Visually Aided Exploration of Interesting Association Rules

Bing Liu, Wynne Hsu, Ke Wang, and Shu Chen

School of Computing
National University of Singapore
Lower Kent Ridge Road, Singapore 119260
{liub, whsu, wangk, chens}@comp.nus.edu.sg
http://www.comp.nus.edu.sg/{~liub, ~whsu, ~wangk}

Abstract. Association rules are a class of important regularities in databases. They are found to be very useful in practical applications. However, the number of association rules discovered in a database can be huge, thus making manual inspection and analysis of the rules difficult. In this paper, we propose a new framework to allow the user to explore the discovered rules to identify those interesting ones. This framework has two components, an interestingness analysis component, and a visualization component. The interestingness analysis component analyzes and organizes the discovered rules according to various interestingness criteria with respect to the user's existing knowledge. The visualization component enables the user to visually explore those potentially interesting rules. The key strength of the visualization component is that from a single screen, the user is able to obtain a global and yet detailed picture of various interesting aspects of the discovered rules. Enhanced with color effects, the user can easily and quickly focus his/her attention on the more interesting/useful rules.

1. Introduction

Association rules, introduced in [2], have received considerable attention in data mining research and applications. The main strengths of association rule mining are that the target of discovery is not pre-determined, and that it is able to find all association rules that exist in the database. Thus, association rules can reveal valuable and unexpected information in the database. However, these strengths are also its weakness, i.e., the number of discovered rules can be huge, in thousands or even tens of thousands, which makes manual inspection of the rules to identify the interesting ones an almost impossible task. Automated assistance is thus needed.

Determining the interestingness of a rule is not a simple task. A rule can be interesting to one person but not interesting to another. The interestingness of a rule is essentially subjective. It depends on the user's existing knowledge about the domain and his/her current interests.

This paper proposes a new interactive and iterative framework to help the user find interesting association rules. The proposed framework consists of two components, an interestingness analysis component and a visualization component. The interestingness analysis component allows the user to specify his/her existing knowledge. It then uses this input knowledge to analyze the discovered rules according to various interestingness criteria, and through such analysis to identify those potentially interesting rules for the user. The visualization component makes it easy for the user to visually ex-

plore the potentially interesting rules. The key strength of the visualization component is that from a single screen, the user is able to obtain a global and yet detailed picture of various interesting aspects of the discovered rules. This enables him/her to visually detect any unusual pattern without the need to browse through a large number of rules. Three main types of information are shown on the screen:
(1) different kinds of potentially interesting rules.
(2) different degrees of rule interestingness and the number of rules in each kind.
(3) interesting items in the conditional part or the consequent part of the rules.
Enhanced with color effects, these types of information can lead the user to easily and quickly explore various aspects of the discovered rules and to focus his/her attention on those truly interesting/useful ones. The whole system works as follows:

Repeat until the user decides to stop
1 the user specifies some existing knowledge or modifies the knowledge specified previously;
2 the system analyzes the discovered rules according to some interestingness criteria;
3 the user inspects the analysis results through the visualization component, saves the interesting rules, and removes those unwanted rules.

2. Association Rules and Subjective Rule Interestingness

2.1 Generalized association rules

Let $I = \{i_1, \ldots, i_w\}$ be a set of items, T be a set of transactions, and G be a set of *taxonomies* or *class hierarchies*. A taxonomy is a directed acyclic graph on the items in I, where an edge represents an *is-a* relationship. A taxonomy example is shown in Fig 1. A *generalized association rule* [15] is an implication of the form $X \to Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The rule $X \to Y$ holds in the transaction set T with confidence c if c% of transactions in T that support X also support Y. The rule has support s in T if s% of the transactions in T contains $X \cup Y$. A transaction t that supports an item in I also supports all its ancestors in I. For example, an association rule could be:

cheese, milk $\to$ Fruit [support = 5%, confidence = 70%],

which says that 5% of people buy cheese, milk and Fruit together, and 70% of the people who buy cheese and milk also buy Fruit (of any kind).

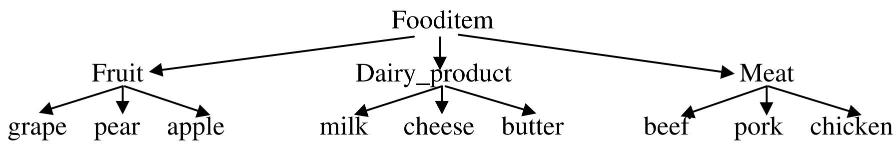

Fig 1. An example taxonomy

2.2 Subjective rule interestingness

Past research has identified two main subjective rule interestingness measures:
Unexpectedness [14, 7]: Rules are interesting if they "surprise" the user.
Actionability [11]: Rules are interesting if the user can do something with them to his/her advantage.

The two measures of interestingness are not mutually exclusive. Interesting rules can be classified into three categories [14]:

1: rules that are both unexpected and actionable,
2: rules that are unexpected but not actionable, and
3: rules that are actionable but expected.

Category 1 and 2 can be handled by finding unexpected rules, and category 3 can be handled by finding the rules that conform to the user's knowledge. This paper proposes a new framework to help the user find these two types of rules, i.e., unexpected rules, and expected rules (or conforming rules).

3. The Interestingness Analysis Component

This component uses the user's existing knowledge to analyze and identify various types of potentially interesting rules from the discovered association rules.

3.1. The specification language

A specification language is designed to enable the user to express his/her existing knowledge. This language focuses on representing the user's existing knowledge about associative relations on items in the database. The basic syntax of the language takes the same format as association rules.

This language has three levels of specifications. Each represents knowledge of a different degree of preciseness. They are: *general impressions*, *reasonably precise concepts*, and *precise knowledge*. The first two levels represent the user's vague feelings. The last level represents his/her precise knowledge. This division is important because a user typically has a mixture of vague and precise knowledge.

The proposed language also uses the idea of class hierarchy (or taxonomy) as in generalized association rules. The hierarchy in Fig 1 can also be represented by:

{grape, pear, apple} ⊂ Fruit ⊂ Fooditem
{milk, cheese, butter} ⊂ Dairy_product ⊂ Fooditem
{beef, pork, chicken} ⊂ Meat ⊂ Fooditem

Fruit, Dairy_product, Meat and Fooditems are classes (or class names). grape, pear, apple, milk, cheese, beef, pork, chicken, #Fruit, #Dairy_product, #Meat and #Fooditems are items. Note that in generalized association rules, class names can also be treated as items, in which case, we append a "#" in front of a class name. Note also that in the proposed language, a class hierarchy does not need to be constructed beforehand. A class can be created any time when needed by using a set of items (see the examples below).

General Impression (GI): It represents the user's vague feeling that there should be some associations among some classes of items, but he/she is not sure how they are associated. This can be expressed with:

gi(<S_1, …, S_m>) [*support*, *confidence*]

where (1) Each S_i is one of the following: an item, a class, or an expression C+ or C*, where C is a class. C+ and C* correspond to one or more, and zero or more instances of the class C, respectively.

(2) A discovered rule: $a_1, \ldots, a_n \rightarrow b_1, \ldots, b_k$, *conforms* to the GI if <$a_1, \ldots, a_n, b_1, \ldots, b_k$> can be considered to be an instance of <S_1, …, S_m>, otherwise it is *unexpected* with respect to the GI.

(3) *Support* and *confidence* are optional. The user can specify the minimum *support* and the minimum *confidence* of the rules that he/she wants to see.

Example 1: The user believes that there exist some associations among {milk, cheese}, Fruit items, and beef (assume we use the class hierarchy in Fig 1). He/she specifies this as:

gi(<{milk, cheese}*, Fruit+, beef>)

{milk, cheese} here represents a class constructed on the fly unlike Fruit. The following are examples of association rules that conform to the specification:

apple → beef
grape, pear, beef → milk

The following two rules are unexpected with respect to this specification:

(1) milk → beef (2) milk, cheese, pear → clothes

(1) is unexpected because Fruit+ is not satisfied. (2) is unexpected because beef is not present in the rule, and clothes is not from any of the elements in the GI.

Reasonably Precise Concept (RPC): It represents the user's concept that there should be some associations among some classes of items, and he/she also knows the direction of the associations. This can be expressed with:

$rpc(<S_1, \ldots, S_m \rightarrow V_1, \ldots, V_g>)$ [*support*, *confidence*]

where (1) Each S_i or V_j is the same as S_i in the GI specification.

(2) A discovered rule, $a_1, \ldots, a_n \rightarrow b_1, \ldots, b_k$, *conforms* to the RPC, if the rule can be considered to be an instance of the RPC, otherwise it is considered as *unexpected* with respect to the RPC.

(3) *Support* and *confidence* are again optional.

Example 2: Suppose the user believes the following:

rpc(<Meat, Meat, #Dairy_product → {grape, apple}+>)

The following are examples of association rules that conform to the specification:

beef, pork, Dairy_product → grape
beef, chicken, Dairy_product → grape, apple

The following two rules are unexpected with respect to the specification:

(1) pork, Dairy_product → grape (2) beef, pork → grape

(1) is unexpected because it has only one Meat item, but two Meat items are needed as we have two Meat's in the specification. (2) is unexpected because Dairy_product is not in the conditional part of the rule.

Precise knowledge (**PK**): The user believes in a precise association. This is expressed with:

$pk(<S_1, \ldots, S_m \rightarrow V_1, \ldots, V_g>)$ [*support*, *confidence*]

where (1) Each S_i or V_j is an item in I.

(2) A discovered rule: $a_1, \ldots, a_n \rightarrow b_1, \ldots, b_k$ [*sup*, *confid*], is *equal* to the PK, if the rule part is the same as $S_1, \ldots, S_m \rightarrow V_1, \ldots, V_g$. Whether it *conforms* to the PK or is *unexpected* depends on the support and confidence specifications.

(3) *Support* and *confidence* need to be specified (they are not optional).

Example 3: Suppose the user believes the following:

pk(<#Meat, milk → apple>) [10%, 50%]

The discovered rule below conforms to the PK quite well because the supports and confidences of the rule and the PK are quite close.

Meat, milk → apple [8%, 53%]

However, if the discovered rule is the following:

Meat, milk → apple [4%, 30%]

then it is less conforming, but more unexpected, because its support and confidence are quite far from those of the PK.

3.2. Analyzing the discovered rules using user's existing knowledge

We now present how to use the user's specifications to analyze the discovered rules. For GIs and RPCs, we only perform syntax-based analysis, i.e., comparing the syntactic structure of the discovered rules with GIs and RPCs. It does not make sense to do semantics-based analysis because the user does not have precise associations in mind. Using PKs, we can perform semantics-based analysis (based on support and confidence) on the discovered rules. Due to space limitations, we could not present this in the paper (see [9] for details).

Let U be the set of user's specifications representing his/her knowledge space. Let A be the set of discovered association rules. The proposed technique analyzes the discovered rules by "matching" and ranking the rules in A in a number of ways for finding different kinds of interesting rules, *conforming rules*, *unexpected consequent rules*, *unexpected condition rules* and *both-side unexpected rules*.

Conforming rules: A discovered rule $A_i \in A$ conforms to a piece of user's knowledge $U_j \in U$ if both the conditional and consequent parts of A_i match $U_j \in U$ well. We use $confm_{ij}$ to denote the degree of *conforming match*.

Purpose: conforming rules show us those discovered rules that conform to or are consistent with our existing knowledge fully or partially.

Unexpected consequent rules: A discovered rule $A_i \in A$ has unexpected consequents with respect to a $U_j \in U$ if the conditional part of A_i matches U_j well, but not the consequent part. We use $unexpConseq_{ij}$ to denote the degree of *unexpected consequent match*.

Purpose: unexpected consequent rules show us those discovered rules that may be contrary to our existing knowledge. These rules are often very interesting.

Unexpected condition rules: A discovered rule $A_i \in A$ has unexpected conditions with respect to a $U_j \in U$ if the consequent part of A_i matches U_j well, but not the conditional part. We use $unexpCond_{ij}$ to denote the degree of *unexpected condition match*.

Purpose: unexpected condition rules show us that there are other conditions that can lead to the consequent of the specification. We are thus guided to explore unfamiliar territories.

Both-side unexpected rules: A discovered rule $A_i \in A$ is both-side unexpected with respect to a $U_j \in U$ if both the conditional and consequent parts of the rule A_i do not match U_j well. We use $bsUnexp_{ij}$ to denote the degree of *both-side unexpected match*.

Purpose: both-side unexpected rules remind us that there are other rules whose conditions and consequents are not mentioned in our specification. It helps us to go beyond our existing concept space.

The values for $confm_{ij}$, $unexpConseq_{ij}$, $unexpCond_{ij}$, and $bsUnexp_{ij}$ are between 1.00 and 0. 1.00 represents the complete match, either the complete conforming or the complete unexpectedness match, and 0 represents no match. Let L_{ij} and R_{ij} be the degrees of condition and consequent match of rule A_i against U_j respectively. We have (for both GIs and RPCs),

$$confm_{ij} = L_{ij} * R_{ij};\qquad unexpConseq_{ij} = \begin{cases} 0 & L_{ij} - R_{ij} \le 0 \\ L_{ij} - R_{ij} & L_{ij} - R_{ij} > 0 \end{cases};$$

$$unexpCond_{ij} = \begin{cases} 0 & R_{ij} - L_{ij} \le 0 \\ R_{ij} - L_{ij} & R_{ij} - L_{ij} > 0 \end{cases};$$

$$bsUnexp_{ij} = 1 - max(confm_{ij}, unexpConseq_{ij}, unexpCond_{ij});$$

We use $L_{ij} - R_{ij}$ to compute the unexpected consequent match degree because we wish to rank those rules with high L_{ij} but low R_{ij} higher. Similar idea applies to $unexpCond_{ij}$. The formula for $bsUnexp_{ij}$ ensures that those rules with high values in any other three categories should have low values here, and vice versa.

Due to the space limitation, we are unable to give the detailed computation methods for L_{ij} and R_{ij}, which depend on whether U_j is a GI or a RPC. The computations can all be done efficiently. See [9] for full details. After $confm_{ij}$, $unexpConseq_{ij}$, $unexpCond_{ij}$, and $bsUnexp_{ij}$ have been computed, we can rank the discovered rules with respect to a U_j. It is also possible to rank the rules with respect to the whole set of specifications U. However, in our applications, we find that such rankings can be quite confusing, and are thus omitted.

4. The Visualization Component

After the discovered rules are analyzed with the method presented in the last section, we want to display those different types of potentially interesting rules to the user. The issue here is how to show the essential aspects of the rules such that we can take advantage of the human visual capabilities to allow the user to identify the truly interesting rules easily and quickly. Let us discuss what the essential aspects are:

1. Types of potentially interesting rules: We should separate them because different types of interesting rules give the user different information.
2. Degrees of interestingness ("match" values): We should group rules according to their degrees of interestingness. This enables the user to focus his/her attention on the most unexpected (or conforming) rules first and to decide whether to view those rules with low degrees of interestingness.
3. Interesting items: We focus on showing the interesting items rather than the rules. This is perhaps the most crucial decision that we have made. In our applications, we find that it is those unexpected items that are most important to the user because due to 1 above, the user already knows what kind of interesting rules he/she is looking. For example, when the user is looking at unexpected consequent rules, it is natural that the first thing he/she wants to know is what are the unexpected items in the consequent parts. Even if we show the rules, the user still needs to look for the unexpected items in the rules.

The main screen in the visualization system contains all the above information. Below, we use an example to describe the visualization system.

The visualization system consists of 4 main modules:

1. *Class hierarchy builder*: it allows the user to build class hierarchies as in Fig 1.
2. *GI viewer*: it allows the user to specify GIs and to visualize the results produced by the interestingness analysis system.
3. *RPC viewer*: it allows the user to specify RPCs and to visualize the results produced by the interestingness analysis system.
4. *PK viewer*: it allows the user to specify PKs and to visualize the results produced by the interestingness analysis system.

Here, we only focus on presenting the *RPC viewer*. Due to space limitations, we are unable to show the others. They are similar in concept to the *RPC viewer*. We will also not discuss the *Class hierarchy builder* since it is straightforward.

4.1. The example setting

Our example uses a RPC specification. The rules in the example are a small subset of rules (857 rules) discovered in an exam results database. This application tries to discover the associations between the exam results of a set of 7 specialized courses (called GA courses) and the exam results of a set of 7 basic courses (called GB courses). A course together with an exam result form an item, e.g., GA6-1, where GA6 is the course code and "1" represents a bad exam grade ("2" represents an average grade and "3" a good grade). The discovered rules and our existing concept specification are listed below.

- **Discovered association rules:** The rules below have only GA course grades on left-hand-side and GB course grades on right-hand-side (we omit their support and confidence).

R1:	GA1-3 → GB2-3	R7:	GA4-1 → GB7-2
R2:	GA4-3 → GB4-3	R8:	GA6-2 → GB7-2
R3:	GA2-3 → GB2-3	R9:	GA5-1, GA2-2 → GB2-2
R4:	GA2-3 → GB5-1	R10:	GA5-2, GA1-2 → GB3-2
R5:	GA6-1 → GB1-3	R11:	GA6-1, GA3-3 → GB6-3
R6:	GA4-2 → GB3-3	R12:	GA7-2, GA3-3 → GB4-3

- Our existing concept specification
  Assume we have the common belief that students good in GA courses are likely to be good in GB courses. This can be expressed as a RPC (also see it in Fig 2):
  Spec1: *rpc*(GA-good → GB-good)
  where the classes, GA-good and GB-good, are defined as follows:
  GA-good ⊃ {GA1-3, GA2-3, GA3-3, GA4-3, GA5-3, GA6-3, GA7-3}
  GB-good ⊃ {GB1-3, GB2-3, GB3-3, GB4-3, GB5-3, GB6-3, GB7-3}

4.2. Viewing the results

After running the system with the above RPC specification, we obtain the screen in Fig 2 (the main screen). We see "RPC" in the middle. To the bottom of "RPC", we have the *conforming rules visualization unit*. To the left of "RPC", we have the *unexpected condition rules visualization unit*. To the right, we have the *unexpected consequent rules visualization unit*. To the top, we have *both-side unexpected rules visualization unit*. Below, we briefly discuss these units in turn with the example.

Conforming rules visualization unit: Clicking on ***Conform***, we will see the complete conforming rules ranking in a pop-up window:

Rank 1:	1.00	R1	GA1-3 → GB2-3
Rank 1:	1.00	R2	GA4-3 → GB4-3
Rank 1:	1.00	R3	GA2-3 → GB2-3
Rank 2:	0.50	R11	GA6-1, GA3-3 → GB6-3
Rank 2:	0.50	R12	GA7-2, GA3-3 → GB4-3

The number (e.g., 1.00, and 0.50) after each rank number is the conforming match value, $confm_{i1}$. The first three rules conform to our belief completely. The last two only conform to our belief partially because GA6-1 and GA7-2 are unexpected. This list of rules can be long in a real-life application. The following mechanisms help the user focus his/her attention, i.e., enabling him/her to view rules with different degrees of interestingess ("match" values) and to view the interesting items.

- On both sides of ***Conform*** we can see 4 pairs of boxes, which represent sets of rules with different conforming match values. If a pair of boxes is colored, it means that there are rules there, otherwise there is no rule. The line connecting "RPC" and a pair of colored boxes also indicates that there are rules under them. The number of rules is shown on the line. Clicking on the box with a value will give all the rules with the corresponding match value and above. For example, clicking on 0.50 shows the rules with $0.50 \leq confm_{i1} < 0.75$. Below each colored box with a value, we have two small windows. The one on the top has all the rules' condition items from our RPC specification, and the one at the bottom has all the consequent items. Clicking on each item gives us the rules that use this item as a condition item (or a consequent item).
- Clicking on the colored box without a value (below the valued box) brings us to a new screen (not shown here). From this, the user sees all the items in different classes involved, and also conforming and unexpected items.

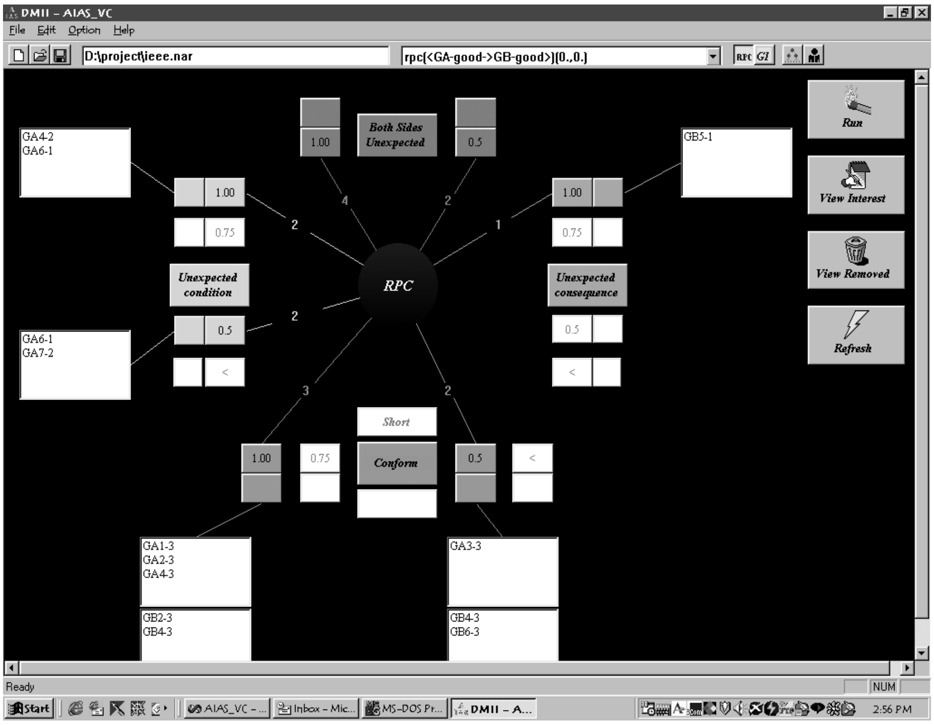

Fig 2. RPC main visualization screen

Unexpected condition rules visualization unit: The boxes here have similar meanings as the ones for conforming rules. From Fig 2, we see that there are 4 unexpected condition rules. Two have the unexpected match value of 1.00 and two have 0.50. The window (on the far left) connected to the box with a match value gives all the unexpected condition items. Clicking on each item reveals the relevant rules. Similarly, clicking on the colored box next to the one with a value shows both the unexpected condition items and the items used in the consequent part of the rules. To obtain all the rules in the category, we can click ***Unexpected Conditions***.

Rank 1:	1.00	R5	GA6-1 $\rightarrow$ GB1-3
Rank 1:	1.00	R6	GA4-2 $\rightarrow$ GB3-3
Rank 2:	0.50	R11	GA6-1, GA3-3 $\rightarrow$ GB6-3
Rank 2:	0.50	R12	GA7-2, GA3-3 $\rightarrow$ GB4-3

1.00 and 0.50 are the $unexpCond_{i1}$ values. Here, we see something quite unexpected. For example, many students with bad grades in GA6 actually have good grades in GB1.

Unexpected consequent rules visualization unit: This is also similar to the conforming rules visualization unit. From Fig 2, we see that there is only one unexpected consequent rule and the unexpected consequent match value is 1.00. Clicking on the colored box with 1.00, we will obtain the unexpected consequent rule:

Rank 1: 1.00 R4 GA2-3 $\rightarrow$ GB5-1

This rule is very interesting because it contradicts our belief. Many students with good grades in GA2 actually have bad grades in GB5.

Both-side unexpected rules visualization unit: We only have two unexpected match value boxes here, i.e., 1.00 and 0.50. Due to the formulas in Section 3.2, rules with $bsUnexp_{ij}$ < 1.00 can actually all be seen from other visualization units. The unexpected items can be obtained by clicking on the box above the one with a value. All the ranked rules can be obtained by clicking ***Both Sides Unexpected***.

Rank 1:	1.00	R7	GA4-1 → GB7-2
Rank 1:	1.00	R8	GA6-2 → GB7-2
Rank 1:	1.00	R9	GA5-1, GA2-2 → GB2-2
Rank 1:	1.00	R10	GA5-2, GA1-2 → GB3-2
Rank 2:	0.50	R11	GA6-1, GA3-3 → GB6-3
Rank 2:	0.50	R12	GA7-2, GA3-3 → GB4-3

From this ranking, we also see something quite interesting, i.e., average grades lead to average grades and bad grades lead to average grades. Some of these rules are common sense, e.g., average to average rules (R8 and R10), but we did not specify them as our existing knowledge (if "average to average" had been specified as our knowledge earlier, these rules would not have appeared here because they would have been removed). This shows the advantage of our technique, i.e., *it can remind us what we have forgotten if the rules are not truly unexpected*.

The system also allows the user to incrementally save interesting rules and remove unwanted rules, and to view them. Whenever a rule is removed or saved (also removed from the original set of rules), the related pictures and windows are updated.

The proposed system has proven to be very useful in a number of applications. In these applications, there are typically thousands of discovered association rules (the smallest rule set has 770 rules). Without the proposed system, it would be very hard for us to analyze these large numbers of rules.

5. Related Work

Traditionally, a query-based approach is used to help the user identify or generate interesting rules. The approach takes many forms, e.g., templates [6], M-SQL [5], DMQL [4], and action hierarchy [1]. Although query languages can be quite different, a query basically defines a set of rules of a certain type (or constraints on the rules to be found). To "execute" a query means to find all rules that satisfy the query. We believe that the query-based approach is insufficient for two main reasons:

1. It is hard to find the truly unexpected rules. It only finds those anticipated rules because what the user's queries are still within his/her existing knowledge space.
2. The user often does not know or is unable to specify completely what interest him/her. He/she needs to be stimulated. The query-based approach does not actively perform this task because it only returns those rules that satisfy the queries.

Our technique not only finds those conforming rules like query-based methods, but also provides three types of unexpected rules. Our approach also helps the user to provide more knowledge to the system by reminding him/her what he/she might have forgotten. If the top ranking rules are not unexpected, then they serve to remind the user what he/she has forgotten. Our visualization component allows the user to easily and quickly explore those interesting rules.

[7, 8] report a related technique for analyzing classification rules [13] using user's existing concepts. However, the technique there cannot be used for analyzing association rules. Association rules require a different specification language and different ways of analyzing and ranking the rules. [7, 8] also do not have visualization systems.

[12] proposes a method of discovering unexpected rules in the rule generation

phase by taking into consideration the user's expectations. This method is, however, not as efficient and flexible as our post-analysis method because the user is normally unable to specify his/her expectations about the domain completely. User interaction with the system is needed in order for him/her to provide a more complete set of expectations and to find more interesting rules. However, user interaction is difficult for the approach in [12] because it is not efficient to run a rule miner whenever the user remembers another piece of knowledge. The association rule mining is typically very time consuming. Post-analysis facilitates user interaction due to its efficiency.

6. Conclusion

This paper proposes an integrated framework for exploration of discovered rules in order to find those interesting ones. The interestingness analysis system uses three types of user's existing knowledge to analyze the discovered rules and to organize them in various ways to expose the user to many interesting aspects of the discovered rules. A simple but powerful visualization system enables the user to view and identify interesting rules easily and quickly.

References

[1] Adomavicius, G. and Tuzhilin, A. "Discovery of actionable patterns in databases: the action hierarchy approach." *KDD-97*, 1997, pp. 111-114.

[2] Agrawal, R., Imielinski, T. and Swami, A. Mining association rules between sets of items in large databases. *SIGMOD-1993*, 1993, pp. 207-216.

[3] Fayyad, U., Piatesky-Shapiro, G. and Smyth, P. "From data mining to knowledge discovery: an overview," In: *Advances in knowledge discovery and data mining*, U. Fayyad, G. Piatesky-Shapiro, P. Smyth and R. Uthurusamy, (eds.), AAAI/MIT Press, 1996, pp. 1-34.

[4] Han, J., Fu, Y., Wang, W., Koperski, K. and Zaiane, O. "DMQL: a data mining query language for relational databases." *SIGMOD Workshop on KDD*, 1996.

[5] Imielinski, T., Virmani, A. and Abdulghani, A. "DataMine: application programming interface and query language for database mining." *KDD-96*, 1996.

[6] Klemetinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A.I. "Finding interesting rules from large sets of discovered association rules." *CIKM-94*, 1994, pp. 401-407.

[7] Liu, B., and Hsu, W. "Post-analysis of learned rules." *AAAI-96*, 1996.

[8] Liu, B., Hsu, W., and Chen, S. "Using general impressions to analyze discovered classification rules." *KDD-97*, 1997, pp. 31-36.

[9] Liu, B., Hsu, W., and Wang, K. "Helping user identifying interesting association rules" *Technical Report,* 1998.

[10] Liu, B., Hsu, W. and Ma, Y. M. "Integrating classification and association rule mining." *KDD-98*, 1998, pp. 80-86.

[11] Piatesky-Shapiro, G., and Matheus, C. "The interestingness of deviations." *KDD-94*, 1994.

[12] Padmanabhan, B., and Tuzhilin, A. "A belief-driven method for discovering unexpected patterns." *KDD-98*, 1998, pp. 94-110.

[13] Quinlan, J. R. *C4.5*: *program for machine learning*. Morgan Kaufmann, 1992.

[14] Silberschatz, A., and Tuzhilin, A. "What makes patterns interesting in knowledge discovery systems." *IEEE Trans. on Know. and Data Eng.* 8(6), 1996.

[15] Srikant, R. and Agrawal, R. "Mining Generalized association rules." *VLDB-1995*, 1995.

[16] Srikant, R., Vu, Q. and Agrawal, R. "Mining association rules with item constraints." *KDD-97*, 1997, pp. 67-73.