining association rules in large databases, In: U. Dayal, P.M.D. Gray, and S. Nishio (eds), *Proceedings of the 21st International Conference on Very Large Data Bases*, Zurich, Switzerland, September 1995, (Morgan Kaufmann, San Francisco, CA, 1995) 432-44.

[20] J. Han, J. Pei, Y. Yin, and R. Mao, Mining frequent pattern without candidate generation: a frequent pattern tree approach, *Data Mining and Knowledge Discovery* 8(1) 2004 53-87.

[21] Y.-C. Li and C.-C. Chang, A new FP-tree algorithm for mining frequent itemsets, In C.-H. Chi and K.-Y. Lam (eds), *Lecture Notes in Compute Science* 3309 (Springer, Berlin, 2004) 266-77.

[22] C.-C. Chang and C.-Y. Lin, Perfect hashing schemes for mining association Rules, *The Computer Journal* 48(2) (2005) 168-79.

[23] R. Agrawal and R. Srikant, Mining quantitative association rules in large relational tables, In: H.V. Jagadish and I.S. Mumick (eds), *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Montreal, Quebec, Canada, June 1996, (ACM Press, New York, 1996) 1-12.

[24] R.J. Miller and Y. Yang, Association rules over interval data, In: J. Peckham (ed.), *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, AZ, May 1997, (ACM Press, New York, 1997) 452-61.

[25] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz, *UCI Machine Learning Repository* (1998). Available at: http://www.ics.uci.edu/~mlearn/MLRepository.html (accessed 1 January 2005).