Tuple-Level Analysis for Identification of Interesting Patterns

Bing Liu, Wynne Hsu, Hing-Yan Lee [*] and Lai-Fun Mun

Department of Information Systems and Computer Science
National University of Singapore
Lower Kent Ridge Road
Singapore 119260

{liub, whsu}@iscs.nus.sg

[*] Japan-Singapore AI Center
Information Technology Institute
11 Science Park Road
Singapore Science Park II
Singapore 0511

hingyan@iti.gov.sg

Abstract

One of the important issues in data mining is the “interestingness” problem. Past research and applications have shown that in many situations a huge number of patterns can be discovered from a database. Most of these patterns are actually useless or uninteresting to the user. But because of the huge number of patterns, it is difficult for the user to identify those interesting to him/her. In this project, we propose a new technique to help the user identify interesting patterns. The user is first asked to provide his/her expected patterns according to his/her past knowledge and/or intuitive feelings. Given these expectations, the system uses a tuple-level fuzzy matching technique to analyze and rank the discovered patterns according to a number of interestingness measures. With this technique, the user can quickly focus on a subset of the discovered patterns with the most application values.

1. Introduction

Past research in knowledge discovery has found that in many applications it is easy to generate hundreds or even thousands of patterns from a database. However, most of these rules patterns are not useful or interesting to the user. Because of the huge number of patterns, it is very difficult for the user to analyze and to identify those that are interesting to him/her. This is known as the "interestingness" problem [1, 8]. To prevent the user from being overwhelmed by a large number of patterns, techniques are needed to help the user identify the interesting ones. This is a real world problem that is faced by our industrial partner. When presented with a huge number of patterns, their customers typically give up because they find it too difficult to analyze the large number of patterns to sieve out the useful ones. In view of this, our industrial partner approached us to study this problem.

The interestingness issue has been an important problem ever since the beginning of data mining research [1]. Many factors that contribute to the interestingness of a discovered pattern have been proposed [e.g., 1, 6, 7]. They include: coverage, confidence, strength, statistical significance, simplicity, unexpectedness, and actionability [7, 10]. The first five factors are the so-called objective factors [7, 10]. They can be handled with techniques that do not require domain knowledge. They have been studied extensively in the literature [e.g., 6, 1, 2]. The last two factors (unexpectedness, and actionability) are called subjective factors [7, 10]. They cannot be measured without knowledge about the application and the user's interests. It is noted in [7, 8] that although objective measures are useful, they are insufficient to determine the interestingness of the discovered patterns. Subjective measures are needed. Subjective interestingness is the focus of this paper.

Designing a domain independent technique for identifying interesting patterns using subjective interestingness measures is a difficult task. Some of the reasons are: (1) in different domains (or even different applications in the same domain), people are interested in different things; (2) given the same database and the discovered patterns, different users may be interested in different subsets of the patterns; (3) even for the same user, at different points in time, his/her interests may also vary due to the specific situation he/she is in at the particular moment.

In this research, we propose a general approach to help the user identify unexpected patterns. This approach is based on the conformity of the discovered patterns to a set of user-specified expected patterns. This conformity can be measured at the pattern-level and at the tuple-level. In [4], we proposed a pattern-level fuzzy matching technique to help the user identify interesting patterns. In this paper, we propose a tuple-level fuzzy matching technique. The key difference between the pattern-level matching and tuple-level matching is that the former only compares patterns, while the latter goes back to the database to analyze the discovered patterns. The new method is more reliable compared to the old method. For example, we have the following discovered pattern and the user-expected pattern:

Discovered pattern (or rule): **If** $X > 9$, $Y < 5$, $Q = 2$ **Then** $R = TRUE$

User-expected pattern (or rule): **If** $X > 9$, $Y < 6$, $P = 4$ **Then** $R = TRUE$

On the surface the two patterns (or rules) seem similar (or conforming). The pattern-level match method in [4] will give a high conforming match value. However, it may be the case that tuples in the database suggest that ($X > 9$, $Y < 5$, $Q = 2$) implies ($X > 9$, $Y < 5$, $P = 4$) — the two patterns are conforming, or that ($X > 9$, $Y < 5$, $P = 4$) implies $R = FALSE$ — the user-expected pattern is actually false, or other possible situations. Without consulting the actual database, we cannot decide which one of the above cases is true.

The new technique can be described as follows: The user is first asked to supply a set of expected patterns according to his/her previous knowledge or intuitive feelings. These expected patterns are then used by the tuple-level fuzzy matching technique to analyze and to rank the discovered patterns through consulting the database tuples. In this technique, a number of rankings can be performed for different purposes. Two main types of ranking are conformity ranking and unexpectedness ranking. The conformity ranking ranks the discovered patterns according to how well they conform to the user's expectations. Hence, a user can simply check the few patterns at the top of the list to confirm or deny his/her intuitions (or previous knowledge). The unexpectedness ranking ranks the discovered patterns according to different kinds of unexpectedness. With the ranking results, the user can find out various types of unexpected patterns. Unexpected patterns are by definition interesting [1, 7, 8, 10]. Thus, our technique helps to solve part of the interestingness problem.

2. Problem Definition and Scope of this Work

From a user's point of view, he/she wants to find interesting patterns from one or more databases D. From a discovery system's point of view, a technique Q discovers a set of patterns $B_{(Q,D)}$ from D. Let $I_{(Q,D)}$ be the set of interesting patterns in $B_{(Q,D)}$. Thus, we have $I_{(Q,D)} \subseteq B_{(Q,D)}$. We define the interestingness problem as follows:

Definition: Given $B_{(Q,D)}$, the set of patterns discovered by Q in D, determine $I_{(Q,D)}$ and rank the patterns in $I_{(Q,D)}$ according to their degrees of interestingness to the user at the particular point in time.

In practice, this is difficult to achieve because the definition of "interestingness" is domain and application dependent, and also user and context dependent. To simplify our task, we only rank the discovered patterns. The identification of $I_{(Q,D)}$ is left to the user. In our method, $I_{(Q,D)}$ will be a subset of patterns at the top of the rankings.

Before presenting our technique, let us define the type of patterns that we will deal with in this paper. Although our technique is general, its algorithms and formulae still depend on the type of patterns. It is well-known that many types of pattern may be discovered from databases. In this paper, we deal with only one type, i.e., classification patterns or rules such as those generated by C4.5 [9] or other induction systems. If other types of patterns are used, our technique needs to be customized for these pattern types.

A classification pattern (or rule) is typically represented as follows:

If $P_1, P_2, P_3, \ldots, P_n$ **then** C (or $P_1, P_2, P_3, \ldots, P_n \rightarrow C$)

where "," means "and", and P_i is a proposition of the following format: *attr OP value*

where *attr* is the name of an attribute in the database, and *value* is a possible value for the attribute *attr* and $OP \in \{=, \neq, <, >, \leq, \geq\}$ is the operator. Each P_i is termed as a **condition** and C is the **conclusion**, which has the same format as P_i. Since all the patterns in this paper are of the if-then rule format, we shall use the terms "patterns" and "rules" interchangeably.

3. The tuple-level matching technique

This technique consists of two steps:

Step 1. The user is asked to provide a set of rules E (with the same syntax as the discovered rules) that he/she expects to find in the database D using his/her previous knowledge or intuitive feelings. These user rules are regarded as fuzzy rules because people's knowledge is normally fuzzy. These fuzzy rules are described with the help of fuzzy linguistic variables and fuzzy sets [11]. See below.

Note that the user does not have to give a complete set of expected rules at the beginning, which is difficult. He/she may try something simple at the beginning and incrementally build up the set of expected rules.

Step 2. The system goes through the database D to check the correctness of each user-expected rule $E_j \in E$ and to rank the rules in B (previous known as $B_{(Q,D)}$) according to their conformity and unexpectedness to the user's expectation E_j.

3.1 Specifying user-expected rules

In Step 1 of the proposed technique, the user specifies a set of expected rules from the database. These expected rules need to be described with the help of fuzzy linguistic variables. Let us now briefly review the definition of a fuzzy linguistic variable [11].

Definition: A fuzzy linguistic variable is a quintuple (x, $T(x)$, U, G, $\tilde{M}$) in which x is the name of the variable; $T(x)$ is the term set of x; that is, the set of names of linguistic values of x, with each value being a fuzzy variable denoted generally by x and ranging over a universe of discourse U; G is a syntactic rule for generating the name, X, of values of x; and M is a semantic rule for associating with each value X its meaning, $\tilde{M}(X)$ which is a fuzzy subset of U. A particular X is called a *term*.

Using fuzzy linguistic variables, a user may specify an expected rule as follows.

If *Grade* = *poor*, *Age* = *old* **then** *Class* = *reject*

Here, *poor*, *old* and *reject* are fuzzy terms. To describe these terms, the user needs to define the semantic rules for *poor*, *old* and *reject*. Assume "*Grade*" takes discrete values and its universe (or domain) is {*A, B, C, D, F*}. The user may define that *poor* grade means:

{(A, 0), (B, 0), (C, 0.2), (D, 0.8) (F, 1) }

where the left coordinate is an element in the universe of "*Grade*", and the right coordinate is the degree of membership of that element in the fuzzy set *poor* (denoted as μ_{poor} [11]), e.g., $\mu_{poor}(D) = 0.8$. *reject* can be described similarly.

On the other hand, "*Age*" is a continuous variable. The semantic rule for *old* takes on the form of a continuous function. To simplify the user's task of specifying the shape of this continuous function, we assume that the function has a curve of the form shown in Figure 1. Thus, the user merely needs to provide the values for a, b, c, and d. Assume the domain of "*Age*" is [1, 80]. Then the user may define *old* by specifying that $a = 50$, $b = 55$, $c = 80$, and $d = 80$.

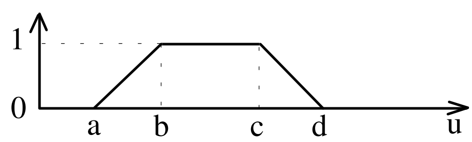

Figure 1. Membership function

3.2 Computing the correctness of the expected rule E_j

Naturally, when the user provides an expected rule $E_j \in E$, he/she wishes to know whether the rule is correct. To compute this value, the system needs to check against the database to see how true E_j is. Since the user's rule E_j is regarded as a fuzzy rule, matching E_j with each tuple in the database is also fuzzy, and it returns a value in the range [0, 1]. Thus, to consider a match to be satisfactory, a positive cutoff value needs to be used.

Let *positive_cutoff* be a value in the range [0, 1] representing the minimal degree of match between the database tuple $D_k \in D$ and $E_j \in E$ on either the conditional or conclusion part to be considered satisfactory. The value of *positive_cutoff* can be specified by the user. Let $|A_j|$ be the number of attributes in the conditional part of E_j. We denote $V_{a,k,j}$ as the match value of the ath condition of E_j with D_k. We then define

$$M_{cond}^{k,j} = \frac{\sum_{|A_j|} V_{a,k,j}}{|A_j|}$$

to be the degree of condition match of D_k and E_j, and $M_{concl}^{k,j} = V_{k,j}$ to be the degree of conclusion match. The detailed computation of $V_{a,k,j}$ (or $V_{k,j}$) is discussed in [4, 5]. Basically, we calculate the degree of membership of the relevant attribute value in D_k with respect to ath fuzzy condition (or proposition) in the conditional (or conclusion) part of E_j.

The **correctness** (or **confidence**) of E_j, M_{confid}^{j} is computed as follows:

$$M_{confid}^{j} = \frac{\sum_{k}\left(M_{cond}^{k,j} * M_{concl}^{k,j}\right)}{\sum_{k} M_{cond}^{k,j}}, \quad M_{cond}^{k,j} \geq positive\_cutoff$$

Interested readers, please refer to [5] for an explanation of the rationale behind the formula.

3.3 Ranking discovered rules according to various interestingness measures

After verifying the correctness of the user-expected rule E_j, we performs different rankings of the discovered rules. With these rankings, the user can find those discovered rules that conform to E_j, those that are unexpected with respect to E_j, and those that are complementary to E_j. We first use an example (Table 1) to illustrate the basic technique of our appraoch.

	P	C	$X_1 \rightarrow Y_1$	$X_2 \rightarrow Y_2$	$X_3 \rightarrow Y_3$	$X_4 \rightarrow Y_4$	$X_5 \rightarrow Y_5$	$X_6 \rightarrow Y_6$
1	✓	✓	✓					
2	✓	✓	✓					
3	✓	✓	✓					
4	✓	✓		✓				
5	✓	✓		✓				
6	✓	✓				✓		
7	✓	×			✓			
8	✓	×		✓				
9	✓	×			✓			
10	✓	×			✓			
11	×	✓		✓				
12	×	✓				✓		
13	×	✓				✓		
14	×	✓				✓		
15	×	×					✓	
16	×	×						✓
17	×	×						✓
18	×	×						✓

Table 1 An example for illustration

The table can be understood as follows: Let the expected rule E_j be $P \rightarrow C$, where P is the set of conditions and C is the conclusion. Let the ith discovered rule, B_i, be $X_i \rightarrow Y_i$, where X_i is the set of conditions and Y_i is the conclusion on the same attribute as C. In this example, we have 6 discovered rules. Each row in the table symbolizes a database tuple (the real tuple is not represented). Column 1 shows the tuple number. Column 2 and 3 show whether a tuple satisfies P and C respectively. If it satisfies P (or C), a ✓ appears, otherwise a × appears. Column 4 to 9 show which tuple is covered (or tagged) by which discovered rule. For each tuple, there is only one $X_i \rightarrow Y_i$ column that has a ✓ because each tuple is tagged (or classified) under one discovered rule only following the idea in C4.5. For ease of understanding and without loss of generality, we have ordered the tuples according to their satisfaction of the user-expected rule $P \rightarrow C$.

From the table, the first thing that we can see is that the expected rule $P \rightarrow C$ is not accurate in general because only 6 out of the 10 tuples that satisfy its conditions also satisfy its conclusion. We can also see from the table that the tuples can be divided into 4 groups.

Group 1 (tuple 1-6): Within this group, tuples are tagged under three discovered rules (Rule 1, 2 and 4). We say these rules ***conform*** to the user-expected rule $P \rightarrow C$ to certain degree because the tuples in this group satisfy $P \rightarrow C$, and Rule 1, 2 or 4. The first two boxs in Table 2 shows the three rules. The numbers in brackets, i.e., x/y, are interpreted as x out of the y tuples originally tagged under discovered rule i are satisfied by $P \to C$. The three rules are ranked according to their x/y values.

Group 2 (tuple 7-10): Within this group, tuples are tagged under two discovered rules (Rule 2 and 3). We say that these two rules have ***unexpected conclusions*** with respect to $P \to C$ to certain degree because the tuples in this group satisfy P, and Rule 2 or 3, but not C.

Group 3 (tuple 11-14): Within this group, tuples are tagged under three discovered rules (Rule 2 and 4). We say that these rules have ***unexpected conditions*** with respect to $P \to C$ to certain degree because the tuples in this group satisfy C, and Rule 2 or 4, but not P.

Group 4 (tuple 15-18): Within this group, tuples are tagged under two discovered rules (Rule 5 and 6). We say that these two rules are ***complementary*** to $P \to C$ to certain degree because the tuples in this group satisfy Rule 5 or 6, but not C and not P.

Table 2 shows the discovered rules ranked according to their x/y values in different groups.

1. Conforming rules Reasons: To certain degree X_i implies P and C.	**3. Unexpected condition rules** Reasons: To certain degree X_i implies $\neg P$ and C.
1. $X_1 \to Y_1$ (3/3) 2. $X_2 \to Y_2$ (2/4) 3. $X_4 \to Y_4$ (1/4)	1. $X_4 \to Y_4$ (3/4) 2. $X_2 \to Y_2$ (1/4)
2. Unexpected conclusion rules Reasons: To certain degree X_i implies P and $\neg C$.	**4. Complementary rules** Reasons: To certain degree X_i implies $\neg P$ and $\neg C$.
1. $X_3 \to Y_3$ (3/3) 2. $X_2 \to Y_2$ (1/4)	1. $X_6 \to Y_6$ (3/3) 2. $X_5 \to Y_5$ (1/1)

Table 2 Ranking the discovered rules

From Table 1 and 2, we can see the basic idea of the proposed approach. The detailed algorithm is given in the next page.

Tag each tuple in the database under the first discovered rule B_i that correctly classified it, and sum the total number of tuples tagged under B_i (denoted as T_i).

For each tuple, D_k, in the database **Do**

    For each expected rule, E_j , of the form $P \rightarrow C$ **Do**

        Compute D_k and E_j match by breaking it down into the conditional part match, $M_{cond}^{k,j}$, and the conclusion part match, $M_{concl}^{k,j}$;

        If $M_{cond}^{k,j} \geq positive\_cutoff$ **Then**

            $RuleM_j = RuleM_j + M_{cond}^{k,j} \times M_{concl}^{k,j}$;

            $CondM_j = CondM_j + M_{cond}^{k,j}$;

        endif;

        If D_k is tagged under a discovered rule $X_i \rightarrow Y_i$ **Then**

            If $M_{cond}^{k,j} \geq positive\_cutoff$ **Then**

                If $M_{concl}^{k,j} \geq positive\_cutoff$ **Then**

                    Increment the number of positively tagged tuples $Conform_{i,j}$;

                Else Increment the number of unexpected conclusion tuples $UnexConcl_{i,j}$;

            Else   **If** $M_{concl}^{k,j} \geq positive\_cutoff$ **then**

                        Increment the number of unexpected conclusion tuples $UnexCond_{i,j}$;

                    Else Increment the number of complementary tuples $Complmt_{i,j}$;

    endfor;

endfor;

For each user-expected rule, E_j **Do**

    $M_{confid}^{j} = \frac{RuleM_j}{CondM_j}$;

    For each discovered rule $X_i \rightarrow Y_i$ **Do**

        $ConformM_{i,j} = \frac{Conform_{i,j}}{T_i}$;

        $UnexConclM_{i,j} = \frac{UnexConcl_{i,j}}{T_i}$;

        $UnexCondM_{i,j} = \frac{UnexCond_{i,j}}{T_i}$;

        $ComplmtM_{i,j} = \frac{Complmt_{i,j}}{T_i}$;

    endfor;

    Using the results above to rank discovered rules according to different interestingness criteria.

endfor;

Notes about the algorithm:

- The algorithm basically consists of three main Steps:

  Step 1 (line 1): Tag each tuple in the database under a discovered rule. As explained earlier, following the C4.5's convention, each tuple is tagged under one discovered rule only (i.e., the first rule that correctly classifies the tuple), except those tuples that are not correctly classified by any discovered rule.

  Step 2 (line 2-18): Compute the intermediate values to be used in Step 3. Note that $RuleM_j$, $CondM_j$, $Conform_{i,j}$, $UnexConcl_{i,j}$, $UnexCond_{i,j}$, and $Complmt_{i,j}$ are all initialized to 0 at the beginning.

  Step 3 (line 19-28): Compute the final results, i.e., the correctness of each user-expected rule, and various *x*/*y* values (see Table 2) for each discovered rule with respect to each user-expected rule. Ranking is then performed using these values for different interestingness categories.

- Complexity of this algorithm can be analyzed as follows: Step 1 can be incorporated into the rule generation program (e.g., C4.5). It is basically linear to the size of the database. Step 2 needs to go through the database once. Assume computing each set of intermediate results from line 4-16 takes constant time. The time complexity of this step is $|D||E|$. The complexity of Step 3 is clearly $O(|E||B|)$ without considering the final ranking, which is a simple sorting process.

  Considering the fact that the user may do the analysis many times because it is unlikely that he/she is able to provide the complete set of expected rules the first time, only Step 2 and 3 need to be performed every time. Step 1 only needs to be done once, which can be part of the rule generator. Thus, the main computation is at Step 2. For more detailed analysis of time and space complexities, see [5].

4. An Illustration

We have tested the system using a number of real databases from our industrial partner. Unfortunately, we are not allowed to use them here. Here, we give a test example using the credit screening database created by Chiharu Sano in UCI repository for machine learning. The rules are generated by C4.5. The value in [ ] is the confidence of the discovered rule. Only one expected rule is considered. In the example, 0.6 is used as the positive cutoff match value. Below are the discovered rules and the user-expected rule.

- **Discovered rules:**

Rule 1: Age > 25, Savings > 7, YR_Work > 2 → Granted Yes.....[94.8%]

Rule 2: Sex = Male, YR_Work > 2 → Granted Yes......................[93.4%]

Rule 3: Jobless = No, Bought = pc → Granted Yes.......................[84.1%]

Rule 4: Bought = medinstru, Age <= 34 → Granted Yes.............[82.0%]

Rule 5: Sex = Female, Age <= 25 → Granted No.........................[56.8%]

Rule 6: Savings <= 7, M_LOAN > 7 → Granted No....................[54.6%]

Rule 7: YR_Work <= 2 → Granted No ....[54.4%]

- **User-expected rule:**

Expected Rule 1: YR_Work <= Few {a = 1, b = 2, c = 3, d = 4}

→ Granted = No {(No, 1), (Yes, 0)}

The running results are reported below:

- **Correctness of the user-expected rule**

Confidence Level: 54.8 % Coverage Of Conditions: 62 cases

Coverage Of Rule: 34 cases

- **Conforming rules and their ranking**

RANK 1 Discovered Rule 7

(C) YR_Work <= 2 → Granted = No.. [54.4%]

Reasons: YR_Work <= Few, Granted = No..........................Match = 100.0% ... (20/20)

RANK 2 Discovered Rule 5

Age <= 25, Sex = Female → Granted = No .... [56.8%]

Reasons: YR_Work <= Few, Granted = No..........................Match = 78.6% ... (11/14)

RANK 3 Discovered Rule 6

M_LOAN > 7, Savings <= 7 → Granted = No.. [54.6%]

Reasons: YR_Work <= Few, Granted = No..........................Match = 60.0% ... (3/5)

Note that (C) before a condition indicates that the condition has a common attribute in the expected rule and the discovered rule. (x/y) after the match value indicates that x of y tuples that are tagged under the discovered rule also satisfy the user-expected rule.

Note that rule 7 is an example of a conforming rule without any unanticipated condition attributes since its only condition attribute (YR_Work) is also the condition attribute in the expected rule. Rule 5 has some unanticipated conditions since *Age* and *Sex* are not in the the expected rule. Although unanticipated, from the database observations, their conditions seem to imply the expected condition 'YR_Work ≤ Few' and conclusion 'Granted = No'.

- **Unexpected conclusion rules and their ranking**

  RANK 1 Discovered Rule 4

  Age <= 34, Bought = medinstru → Granted = Yes .... [82.0%]

  Reasons: YR_Work <= Few, NOT(Granted = No) ....Match = 100.0%.... (3/3)

  RANK 2 Discovered Rule 3

  Bought = pc, Jobless = No → Granted = Yes .... [84.1%]

  Reasons: YR_Work <= Few, NOT(Granted = No)................Match = 80.0% ... (4/5)

  The reason for the unexpected conclusion is simple. According to the database, the conditions (e.g., Age <= 34, Bought = medinstru) in the discovered rule seems to imply to certain degree the condition of the expected rule (e.g., YR_Work <= Few), but not its conclusion.

- **Unexpected condition rules and their ranking**

  There is no unexpected condition rules because of low matching values.

- **Complementary Rules**

  RANK 1 Discovered Rule 1

  Age > 25, Savings > 7, (C) YR_Work > 2 → Granted = Yes .... [94.8%]

  Reasons: NOT(YR_Work <= Few), NOT(Granted = No) .....Match = 85.5% ... (47/55)

  RANK 2 Discovered Rule 2

  Sex = Male, (C) YR_Work > 2 → Granted = Yes .... [93.4%]

  Reasons: NOT(YR_Work <= Few), NOT(Granted = No) .....Match = 66.7% ... (6/9)

  The complementary rules can be similarly explained as above.

5. Related Work

Subjective interestingness has long been identified as an important problem in data mining [1, 8, 7, 10, 6]. However, most data mining techniques and tools only deal with objective interestingness. It has been noted in [7, 8] that objective measures are insufficient for determining the interestingness of the discovered patterns. Subjective measures are needed. Two main subjective measures are: unexpectedness, and actionability [7, 10, 1].

[7] studied the issue of interestingness in the context of a health care application. A knowledge discovery system, KEFIR, is built. The system analyzes health care information to uncover "key findings". The degree of interestingness of a finding is estimated by the amount of benefit that could be realized if an action can be taken. Domain experts provide the recommended actions to be taken for various findings. For each action, the expert also estimates the

probability of success. Once a finding is discovered, the system computes the estimated benefit for taking an recommended action. This approach presents a good method for incorporating the subjective interestingness into an application system. However, the approach is application specific as its domain knowledge (from experts) is hard-coded in the system as production rules. The system cannot be used for any other application. Furthermore, the system mainly deals with actionability issue.

Our method is general, i.e., domain independent. A pattern analysis system based on our technique can be attached to each data mining tool to help the user identify interesting patterns. Though domain-specific systems such as KEFIR are still more effective, the costs of building such systems are very high.

[10] proposed to use probabilistic beliefs and belief revision as the framework for describing subjective interestingness. Specifically, a belief system is used for defining unexpectedness. However, [10] is just a proposal, and no system was implemented. For an actual implementation, a great deal of details has to be studied. Our proposed approach has been implemented and tested. In addition, our approach allows the user to specify his/her beliefs (or expectations) in fuzzy terms which we feel are more natural and intuitive than the complex probabilities that the user has to supply in [10].

In [4], we proposed a method for finding unexpected patterns based on a pattern-level match technique. This method matches the discovered patterns against the user-expected patterns without consulting the database. First, it matches the attribute names and the attribute values, and then it combines them in different ways to rank the discovered patterns according to their conformity and unexpectedness to the user-expected patterns. The advantage of this method is that it is efficient because no database access is involved. However, it has some limitations. The main limitation is that it computes the match based solely on the common attributes and common value ranges in the discovered pattern and the user-expected pattern. Ranking based on this computation can be unreliable (see the introduction). The approach proposed in this paper is more reliable because it verifies each claim by consulting the actual database.

6. Conclusion

In this paper, we study the subjective interestingness issue in data mining from a domain independent perspective. A tuple-level matching method for ranking the discovered rules)or patterns) according to their interestingness is proposed. The method is characterized by asking the user to input his/her expected rule (or patterns), and then the system analyzes and ranks the discovered rules by consulting the database. This method, among other things, can help the user identify unexpeced patterns. Our future research will also study how to help the user identify actionable patterns, and how to use domain knowledge in the ranking processes.

Acknowledgment

We would like to thank Gui-Jun Yang for implementing the user interface, and Hwee-Leng Ong and Angeline Pang from Information Technology Institute for useful discussions.

References

[1] W. J. Frawely, G. Piatesky-Shapiro, C. J. Matheus. Knowledge discovery in databases: an overview. In G. Piatetsky-Shapiro and W.J Frawley (eds), *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991, 1-27.

[2] J. Han, Y. Cai, and N. Cercone. Data driven discovery of quantitative rules in relational database. *IEEE Trans. Knowl. & Data Eng.* 5, 1993, 29-40.

[3] W. Klosgen. Problems for knowledge discovery in databases and their treatment in the statistics interpreter explora. *International Journal of Intelligent Systems*, 7(7), 1992, 649-673.

[4] B. Liu and W. Hsu. Post-analysis of learned rules. To appear in *AAAI-96*.

[5] B. Liu, W. Hsu, and L.F. Mun. Finding interesting patterns using user expectations. *DISCS Technical Report*, 1996.

[6] J. A. Major, J. J. Mangano. Selecting among rules induced from a hurricane database. *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, 1993, 30-31.

[7] G. Piatesky-Shapiro and C. J Matheus. The interestingness of deviations. *Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, 1994, 25-36.

[8] G. Piatetsky-Shapiro, C. Matheus, P. Smyth, and R. Uthurusamy. KDD-93: progress and challenges ... *AI magazine*, Fall, 1994, pg. 77-87.

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

[10] A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 1995, 275-281.

[11] H. -J. Zimmermann. *Fuzzy set theory and its applications*. Kluwer Academic Publishers, 1991.