Identifying Relevant Databases for Multidatabase Mining

Huan Liu, Hongjun Lu, Jun Yao

Department of Information Systems and Computer Science
National University of Singapore
Kent Ridge, Singapore 119260
{liuh,luhj,yaojun}@iscs.nus.edu.sg

Abstract. Various tools and systems for knowledge discovery and data mining are developed and available for applications. However, when we are immersed in heaps of databases, an immediate question facing practitioners is where we should start mining. In this paper, breaking away from the conventional data mining assumption that many databases be joined into one, we argue that the first step for multidatabase mining is to identify databases that are most likely relevant to an application; without doing so, the mining process can be lengthy, aimless and ineffective. A relevance measure is thus proposed to identify relevant databases for mining tasks with an objective to find patterns or regularities about certain attributes. An efficient implementation for identifying relevant databases is described. Experiments are conducted to validate the measure's performance and to show its promising applications.

Keywords: multiple databases, data mining, query, relevance measure

1 Introduction

While researchers are trying to develop efficient algorithms to cope with a large volume of data [1, 5, 18], little work has been devoted to the *data aspect* in knowledge discovery process. Although most data mining algorithms assume a single data set, for real world application practitioners have to face the problem of discovering knowledge from systems with multiple databases. To mine knowledge from multiple databases, a brute force approach is to join the related tables into a single large table, upon which existing data mining techniques or tools can be applied. There are several problems for this approach in real world application. First, database integration itself is still a problematic area, especially where the source domains differ. Second, the size of the joined table will increase. The increase in size not only prolongs the running time of mining algorithms, but also affects the behavior of mining algorithms. Third, among a large number of generated patterns in a database, it is often difficult for users to search for useful ones [9, 13, 16]. The useless or uninteresting patterns are generated as there are quite a number of irrelevant databases when joining. Therefore, as in any effective knowledge discovery process, the first important step in mining from multiple databases is indeed to select those databases that are *relevant* to the specific mining task.

In this paper, we will address the relevance problem: identifying databases relevant to a particular data mining task in multiple databases. Without the loss of generality, we call each database a relation, or a table and assume that (1) a specific mining task is related to the property of certain attributes, which can be expressed using query predicates and (2) the higher order correlations of attributes with a query are at least partially reflected in their first order correlations. The problem of identifying relevant databases can be stated as follows:

> Given n data tables (relations), $D_k (1 \leq k \leq n)$, each of which consists of a number of attributes. A query predicate Q is expressed in the form of "A_i *relop* C" where A_i is an attribute, *relop* is one of the relational operators $(=, >, <, \leq, \geq, \neq)$, and C is a value in its domain. Identifying relevant databases is to select *relevant* tables (relations) that contain specific, reliable and statistically significant information pertaining to the query.

We start with mining tasks with query predicates for two reasons. (1) Queries are easy to form and are most familiar and natural when dealing with databases: user's interest, prior knowledge and intention can be conveyed by means of such queries; (2) among various data mining tasks (classification, characterization, generalization, association, etc.), most of them aim at discovering knowledge that can be expressed as relationships among values of attributes [3, 11, 6, 15, 17]. What we are concerned is to have a relatively simple and fast method by which we can identify the databases (tables) that are relevant to a mining task. After that, various existing techniques can be used to conduct mining tasks. Major contributions of our work include:

- illustrating the importance and necessity of identifying relevant databases when mining from multi-databases;
- establishing a quantitative measure that allows us to identify relevant database (tables) before mining; and
- developing an efficient implementation of computing the relevance measure.

The remainder of the paper is organized as follows. Section 2 gives a motivating example to further illustrate the issue of relevance. Section 3 presents a relevance measure to identify relevant databases. Section 4 describes an algorithm that identifies relevant databases using the proposed measure. Section 5 presents some preliminary results of using the measure to a real database. Section 6 discusses the relationship between our work and some related work. Section 7 concludes the paper.

2 A Motivating Example

In this section, we will use two artificial databases to argue that, for data mining tasks involving multiple databases, it is better to identify relevant database first before applying mining techniques.

Our example databases are about people's diet habits. It is user's intention to discover some *useful* knowledge about Chinese food habits from the databases. Here and in most KDD applications, being useful means something that should potentially lead to some useful actions by a user [4]. Specific information is one type of usefulness. In our example, the useful knowledge should be some diet habits specific to Chinese. Two tables (databases) with desired attributes, are as shown in Tables 1 and 2.

Table 1. Database 1 of Diet Habits

ID	ethnic-group	alcohol	on-diet	snack-between-meals	favorite-non-veg
950351	Chinese	never	no	sometimes	fish
950301	Chinese	never	no	seldom	pork
950282	Chinese	sometimes	no	seldom	fish
940112	Chinese	often	no	often	chicken
940023	Chinese	sometimes	no	sometimes	beef
938976	Chinese	sometimes	no	sometimes	egg
950612	American	never	no	seldom	beef
950122	American	often	no	sometimes	beef
940227	American	sometimes	no	sometimes	chicken
938567	American	sometimes	no	often	pork
950348	Indian	often	no	sometimes	fish
950312	Indian	sometimes	no	seldom	pork
950123	Indian	never	no	sometimes	chicken
940247	Indian	sometimes	no	sometimes	fish
940100	Indian	never	no	sometimes	beef

Let's look at Table 1 first. It is difficult to draw a conclusion that the database contains information specific to Chinese. In such a case, it is fair to say that the database is irrelevant with respect to the query about Chinese.

It is easy to see that database in Table 2 is relevant to the query about Chinese since we can at least derive such a statement (rule)

"If ethnic group is Chinese, the main food is rice"

since all records about Chinese show that the main food is rice whereas records about American and Indian do not indicate such a habit. We have seen that although both databases contain information about the diet habits of Chinese, one of them contains relevant knowledge but the other does not.

Table 2. Database 2 of Diet Habits

ID	ethnic-Group	main-food	regular eating times	drink	vegetarian
950578	Chinese	rice	3	tea	no
950351	Chinese	rice	3	tea	no
950301	Chinese	rice	3	tea	no
950282	Chinese	rice	3	tea	no
940226	Chinese	rice	3	cola	no
940112	Chinese	rice	3	tea	yes
950612	American	bread	3	coffee	no
950122	American	bread	3	cola	no
940227	American	rice	3	cola	yes
940121	American	bread	3	tea	no
950348	Indian	bread	3	coffee	no
950312	Indian	bread	3	coffee	yes
950123	Indian	rice	3	cola	no
940247	Indian	rice	3	coffee	no
940109	Indian	bread	3	tea	no

We were able to manually pick one database over the other in the above simple example because the databases are small. We need a method for large databases for selection. Let's examine the existing methods to see if some of them can serve the purpose of data mining, pretending no knowledge about which database contains interesting information.

1. The first solution is to combine two tables together, i.e., joining them on the common attribute *ID* before applying mining algorithms. There are some problems of doing so. (1) the resulting tuples will have 10 attributes besides *ID*, i.e., as twice as many in the original databases; (2) the resulting table will have more tuples. In our example, the number of samples increases to 19 (4 more than the original one); and (3) for certain samples, some attributes will have missing values since they only appear in one of the databases. All these will surely make the subsequent mining task more complex.
2. The second solution could be issuing database queries to retrieve information from the databases and see which one should be focused on. In our example, if an SQL selection query with condition "**WHERE** ethnic-group = Chinese " is posed, both databases will return 6 tuples. There is no indication whether one of them is relevant or not.
3. We can apply some data mining algorithms to the two databases, some patterns will be generated from the irrelevant database. For instance, using the concept of *support* and $confidence$ level in mining association rules, we can have rules like:

   $ethnic\text{-}group$ = Chinese $\rightarrow$ $on\text{-}diet$ = no

   with 40% support and the confidence level of 100%. We do not think such information is really interesting since *on-diet* = no in fact holds for all tuples. We can also apply certain classification algorithms such as CART, CN2, and C4.5 [2, 3, 15] to the database specifying attribute *ethnic-group* as the class label. We will get some classification rules with certain accuracy too but not interesting.

From the above observations, several possible solutions seem inadequate to efficiently distinguish the irrelevant database from the relevant one. It is clear that there is a necessity of identifying relevant databases first when mining from multiple data sources.

3 Relevance Measurement

In this section, we discuss a quantitative measure of relevance. The relevant databases identified by the measures should contain specific information pertaining to our mining task, which is reliable and statistically significant.

We call A *relop* C a *selector*, where A is an attribute name which is not referenced by the query predicate Q, $relop \in \{=, <, >, \leq, \geq, \neq\}$, and C is a constant value in the domain of A. We define the *relevance factor* of selector s with respect to Q as:

$$RF(s, Q) = \Pr(s|Q)\Pr(Q)\log\frac{\Pr(s|Q)}{\Pr(s)} \tag{1}$$

where $\Pr(Q)$ and $\Pr(s)$ are priors and estimated by the ratios how frequently they appear in a database; and $\Pr(s|Q)$ is the posterior about the frequency ratio of s appearing given that Q occurs. The rationale of defining relevance as in Equation 1 is as follows:

$\frac{\Pr(s|Q)}{\Pr(s)}$ shows degree of the deviation of the posterior from the prior. This ratio tells us different relationships between $\Pr(s|Q)$ and $\Pr(s)$.

Case 1 If $\frac{\Pr(s|Q)}{\Pr(s)}$ is close to 1, i.e., $\Pr(s|Q) \simeq \Pr(s)$, s is independent of Q;
Case 2 If $\frac{\Pr(s|Q)}{\Pr(s)}$ is close to 0, i.e., $\Pr(s|Q)$ is almost 0, s rarely occurs given Q;
Case 3 If $\frac{\Pr(s|Q)}{\Pr(s)}$ is less than 1, s is not frequent enough using $\Pr(Q)$ as reference;
Case 4 If $\frac{\Pr(s|Q)}{\Pr(s)}$ is greater than 1, then s occurs more often given Q than without Q, hence s and Q are correlated.

It is clear that in the context of finding relevant databases, we are only interested in the last case. Therefore, the logarithm function is taken since $log\frac{\Pr(s|Q)}{\Pr(s)}$ is either 0 or less than 0 for cases 1 and 3. Adding a weight $\Pr(s|Q)$ to the measure, we make sure that in case 2, RF is also close to 0. $\Pr(s|Q)$ also indicates how valid the correlation between s and Q is. Another weight $\Pr(Q)$ is introduced to take into account how frequently Q occurs in the database. This is because

Case 5 If $\Pr(Q)$ is close to 0, Q seldom occurs in the database, hence it may not be statistically significant to conclude something related to Q.

Thus, we develop a measure RF that is able to distinguish all the five cases. The higher the value RF, the more relevant a selector s with respect to query Q. If RF for all selectors of a database is close to or less than 0, the database is irrelevant to Q.

With the above definition about the relevance factor of selectors, we can have a definition about the relevance of databases.

A selector s is relevant to Q if $RF(s, Q) > \delta$, where $\delta(> 0)$ is the given threshold.
A table is relevant to Q if there exists at least one selector s_j, $(A_i \; relop \; C)$ where s_j is relevant to Q.

Now we have a quantitative approach to distinguishing irrelevant databases from the relevant ones. Let us compute the values of RF for the example databases in the previous section. Given *Q: Ethnic-Group* = Chinese. The maximum RF in Table 1 value is 0.088. If we set δ to 0.1, the database will be considered as irrelevant one since there does not exist any selector relevant to Q. With the same value of δ, two relevant selectors in Table 2, *main food* = *rice* and *drink* = *tea* with RF values as 0.294 and 0.278 respectively, are considered relevant to Q. We conclude that the database is relevant. From the example of these tables, we can see that the definition can tell relevant database from irrelevant one.

One issue in using the relevance measure is how to choose the threshold δ. The definition of RF consists of two parts: $\Pr(s|Q) \times \Pr(Q)$ - the term before the logarithm and $\Pr(s|Q)/\Pr(s)$ - the logarithmic term. In our definition, the selectors with larger values for both parts will be considered more specific to a query. In Figure 1, we draw $\Pr(Q \wedge s) \times log \Pr(s|Q)/\Pr(s) = \delta_i$ for $\delta_i = 0.05, 0.1, 0.2$ and 0.5 using $\Pr(Q \wedge s)$ and $\Pr(s|Q)/\Pr(s)$ as variables for the y-axis and x-axis respectively. For a threshold value δ_i , a database is identified as relevant one only if it contains at least one selector s such that $\Pr(Q \wedge s)$ and $\Pr(s|Q)/\Pr(s)$ fall into the right-upper region of the curve in Figure 1. However, $\Pr(Q \wedge s)$ is usually not very high in real databases, especially in large databases. The value of RF will decrease rapidly as the value of $\Pr(Q \wedge s)$ decreases. So we need to lower the threshold when database size is large.

For most data mining tasks, $\Pr(Q \wedge s)$, i.e., the percentage of samples for which both Q and s hold should not be too low, since this expectation value indicates the relative significance in terms of statistics. But it is also not obvious what the lower bound of $\Pr(Q \wedge s)$ is. In order to make sure that $\Pr(Q \wedge s)$ be not too small, the query predicate should cover a sufficient number of tuples in the database in the first place, i.e., $\Pr(Q)$ should not be too small. Usually, we know the total number of samples in a database. Therefore it is relatively easy to specify a lower bound of $\Pr(Q)$ (the percentage of samples covered by a query). We can prove that, given $\Pr(Q)$ the following holds: $RF \leq \Pr(Q) \, log \, (1/\Pr(Q))$

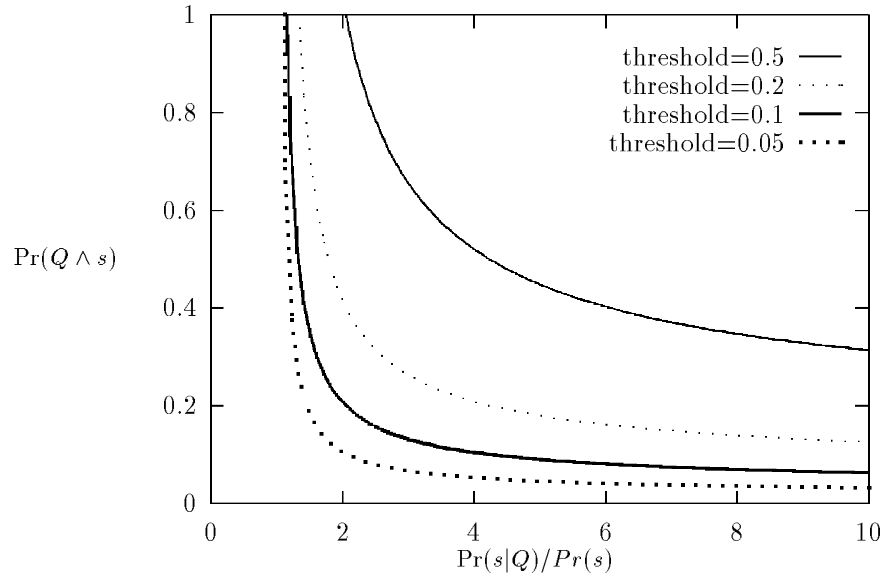

Fig. 1. $\Pr(Q \wedge s)$ as the function of $\Pr(s|Q)/\Pr(s)$ on different threshold values

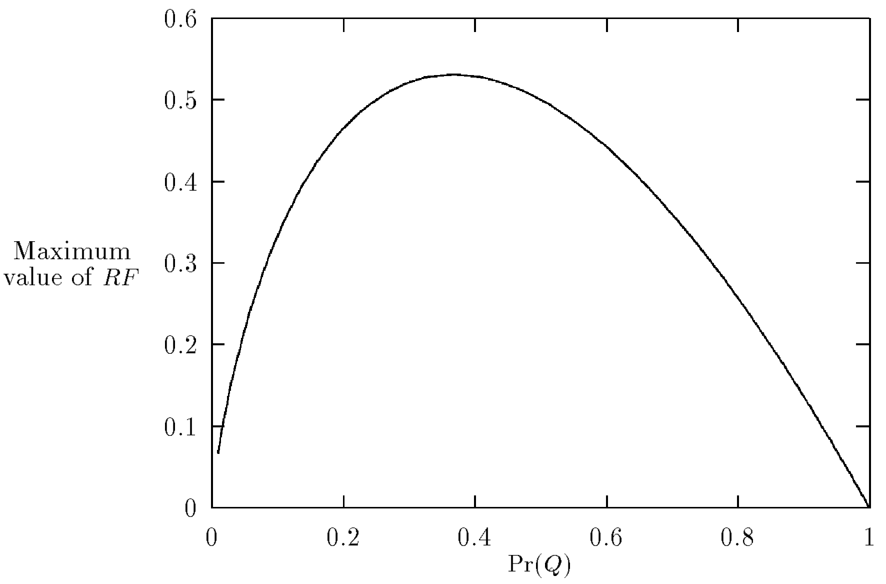

Fig. 2. Maximum value of RF as the function of $\Pr(Q)$

Therefore, given an estimate of $\Pr(Q)$, we can find the maximum value of RF from Figure 2. To identify a relevant database, we can choose δ depending on our expectation of the degree of relevance. With a small δ, more databases will be considered as relevant. It might be reasonable to set threshold value δ about one-third of the maximum value of RF. In the previous example, Q is *ethnic-group* $=$ *Chinese* and $\Pr(Q)$ is 40%. The corresponding maximum value of RF is about 0.50. If we choose 0.16 as the threshold value for both δ, RF values in the second database all are greater than 0.16 and identified as the relevant one. If we check the values of RF in the first database, neither of them is greater than 0.16. Therefore it is irrelevant.

4 Identifying Relevant Databases

With the relevance factor RF, we can determine whether a database is relevant to a query predicate before applying data mining algorithms. To compute $RF(s,\ Q)$, we only need to count three values, $\Pr(Q)$, $\Pr(s)$ and $\Pr(s \wedge Q)$. To determine whether a database is relevant to Q, we need to test all selectors, which can be done by scanning the table once. Let us assume that there are $m+1$ attributes, $A_0, ..., A_m$. Attribute A_0 is referenced by Q. For each of the other attributes, A_i, $1 \leq i \leq m$, a table S_i is maintained to keep track of selectors and the related counters. An entry of S_i is a triple of $(S\_value, S\_counter, SQ\_counter)$. S_value is the value of selector, $S\_counter$ records the number of tuples for which $A_i = S\_value$ is true. $SQ\_counter$ records the number of tuples for which both Q and $A_i = S\_value$ are true. With S_i table, we can determine the relevance of a database by scanning it once. This process has two parts: (1) reading each record in the database; and (2) searching for the entry of selectors and update the counters for each attribute in the record. The cost of part 1 is proportional to the number of records in the database. As for part 2, with a proper data structure, for example, using hashing function for the selector tables, the search for selector entries can be kept as constant, regardless of the number of selectors of an attribute. Therefore, the run time for the overall calculation is $O(NM)$ where N is the number of records in the database and M is the number of attributes.

5 Experiments

In order to have a better understanding of the issue of relevance and the relevance factor, we conducted some preliminary experiments on real-world databases using the relevance measure. To simplify the task of finding meaningful databases, we transform the problem of identifying relevant databases from multiple databases to a problem of determining the relevance of a single database to different query predicates. One of the databases tested is the NSERC (Natural Science and Engineering Research Council of Canada) research grant database. The database contains 8 tables. We take the largest table, NSERC table, as the example. The table contains information on amount of awards with 14 attributes as follows: *Id, Sysid, Sortname, Dept, Organization-id, Fyr, Compyr, Award, Grand-code, Ctee, Install, Acd3, Discipline-code* and *Cnt*.

One of our objectives is to determine whether the table contains interesting information related to the grant amount *Award*. First we need to determine the threshold. We estimated the maximum RF as follows: The total number of tuples in the table is 18510. To make a query predicate have statistical significance, we require that the query covers at least 3% of the tuples (about 500). So we set the lower bound for $\Pr(Q)$ as 0.03. From Figure 2 we can find that the maximum RF is about 0.15. Therefore we set the threshold value as 0.05.

Table 3. Experiments with NSERC data

Query predicate Q	Selector S where $RF(s,Q) > \delta$	$RF(s,Q)$
$0 \leq Award < 20000$	*Grand-code* =OGPIN	0.096328
	Sysid =G	0.088566
	Acd3 =1201	0.060338
$20000 \leq Award < 40000$	*Grand-code* =OGPIN	0.091384
	Sysid =G	0.091384
$40000 \leq Award < 60000$	Nil	
$60000 \leq Award < 90000$	Nil	
$90000 \leq Award$	Nil	

Since attribute *Award*, the amount of award is continuous, we construct query predicates in the form of: $a_1 \leq Award < a_2$. Table 3 lists the testing results of five query predicates. We can see that, for two queries, with the award amount between [0, 20000) and [20000, 40000), we obtain some selectors whose RF values are greater than the specified threshold 0.05. For other three query predicates, no such selector is obtained. In other words, NSERC table is relevant to the first two queries, but irrelevant to the last three queries. That is, we are able to find interesting patterns about award amount in the first two ranges but not in the last three.

To verify this, let us take a close look at selector *Grand-code* = OGPIN (OGPIN refers to individual research grant type), since the RF value of *Grand-code* = OGPIN for both query predicates are greater than the threshold. Figure 5 shows the distributions of amount of award for top five grant codes: OGPIN, EQPEQ, INFIF, SAPIN, and OGPGP. The area between a curve for a grant code and the x-axis is the number of tuples with that code. As we can see, most tuples with grant code OGPIN have the amount of Award in the range of (0, 40000), that is the distribution of award amount for OGPIN has a speciality within this range, therefore the table is identified as relevant to the queries in these two ranges. Viewing these distributions and their shapes can also determine which code is relevant to the query. However, it would involve human interpretation. It is not practical when the number of codes is large and with many databases to examine. Our measure implements human interpretation in an automatic manner.

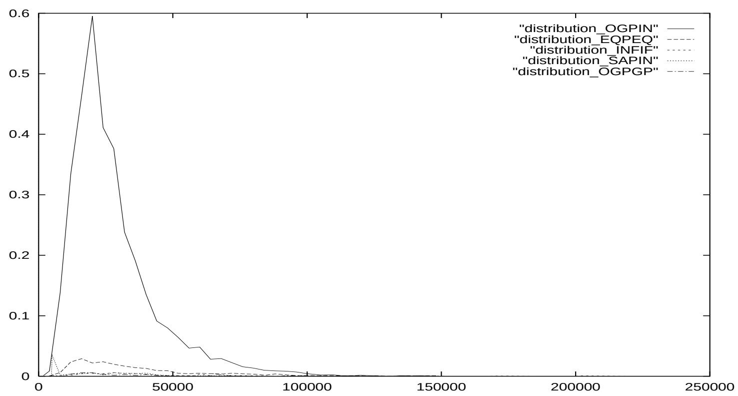

Fig. 3. Distributions of amount of Award

6 Related Work

To our knowledge, little work on mining from multidatabase has been reported in the literature. However, research work on interestingness of discovered knowledge seems quite related to this work, though the objectives and methods are different.

As mentioned before, applying data mining techniques to a given database can usually generate some knowledge in the forms such as patterns or rules. Typically, the number of such patterns is large and only a small fraction of which may be of interest to the users. Research work on interestingness of knowledge tried to distinguish the potentially interesting patterns from others. Some researchers proposed approaches that determine interestingness of a discovered rule based on user's feedback [16, 10, 14, 8], hence the measurement of interestingness is *subjective*. Researchers also developed various *objective* interestingness measures based on the statistics underlying the discovered patterns [12, 7]. With the quantitative objective interestingness measure, discovered patterns can be ranked and less interesting ones can be filtered out. Kamber and Shinghal proposed a measure, IC^{++}, which is claimed to have the best property regarding evaluating the interestingness of characteristic rules.

While interestingness is an important issue for knowledge discovery, we argue that it is also important to determine whether a database contains potentially interesting information before applying data mining techniques. When we filter out those irrelevant databases, we should not discard the databases that may contain interesting information. To check on this, we apply our method to the example given by [7]. The results show that the relevance measure serves the purpose of identifying relevant databases: For the database relevant to the query, interesting rules can be found. Otherwise, the discovered rules are not interesting. Notice that mining rules is more costly than identifying relevant databases. It is quite reasonable to identifying the relevant databases at first before mining algorithms applied.

7 Conclusions

In this paper, we addressed one of the issues related to data mining from multi-database systems: identifying those databases relevant to a data mining task from a set of semantically related databases. We argue that effective data mining from multi-databases should involve a preselection phase. Data mining techniques are only applied to those relevant databases. In order to quantify the relevance of a database, a measurement, RF, relevance factor, is proposed. The beauty of RF is that, (1) its value can indicate statistical significance of different aspects of an attribute value related to the query predicate; and (2) it can be efficiently computed by scanning the database only once.

Some preliminary experiments were conducted to test the method proposed in the paper and the measure was cross validated by the state-of-art interesting measure in the literature. One future direction of research is providing a better guideline for choosing the threshold value since sometimes it may be not easy to determine a proper threshold value. Also, we look forward to more comprehensive experiments on real world data.

Acknowledgement
We like to thank Jiawei Han who makes the NSERC database available to us.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914–925, Dec 1993.
2. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression.* Wadsworth & Brooks/Cole Advanced & Books Software, 1984.
3. P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.
4. U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press / The MIT Press, 1996.
5. J. Han and Y. Fu. Attribute-oriented induction in data mining. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 399–421. AAAI Press / The MIT Press, 1996.
6. J. Hong and C. Mao. Incremental discovery of rules and structure by hierarchical and parellel clustering. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 177–194. AAAI / The MIT Press, 1991.

7. M. Kamber and R. Shinghal. Evaluating the interestingness of characteristic rules. In *Proceedings of the Second International Conference on Data Mining (KDD-96)*, pages 263–266. AAAI Press, 1996.
8. B. Liu and W. Hsu. Post-analysis of learned rules. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI-96*, pages 828–834. AAAI press/ The MIT press, August 1996.
9. J. A. Major and J. Mangano. Selecting among rules induced from a hurricane database. In G. Piatetsky-Shapiro, editor, *Proceedings of AAAI-93 workshop on Knowledge Discovery in Databases*, pages 28–44, 1993.
10. C. J. Matheus, G. Piatetsky-Shapiro, and D. McNeill. Selecting and reporting what is interesting. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 495–514. AAAI Press / The MIT Press, 1996.
11. R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system aq15 and its testing application to three medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 1041–1045, 1986.
12. G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229–248. AAAI / The MIT Press, 1991.
13. G. Piatetsky-Shapiro, C. Matheus, P. Smyth, and R Uthurusamy. KDD93: progress and challenges. In *AI magazine*, pages 77–87, Fall 1994.
14. G. Piatetsky-Shapiro and C. J. Matheus. The interestingness of deviations. In *AAAI-84 Workshop on Knowledge Discovery in Databases*, pages 25–36, 1994.
15. J. R. Quinlan. *C4.5: Program for machine learning*. Morgan Kaufmann, 1993.
16. A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 275–281, 1995.
17. R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1996.
18. R. Zembowicz and J. M. Zytkow. From contigency tables to various forms of knowledge in databases. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 329–349. AAAI Press / The MIT Press, 1996.