# On NIS-Apriori Based Data Mining in SQL

Sakai Hiroshi, Liu Chenxi, Zhu Xiaoxin, Nakata Michinori

**Journal or Publication Title:** Lecture Notes in Computer Science  
**Volume:** 9920  
**Page Range:** 514-524  
**Year:** 2016-09-29  
**URL:** http://hdl.handle.net/10228/00006391  
**doi:** info:doi/10.1007/978-3-319-47160-0_47

|著者|鳥取大学大学院情報科学研究科情報システム工学専攻博士課程第二種

|研究分野|情報工学

|所属機関|九州産業大学大学院情報科学研究科情報システム工学専攻

|問い合わせ先|鳥取大学大学院情報科学研究科情報システム工学専攻

|doi|info:doi/10.1007/978-3-319-47160-0_47|
On NIS-Apriori based Data Mining in SQL

Hiroshi Sakai\(^1\), Chenxi Liu\(^1\), Xiaoxin Zhu\(^2\), and Michinori Nakata\(^3\)

\(^1\) Graduate School of Engineering, Kyushu Institute of Technology
Tobata, Kitakyushu 804-8550, Japan
sakai@ms.kyutech.ac.jp, p350932s@mail.kyutech.jp

\(^2\) Graduate School of of Engineering, Harbin Institute of Technology
Westdazhi Str. Nangang Dist. Harbin City, China
XiaoxinZhu@hit.edu.cn

\(^3\) Faculty of Management and Information Science,
Josai International University
Gunno, Togane, Chiba 283-0002, Japan
nakatam@ieee.org

Abstract. We have proposed a framework of Rough Non-deterministic Information Analysis (RNIA) for tables with non-deterministic information, and applied RNIA to analyzing tables with uncertainty. We have also developed the RNIA software tool in Prolog and getRNIA in Python, in addition to these two tools we newly consider the RNIA software tool in SQL for handling large size data sets. This paper reports the current state of the prototype named NIS-Apriori in SQL, which will afford us more convenient environment for data analysis.

Keywords: Association rules, NIS-Apriori algorithm, SQL, Prototype, Uncertainty.

1 Introduction

We have been coping with rough sets [7], non-deterministic information [6, 7], the \textit{Apriori} algorithm [1, 12], the software tool in Prolog [9], and getRNIA in Python [11, 15]. Recently, we are considering a software tool in SQL in order to handle large size data sets.

In rough sets, we usually employ Deterministic Information Systems (DISs) with deterministic attribute values. We can see every DIS is a standard table. For handling information incompleteness in tables [2, 4, 6, 7, 9], we employ Non-deterministic Information Systems (NISs) with non-deterministic values and missing values. By changing DIS to NIS, several new issues occurred, for example the possible equivalence classes, the minimum and the maximum degree of data dependency, the certain and the possible rules, and so on [9]. At the same time, one computational problem occurred, namely the computational complexity may increase exponentially due to the case analysis on NIS. However, in rule generation, we proved some properties and escaped from the exponential order problem [10, 11]. Due to this result, the rule generator in Prolog [10] and getRNIA in Python [15] were implemented.
In this paper, we focus on the rule generator in SQL, because SQL has the high versatility. Furthermore, several algorithms including Apriori were investigated in [12]. For handling large size data sets, we think SQL will be more suitable than the previous languages, Prolog and Python. Recently, the 'sparse' property of the data sets is considered [2, 14]. The density of the important part in the data sets may not be unique, and we may ignore the meaningless part. In the sparse matrix, we may employ the special format for reducing the data size. The use of this sparse property will be another approach to large size data sets.

As for this work, we need to specify that this is not the first trial, and the first trial was done in [13]. We follow the result in [13], and consider a rule generator which we name NIS-Apriori in SQL.

This paper is organized as follows: Section 2 surveys RNIA and rule generation. Section 3 investigates NIS-Apriori in SQL and its prototype system. Section 4 concludes this paper.

2 RNIA and Rule Generation

At first, we clarify the rules in DIS. For a fixed decision attribute Dec and a set CON of attributes, we see an implication \( \tau : \wedge_{\forall \in \text{CON}} [A, val_A] \Rightarrow [Dec, val] \) is (a candidate of) a rule, if \( \tau \) satisfies the next two constraints.

\[
\begin{align*}
\text{support}(\tau) &= N(\tau)/(|OB|) \geq \alpha, \\
\text{accuracy}(\tau) &= N(\tau)/N(\wedge_{\forall \in \text{CON}} [A, val_A]) \geq \beta,
\end{align*}
\]

Here, \( N(*) \) means the number of objects satisfying the formula \(*\), \( OB \) means a set of all objects.

Then, we briefly survey rule generation in RNIA. Figure 1 shows NIS \( \Psi_1 \), where we see [high,veryhigh] and nil. Here, [high,veryhigh] is non-deterministic information, namely either high or veryhigh is the actual value, and nil is missing value. Each nil may take every possible value in the attribute.

In \( \Psi_1 \), we replace each non-deterministic information and nil with a possible value, and we obtain a table with deterministic information. We named it a derived DIS from NIS. Let \( DD(\Psi) \) be a set of all derived DISs from \( \Psi \). We see an actual DIS \( \phi_{\text{actual}} \) exists in \( DD(\Psi) \). For \( \Psi_1 \), \( DD(\Psi_1) \) consists of \( 4008 (=3^2 \times 2^3) \) derived DISs. Based on \( DD(\Psi) \), we proposed the certain and the possible rules below:

Definition 1. [10] For NIS \( \Psi \) and the decision attribute Dec, we fix the threshold values \( \alpha \) and \( \beta \) (0 < \( \alpha, \beta \leq 1.0 \)).

1. We say \( \tau \) is a certain rule, if \( \tau \) satisfies \( \text{support}(\tau) \geq \alpha \) and \( \text{accuracy}(\tau) \geq \beta \) in each \( \phi \in DD(\Psi) \).
2. We say \( \tau \) is a possible rule, if \( \tau \) satisfies \( \text{support}(\tau) \geq \alpha \) and \( \text{accuracy}(\tau) \geq \beta \) in at least one \( \phi \in DD(\Psi) \).

Definition 1 seems natural, but we have the computational complexity problem, because the number of elements in \( DD(\Psi) \) increases exponentially. In \( \Psi_1 \),
the number is $4000$, and the number is more than $10^{100}$ in Mammographic data set in UCI machine learning repository [3]. For this computational problem, we defined two sets for a descriptor $[A, val]$ below:

$$
\begin{align*}
\inf([A, val]) &= \{ x: object \mid \text{the value of } x \text{ for } A \text{ is a singleton set } \{val}\}, \\
\sup([A, val]) &= \{ x: object \mid \text{the value of } x \text{ for } A \text{ is a set including } val\}, \\
\inf(\land_{A \in CON}[A, val]) &= \cap_{A \in CON}\inf([A, val]), \\
\sup(\land_{A \in CON}[A, val]) &= \cap_{A \in CON}\sup([A, val]).
\end{align*}
$$

For example, $\inf([\text{head, yes}]) = \{x2, x4, x6, x8\}$ and $\sup([\text{head, yes}]) = \inf([\text{head, yes}]) \cup \{x1, x5\}$ hold in $\Psi$. The actual equivalence class is between two sets. For $NIS \Psi$, an implication $\tau$, and $\minsupp(\tau)$ and $\minacc(\tau)$ defined by $\min_{\phi \in DD(\Psi)}\{\text{support}(\tau) \text{ by } \phi\}$ and $\min_{\phi \in DD(\Psi)}\{\text{accuracy}(\tau) \text{ by } \phi\}$, we have the following which do not depend upon the number of $DD(\Psi)$.

$$
\begin{align*}
\tau: \land_{A \in CON}[A, val] &\Rightarrow [\text{Dec}, val], \\
\minsupp(\tau) &= \inf([\land_{A \in CON}[A, val]] \cap \inf([\text{Dec}, val]))/|OB|, \\
\minacc(\tau) &= \inf([\land_{A \in CON}[A, val]] \cap \inf([\text{Dec}, val]))/\inf([\land_{A \in CON}[A, val]]), \\
\text{OUTACC} &= \{\sup([\land_{A \in CON}[A, val]] \cap \inf([\land_{A \in CON}[A, val]])/\inf([\text{Dec}, val])\}. \\
\end{align*}
$$

The $\text{OUTACC}$ means a set of objects, from which we can obtain an implication $\tau': \land_{A \in CON}[A, val] \Rightarrow [\text{Dec}, val']$ (val $\neq$ val'). Similarly, we can calculate $\maxsupp(\tau)$ and $\maxacc(\tau)$. We can also prove that there exists $\phi_{\min} \in DD(\Psi)$ which makes both $\text{support}(\tau)$ and $\text{accuracy}(\tau)$ the minimum. There exists $\phi_{\max} \in DD(\Psi)$ which makes both $\text{support}(\tau)$ and $\text{accuracy}(\tau)$ the maximum. Based on these results, we have the chart in Figure 3 and Theorem 1.

**Theorem 1.** For an implication $\tau$, we have the following.

1. $\tau$ is a certain rule, if and only if $\minsupp(\tau) \geq \alpha$ and $\minacc(\tau) \geq \beta$.
2. $\tau$ is a possible rule, if and only if $\maxsupp(\tau) \geq \alpha$ and $\maxacc(\tau) \geq \beta$. 

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
\textbf{object} & \textbf{temp(temperature)} & \textbf{head(hoe)} & \textbf{mal(ena)} & \textbf{flu} & \textbf{object} & \textbf{attri val} \textbf{det} \\
\hline
x1 & high & nil & no & yes & x1 & temp & high & 1 \\
\hline
x2 & high & nil & yes & yes & x1 & head & yes & 2 \\
\hline
x3 & nil & no & no & nil & x1 & head & no & 2 \\
\hline
x4 & high & yes & nil & nil & x1 & flu & yes & 1 \\
\hline
x5 & high & nil & yes & no & x1 & flu & yes & 1 \\
\hline
x6 & normal & yes & nil & nil & x2 & head & yes & 1 \\
\hline
x7 & normal & no & yes & no & x2 & head & yes & 1 \\
\hline
x8 & nil & yes & nil & yes & x2 & head & yes & 1 \\
\hline
\end{tabular}
\end{table}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig1.png}
\caption{An exemplary $\Psi_1$.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig2.png}
\caption{A part of $\Psi_1$ in NRDF format.}
\end{figure}
Fig. 3. The distribution of each point \((support(\tau), accuracy(\tau))\) by \(\phi \in DD(\Psi)\).

(3) Even though the certain rules and the possible rules depend upon \(DD(\Psi)\), the conditions to check them do not depend upon \(DD(\Psi)\).

Based on Theorem 1, we can escape from the exponential order problem. Without Theorem 1, it will be hard to handle Mammographic data set, which has more than \(10^{100}\) derived DISs.

3 NIS-Apriori in SQL

3.1 NIS-Apriori Algorithm

The Apriori algorithm was proposed by Agrawal, and this is the representative algorithm in data mining [1, 12]. This algorithm handles transaction data, and each transaction is given as a set of items. We identify each descriptor in table data with an item, then we can consider the Apriori algorithm in tables [10, 11]. In certain rule generation, we compare the minimum point in Figure 3 with the threshold values \(\alpha\) and \(\beta\). On the other hand, we compare the maximum point in Figure 3 with the threshold values \(\alpha\) and \(\beta\). Since the management of the implications is almost the same as in case of the Apriori algorithm and the calculation does not depend upon \(|DD(\Psi)|\), we figure out that the computational complexity of the NIS-Apriori algorithm is about twice the complexity of the Apriori algorithm.

3.2 The NRDF Format

In data sets, we usually have the csv format. This is very familiar, however the name of the attribute and the number of all attributes may be different in each data set. For handling various types of data sets, it is useful to employ another unified format. Otherwise, the program is depending upon the number of the attributes and the name of the attribute.

Based on [13], we employ the NRDF format, which is the extended RDF (resource description framework) format, for any data set. This RDF format
may be called as the EAV (entity-attribute-value) format \cite{5,14}. In \cite{5}, the KDD-related tasks of attribute selection and decision tree induction were implemented based on the EAV format.

The NRDF format employs 4 attributes, object, attrib, value, and det. Figure 2 shows a part of the NRDF format of $\Psi_1$. In order to specify non-deterministic information, we added the 4th column det. The value of det means the number of possible values. If det=1, this means that the value is deterministic. Otherwise, we know the value is non-deterministic and the number of values by det.

3.3 Step 1: Rule Generation in the Form of $P_1 \Rightarrow \text{Dec}$

In Step 1, the procedure step1 generates the certain and the possible rules in the form of $P_1 \Rightarrow \text{Dec}$. This procedure consists of the following:

1. create table condi (the condition of the rules),
2. create table con_des (the descriptors for the condition),
3. create table dec_des (the descriptors for the decision),
4. create table impl1 (the implications with inf, sup, inacc, outacc),
5. create table crule1 (the certain rules),
6. create table prule1 (the possible rules),
7. create table crest1 (the candidates of Step 2),
8. create table prest1 (the candidates of Step 2).

At first, a file mdf in the NRDF format in Figure 2 is stored in the system (Figure 4). In Figure 4, we execute ‘call step1(‘flu’, 8, 0.1, 0.8)’, which means the decision attribute is ‘flu’, the number of the objects is 8, the support value is 0.1, and the accuracy value is 0.8. It took about 0.33 (sec) for executing the procedure step1 in windows PC, and all tables in Figure 5 were generated.

In Figure 5, two tables con_des and dec_des store the set of descriptors on the condition part and the set of descriptors on the decision part, respectively.

Fig. 4. SQL query execution, where Japanese characters were erased.

Fig. 5. All created tables.
The procedure `step1` generates the Cartesian Products by using `con_des` and `dec_des`, and adds `inf`, `sup`, `inacc` and `outacc` to the table `implil` (Figure 6).

Based on `implil`, the procedure `step1` calculates `minsupp` and `minacc` for each tuple and compares them with the threshold values `alpha` and `beta`. If `minsupp` ≥ `alpha` and `minacc` ≥ `beta`, the procedure `step1` adds this tuple to the table `crule1`. If `minsupp` ≥ `alpha` and `minacc` < `beta`, the procedure adds this tuple to the table `crest1` (Figure 7). On the other hand, the procedure calculates `maxsupp` and `maxacc` for each tuple and compares them with the threshold values `alpha` and `beta`. If `maxsupp` ≥ `alpha` and `maxacc` ≥ `beta`, the procedure adds this tuple to the table `prule1` (Figure 8). If `maxsupp` ≥ `alpha` and `maxacc` < `beta`, the procedure adds this tuple to the table `prest1`. The following is the SQL procedure for generating the table `prule1`.

The procedure for `prule1` in Step 1:
```sql
create table prule1 (atti varchar, val1 varchar, deci varchar, deci_value varchar, maxsupp decimal, maxacc decimal);
select implil.atti, implil.val1, implil.deci, implil.deci_value,
implil.sup/ob as maxsupp, implil.sup/(con_des.inf+inacc) as maxacc
from implil, con_des
where implil.atti=con_des.atti and implil.val1=con_des.val1
having maxsupp >=alpha and maxacc >=beta;
```

In Step 1, the most complicated part is to generate the table `implil`. After obtaining the Cartesian Products `impl`, `step1` sequentially adds `inf`, `sup`, `inacc`, and `outacc` to `implil`. If `inf([A, valA] ∨ [Dec, val])` is an empty set, this tuple is not stored in the temporary table data set. Even though it is necessary to add `inf=0` to the table `implil`, the value NULL is added to `implil` in this case. Therefore, `step1` replaces NULL with 0 after adding `inf` information to `implil`. The same occurs for `sup`, `inacc`, and `outacc`. In the current implementation, we faithfully simulated the NIS-Apriori algorithm, and there are ineffective procedures including the above case. It is necessary to reduce such ineffective part.
3.4 Step 2: Rule Generation in the Form of $P_1 \land P_2 \Rightarrow \text{Dec}$

In Step 2, the procedure step2 generates the certain and the possible rules in the form of $P_1 \land P_2 \Rightarrow \text{Dec}$. Since $\text{support}(P_1 \land P_2 \Rightarrow \text{Dec}) \leq \text{support}(P_1 \Rightarrow \text{Dec})$ holds, it is enough to consider the implications $P_1 \land P_2 \Rightarrow \text{Dec}$ satisfying $(P_1 \Rightarrow \text{Dec}), (P_2 \Rightarrow \text{Dec}) \in \text{crest1}$ in certain rule generation.

We execute ‘call step2(‘flu’, 8, 0.1, 0.8)’ again, and it took about 0.39 (sec) for executing the procedure step2. Then, all tables in Figure 10 were generated. In Figure 10, two tables cimpli2 and pimpli2 store the tuples with $\text{inf}$, $\text{sup}$, $\text{inacc}$ and $\text{outacc}$, respectively. Tables crule2 and prule2 store the certain rules and the possible rules in the form of $P_1 \land P_2 \Rightarrow \text{Dec}$. Similarly to the tables crest1 and prest1, crest2 and prest2 are generated for Step 3. In Step 2, we obtained a certain rule in Figure 9 and 12 possible rules in prule2.

The rule generation in Step 3 is the same as in case of Step 2. Like Step 2, we execute ‘call step3(‘flu’, 8, 0.1, 0.8)’, then the procedure step3 generates rules.

3.5 An Implementation of NIS-Apriori in SQL

This prototype system is implemented on desktop PC and note PC by using the phpMyAdmin tool [8]. Currently, we made three procedures step1, step2, and step3 by using SQL command procedures. The data size of this file including all procedures is about 40KB in the text format. Since SQL command procedure is familiar, we will be able to use this prototype in the most of PC with SQL. Actually, we employed both desktop PC and note PC simultaneously for this
implementation. We can also handle any DIS as a special case of NIS. In the
NRDF format, we specify det=1 in each tuple, then we have the same rules in
certain rule generation and possible rule generation.

3.6 The Difference between Two Software Tools RNAI in Prolog
and NIS-Apriori in SQL

Figure 11 is the execution log for NIS $\psi_1$ by RNAI in Prolog. Except the redundant case of the rules, we examined the result by RNAI in Prolog is equal to the result by NIS-Apriori in SQL. This will be an assurance that two software tools were implemented correctly.

Now, we have to remark the difference between the data structures of two software tools. RNAI in Prolog employs two blocks $inf$ and $sup$, and internally manages them for each calculation. On the other hand, NIS-Apriori in SQL does not employ them directly, and employs the total search of the data set. These two points are the big difference between two software tools. We explain these two points below.

RNAI in Prolog generates $inf([A,val_A])$ and $sup([A,val_A])$ information for each descriptor $[A,val_A]$, and $inf(\tau)$ and $sup(\tau)$ are generated for each $\tau$. For example, the set $inf([A,val_A] \land [Dec,val])$ is defined by $inf([A,val_A] \cap inf([Dec,val])$, and it is stored as a temporary set. RNAI in Prolog makes use of $inf(\tau)$ and $sup(\tau)$, and generates rules. However, the use of $inf(\tau)$ and $sup(\tau)$ for each $\tau$ may be a heavy load. Figure 12 is the beginning of the log data for Marumonographic data set [3]. In Figure 12, we see that 427 objects support this rule, and every number of the object, i.e., 3, 5, $\cdots$, 960, is stored in the list. Even though Prolog has a list processing functionality, the manipulation of such large size lists will be a heavy load.

On the other hand, NIS-Apriori in SQL does not store every number of the object, but stores the amount of objects (Figure 13). For obtaining $inf(\tau)$ and
Fig. 12. The list of the objects supporting an implication.

<table>
<thead>
<tr>
<th>s1</th>
<th>v1</th>
<th>a2</th>
<th>v2</th>
<th>a3</th>
<th>v3</th>
<th>dec</th>
<th>decvalue</th>
<th>inf</th>
<th>sup</th>
<th>inacc</th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>50</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>110.000</td>
<td>100.000</td>
<td>10.000</td>
<td></td>
</tr>
<tr>
<td>age</td>
<td>50</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>45.000</td>
<td>0.000</td>
<td>7.000</td>
<td></td>
</tr>
<tr>
<td>sex</td>
<td>60</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>125.000</td>
<td>100.000</td>
<td>15.000</td>
<td></td>
</tr>
<tr>
<td>sex</td>
<td>60</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>75.000</td>
<td>0.000</td>
<td>10.000</td>
<td></td>
</tr>
<tr>
<td>sex</td>
<td>60</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>97.000</td>
<td>0.000</td>
<td>12.000</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 13. Total contents in cimpli2 for Mammographic data set.

\(sup(\tau)\), \textit{NIS-Apriori} in SQL executes the total search in the NRDF data set. As we have described, the most complicated part is to add \textit{inf}, \textit{sup}, \textit{inacc}, and \textit{outacc} information to the Cartesian Products. For this part, we need to employ the total search of the NRDF data set instead of manipulating \textit{inf}(\tau) and \textit{sup}(\tau), but we can escape from the manipulation on the large size lists. In the application of \textit{NIS-Apriori} in SQL to Mammographic data set, we obtained the same result by RNA in Prolog. However, the execution time by the implemented \textit{NIS-Apriori} in SQL was not good. It took about 1 (min) for Step 1. It is necessary to revise the current procedure, especially the generation of \textit{impli1}, \textit{cimpli2}, and \textit{plimpli2}.

4 Concluding Remarks

This paper briefly described the background of RNA for handling information incompleteness in table data, and we newly focused on SQL system for handling large size data sets. As for this prototype, we have the following consideration.
(1) Since SQL has the high versatility, \textit{NIS-Apriori} in SQL will offer the useful environment for analyzing tables with non-deterministic values.
(2) Both RNA in Prolog and \textit{getRNA} in Python internally store a list for each implication. For large size data sets, the manipulation of these lists will be a heavy load. On the other hand, \textit{NIS-Apriori} in SQL does not employ such lists, but it employs the total search of the data sets. In two strategies, i.e., the list manipulation strategy and the total search strategy, we figure out that the list
manipulation strategy will be suitable to rule generation for small size data sets, and the total search strategy will be suitable to rule generation for large size data sets.

(3) In the prototype, we faithfully simulated the NIS-Apriori algorithm, so the procedures in SQL might generate the meaningless tables. It is necessary to revise this point.

Acknowledgment: The authors would be grateful to the anonymous referees for their useful comments. This work is supported by JSPS (Japan Society for the Promotion of Science) KAKENHI Grant Number 26330277.

References

http://mlearn.ics.uci.edu/MLRepository.html
http://getRNAI.org