Towards Multidatabase Mining: Identifying Relevant Databases

Huan Liu, Hongjun Lu, Jun Yao *

hliu@asu.edu, luhj@cs.ust.hk.

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 is where we should start mining. It is not true that the more databases, the better for data mining. It is only true when databases involved are relevant to a task at hand. 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 measure of relevance is thus proposed for mining tasks with an objective to find patterns or regularities about certain attributes. An efficient algorithm for identifying relevant databases is described. Experiments are conducted to verify the measure's performance and to exemplify its application.

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

*Published in IEEE TKDE 13:(4), 541–553, July/August, 2001.

1 Introduction

With more and more databases created, an increasingly pressing issue is how to make efficient use of them. To address the issue, there is a recent surge of research interest on knowledge discovery and data mining [1, 10, 7, 33]. While researchers are trying to develop efficient algorithms to cope with large volumes of data, little work has been devoted to the *data aspect* in the knowledge discovery process. In most organizations, data is rarely specially collected and stored for the purpose of mining knowledge, but usually as the byproducts of other tasks [32]. Furthermore, with the development of technologies, it is not uncommon that an organization has a large number of database systems and diverse data sources.

Although most data mining algorithms assume a single data set, for real world applications, practitioners have to face the problem of discovering knowledge from multiple databases. In order to do so, one way is to employ a brute force approach to join the available 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 applications. First, database integration itself is still a problematic area, especially where the source domains differ. Second, all tables with foreign key references need to be joined together to produce a single combined table. The size of the resulting table, in terms of both the number of records and the number of attributes, will be much larger than the original individual tables. The increase of data size not only prolongs the running time of mining algorithms, but also affects the behavior of mining algorithms. From the viewpoint of statistics, joining one relevant database with an irrelevant one will result in a more difficult task to find useful patterns as search space is enlarged by irrelevant attributes. For example, there are two binary-valued databases and each has N attributes, assuming that one database is irrelevant and $N/2$ attributes can be found in both databases. Simply working on the relevant database, the hypothesis space is 2^{2^N}; after joining, it is $2^{2^{3N/2}}$. For this simplified analysis, we have not yet considered the factor of missing values due to joining. This factor will certainly increase the difficulty of data mining, too. Third, if databases are joined and data mining algorithms are applied, the users face the problem of identifying interesting patterns from a large number of discovered rules. In practice, it is too easy to discover a huge number of patterns in a database [20, 26, 28], however, it is difficult for users to search in all the discovered patterns for useful ones. The redundant, useless or uninteresting patterns can be even more easily generated when there are quite a number of databases irrelevant to the mining task. Therefore, as in any effective knowledge discovery process, the first important step in mining multiple databases is indeed to select those databases that are *relevant* to a 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 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. It is important to note that unlike the conventional query processing, our task is to select the databases relevant to a given query predicate rather than identifying the databases matching the query.

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.

By being reliable and significant, we mean that the information is consistent and accurate to a certain degree, and not due to sheer chance even when its occurrence is not frequent.

We start mining tasks with query predicates for two reasons: (1) although users may not exactly know what kind of knowledge to mine, they usually have certain rough ideas. Queries are easy to form, and they are most familiar and natural to database users: user's interest, prior knowledge and intention can be conveyed by means of 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 [4, 23, 11, 27, 30, 8]. For example, classification rules, characteristic rules and association rules can be expressed in the form of $A \rightarrow B$, where both A and B are conjunctions of attribute values.

Queries can easily take care of these forms of knowledge.

There are two key issues concerning solutions to the selection problem: (1) the solution should be efficient in using databases, and (2) only the relevant databases should be chosen for mining tasks. Note that it is not our intention to propose yet another data mining method [31]. What we are concerned about 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 multiple 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 illustrate the issue of relevance. A relevance measure that can tell the relevant tables from the irrelevant ones is presented in Section 3. 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. Finally, Section 7 concludes the paper.

2 A Motivating Example

Before we design the relevance measure for identifying relevant databases, let's look at a simple example. We will use two small databases to argue that for data mining tasks involving multiple databases, it is better to identify relevant database first before applying mining techniques.

2.1 Two Artificial Databases

Our example databases are about people's diet habits. It is user's intention to discover some *useful* knowledge about Chinese diet habits from the databases. Here and in most KDD applications, being useful means something that should potentially lead to some useful actions by a

Table 1: Database 1 of Diet Habits

ID	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	Russian	never	no	seldom	beef
950122	Russian	often	no	sometimes	beef
940227	Russian	sometimes	no	sometimes	chicken
938567	Russian	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

Table 2: Database 2 of Diet Habits

ID	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	Russian	bread	3	coffee	no
950122	Russian	bread	3	cola	no
940227	Russian	rice	3	cola	yes
940121	Russian	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

Table 3: Analysis of data in Database 1

group		Chinese	Russian	Indian	Total
No Tuples		6	4	5	15
alcohol	never	2	1	1	4
	sometimes	3	2	2	7
	often	1	1	2	4
on-diet	no	6	4	5	15
snack-between-meals	seldom	2	1	1	4
	sometimes	3	2	4	9
	often	1	1	0	2
favorite-non-veg.	pork	1	1	1	3
	chicken	1	1	1	3
	beef	1	2	1	4
	fish	2	0	2	4
	egg	1	0	0	1

user [6]. 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. Let us look at Table 1 first. From the attribute values summarized in Table 3, we observe the following:

- No person is on diet, regardless of his ethnic group;
- For both Chinese and Russian, 50% of records show that they take alcohol sometimes;
- Similarly, 50% of Chinese and Russian have snacks between regular meals; and
- It seems difficult to arrive at a convincing conclusion on favorite non-vegetable food since a very small number of tuples are available: there are five different foods listed and the largest group only has six records available.

From the above observations, we can see that, the database in fact does not contain 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 the group is Chinese, the main food is rice"

since all records about Chinese show that the main food is rice whereas records about Russian 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 information but the other does not.

2.2 Possible Solutions for Mining Multiple Databases

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 method and check if some of them can serve the purpose of selection, 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., twice as many as in the original databases; (2) the resulting table will have more tuples. In our example, the number of tuples increases to 19 (4 more than the original one); and (3) for certain tuples, 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** 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 for association rules [9], we can have rules like:

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

   with 40% support and 100% confidence. We do not think such information is really interesting since *on-diet* $=$ no in fact holds for all tuples. In other words, what we found is not specific to Chinese. We can also apply certain classification algorithms such as CART, CN2, and C4.5 [3, 4, 27] to each database specifying attribute *group* as the class label. Using C4.5, for example, we can induce classification rules from both databases whose accuracy rates are better than that of just choosing the most probable class label. Therefore, based on whether any rules can be induced, we cannot distinguish the relevant database from the irrelevant one.

From the above, commonly used methods do not seem to work for database selection, or efficiently distinguishing the irrelevant database from the relevant one. We need a new method to identify relevant databases first and to facilitate efficient mining multiple databases later.

3 Relevance Measurement

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

3.1 Relevance Factor of Selector

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 such as "drink = tea" with respect to Q such as "group = Chinese" below:

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

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 that $\frac{\Pr(s|Q)}{\Pr(s)}$ 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 when Q is given;

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. This is because

Case 5 If $\Pr(Q)$ is close to 0, then Q seldom occurs in the database. Hence it is not statistically significant to conclude that something is 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. The changes in the proportion of irrelevant information can also be seen from the RF definition (1). Assuming that $\Pr(Q)$ and $\Pr(s)$ are fixed, we can see that if Q and s are positively relevant (i.e., $\Pr(s|Q) > \Pr(s)$), RF is positive; if Q and s are negatively relevant (i.e., $\Pr(s|Q) < \Pr(s)$); if Q and s are irrelevant (i.e., independent of each other), RF is 0.

3.2 Relevant Databases

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

Definition I:

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: Group* = Chinese, Tables 4 and 5 list the RF values for all the selectors in Tables 1 and 2 respectively. $\Pr(s|Q)$ is also given as reference.

From Table 4 we can see that the maximum RF value is 0.088. If we set δ to 0.1, the database will be considered as irrelevant since there does not exist any selector relevant to Q.

Let's check Table 5. With the same value of δ, two relevant selectors, *main food* = *rice* and *drink* = *tea*, are considered relevant to Q. We conclude that the database is relevant. From the two simple illustrative examples, we can see that the measure RF can tell relevant database from irrelevant one.

Table 4: Relevance Factors of RF in Database One

No	Selector s	$RF(s,Q)$	$Pr(s\|Q)$
1	alcohol=never	0.000000	0.333333
2	alcohol=sometimes	0.019907	0.500000
3	alcohol=often	-0.017536	0.166667
4	on diet=no	0.000000	**1.000000**
5	snacks between meal=sometimes	-0.052607	0.500000
6	snacks between meal=seldom	0.042924	0.333333
7	snacks between meal=often	0.021462	0.166667
8	favorite non-veg=fish	0.042924	0.333333
9	favorite non-veg=pork	-0.017536	0.166667
10	favorite non-veg=chicken	-0.017536	0.166667
11	favorite non-veg=beef	-0.045205	0.166667
12	favorite non-veg=egg	**0.088129**	0.166667

Table 5: Relevance Factors of RF in Database Two

No	Selector s	$RF(s,Q)$	$Pr(s\|Q)$
1	main food=rice	**0.294786**	**1.000000**
2	regular eating times=3	0.000000	**1.000000**
3	drink=tea	**0.278834**	0.833333
4	drink=cola	-0.045205	0.166667
5	vegetarian=no	0.019631	0.833333
6	vegetarian=yes	-0.017536	0.166667

In order to determine whether a database is relevant to a given query predicate Q, we need to examine every attribute and values. For categorical attributes, selectors can be formed for each distinct value. For non-categorical attributes, we can either employ the idea of ranges or discretize attribute values [19], which divides attribute values into a limited number of intervals and maps values in each interval to an integer. When discretization is needed, we resort to Chi2 - a discretization system based on the χ^2 statistics and an inconsistency measure [17]. The system is available for research purposes upon request.

3.3 Choosing the Threshold

One issue in using the relevance measure is how to choose the threshold δ. In this section, we study in detail how to choose threshold value for RF. 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. As mentioned before, $\Pr(s|Q)$ indicates how strong the correlation between s and Q is. By multiplying $\Pr(s|Q)$ by $\Pr(Q)$, the first part is in fact $\Pr(Q \wedge s)$, i.e., the ratio between the number of tuples for which both Q and s hold and the total number of tuples in the database. The second part $\Pr(s|Q)/\Pr(s)$, which can be estimated using the ratio of the number of tuples for which both s and Q hold and the number of tuples where s holds, reflects specificity of s to Q with respect to the whole database. 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 to satisfy $RF = \Pr(Q \wedge s) \times \log \Pr(s|Q)/\Pr(s) \geq \delta_i$, the values of $\Pr(Q \wedge s)$ and $\Pr(s|Q)/\Pr(s)$ should be in the right-upper part of the curve ($y = \delta_i/\log x$). In other words, a database is identified as relevant 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. Note that the value of $\Pr(s|Q)/Pr(s)$ which reflects the degree of deviation of the posterior from the prior could be very large. 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, $\Pr(Q \wedge s)$, the percentage of tuples for which both Q and s hold,

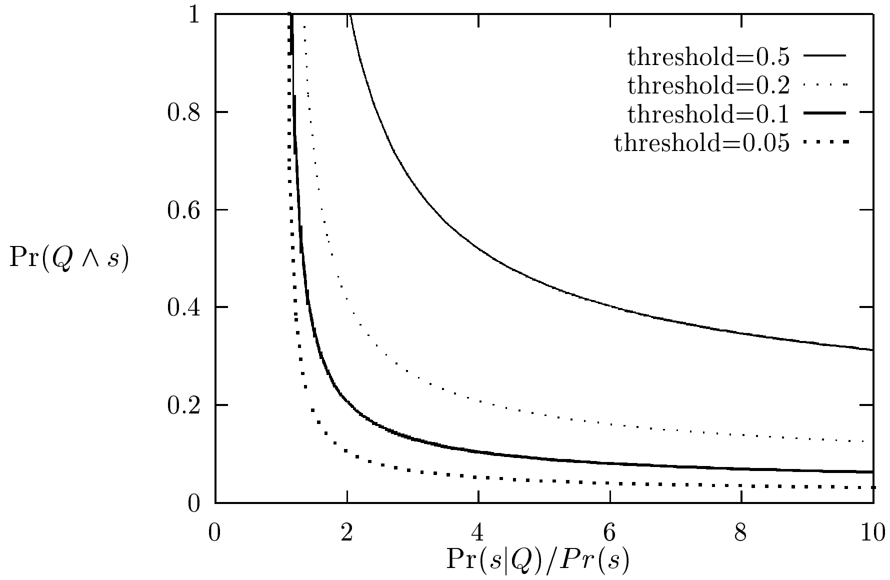

Figure 1: $\Pr(Q \wedge s)$ as the function of $\Pr(s|Q)/\Pr(s)$ on different threshold values

should not be too low, since this value indicates the relative significance in terms of statistics. However, it is still unclear 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 tuples in a database. Therefore it is relatively easy to specify the lower bound of $\Pr(Q)$ (the percentage of tuples covered by a query). We can show that, with the estimated $\Pr(Q)$, we can obtain an upper bound for RF.

Theorem I. Given $\Pr(Q)$, the following holds:

$$RF \;\leq\; \Pr(Q) \;\log\; (1/\Pr(Q)) \tag{2}$$

Proof:

Let us define $\Pr(Q) = \frac{N_q}{N}$; $\Pr(s) = \frac{N_s}{N}$; $\Pr(s|Q) = \frac{N_{s\wedge q}}{N_q}$, then

$$\begin{aligned} RF &= \Pr(Q \wedge s) \log(\frac{N_{s\wedge q}}{N_q} \times \frac{N}{N_s}) \\ &= \Pr(Q \wedge s) \log(\frac{N_{s\wedge q}}{N_s} \times \frac{N}{N_q}) \end{aligned}$$

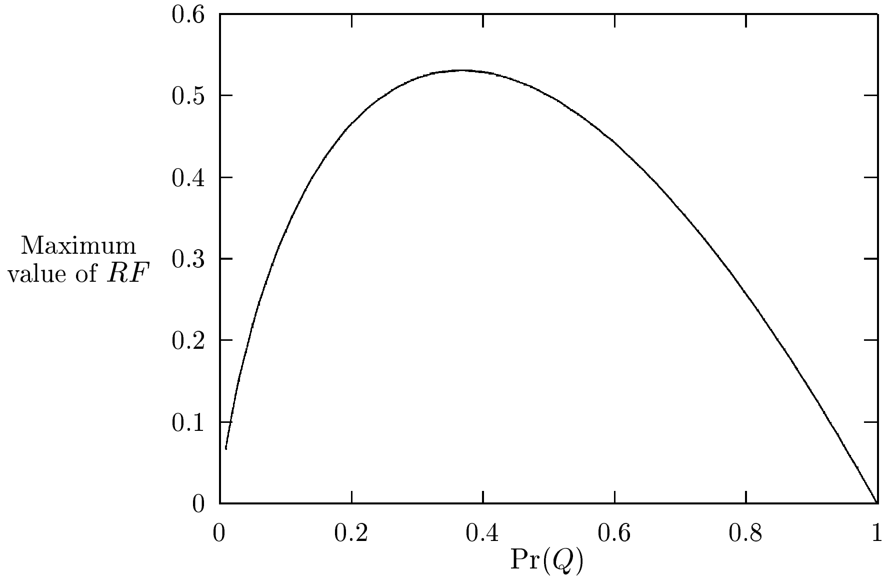

Figure 2: Maximum value of RF as the function of $\Pr(Q)$

$$= \Pr(Q \wedge s) \log(\frac{N_{s\wedge q}}{N_s} \times \frac{1}{\Pr(Q)})$$

Because $\Pr(Q \wedge s) \leq \Pr(Q)$ and $\frac{N_{s\wedge q}}{N_s} \leq 1$, hence

$$RF \leq \Pr(Q) \log \frac{1}{\Pr(Q)}.$$

Therefore, given an estimate of $\Pr(Q)$, we can find the maximum value of RF from Figure 2. This gives a range within which a possible threshold can be chosen. We can choose δ depending on our expectation of the degree of relevance. In general, with a smaller δ, more databases will be considered as relevant. We suggest to initially set δ as one-third of RF_{max}. It can then be adjusted accordingly. In our example, Q is $group = Chinese$ and $\Pr(Q)$ is 40%. The corresponding maximum value RF_{max} is about 0.50 from Figure 2. If we choose $\delta = 0.16$ for both databases, two RF values in the second database are greater than 0.16. According to Definition I, the second database will be identified as relevant. If we check the values of RF in the first database, none is greater than 0.16. Therefore, it is irrelevant.

4 Algorithm for Identifying Relevant Databases

With the relevance measure RF, we can determine whether a database is relevant to a query predicate before applying data mining algorithms. In this section, we present an algorithm that determines whether a database is relevant to a query predicate Q according to Definition I.

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. The algorithm is listed in Figure 3.

Algorithm RelevantDB scans the table and finds all possible selectors and inserts them into the S tables (line 1-15). Related counters are updated. After the scanning, function $ComputeRF$ is called for each selector to compute its relevance factor, RF (line 18-20). If RF is greater than the given threshold, the algorithm stops and returns $TRUE$ (line 21-22). If no selector has RF greater than δ, $FALSE$ is returned (line 25), i.e., the database is irrelevant to Q. As an example, Figure 4 lists the contents of the S tables for each attribute after applying algorithm *RelevantDB* to the first example database in Section 2. From the S tables, relevance factor RF for each selector can be computed.

Algorithm *RelevantDB* 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 some 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 algorithm is $O(NM)$ where N is the number of records in the database and M is the number of attributes.

Algorithm RelevantDB

input:	$D(A_0, A_1, ..., A_m)$,	// the database
	$Q : A_0$ *relop* Q_value,	// the query predicate
	δ : threshold;	// the threshold
output:	$True$ or $False$;	// TRUE if D is relevant, FALSE otherwise
$Q\_counter$:	number of tuples for which Q holds;	

```
begin
    /* search database to establish counters */
    foreach record R do begin
        if R.A_0 = Q_value then
            increase Q_counter by 1;
        for attribute A_i, 1 ≤ i ≤ m do begin
            search record t in Si where t.S_value = R.A_i;
            if found then
                increase t.S_counter by 1;
            else
                insert (R.A_i, 1, 0) into S_i;
            if R.A_0 = Q_value then
                increase t.SQ_counter by 1;
        end
    end;

    /* compute RF(s, Q) */
    foreach table S_i(1 ≤ i ≤ m) do
        foreach record t do begin
            RF = ComputeRF(Q_counter, t.S_counter, t.SQ_counter);
            if RF ≥ δ then
                return TRUE;
        end;

    return FALSE;
end;
```

Figure 3: Algorithm RelevantDB

alcohol

S_value	*S_counter*	*SQ_counter*
never	4	2
sometimes	7	3
often	4	1

on_diet

S_value	*S_counter*	*SQ_counter*
no	15	6

snack-between-meals

S_value	*S_counter*	*SQ_counter*
seldom	4	2
sometimes	9	3
often	2	1

favorite-non-veg.

S_value	*S_counter*	*SQ_counter*
pork	3	1
chicken	3	1
beef	4	1
fish	4	2
egg	1	1

Figure 4: Applying RelevantDB to Example DB1

5 Experiments

In order to have a better understanding of the issue of relevance and the relevance factor, we conducted two sets of experiments on the relevance measure.

5.1 Benchmark data sets

There are a large number of benchmark data sets available for evaluating data mining algorithms [22]. However, no similar data sets are available for evaluating measures such as the one proposed in this paper. To obtain multiple possibly relevant data sets, one approach is to vertically partition a data sets into a number of data sets each of which contains a certain number of attributes. It is hoped that some data sets obtained are relevant to a particular query predicate and some are not so that the effectiveness of the proposed measure can be evaluated. In experiments, we use the data sets from the UC Irvine repository [2]. The multiple databases are generated from the Wisconsin breast cancer data and mushroom data. We use this procedure to construct multiple databases:

1. run a feature ranking algorithm - Relief [14, 5] to rank the attributes of a data set[1];

2. divide (vertically partition) the data set into several data sets according to the rankings of the features; and

3. specify queries according to class labels and see which data sets are identified as relevant.

We should expect that data sets with important features identified by Relief are relevant to some queries regarding the class.

Results of the Wisconsin breast cancer data:

The data has 10 attributes and the first attribute is ID. Relief ranks importance attributes of the Breast Cancer data in a decreasing order: 7, 3, 4, 8, 6, 2, 5, 9, 10, 1. So we obtain three data sets: D1 has attributes 7, 3, 4 plus class, D2 has attributes 8, 6, 2 plus class, and D3 has attributes 5, 9, 10, 1 plus class.

The class has two values: "benign", "malignant". In general, people are more concerned about the instances with "malignant" class value. If we specify our query as class = "malignant" and choose $\delta = 0.17$ for three data sets (there are 241 instances which has class label = "malignant", so $\Pr(Q) = 0.34$ and $RF_{max} = 0.51$, thus $\delta = 0.51/3 = 0.17$) , we only got one selector: Bare Nuclei (attribute 7) = "10" which RF value is 0.277395 from data set 1. Thus, data set D1 was identified as relevant. When we specified the query as class = "benign", three data sets are identified as relevant since selectors with RF larger than 0.17 are obtained from three data sets (Table 6). We ran C4.5rules [27] and confirmed that classification rules for "benign" cases contain attributes 2, 3, 5, 7 and 9 from all the three data sets.

Results of the Mushroom data:

This data has 22 attributes. Relief ranks importance order of attributes as: 5, 20, 11, 8, 19, 4, 10, 22, 9, 12, 13, 21, 7, 3, 2, 15, 14, 18, 6, 17, 16, 1. We divide the Mushroom data into four data sets: besides a class attribute, D1 has attributes 5, 20, 11, 8, 19, D2 has attributes 4, 10, 22, 9, 12, 13, D3 has attributes 21, 7, 3, 2, 15, 14, and D4 has attributes 18, 6, 17, 16, 1.

We posed queries based on both class values: "edible", "poisonous" and specified $\delta = 0.16$. The values of $\Pr(Q)$ for "edible", "poisonous" are 0.518 and 0.482, and their RF_{max} values are

[1] Relief used here is our implementation. It relies on repeated sub-sampling to weigh each feature. It needs to access data many times and shows good results in removing irrelevant features.

Table 6: RF values for selectors found for Query class = "benign" of the Wisconsin breast cancer data.

D1	bare nuclei (attribute 7) = 1	RF=0.307321
	uniformity of cell size (attribute 3) = 1	RF=0.323374
	uniformity of cell shape (attribute 4) = 1	RF=0.302165
D2	single epithelial cell size (attribute 6) = 2	RF=0.270725
D3	marginal adhesion (attribute 5) = 1	RF=0.263845
	normal nucleoli(attribute 9) = 1	RF=0.270205

Table 7: RF values for selectors found for Query 1 of the mushroom data.

D1	odor (attribute 5)= "foul"	RF=0.279920
	spore-print-color (attribute 20)= "chocolate"	RF=0.196877
	gill-size (attribute 8)= "narrow"	RF=0.240120
	ring-type(attribute 19)= "large"	RF=0.167952
D2	bruises (attribute 4)= "no"	RF=0.212519
	gill-color (attribute 9)= "black"	RF=0.223936
	stalk-surface-above-ring (attribute 12)= "silky"	RF=0.263952
	stalk-surface-below-ring (attribute 13)= "silky"	RF=0.255164
D3	population (attribute 21)= "several"	RF=0.19225
D4	none	

Table 8: *RF* values for selectors found for Query 2 of the mushroom data.

D1	odor (attribute 5) = "none"	RF=0.377183
	gill-size (attribute 8)= "broad"	RF=0.208157
	spore-print-color (attribute 20)= "brown"	RF=0.166312
	ring-type (attribute 19)= "pendant"	RF=0.239353
D2	bruises (attribute 4)= "bruises"	RF=0.221617
	stalk-surface-above-ring(attribute 12)= "smooth"	RF=0.197662
	stalk-surface-below-ring (attribute 13)= "smooth"	RF=0.172113
D3	none	
D4	none	

about 0.48, one third of which is 0.16 as δ. For Query 1: class = "poisonous" and Query 2: class = "edible", we have found the selectors shown in Tables 7 and 8 respectively. From the results, it is clear that for Query 1: class = "poisonous", data sets D1, D2 and D3 are identified as relevant. It should be noted, however, that for D1 and D2, there are four selectors found for each, but there is only one selector found (population (attribute 21) = "several", RF = 0.19225) in D3. According to the ranking by Relief, attribute 21 is the most important attribute in D3. As for Query 2: class = "edible", data sets D1 and D2 are identified as relevant, D3 and D4 are irrelevant. As we see, the experimental results are consistent with the relevance of the data sets in the two benchmark data sets. We also ran C4.5rules to confirm if any features in D3 and D4 are found in the classification rules. The result is negative: only features in D1 and D2 are contained in the rules. After the above experiment, we have some understanding of the measure. Now let us engage another database in further experiment.

5.2 A real-world data set

The NSERC (Natural Science and Engineering Research Council of Canada) research grant database is used. It contains 8 tables. We briefly explain the largest table, NSERC table. The table contains information on amount of awards with 14 attributes as follows: *Id, Sysid, Sort-*

Table 9: Experiments with NSERC data

Query predicate Q	Selector S where $RF(s,Q) > \delta$	$RF(s,Q)$
$0 \leq Award < 20000$	*Grant-code* =OGPIN	0.096328
	Sysid =G	0.088566
	*Acd*3 =1201	0.060338
$20000 \leq Award < 40000$	*Grant-code* =OGPIN	0.091384
	Sysid =G	0.091384
$40000 \leq Award < 60000$	Nil	
$60000 \leq Award < 90000$	Nil	
$90000 \leq Award$	Nil	

name, Dept, Organization-id, Fyr, Compyr, Award, Grant-code, Ctee, Install, Acd3, Discipline-code and *Cnt*. 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 that of different query predicates for a single database.

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. As we discussed in Section 3.3, 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, one third of it is 0.05 as δ.

Since the amount of award is continuous, we construct query predicates in the form of ranges such as $a_1 \leq Award < a_2$. Table 9 lists the testing results of five query predicates. 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 the other three query predicates, no such selector is obtained. In other words, the NSERC table is relevant to the first two queries, but irrelevant to the last three queries. That is, we can find interesting patterns about the award amount in the first two ranges but not in the last three.

To verify this finding, let us take a closer look at selector *Grant-code* = OGPIN (OGPIN

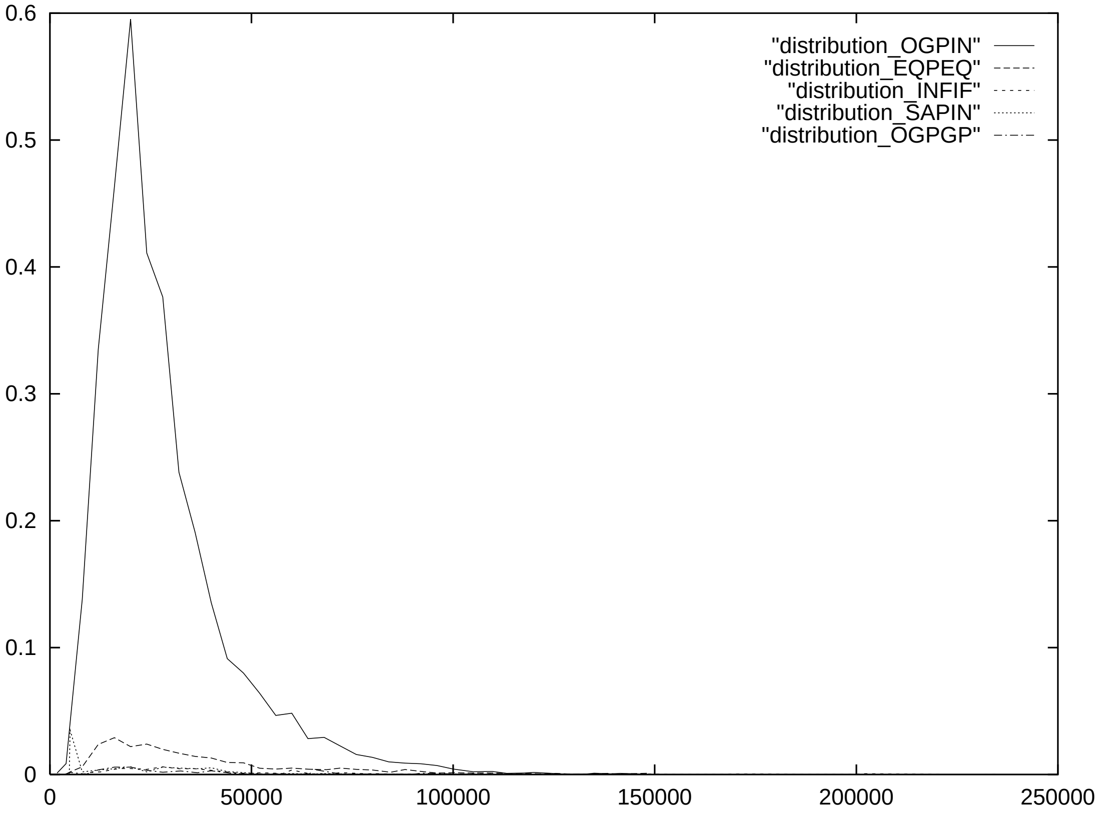

Figure 5: Distributions of Award Amount

refers to an individual research grant type), since the *RF* value of *Grant-code* = OGPIN for both query predicates are greater than the threshold. Figure 5.2 shows the distributions of award amount 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 award amount in the range of (0, 40000), that is the distribution of the award amount for OGPIN is special 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 impractical when the number of codes is large and with many databases to examine. Our measure serves the purpose of human interpretation in an automatic manner.

6 Related Work

To our knowledge, little work on mining multiple databases 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 was mentioned earlier, applying data mining techniques to a given database can usually generate some knowledge (or patterns, rules). Typically, the number of such patterns is large and only a small fraction of which may be of interest to the user. Research work on interestingness of knowledge tries to distinguish the potentially interesting patterns from others. There are a number of approaches studying the interestingness problem. Some researchers proposed approaches that determine interestingness of a discovered rule based on user's feedback [28, 21, 15], hence the measurement of interestingness is *subjective*. Some researchers developed various *objective* interestingness measures based on the statistics underlying the discovered patterns [25, 13, 12]. With the quantitative objective interestingness measure, discovered patterns can be ranked and less interesting ones can be filtered out. Kamber and Shinghai [13] gave a good survey on such objective interestingness measures. They also proposed a measure, IC^{++}, which is claimed to have the best property regarding evaluating the interestingness of characteristic rules.

Table 10: Relevance factor for rules in Kamb96

Rule	Interesting by IC^{++}	$RF(B_i, A_i)$
$A_1 \rightarrow B_1$	No	0.00
$A_2 \rightarrow B_2$	Yes	0.43
$A_3 \rightarrow B_3$	Yes	0.45
$A_4 \rightarrow B_4$	No	0.00

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 relevance measure to the example given in [13]. They presented preliminary results about interestingness of characteristic rules mined from a synthetic database of 4,000 tuples described by 10 attributes. Among four mined rules, IC^{++} indicated that two are interesting and the other two are not interesting. Based on the statistics given, we computed the relevance factors for each rule by taking its left hand side as the query predicate and the right hand side as the selector. The results are shown in Table 10. It can be seen that, our method indeed serves the purpose: If RF is small (0 in case of Rule 1 and 4), the database is irrelevant, and there is no need to mine patterns related to the predicates (A_1, A_4). Recall that mining rules is more costly than calculating RF. If we insist to proceed, there might be some rules found ($A_1 \rightarrow B_1$ and $A_4 \rightarrow B_4$), but by applying the interestingness measure IC^{++}, they would be considered uninteresting as shown in the example given by [13]. A major difference between IC^{++} and RF is that the former is applied as a post-processing filter, and the latter a pre-processing filter.

Another piece of related work can be found in [29] in which they developed a cross entropy based approach to rule induction from databases. It not only learns rules for a given concept (classification), but it simultaneously learns rules relating multiple concepts - known as generalized rule induction. Briefly put, the ITRULE algorithm repeatedly modifies the K rules by calculating their J-measure through depth-first search over possible left-hand sides, starting with the first-order conditions. The commonality between our measure and the J-measure is that both rely on information computation. Our measure stems directly from the need to cover the five cases in Section 3 in order to select databases instead of rule induction. An important difference is that our measure requires just one pass over a database and no revision is required.

7 Conclusion

In this paper, we address one issue of 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 multiple databases should involve a pre-selection phase. In that phase, those databases relevant to the mining task are identified. Data mining techniques are then only applied to those relevant databases. The pre-selection phase is essential since joining irrelevant database with relevant ones will enlarge the search space, reduce the statistical significance of knowledge to be discovered, and may lead to too many uninteresting patterns. This work underlines our belief that less can mean more [18].

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. Given also is an algorithm that implements the measurement to check whether a database is relevant to a given query predicate. Experiments were conducted on both benchmark datasets and a real-world database to test the method proposed in the paper and the measure was cross validated by the state-of-the-art interesting measure (IC^{++}) in the literature.

Further work will be generalizing the measure to overcome some limitations of the measure. We consider two directions. First, our measure focuses on individual attributes due to the efficiency issue. Therefore, the measure assumes that higher order interdependence among the attributes is somehow reflected in first order attributes. However, further analysis shows the monotonic nature of RF when $\Pr(Q) \leq \Pr(s)$. It may not be so when $\Pr(Q) > \Pr(s)$. Therefore, the above assumption is still required for a general case. Investigation continues. Exhaustive search is infeasible due to its computational complexity. Educated random search may be one way out [24]. Second, moving towards attribute-based identification from selector-based. Although we presented a guideline for choosing the threshold value in Section 3, a proper threshold value is also determined by other factors. Because if query Q covers many selectors of an attribute, $\Pr(Q \wedge s)$ may be much less than $\Pr(Q)$, the upper bound of RF by which we choose the threshold may be relatively too loose. Also, it is possible that one database is identified as relevant just because it contains only one selector with some high value of RF and another database is filtered out which may contain many selectors with values of RF less than but near the threshold value. In order to prevent this from happening, we plan to introduce a modified measure that sums up RF's whose values are ϵ-within the threshold δ, where ϵ can be a function of δ, e.g., its one-third.

One immediate application of this work is for information providers to screen databases they are offered. Imagine that an information dealer for a special trade wants to buy data related to the trade. The dealer pays accordingly for relevant databases and declines irrelevant ones. Our measure RF provides a way for the dealer to screen what to buy or decline.

In addition to the practical value of this work, more opportunities open up for research. One line of research is to exploit irrelevant databases identified by the measure RF. We may probe further whether these databases can be merged for reuse. This occurs when we face a particular application with few relevant databases. Obviously, database cleaning and merging requires some methods beyond RF. Throughout this paper, we assume that a database for a specific purpose resides in one place. As the web technology advances, however, there are cases that one database is distributed in several places. Although it is always possible to consider one database instead of its portions, it is also desirable to deal with each potion. This necessitates an extension to the current form of RF as it is clear that RF cannot directly handle the databases in distributed environments.

Acknowledgments

We would like to thank Jiawei Han who has made the NSERC database available for our experiments, Manoranjan Dash for fruitful discussions, and Mohammad Mahmud Hasan for running some confirmation experiments. The comments and suggestions by three anonymous reviewers have greatly helped improve this work and shed lights on insights gained from this project.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Trans. on Knowledge and Data Engineering*, 5(6):914–925, Dec 1993.

[2] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression.* Wadsworth & Brooks/Cole Advanced & Books Software, 1984.

[4] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[5] M. Dash and H. Liu. Feature selection methods for classifications. *Intelligent Data Analysis: An International Journal*, 1(3), 1997. `http://www-east.elsevier.com/ida/free.htm`.

[6] 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.

[7] 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 495–515. AAAI Press / The MIT Press, 1996.

[8] J. Han, Y. Cai, and N. Cercone. Data driven discovery of quantitative rules in relational database. *IEEE Transactions on Knowledge and Data Engineering*, 5(1):29–40, 1993.

[9] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of the 21st VLDB Conference Zurich, Swizerland*, pages 420–431, 1995.

[10] 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.

[11] 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.

[12] F. Hussain, H. Liu, E. Suzuki, and H. Lu. Exception rule mining with a relative interestingness measure. In *Proceedings of the Fourth Pacific-Asia conference on Knowledge Discovery and Data mining (PAKDD)*, 2000.

[13] 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.

[14] K. Kira and L.A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 129–134. Menlo Park: AAAI Press/The MIT Press, 1992.

[15] 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, Menlo Park, California, 1996. AAAI Press / The MIT Press.

[16] H. Liu and H. Motoda, editors. *Feature Extraction, Construction and Selection: A Data Mining Perspective.* Boston: Kluwer Academic Publishers, 1998.

[17] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery Data Mining.* Boston: Kluwer Academic Publishers, 1998.

[18] H. Liu and H. Motoda. *Less Is More*, pages 3 – 12. In [16], 1998.

[19] H. Liu and R. Setiono. Feature selection via discretization. *IEEE Trans on Knoweldge and Data Engineering*, 9(4):642–645, July/August 1997.

[20] 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.

[21] 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.

[22] C.J. Merz and P.M. Murphy. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Irvine, CA: University of California, Department of Information and Computer Science, 1996.

[23] 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.

[24] R. Motwani and P. Raghavan. Randomized algorithms. In A.B. Tucker, Jr., editor, *The Computer Science and Engineering Handbook*, pages 141–161. CRC Press and ACM, 1997.

[25] 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.

[26] G. Piatetsky-Shapiro, C. Matheus, P. Smyth, and R Uthurusamy. KDD93: progress and challenges. In *AI magazine*, pages 77–87, Fall 1994.

[27] J. R. Quinlan. *C4.5: Program for machine learning.* Morgan Kaufmann, 1993.

[28] 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.

[29] P. Smyth and R.M. Goodman. An information theorectic approach to rule induction from databases. *IEEE Trans. on Knowledge and Data Engineering*, 4(4):301 – 316, August 1992.

[30] 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.

[31] R. Uthurusamy. From data mining to knowledge discovery: Current challenges and future directions. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 561–569. AAAI Press / The MIT Press, 1996.

[32] G. Wiederhold. Foreword: On the barriers and future of knowledge discovery. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages vii–xi. AAAI Press / The MIT Press, 1996.

[33] 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.