Analyzing the Subjective Interestingness of Association Rules

Bing Liu, Wynne Hsu, Shu Chen and Yiming Ma

School of Computing
National University of Singapore
3 Science Drive 2
Singapore 117543

{liub, whsu, maym}@comp.nus.edu.sg

Abstract

Association rules are a class of important regularities in databases. They are found to be very useful in practical applications. However, association rule mining algorithms tend to produce a huge number of rules, most of which are of no interest to the user. Due to the large number of rules, it is very difficult for the user to analyze them manually in order to identify those truly interesting ones. In this paper, we propose a new approach to assist the user in finding interesting rules (in particular, unexpected rules) from a set of discovered association rules. This technique is characterized by analyzing the discovered association rules using the user's existing knowledge about the domain and then ranking the discovered rules according to various interestingness criteria, e.g., conformity and various types of unexpectedness. This technique has been implemented and successfully used in a number of applications.

Keywords: subjective interestingness, association rules, interestingness analysis in data mining.

1. Introduction

The interestingness issue has long been identified as an important problem in data mining. It refers to finding rules that are interesting/useful to the user, not just any possible rule [e.g., 1, 11, 12, 21, 23, 24, 27, 30]. The reason for its importance is that, in practice, it is all too easy for a data mining algorithm to discover a glut of rules, and most of these rules are of no interest to the user [11, 12, 21, 27, 30]. This is particularly true for association rule mining [e.g., 2, 3, 7, 14, 16, 28], which often produces a huge number of rules. The huge number of rules makes manual inspection of the rules very difficult. Automated assistance is needed. This paper presents an interestingness analysis system (IAS) to help the user identify interesting association rules.

1.1 Rule interestingness measures

Past research in data mining has shown that the interestingness of a rule can be measured using objective measures and subjective measures [e.g., 27, 11]. Objective measures involve analyzing the rule's structure, predictive performance, and statistical significance [e.g., 27, 21, 17, 14, 2, 3]. In association rule mining, such measures include *support* and *confidence* [2, 28, 3]. However, it is noted in [21] that objective measures are insufficient for determining the interestingness of a discovered rule. Subjective measures are needed. Subjective interestingenss is the topic of this paper. Two main subjective interestingness measures are: unexpectedness [11, 27] and actionability [21, 27].

- *Unexpectedness*: Rules are interesting if they are unknown to the user or contradict the user's existing knowledge (or expectations).
- *Actionability*: Rules are interesting if the user can do something with them to his/her advantage.

Although both unexpectedness and actionability are important, actionability is the key concept in most applications because actionable rules allow the user to do his/her job better by taking some specific actions in response to the discovered knowledge [21, 27]. Actionability is, however, an elusive concept because it is not feasible to know the space of all rules and the actions to be attached to them [27]. Fortunately, the two measures are not mutually exclusive. Interesting rules can be classified into three categories: (1) rules that are both unexpected and actionable; (2) rules that are unexpected but not actionable; and (3) rules that are actionable but expected.

In this research, we only focus on unexpectedness. Actionability is partially handled through unexpectedness because actionable rules are either expected or unexpected. Thus, the proposed technique aims to find expected and unexpected association rules. Expected rules are also called conforming rules as they conform to the user's existing knowledge or expectations.

1.2 Generalized association rules

Before discussing our proposed technique, let us first introduce the concept of association rules, in particular, *generalized association rules* [28]. The generalized association rule model is more general than the original association rule model given in [2].

The (generalized) association rule mining is defined as follows: Let $I = \{i_1, \ldots, i_w\}$ be a set of items. Let G be a directed acyclic graph on the items. An edge in G represents an is-a relationship. Then, G is a set of *taxonomies*. A taxonomy example is shown in Figure 1. Let T be a set of transactions, where each transaction t is a set of items such that $t \subseteq I$. A (*generalized*) *association rule* is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. The rule $X \rightarrow Y$ 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 $X \cup Y$.

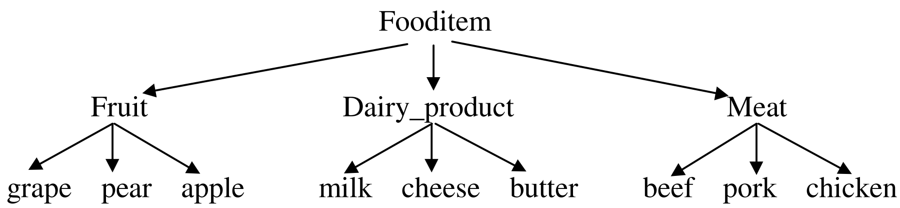

Figure 1: An example taxonomy

For example, an association rule could be:

grape $\rightarrow$ apple [support = 10%, confidence = 60%],

which says that 10% of people buy *grape* and *apple* together, and 60% of the people who buy *grape* also buy *apple*. This rule only involves items at the bottom level of the taxonomy. We can also have rules that involve items of more than one level. For example, Fruit → Dairy_product [support = 2%, confidence = 67%]

Fruit, milk → Meat [support = 3%, confidence = 65%]

1.3 Summary of the proposed technique

The basic idea of our technique is as follows: The system first asks the user to specify his/her existing knowledge, e.g., beliefs or concepts, about the domain. It then analyzes the discovered rules to identify those potentially interesting ones (e.g., unexpected rules). The proposed technique is an interactive and iterative post-processing technique (see Section 3). It consists of three components:

1. *A specification language*: it allows the user to specify his/her various types of existing knowledge.
2. *An interestngness analysis system*: it analyzes the discovered association rules using the user's specifications, and through such analysis, to identifiy: *conforming rules*, *unexpected consequent rules*, *unexpected condition rules* and *both-side unexpected rules*.
3. *A visualization system*: it enables the user to visually detect interesting rules easily.

The proposed technique has been implemented and successfully applied to a number of applications. The system is called IAS. It can be downloaded from: http://www.comp.nus.edu.sg/~dm2.

The paper is organized as follows: In the next section, we discuss the related work. Section 3 presents the proposed technique. Section 4 describes the visualization system using an example. Section 5 evaluates the proposed technique. Section 6 concludes the paper.

2 Related Work

Existing research in rule interestingness focuses on either objective interestingness or subjective interestingness. Objective interestingness analyzes rules' structure, predictive performance, statistical significance, etc [e.g., 2, 3, 14, 16, 18, 22, 25, 31, 32]. Objective interestingness will not be discussed further, as it is not the focus of this paper. This paper studies subjective interestingness. We assume that objective interestingness analysis [14, 3, 32] has been performed to remove those redundant and/or insignificant rules.

Most existing approaches to finding subjectively interesting association rules ask the user to explicitly specify what types of rules are interesting and uninteresting. The system then generates or retrieves those matching rules. [10] proposes a template-based approach. In this approach, the user specifies interesting and uninteresting association rules using templates. A template describes a set of rules in terms of items occurred in the conditional and the consequent parts. The system then retrieves the matching rules from the set of discovered rules.

[29] proposes an association rule mining algorithm that can take item constraints specified by the user in the rule mining process so that only those rules that satisfy the constraints are generated. [20] extends this approach further to allow much more sophisticated constraints to be specified by the user. It also uses the constraints to optimize the association rule mining process. The idea of using constraints in the rule mining process is important as it avoids generating irrelevant rules.

Along the similar line, there are also a number of works based on data mining queries. For example, M-SQL in [8], DMQL in [7], and Metaqueries in [26]. A data mining query basically defines a set of rules of a certain type (or constraints on the rule to be found). To "execute" a query means to find all rules that satisfy the query.

All the above methods view the process of finding subjectively interesting rules as a *query-based process*, although the queries may be considered during rule generation or after all rules have been discovered. Query-based methods have the following problems.

1. It is hard to find the truly unexpected rules. They can only find those anticipated rules because queries can only be derived from the user's existing knowledge space. Yet, many rules that do not satisfy the user's queries may also be of interest. It is just that the user has never thought of them (they are unexpected or novel) or has forgotten about them.
2. The user often does not know or is unable to specify completely what interest him/her. He/she needs to be stimulated or reminded. Query-based approaches do not actively perform this task because they only return those rules that satisfy the queries.

Our proposed technique not only identifies those conforming rules as query-based methods, but also provides three types of unexpected rules. Thus, the user is exposed to more possible interesting aspects of the discovered rules rather than only focusing on his/her current interests (which he/she may not be sure). If the unexpected rules are not truly unexpected, they serve to remind the user what he/she has forgotten. IAS's visualization system also helps the user explore interesting rules easily.

In [11, 12], we reported two techniques for analyzing the subjective interestingness of classification rules. However, those techniques cannot be applied to analyzing association rules. Association rules require a different specification language and different ways of analyzing and ranking the rules.

[23, 24] proposes a method of discovering unexpected patterns that takes into consideration a set of expectations or beliefs about the problem domain. The method discovers unexpected patterns using these expectations to seed the search for patterns in data that contradict the beliefs. However, this method is in general not as efficient and flexible as our post-analysis method unless the user is able to specify his/her beliefs or expectations about the domain completely beforehand, which is very difficult, if not impossible [4, 5]. Typically, 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. Our post-analysis method facilitates user-interaction because of its efficiency. The approach given in [23, 24] also does not handle user's rough or vague feelings, but only precise knowledge (see Section 3.1). User's vague feelings are important for identifying interesting rules because in our applications we found that the user is more likely to have such forms of knowledge than precise knowledge. The definitions of vague feelings and precise knowledge will be given in Section 3.1.

The system WizWhy [31] also has a method to produce unexpected rules. Its method, however, is based on objective interestingness as its analysis does not depend on individual users. It first computes the expected probability of a rule assuming independence of each of its conditions. It then compares this expected probability with the rule's actual probability to compute its unexpectedness.

[27] proposes to use belief systems to describe unexpectedness. A number of formal approaches to the belief systems are presented, e.g., Bayesian probability and Dempster-Shafer theory. These approaches require the user to provide complex belief information, such as conditional probabilities, which are difficult to obtain in practice.

There are also existing techniques that work in the contexts of specific domains. For example, [21] studies the issue of finding interesting deviations in a health care application. Its data mining system, KEFIR, analyzes health care information to uncover "key findings". A domain expert system is constructed to evaluate the interestingness (in this case, actionability) of the "key findings". The approach is, however, application specific. It also does not deal with association rules. Our method is general. It does not make any domain-specific assumptions.

3. IAS: Interestingness Analysis System

We now present IAS. Basically, IAS is an interactive and iterative technique. In each iteration, it first asks the user to specify his/her existing knowledge about the domain. It then uses this knowledge to analyze the discovered rules according to some interestingness criteria, conformity and various types of unexpectedness, and through such analysis to identify those potentially interesting rules. The IAS 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 their conformity and unexpectedness;
3 the user inspects the analysis results through the visualization system, saves the interesting rules, and removes those unwanted rules.

3.1. The specification language

IAS has a simple specification language 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. It is intuitive and simple, which is important for practical applications.

The language allows three types of specifications. Each represents knowledge of a different degree of preciseness. They are:

- *general impressions*,
- *reasonably precise concepts*, and
- *precise knowledge*.

The first two types of knowledge represent the user's vague feelings. The last type represents his/her precise knowledge. This division is important because human knowledge has *granularities*. It is common that some aspects of our knowledge about a domain are quite vague, while other aspects are very precise. For example, we may have a vague feeling or impression that some *Meat* items and *Fruit* items should be associated, but have no idea what items are involved and how they are associated. However, we may know precisely from past experiences or a previous data mining session that buying *bread* implies buying *milk* with a support of around 10% and confidence of around 70%.

It is crucial to allow different types of knowledge to be specified. This not only determines how we can make use of the knowledge, but also whether we can make use of all possible knowledge from the user. For example, if a system can only handle precise knowledge, then the user who does not have precise knowledge but has only vague impressions cannot use it.

The proposed specification language also make use of the idea of class hierarchy (or taxonomy), which is the same as the one used in generalized association rules [28]. We represent the hierarchy in Figure 1 as follows:

{grape, pear, apple} $\subset$ Fruit $\subset$ Fooditem

{milk, cheese, butter} $\subset$ Dairy_product $\subset$ Fooditem

{beef, pork, chicken} $\subset$ Meat $\subset$ 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, but can be created on the fly when needed.

We now discuss the three types of knowledge that the user may input. The next sub-section shows how these types of knowledge are used in finding conforming and unexpected rules.

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:

$gi(<S_1, \ldots, S_m>)$ [*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 $<S_1, \ldots, S_m>$, otherwise it is *unexpected* with respect to the GI.

(3) This impression actually represents a disjunctive propositional formula. Each disjunct is an implication. For example, "$gi(<a, \{b, c\}+>)$" can be expanded into the following (note that {b, c} is treated as a constructed class without a name):

$$(a \rightarrow b) \vee (a \rightarrow c) \vee (b \rightarrow a) \vee (c \rightarrow a) \vee$$
$$((a \wedge b) \rightarrow c) \vee ((a \wedge c) \rightarrow b) \vee ((b \wedge c) \rightarrow a) \vee$$
$$(a \rightarrow (b \wedge c)) \vee (b \rightarrow (a \wedge c)) \vee (c \rightarrow (a \wedge b))$$

A discovered association rule *conforms* to the impression if the rule is one of the disjuncts. We can see that the formula is much more complex than the GI.

(4) *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: The user believes that there exist some associations among {milk, cheese}, Fruit items, and beef (assume we use the class hierarchy in Figure 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 of the GI specification.

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) 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 *unexpected* with respect to the RPC.

(3) Similar to a GI, an RPC also represents a complex disjunctive propositional formula.

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

Example 2: Suppose the user believes the following:

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

Note that #Dairy_product here refers to an item, not a class. 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 association rules are unexpected with respect to the specification:

(1) pork, Dairy_product → grape

(2) beef, pork → grape

(3) beef, pork → milk

(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. (3) is unexpected because Dairy_product is not in the conditional part of the rule, and *milk* is not in the consequent of the RPC specification.

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.

(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 (not optional).

Example 3: Suppose the user believes the following:

pk(<#Meat, milk → apple>) [10%, 30%]

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%, 33%]

However, if the discovered rule is the following:

Meat, milk → apple [1%, 10%]

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

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

After the existing knowledge of the user is specified, the system uses it to analyze the discovered rules. For GIs and RPCs, we 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 any precise associations in mind. Using PKs, we can perform semantics-based analysis, i.e., to perform support and confidence comparisons of the user's specifications against the discovered rules that are equal to the specifications. This process is quite straightforward and will not be discussed here. See [15] for details.

Let U be the set of user's specifications representing his/her knowledge space, A be the set of discovered association rules. The proposed technique "matches" and ranks the rules in A in a number of ways for finding different types of interesting rules, *conforming rules*, *unexpected consequent rules*, *unexpected condition rules* and *both-side unexpected rules*. Below, we define them intuitively and explain the purposes they serve. The computation details will follow.

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 those of $U_j \in U$ well. We use $confm_{ij}$ to denote the degree of *conforming match*.

Purpose: ranking of conforming rules shows us those 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 that of U_j well, but not the consequent part. We use $unexpConseq_{ij}$ to denote the degree of *unexpected consequent match*.

Purpose: ranking of unexpected consequent rules shows us those discovered rules that are contrary to our existing knowledge (fully or partially). 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 that of U_j well, but not the conditional part. We use $unexpCond_{ij}$ to denote the degree of *unexpected condition match*.

Purpose: ranking of unexpected condition rules shows us that there are other conditions which can lead to the consequent of our specified knowledge. We are thus guided to explore the unfamiliar territories, i.e., other associations that are related to our existing knowledge.

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 those of U_j well. We use $bsUnexp_{ij}$ to denote the degree of *both-side unexpected match*.

Purpose: ranking of both-side unexpected rules reminds us that there are other rules whose conditions and consequents have never been mentioned in our specification(s). It helps us to go beyond our existing concept space.

The values for $confm_{ij}$, $unexpConseq_{ij}$, $unexpCond_{ij}$, and $bsUnexp_{ij}$ are between 0 and 1. 1 represents a complete match, either a complete conforming or a 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. $confm_{ij}$, $unexpConseq_{ij}$, $unexpCond_{ij}$, and $bsUnexp_{ij}$ are computed as follows:

$$confm_{ij} = L_{ij} * R_{ij};$$

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

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

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

Note that 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 $unexpectCond_{ij}$. The formula for $bsUnexp_{ij}$ is basically to make sure that those rules with high values in any other three categories should have lower values here, and vice versa.

We now show how to compute L_{ij} and R_{ij} for GI and RPS specifications. Let $I = \{i_1, \ldots, i_w\}$ be the set of items in the database. Let LN_i and RN_i be the total numbers of items in the conditional and consequent parts of A_i respectively. Let the discovered rule A_i be

$$a_1, \ldots, a_n \rightarrow b_1, \ldots, b_k$$

1. U_j is a *general impression* (GI):

$gi(<S_1, \ldots, S_m>)$

Let SN_j be the total number of elements in the GI, where a class with a "*", i.e., C*, is not counted. Let LM_{ij} and RM_{ij} be the numbers of items in the conditional and consequent parts of A_i that match $\{S_1, \ldots, S_m\}$ respectively. Let SM_{ij} be the number of elements in $\{S_1, \ldots, S_m\}$ that have been matched by A_i (again, matching with a C* is not counted).

We define that an item $a_p \in \{a_1, \ldots, a_n\}$ matches $S_q \in \{S_1, \ldots, S_m\}$ (the same applies to $b_p \in \{b_1, \ldots, b_k\}$ and S_q match) as follows:

(i) if $S_q \in I$ and $a_p = S_q$, or

(ii) if S_q is a class C, and only exactly one $a_p \in$ C (or exactly one a_p is an instance of C), or

(iii) if S_i is C+ or C*, and $a_p \in$ C.

L_{ij} and R_{ij} are computed as follows:

if $\frac{LM_{ij}}{LN_i} > \frac{RM_{ij}}{RN_i}$ **then**

$$L_{ij} = \min(\frac{LM_{ij}}{LN_i}, \frac{SM_{ij}}{SN_j});$$

$$R_{ij} = \frac{RM_{ij}}{RN_i};$$

else $$R_{ij} = \min(\frac{RM_{ij}}{RN_i}, \frac{SM_{ij}}{SN_j});$$

$$L_{ij} = \frac{LM_{ij}}{LN_i};$$

Note that if $SN_j = 0$ then $\frac{SM_{ij}}{SN_j} = 1$.

2. U_j is a *reasonably precise concept* (RPC):

$rpc(<S_1, \ldots, S_m \rightarrow V_1, \ldots, V_g>)$.

Let LSN_j and RVN_j be the total numbers of elements in the conditional and consequent parts of the RPC respectively, where a class with a *, e.g., C*, is not counted. Let LM_{ij} and RM_{ij} be the numbers of items in the conditional and consequent parts of A_i that match $\{S_1, \ldots, S_m\}$ and $\{V_1, \ldots, V_g\}$ respectively. Let LSM_{ij} and RVM_{ij} be the numbers of elements in $\{S_1, \ldots, S_m\}$ and $\{V_1, \ldots, V_g\}$ that have been matched by the conditional and consequent parts of A_i respectively (matching with C* is not counted).

The meaning of matching is the same as above for the GI, except that here the conditional and the consequent parts of A_i are considered separately with respect to $\{S_1, \ldots, S_m\}$ and $\{V_1, \ldots, V_g\}$.

L_{ij} and R_{ij} are computed as follows:

$$L_{ij} = \min(\frac{LM_{ij}}{LN_i}, \frac{LSM_{ij}}{LSN_j});$$

$$R_{ij} = \min(\frac{RM_{ij}}{RN_i}, \frac{RVM_{ij}}{RVN_j});$$

Note that if $LSN_{ij} = 0$ (or $RVN_{ij} = 0$) then $\frac{LSM_{ij}}{LSN_j} = 1$ (or $\frac{RVM_{ij}}{RVN_j} = 1$).

After $confm_{ij}$, $unexpConseq_{ij}$, $unexpCond_{ij}$, and $bsUnexp_{ij}$ have been computed, we rank the discovered rules using these values.

Ranking the rules with respect to each individual $U_j \in U$: For each $U_j \in U$, we simply use the $confm_{ij}$, $unexpConseq_{ij}$, $unexpCond_{ij}$ and $bsUnexp_{ij}$ values to sort the discovered rules in A in a descending order to obtain the four rankings. In each ranking, those rules that do not satisfy the support and confidence requirements of U_j are removed.

Ranking the rules with respect to the whole set of specifications *U*: Formulas for these rankings are also designed and implemented. However, in our applications, we find that it is less effective to use these rankings because all the conforming rules or unexpected rules with respect to all the specifications in U are lumped together, thus making them hard to understand. Ranking with respect to individual specification is more effective and easy to understand. However, ranking of the discovered rules in A with respect to the whole set U is useful for finding rules whose conditional and consequent parts are both unexpected, namely, *both-side unexpected* rules.

Both-side unexpected: Both the conditional and consequent parts of the rule $A_i \in A$ are unexpected with respect to the set U. The match value $BsUnexp_i$ of A_i is computed with:

$$BsUnexp_i = 1 - max(Cfm_i, UCond_i, UConseq_i),$$

where $Cfm_i = max(confm_{i1}, confm_{i2}, \ldots, confm_{i|U|})$,

$UConseq_i = max(unexpConseq_{i1}, unexpConseq_{i2}, \ldots, unexpConseq_{i|U|})$,

$UCond_i = max(unexpCond_{i1}, unexpCond_{i2}, \ldots, unexpCond_{i|U|})$.

Clearly, $0 \leq BsUnexp_i \leq 1$. This formula ensures that those rules that have been ranked high in other rankings will not be ranked high here.

Time complexity: Assume the maximal number of items in a discovered rule is N; the number of existing concept specifications is $|U|$, and the number of discovered rules is $|A|$. Computing LM_{ij} and RM_{ij} can be done in $O(N)$. Without considering the final ranking which is a sorting process, the runtime complexity of the algorithm is $O(N|U||A|)$. Since N is small (at most 6 in our applications) and $|U|$ is also small (most of the time we only use each individual specification for analysis), the computation is very efficient.

4. The Visualization System of IAS

After the discovered rules have been analyzed, IAS displays different types of potentially interesting rules to the user. The key here is to show the essential aspects of the rules such that it can take advantage of the human visual capabilities to enable the user to identify the truly interesting rules easily and quickly. Let us discuss what are the essential aspects:

1. Types of potentially interesting rules: Different types of interesting rules should be separated because they give the user different kinds of interesting knowledge.
2. Degrees of interestingness ("match" values): Rules should be grouped 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 lower degrees of interestingness.
3. Interesting items: Showing the interesting items in a rule is more important than the whole rule. 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 whole set of rules, the user still needs to look for the unexpected items in the rules.

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

4.1. An example

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 poor 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 some GA courses are likely to be good in some GB courses. This can be expressed as a RPC:

  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 Figure 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 will discuss these units in turn with the example.

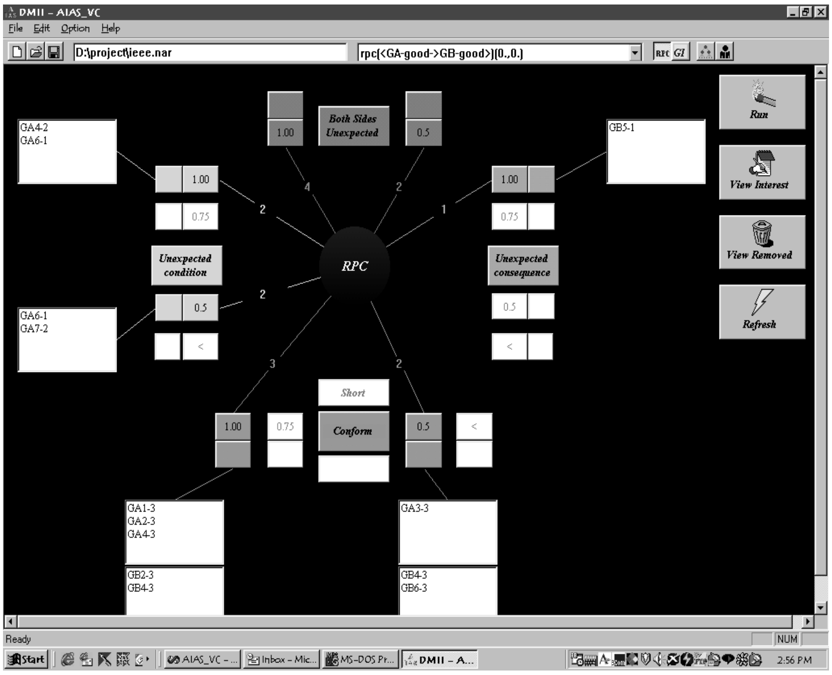

Figure 2. RPC main visualization screen

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 since GA6-1 and GA7-2 are unexpected. This list of rules can be long in an application. The following mechanisms help the user focus his/her attention, i.e., enabling him/her to view rules with different degrees of interestingness ("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 $0.50 \le confm_{i1} < 0.75$. 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 screen, the user sees all the items in different classes involved, and also conforming and unexpected items.

Unexpected condition rules visualization unit: The boxes here have similar meanings as the ones for conforming rules. From Figure 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 Condition***.

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 Figure 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 $\rightarrow$ GB7-2
Rank 1:	1.00	R8	GA6-2 $\rightarrow$ GB7-2
Rank 1:	1.00	R9	GA5-1, GA2-2 $\rightarrow$ GB2-2
Rank 1:	1.00	R10	GA5-2, GA1-2 $\rightarrow$ 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 visualization system also allows the user to incrementally save interesting rules and to remove unwanted rules. Whenever a rule is removed or saved (also removed from the original set of rules), the related pictures and windows are updated.

5. Evaluation

The IAS system is implemented in Visual C++. Our association rule mining system is based on the generalized association rule mining algorithm in [28]. Those redundant and/or insignificant rules are removed using the pruning technique in [14] (objective interestingness analysis).

Since there is no existing technique that is able to perform our task, we could not carry out a comparison. Most existing methods [10, 7, 8, 18, 20, 29] only produce conforming rules but not unexpected rules. Although the system described in [23, 24] produces unexpected association rules, it is not an interactive post-analysis system, and it does not handle RPC and GI specifications.

As the proposed technique deals with subjective interestingness, it is difficult to have an objective measure of its performance. We have carried out a number of experiments involving our users (2) and students (6) to check whether the rankings do reflect people's intuitions of subjective interestingness, in particular, unexpectedness.

In the experiments, we used 3 application datasets (see below), and each subject is asked to specify 10 pieces of existing knowledge for each dataset and to view the ranking results. In the process, we found that some subjects do occasionally disagree with the relative ranking. For example, a subject may believe that a particular rule should be ranked above its neighbor. There were 5 such cases. However, this (i.e., slightly different relative ranking) is not a problem. We do expect such minor disagreements because we are dealing with a subjective issue here. The important thing is that everyone agrees that the technique is able to bring those interesting rules to the top of the list.

Our system has been successfully used in three real-life applications, one educational application, one insurance application and one medical application. Due to confidentiality agreements, we could not give details of these applications. In the applications, the smallest rule set has 770 rules. Most of them have one to two thousand rules. When our users first saw a large number of rules, they were overwhelmed. Our tool makes it much easier for them to analyze these discovered rules. Initially, they were only interested in finding a few types of rules to confirm (or verify) their hypotheses. However, they ended up finding many interesting rules that they had never thought of before as a result of the various unexpectedness rankings. The rules used in the example of Section 4 are from one of our applications (the items appeared in the rules were encrypted).

6. Conclusion and Future Work

This paper proposes a new approach for helping the user identify interesting association rules, in particular, expected and unexpected rules. It consists of an intuitive specification language and an interestingness analysis system. The specification language allows the user to specify his/her various types of existing knowledge about the domain. The interestingness analysis system analyzes the discovered association rules using the user's specifications to identify those potentially interesting ones for the user. The new method is more general and powerful than the existing methods because most existing methods only produce the conforming rules, but not the unexpected rules of various types. Unexpected rules are by definition interesting.

In our future work, we will investigate more sophisticated representation schemes and analysis methods such that we not only can perform analysis at individual rule level but also at higher levels, e.g., to determine whether a set of rules is interesting as a group to the user, and to infer interesting knowledge from the discovered rules.

Acknowledgement: We would like to thank many people, especially Minqing Hu, Ken Wong and Yiyuan Xia, for their contributions to the project. The project is funded by National Science and Technology Board (NSTB) and National University of Singapore (NUS).

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. and Srikant, R. Fast algorithms for mining association rules. *VLDB-94*, 1994.

[3]. Bayardo, R. and Agrawal, R. "Mining the most interesting rules." *KDD-99*, 1999.

[4]. Boose, J. "A survey of knowledge acquisition techniques and tools." In B. Buchanan & D. Wilkins (ed.) *Knowledge Aacquisition and Learning*, 1993, pp. 39-56.

[5]. Buchanan, B. and Wilkin, D. (eds.). *Readings in knowledge acquisition and learning*. Morgan Kaufmann, 1993.

[6]. 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.

[7]. Han, J., Fu, Y., Wang, W., Koperski, K. and Zaiane, O. "DMQL: a data mining query language for relational databases." *Proceedings of the SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 1996.

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

[9]. Kamber, M., and Shinghal, R. 1996. "Evaluating the interestingness of characteristic rules."

KDD-96, 1996, pp. 263-266.

[10]. Klemetinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A.I. 1994. "Finding interesting rules from large sets of discovered association rules." *Proceedings of the Third International Conference on Information and Knowledge Management,* 1994, pp. 401-407.

[11]. Liu, B., and Hsu, W. "Post-analysis of learned rules." *AAAI-96*, 1996, pp. 828-834.

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

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

[14]. Liu, B., Hsu, W. and Ma, Y. "Pruning and summarizing the discovered associations." *SIGKDD-99*, pp. 125-134, 1999.

[15]. Liu, B., Hsu, W, and Chen, S. "Helping the user identify unexpected association rules." *Technical Report*, School of Computing, 1999.

[16]. Liu, B., Hu, M. and Hsu, W. "Multi-level organization and summarization of the discovered rules," *KDD-2000*, 2000.

[17]. Major, J., and Mangano, J. "Selecting among rules induced from a hurricane database." *KDD-93*, 1993, pp. 28-41.

[18]. Meo, R., Psaila, G. and Ceri, S. "A new SQL-like operator for mining association rules." *VLDB-96*, 1996, pp. 122-133.

[19]. Mills, F. Statistical Methods, Pitman, 1955.

[20]. Ng. R. T., Lakshmanan, L. V. S., Han, J., and Pang, A. "Exploratory mining and pruning optimizations of constrained associatino rules." *SIGMOD-98*, 1998.

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

[22]. Piatesky-Shapiro, G. "Discovery, analysis and presentation of strong rules." In Piatesky-Shapiro, G., and Frawley, W (Eds) *Knowledge Discovery in Databases*, pp. 229-248, MIT Press, Cambridge, MA, 1991.

[23]. Padmanabhan, B., and Tuzhilin, A. "A belief-driven method for discovering unexpected patterns." *KDD-98*, 1998, pp. 74-100.

[24]. Padmanabhan, B., and Tuzhilin, A. "Small is Beautiful: Discovering the Minimal Set of Unexpected Patterns." *KDD-2000*, 2000.

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

[26]. Shen, W-M. Ong, K-L. Mitbander, B. and Zaniolo, C. "Metaqueries for data mining." *Advanced in knowledge discovery and data mining*, AAAI Press, chapter 15, 1996, pp. 375-398.

[27]. Silberschatz, A., and Tuzhilin, A. "What makes patterns interesting in knowledge discovery systems." *IEEE Trans. on Know. and Data Eng.* 8(6), 1996, pp. 970-974.

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

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

[30]. Tuzhilin, A. "A pattern discovery algebra." *SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 1997, pp. 71-76.

[31]. WizWhy, http://www.wizsoft.com.

[32]. Zaki, M. "Generating non-redundant association rules," *KDD-2000*, 2000.