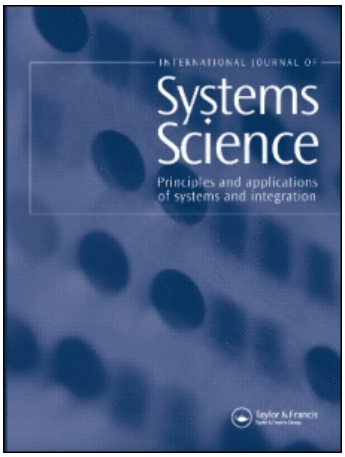
This article was downloaded by: *[Canadian Research Knowledge Network]*
On: *1 October 2008*
Access details: *Access Details: [subscription number 783016891]*
Publisher *Taylor & Francis*
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



International Journal of Systems Science
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713697751

Hierarchical grouping of association rules and its application to a real-world domain

A. An [a]; S. Khan [a]; X. Huang [b]
[a] Department of Computer Science, York University, Toronto, Ontario M3J 1P3, Canada [b] School of Information Technology, York University, Toronto, Ontario M3J 1P3, Canada

Online Publication Date: 20 October 2006

To cite this Article An, A., Khan, S. and Huang, X.(2006)'Hierarchical grouping of association rules and its application to a real-world domain',International Journal of Systems Science,37:13,867 — 878
To link to this Article: DOI: 10.1080/00207720600891661
URL: http://dx.doi.org/10.1080/00207720600891661

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.informaworld.com/terms-and-conditions-of-access.pdf

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

International Journal of Systems Science
Vol. 37, No. 13, 20 October 2006, 867–878

Taylor & Francis
Taylor & Francis Group

Hierarchical grouping of association rules and its application to a real-world domain

A. AN*†, S. KHAN† and X. HUANG‡

†Department of Computer Science, York University, Toronto, Ontario, M3J 1P3, Canada
‡School of Information Technology, York University, Toronto, Ontario, M3J 1P3, Canada

(*Received 4 September 2005; in final form 9 April 2006*)

One common problem in association rule mining is that often a very large number of rules are generated from the database. The sheer volume of these rules makes it difficult, if not impossible, for human users to analyze and make use of the rules. In this article, we propose two algorithms for grouping and summarizing association rules. The first algorithm recursively groups rules according to the structure of the rules and generates a tree of clusters as a result. The second algorithm groups the rules according to the semantic distance between the rules by making use of a semantic tree-structured network of items. We propose an algorithm for automatically tagging the semantic network so that the rules can be represented as directed line segments in a two-dimensional space and can then be grouped according to the distance between line segments. We also present an application of the two algorithms, in which the proposed algorithms are evaluated. The results show that our grouping methods are effective and produce good grouping results.

Keywords: Data-mining; Real-world application; Grouping association rules

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

1. Introduction

The problem of association rule mining that discovers an interesting class of database regularities was first introduced in Agrawal *et al.* (1993). Today, a variety of efficient association rule mining algorithms exist (Agrawal and Srikant (1994), Han *et al.* (2000) and Savasere *et al.* (1995)). However, these algorithms often produce considerable amounts of rules or patterns and create another data-mining problem. The amount of rules produced by the algorithms is often so huge that it is virtually impossible for human users to analyze these rules in order to identify useful ones. Some solutions have been proposed to overcome this problem. One line of research is constraint-based data-mining, in which only patterns that satisfy certain constraints are generated. For example, Srikant *et al.* (1997) integrated user-specified constraints that are boolean expressions over the presence or absence of items into candidate generation process of the *a priori* algorithm. Pei *et al.* (2001) incorporated constraints into the frequent pattern generation process of the *FP-tree* algorithm. These constraint-based mining algorithms not only reduce the number of generated rules, but also speed up the algorithm dramatically. Another way to reduce the number of discovered patterns is to post-prune rules according to rule structures or statistical tests. Shah *et al.* (1999) proposed a set of *pruning rules* to eliminate structurally and semantically redundant rules from the set of mined patterns. Toivonen *et al.* (1995) computed a subset of rules, called a *structural rule cover*, to reduce the number of rules and further grouped the rules in the cover using clustering. Cristofor and Simovici (2002) defined another type of rule cover, called *informative cover*, to group and

*Corresponding author. Email: aan@cs.yorku.ca

International Journal of Systems Science
ISSN 0020–7721 print/ISSN 1464–5319 online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207720600891661

summarize related rules. Liu *et al.* (1999) applied the χ^2 test to remove insignificant associations and then found a subset of unpruned rules, called *direction setting rules*, to form a summary of the discovered association rules. To prune uninteresting rules, researchers have used different measures of interestingness to assess the significance of a rule. For example, Piatetsky-Shapiro (1991) proposed to use a measure, called *RI* (Rule Interestingness), to measure the interestingness of a rule. Tan and Kumar (2000) defined a measure, called *IS*, derived from statistical correlation between the antecedent and consequent of a rule. Blanchard *et al.* (2005) used information-theoretic measures to assess association rule interestingness. Hilderman and Hamilton (1999) gave a survey of various interestingness measures used for association rules. By ranking the discovered rules according to their degree of interestingness, only interesting rules are presented and non-interesting ones are pruned.

In our previous work (Huang *et al.* 2002), we conducted an empirical study on learning association rules from a Web log data set. A huge number (tens of thousands or millions) of rules are generated and many of them are not interesting. To reduce the number of rules and identify the interesting ones, we applied some pruning rules proposed in Shah *et al.* (1999) to remove some structurally or semantically redundant rules, and used some statistical interestingness measures to identify interesting rules. However, the number of interesting rules resulting from these two steps is still large. Further organization or grouping of rules is necessary in order for human users to digest and make use of the rules. In this article, we propose two algorithms for grouping association rules. The algorithms can take as input, the rules resulting from other rule-pruning methods and further group the rules into clusters according to some criteria. The first algorithm, called the objective grouping algorithm (the *OG* algorithm), is based on the concept of *rule covers* of rules and groups the rules according to the syntactic structure of the rules. The second algorithm, referred to as the subjective grouping algorithm (the *SG* algorithm), incorporates domain knowledge and groups the rules according to the semantic information of the objects in the rules. Both algorithms group similar rules together and thus provide users with a high-level overview of the rules and with the capability of the exploratory top–down analysis of the rules, which helps the user to better understand the rules.

The article is organized as follows: in section 2, we give a literature review on the topic of grouping association rules: in section 3 and 4, we give the necessary details of the proposed algorithms, the objective grouping algorithm and the subjective grouping algorithm, respectively: section 5 presents an application in which we empirically evaluate the two proposed algorithms and we conclude the article in section 6.

2. Related work

One of the first approaches to clustering association rules was presented in Lent *et al.* (1997), where discovered rules in the 2D space are clustered using heuristic methods based on geometric properties of 2D grids. The problem of the approach is that it is limited to only the rules with two fixed attributes in their antecedents. Another approach presented in Wang *et al.* (1998) considers association rules with any number of numeric and categorical attributes in their bodies, and groups rules with similar structures by merging adjacent intervals of numeric values in a bottom-up manner. The approach lifts the 2D restriction, but grouping is only based on numeric attributes.

Toivonen *et al.* (1995) presented another approach to grouping association rules, in which rules are grouped into clusters according to a distance measure. In their approach, the distance between two association rules is defined as the number of transactions on which the two rules differ. A limitation of this approach is that rules that belong to the same cluster may have substantially different structures, and thus it is difficult to describe the rule cluster to the user. Adomavicius and Tuzhilin (2001) described another similarity-based approach to grouping association rules. In their approach, the similarity measure is based on the concept of attribute hierarchies. The attribute hierarchy is a tree structure provided by the human expert. The leaves of the tree consist of all the attributes of the data set and the non-leaf nodes in the tree are specified by the human expert and are obtained by combining several lower-level nodes into one parent node. By specifying a rule aggregation level, the rules are generalized using the non-leaf nodes at the aggregation level, and the rules with the same aggregated rule are grouped together. The benefit of this approach is that each group can be described by the aggregated rule. However, this approach requires the intensive user interaction during the grouping process. The user must specify the aggregation level. When the attribute hierarchy is huge, the user may not have a clear idea about what would be the appropriate aggregation level.

In the next two sections, we present two association rule-clustering algorithms. The two algorithms require little intervention from the user during the rule-grouping process and result in clusters that are descriptive. The algorithms can be applied to association rules that are derived from a database containing a large number of different items or attributes.

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

3. Grouping rules: an objective approach

The first algorithm that we present groups association rules according to only the structure of the rules (without domain knowledge or user interaction), and is therefore called the objective grouping (OG) algorithm. In the absence of any domain knowledge, a reasonable way to cluster rules is to group together those rules that share an item in the antecedent and an item in the consequent. We use this idea in the OG algorithm to group rules. Next, we first formally define some concepts used in the algorithm and then describe the algorithm.

3.1 *Definitions*

Let I be a set of all items in a domain and D be a set of transactions over I. A transaction is a subset of I. The association rule is an implication of the form $A \rightarrow B$, where $A \subseteq I$, $B \subseteq I$, and $A \cap B = \emptyset$. The rule holds on the transaction set D with support s and confidence c if s% of the transactions in D contains $A \cup B$, and c% of the transactions that contain A also contain B. A is called the antecedent of the rule and B is called the consequent of the rule.

Let R be a set of association rules over I. Let $a, b \in I$ and $\rho = \{a\} \rightarrow \{b\}$. Note that ρ may or may not be in R. The *cover* of ρ in R, $cover_R(\rho)$, is defined as†

$$cover_R(\rho) = \{r \in R | r = A \rightarrow B, a \in A, b \in B\}.$$

Intuitively, $cover_R(\rho)$ contains all the rules in R that have a in their antecedent and b in their consequent. Again, note that $\rho \notin cover_R(\rho)$ when $\rho \notin R$. The *seed rule* of the cover of ρ in R is defined to be ρ. The *size* of the cover of ρ in R, $size_R(\rho)$, is the number of rules in $cover_R(\rho)$, i.e., $size_R(\rho) = |cover_R(\rho)|$, where $|X|$ is the cardinality of the set X.

Using these concepts, we next present our OG algorithm.

3.2 *The objective grouping algorithm*

The basic idea of the OG algorithm is to recursively group rules with common items in their antecedents and consequents until some criteria are satisfied. The result of the algorithm is a tree-structured hierarchy of clusters, in which each leaf node is a rule and each non-leaf node is a cluster that contains all the rules in its children. In addition, each cluster has a unique label or group name, which is the ancestor rule of the cluster. For example, given a set of association rules

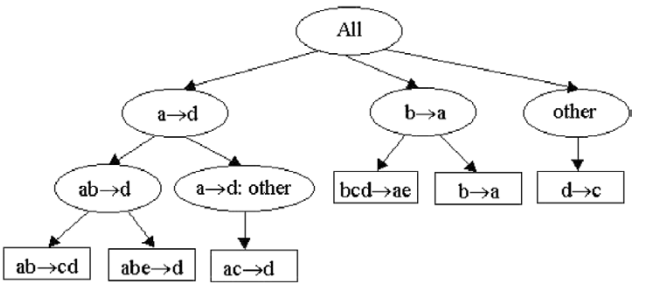


Figure 1. A sample output of the OG algorithm.

$\{ab \rightarrow cd, bcd \rightarrow ae, abe \rightarrow d, ac \rightarrow d, b \rightarrow a, d \rightarrow c\}$, the OG algorithm can generate a tree of clusters shown in figure 1, where non-leaf nodes denote clusters. For example, the node labeled $ab \rightarrow d$ denotes the cluster that contains all the rules with a and b in its antecedent and d in its consequent. The cluster that has 'other' in its label means that it is the remaining set by taking out the rules in its siblings from its parent.

The OG algorithm is a recursive algorithm. Given a set of association rules R, in each step the OG algorithm searches for a seed rule ρ whose cover has the largest size in R and then groups the rules in the cover of ρ. We allow both the cases in which ρ may or may not be in R. The algorithm is presented in figure 2.

The OG algorithm takes four inputs, which are a set of items (I), a set of association rules over I (R), an integer that denotes the maximum number of rules in a group or cluster (*threshold*) and an integer that denotes the depth of recursive call (*depth*). The *depth* parameter should be 1 when the algorithm is first called. The algorithm works as follows. As long as the number of ungrouped rules is greater than a certain predefined limit‡, it tries to group the rules in the following manner. First, it searches for a seed rule ρ (with single item antecedent and single item consequent) that has the *depth*'th largest size of cover in R by enumerating over all possible combinations. Here, rather than considering the largest cover, we search for the *depth*'th largest cover to avoid grouping items using the same seed rule over and over again, in a recursive call. The reason behind this is that we have already classified the largest cover into one group in one of the upper levels of recursion. After a seed rule ρ is selected, the cover of ρ in R is computed and all the rules in the cover are grouped into a single cluster, labeled as group ρ.

Next, if $cover_R(\rho)$ has more than *threshold* elements, we recursively group them. To reduce the complexity

†Our definitions of *over* and *seed rule* are recast from the definitions of *coverage list* and *ancestor rule* in Sahar (1999).
‡In the algorithm, we reuse the *threshold* to define this limit. Another threshold can be used for this purpose.

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

Algorithm: **Objective_Grouping**

Input: I = a set of items;

R = a set of association rules over I;

$threshold$ = an integer denoting the maximum number of rules in a group;

$depth$ = an integer that denotes the depth of recursive call.

Output: $Output$=the grouped version of R.

Begin

(1) $Output \leftarrow \emptyset$;
(2) **While** $|R| > threshold$ **Begin**
(3) **For** $i = 1$ to $|I|$
(4) **For** $j = 1$ to $|I|$
(5) $count_{i,j} = 0$;
(6) **For** $i = 1$ to $|R|$
(7) **For** each item a in the antecedent
(8) **For** each item b in the consequent
(9) $count_{a,b} \leftarrow count_{a,b} + 1$;
(10) **If** the $depth$'th largest $count_{a,b} = 0$, **Return** R;
(11) $(x, y) \leftarrow$ the index of the $depth$'th largest $count_{a,b}$
(12) $\rho \leftarrow$ rule $\{x\} \rightarrow \{y\}$;
(13) Compute $cover_R(\rho)$;
(14) Group all the rules in $cover_R(\rho)$ with label "ρ"
(15) **If** $|cover_R(\rho)| > threshold$ **Begin**
(16) $I_\rho \leftarrow$ all the items appearing in $cover_R(\rho)$;
(17) $cover_R(\rho) \leftarrow$ **Objective_Grouping**$(I_\rho, cover_\rho, threshold, depth + 1)$;
(18) **End**
(19) $Output \leftarrow Output \cup cover_R(\rho)$;
(20) $R \leftarrow R - cover_R(\rho)$;
(21) **End**
(22) Group R with label "other";
(23) $Output \leftarrow Output \cup other$;
(24) **Return** $Output$;

End

Figure 2. The objective grouping algorithm.

of the process, when we recursively call the OG algorithm, we reduce the size of the itemset I by keeping only the items that appear in $cover_R(\rho)$. For the rest of the rules in R, i.e., for $R - cover_R(\rho)$, we repeat this procedure until the number of leftover rules is less than *threshold*. Finally, we group and label these leftovers as the 'other' group and terminate the procedure.

Note that the OG algorithm is a greedy algorithm. In each level of recursion, it selects a seed with the locally largest cover, groups the rules in this largest cover together, and removes the rules in the cover from R to avoid further consideration. Therefore, the sibling clusters (which have the same parent in the final cluster tree) do not overlap. For example, if rule $\{a, b\} \rightarrow \{c, d\}$ is clustered into, say, group $\{a\} \rightarrow \{c\}$, it won't be in group $\{b\} \rightarrow \{d\}$ because the first group is larger. This design choice is for the efficiency reason. If efficiency is not an issue, we can easily modify the algorithm to allow overlapping clusters at the same level. The advantage of the OG algorithm is that it produces a hierarchy of clusters by doing recursive grouping. This provides the user the capability to perform top–down analysis of rules. In addition, it groups rules according to only the structure of the rules and thus requires little user intervention.

4. Grouping rules: a subjective approach

The OG algorithm does not incorporate domain knowledge. However, domain knowledge can often provide better insight into the discovered rules. In this section, we introduce an algorithm, called the SG algorithm, that makes use of domain knowledge. The algorithm groups the rules according to the semantic distance between the rules. It uses a taxonomy or a special type of semantic network to calculate the distance between two rules. Before we present the SG algorithm, we discuss the structure and properties of the semantic network used by this algorithm.

4.1 *Labeled semantic network*

The domain knowledge used in our SG algorithm is a tagged semantic network, which is a special type of taxonomy or *is-a*-hierarchy. The semantic network is defined or provided by the domain experts and has the following properties:

1. The taxonomy is a tree-like structure. It can contain one or more trees. Each node in a tree represents an object or entity. The upper-level nodes represent generalization of their children while the lower-level nodes are specialization of their parents.
2. Both leaf and non-leaf nodes of the taxonomy can be present in the antecedent and the consequent of a rule. This applies to the cases where some items in the rule have been generalized according to the taxonomy (in order to reduce the number of rules). It also applies to some Web mining applications

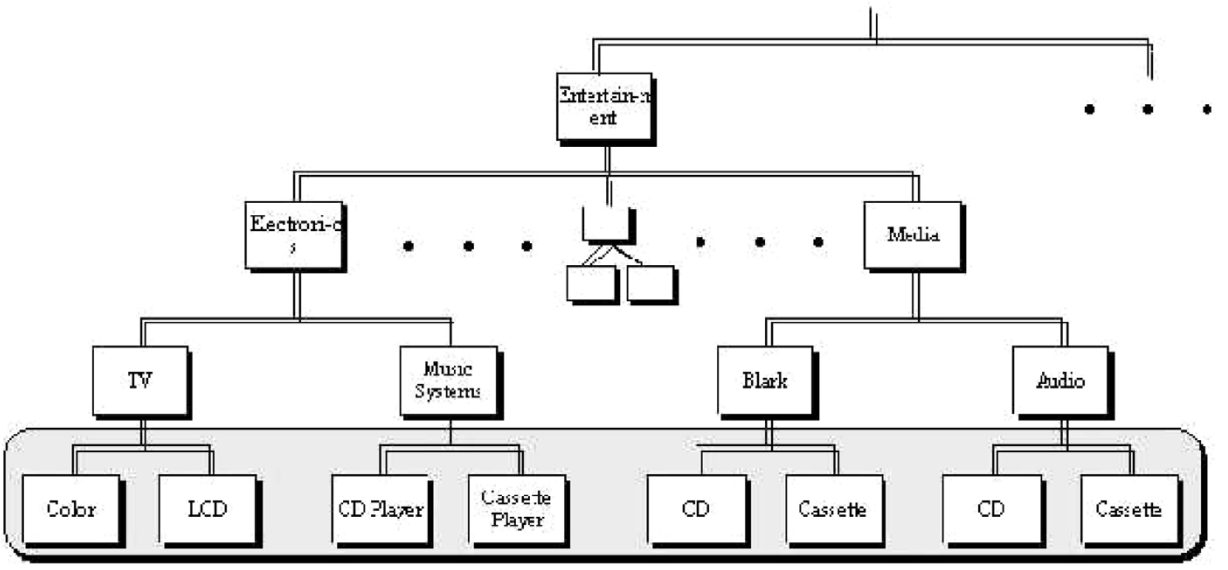


Figure 3. An example of semantic networks.

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

where rules or patterns involve 'folder' pages that contain links to its children pages.

3. Each node of the taxonomy is associated with a pair of numbers, which represents the relative position of the node in the taxonomy. We call this pair of numbers the *relative semantic position* (RSP) of the node. Generally, if two objects are closer to each other in terms of their semantic distance, the RSPs assigned to them are also closer and vice versa. RSPs can be either specified by the domain experts, or automatically generated by our algorithm.
4. A tree in the taxonomy can be an unbalanced tree.

Figure 3 illustrates an example of a taxonomy tree in an electronic store. Next, we describe a procedure to assign an RSP to each node in the taxonomy. Then we explain how the elements of this semantic network can be used to group similar rules.

4.2 *Assigning relative semantic positions*

As we have mentioned previously, a RSP represents the semantic position of the object in the network. Hence, two objects that are semantically closer to each other (e.g., tea and coffee) should be assigned two closer RSPs than two objects that represent two very different concepts (e.g., tea and baseball). A RSP can be assigned in one of the two ways. The first and probably the more accurate but costly method is to take help from a domain expert to assign an RSP to each of the nodes of the tree(s). Unfortunately, this process is highly user-intensive. Moreover, domain experts are often not readily available or are extremely costly. To come around with this problem, we propose a second way that automatically determines an RSP for each node by exploiting the structure of the semantic network. This procedure requires that, at a minimum, the user provide the semantic network as an input.

We define the RSP of a node to consist of two numbers, denoted as (*hpos*, *vpos*), where *hpos* represents the horizontal position of the node in the tree and *vpos* represents the vertical position of the node in the tree. We use the level of the node in the tree to represent the vertical position of the node. For example, the *vpos* for the root of a tree is 1, the *vpos* for each of the children of the root is 2, and so on. The *hpos* value is designed to be unique for the node. We use the position of the node in the tree's in-order traversal sequence to represent the *hpos* of the node. For example, a tree with the *hpos* value for each node is shown in figure 4. The benefit of this method for assigning RSPs is that the tree can be easily visualized using the RSP values in a 2D space. For example, the example tree in figure 4 can be visualized using the RSPs in a 2D space, which is shown figure 5. We later represent association rules in this same space.

Figure 6 describes our algorithm for assigning RSPs for a single tree. If the semantic network is a forest (i.e., it contains multiple trees), we can either make the forest a single tree by adding a root node on top of all the trees, or assign RSPs for each tree individually but with different value ranges. As we can see in figure 6, determining an RSP for each node automatically is a fairly straightforward procedure. The idea is to create a completely balanced tree by adding artificial nodes to the tree, do an in-order traversal of the

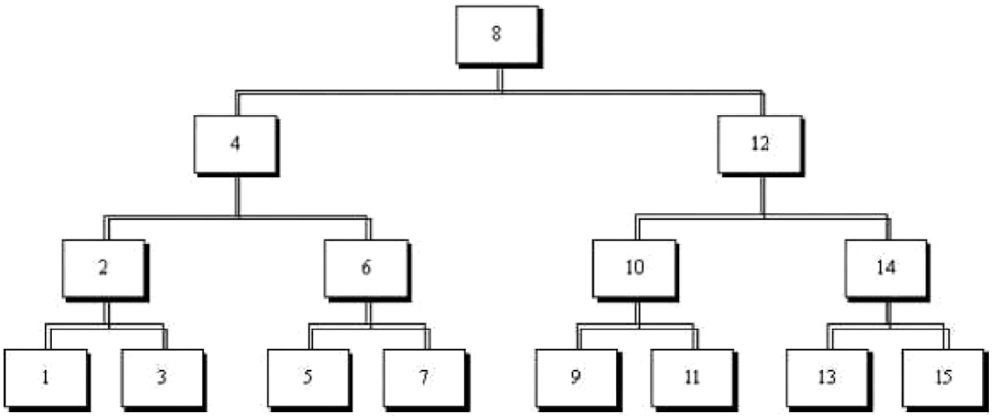


Figure 4. An example tree with *hpos* values.

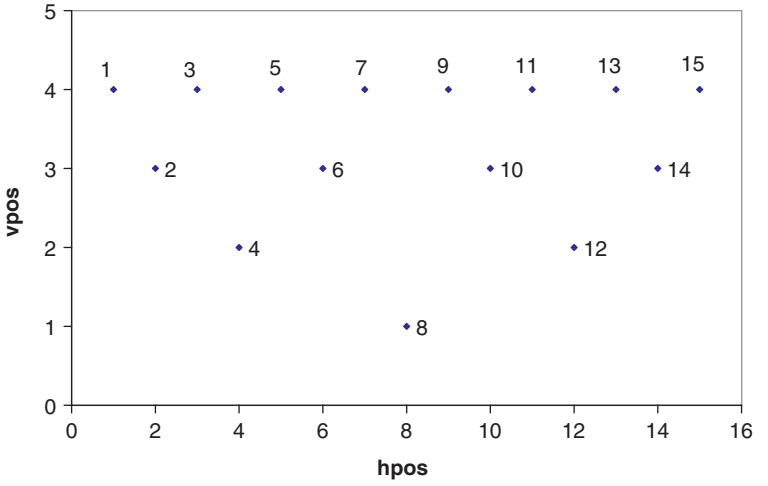


Figure 5. The example tree visualized using RSPs.

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

balanced tree, and then remove the artificial nodes. The reason for creating a balanced tree is to maximize the distance between nodes in different subtrees. While creating a balanced tree, we assign 'level' to each node as its *vpos* value. While performing in-order traversal of the tree, we assign gradually monotonically increasing integers to the nodes of the tree as their *hpos* values. Figure 7 illustrates an unbalanced tree with its RSPs assigned by our algorithm. Its balanced version of the tree is the one in figure 4. Though we only illustrate the case of binary tree here, this also works for trees where each node can have an arbitrary number of children.

4.3 *The subjective grouping algorithm*

Having discussed a method for assigning an RSP to each node of the taxonomy, we now have our ground set for the SG algorithm. In the SG algorithm, we first use RSPs to represent the objects or items in each association rule. We then calculate the average RSP of all the elements in a rule's antecedent and the average RSP of all the elements in the rule's consequent. The rule is then represented by two mean RSPs. Since each of the two mean RSPs corresponds to a point in a 2D space, the rule can be further represented by a directed line segment (pointing from the antecedent mean to the consequent mean) in the 2D space. For example, consider the following rule described by the RSPs of its objects: $\{(2,3),(4,2)\} \rightarrow \{(9,4),(10,3)\}$. We can represent the rule using the mean RSPs of its antecedent and consequent as $\{3,2.5\} \rightarrow \{(9.5,3.5)\}$ and further depict the rule using a directed line segment, as shown in figure 8.

Once the rules are represented using line segments, the problem of grouping association rules is converted

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

Algorithm: **Relative_Semantic_Position**

Input: UT = an untagged tree;

n = a reference number that will be the smallest RSP in the tree.

Output: TT = the tagged version of the same tree.

Begin

(1) $TT \leftarrow$ Introduce artificial nodes to UT to make it a completely balanced tree, and at the same time, assign the level of each node to its *vpos*;

(2) $TT \leftarrow$ **In_order_traversal_RSP**(TT, n);

(3) $TT \leftarrow$ Remove all artificial nodes from TT;

(4) **Return** TT;

End

Algorithm: **In_order_traversal_RSP**

Input: UBT = an untagged balanced tree;

n = a reference number that will be the smallest RSP in the tree.

Output: the tagged version of the same tree.

Begin

(1) **If** UBT contains a single node m **Begin**

(2) Assign n to the *hpos* of node m;

(3) $n \leftarrow n + 1$;

(4) **Return** UBT;

(5) **End**

(6) **Else Begin**

(7) $k \leftarrow$ number of subtrees of the root;

(8) **For** $i = 0$ to $k/2$

(9) $i$th_Subtree_of_$UBT \leftarrow$ **In_order_traversal_RSP**($i$th_Subtree_of_UBT, n);

(10) Assign n to the *hpos* for the root;

(11) $n \leftarrow n + 1$;

(12) **For** $i = (k/2) + 1$ to k

(13) $i$th_Subtree_of_$UBT \leftarrow$ **In_order_traversal_RSP**($i$th_Subtree_of_UBT, n);

(14) **Return** UBT;

(15) **End**

End

Figure 6. Algorithms for assigning RSPs.

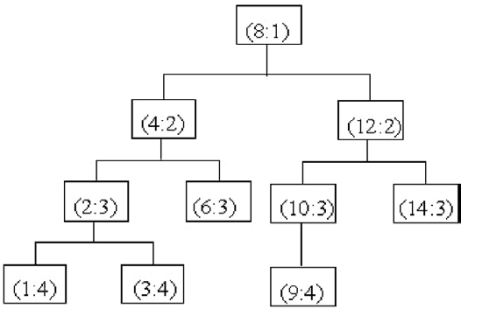


Figure 7. An unbalanced tree with RSP values.

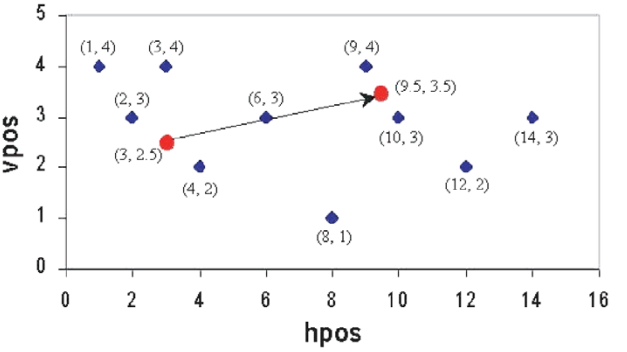


Figure 8. A rule represented by a line segment in the space showing the taxonomy of objects.

to the problem of clustering line segments. We can use a standard clustering algorithm to cluster the line segments, and modify the distance function used in the clustering algorithm to measure the distance between two line segments. The objective of clustering line segments in our case is to group segments that are close to each other and have similar length and orientation. For this purpose, we define a distance function that takes into account the angle between the two lines, the distance between the center points of two segments and the difference between the lengths of the two segments. The distance function is defined as:

$$\begin{aligned}\text{Distance}(s_1, s_2) = w_1 \times (1 - cos(s_1, s_2)) + w_2 \\ \times \text{NDist}(c_1, c_2) + w_3 \\ \times \text{NDiff}(\text{length}(s_1), \text{length}(s_2))\end{aligned}$$

where s_1 and s_2 are two line segments, $cos(s_1, s_2)$ takes the cosine of the angle between s_1 and s_2, c_1 and c_2 are the center points of s_1 and s_2, respectively, NDist(c_1, c_2) represents normalized distance between c_1 and c_2, *N*Diff(x_1, x_2) denotes the normalized difference between x_1 and x_2, *length*(x) computes the length of segment x, and w_1, w_2, and w_3 are weights representing the relative importance of the three factors.† Using this distance function, we can group together lines

†The values of weights can be adjusted according the application need. In the experiments reported in this article, we assign value 1 to all the three weights, assuming the three factors (the angle between the two lines, the distance between the center points of the two line segments and the difference between the lengths of the segments) are equally important.

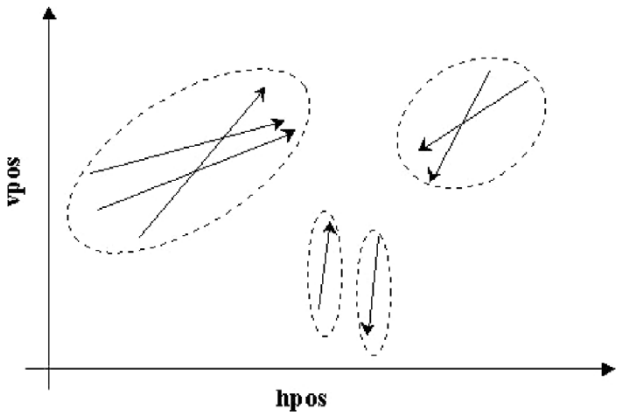


Figure 9. Clustered line segments.

Algorithm: Subjective_Grouping

Input: a tagged tree or forest;
a set of mined association rules R;

Output: a set of grouped association rules

Begin

(1) **For** each rule in R **Begin**
(2) Replace each item in the rule with its RSP;
(3) Compute the mean of RSPs of all the items in the antecedent of the rule;
(4) Compute the mean of RSPs of all the items in the consequent of the rule
(5) Add the two means and the rule id as a record in table T
(6) **End**
(7) Call a clustering algorithm to group the line segments (represented by the two means) in T, using the distance function discussed earlier;
(8) Label each group with the mean of RSPs in the antecedents and the mean of RSPs in the consequents of the rules in the group;
(9) **Return** the grouped association rules;

End

Figure 10. The subjective grouping algorithm.

with similar starting points and similar end points. Figure 9 illustrates some possible clusters of rules resulting from the use of this distance function.

The SG algorithm is presented in figure 10. It groups together rules with similar antecedents and similar consequents, and labels each group by the mean RSPs of the rules in the group, which indicate the position of the group in the semantic network. Depending on the clustering method used for grouping line segments, the algorithm can generate a hierarchy of clusters if a hierarchical clustering method is used.

5. Evaluation in a real-world application

We have applied the OG and SG algorithms to group association rules discovered in a Web mining application. The data set used in the application is the Web log data produced by a commercial product, named Livelink†. Livelink is a Web-based system that provides automatic management and retrieval of a wide variety of information objects (such as a pdf file, a power point file, a page for a project description, and so on) over an intranet or extranet. The log data set that we used in our experiments describes more than 3,000,000 requests made to a Livelink server from around 5000 users during a period of 2 months. Each request corresponds to a log entry in the log file, which records the information objects being requested, the time the request was made, where the request came from, etc. The objective of our application is to discover interesting Livelink usage patterns so that these patterns can be used to improve the organization of the information managed by Livelink and to test hypotheses about the effects of different design variables on Livelink user behavior.

To extract interesting patterns from the Livelink log data, we applied an association rule-mining algorithm that generates a complete set of association rules satisfying a minimum support and a minimum confidence. Depending on the threshold values, the number of discovered rules varies from a few to billions. If the thresholds are high, the small number of generated rules are obvious patterns which are not interesting to our domain experts. Only when the support threshold is low can interesting rules be generated. However, the total number of rules is huge. For example, for a support threshold of 0.0025 and a confidence threshold of 0.5, 74,565 rules are generated from the association rule-mining program. In order to identify interesting rules and better understand the discovered patterns, we conduct the following steps to post-process the discovered rules:

1. identify interesting rules according to an interestingness measure;
2. post-prune the rules according to the rule structure;
3. grouping the interesting rules using the OG and SG algorithms.

5.1 *Ranking and post-pruning rules*

To identify interesting rules, we use an interestingness measure to assign an interestingness value to each rule and then rank the rules according to the value. The interestingness measure that we use is called MD (Measure of Discrimination), which has been evaluated

†Developed and sold by Open Text Corporation (http://www.opentext.com).

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

to be one of the best measures for ranking association rules in our previous study Huang *et al.* (2002). Given an association rule $A \rightarrow B$, where A and B are a set of items, the MD measure is defined as

$$\mathrm{MD} = \log \frac{P(A|B)(1 - P(A|\overline{B}))}{P(A|\overline{B})(1 - P(A|B))},$$

where P denotes probability.

The interestingness measure can help identify interesting rules by putting the interesting rules at the top-ranking list. However, there are many redundant rules in the top-ranking list. By redundant rules we mean that multiple rules contain the same semantic information and hence some of them are redundant. To remove redundant rules, we adopt two *pruning rules* from Shah *et al.* (1999) and adapt the rules to use with interestingness measures. Our pruning rules are as follows:

- **Pruning Rule 1:** If there are two rules of the form $A \rightarrow C$ and $A \wedge B \rightarrow C$, and the interestingness value of rule $A \wedge B \rightarrow C$ is not significantly better than rule $A \rightarrow C$, then rule $A \wedge B \rightarrow C$ is redundant and should be pruned.
- **Pruning Rule 2:** If there are two rules of the form $A \rightarrow C_1$ and $A \rightarrow C_1 \wedge C_2$, and the interestingness value of rule $A \rightarrow C_1$ is not significantly better than rule $A \rightarrow C_1 \wedge C_2$, then rule $A \rightarrow C_1$ is redundant and should be pruned.

A rule R_1 is significantly better than rule R_2 if $(\mathrm{IV}(R_1) - \mathrm{IV}(R_2)/\mathrm{IV}(R_2)) > 5\%$, where $\mathrm{IV}(R_1)$ and $\mathrm{IV}(R_2)$ are the interestingness values for R_1 and R_2, respectively.

The use of pruning rules is effective. The number of rules was reduced significantly as the result of pruning. For example, for the rules generated with the support threshold of 0.0025 and confidence threshold of 0.5, the number of rules was reduced from 74,565 to 997. However, the number of rules is still too large for human users to analyze and understand them. To further reduce the number of rules, we only take the first 100 rules ranked high by the interestingness measure and present them to the domain experts.

5.2 *Grouping rules*

By analyzing the top 100 rules, our domain experts identify that many of these rules have similar structures or close to each other semantically in the context of the taxonomy of the information objects. They suggest that these rules should be grouped and summarized. To achieve this purpose, we applied the OG and SG algorithms to the rules. Next, we present the results from these two algorithms, and compare the results with the grouping result from our domain experts.

5.2.1 Grouping results by human expert. Our domain expert groups the 100 rules into 15 groups. The smallest group contains only one rule and the largest group has 26 rules. The distribution of the group size is shown in table 1. As we can see from the table, there are three big groups in the 100 rules and other groups are much smaller. Each of the groups is about a subject and the rules within a group often have information objects in common.

5.2.2 The results of objective grouping. Given a set of association rules, the only input to the OG algorithm that is needed from the user is the maximum number of rules that a cluster can contain. In our experiment, we set this threshold to be two different numbers. The results of these two runs (together with the SG result) are shown in table 2. Each run of the OG algorithm produces a tree of clusters. The table shows the number of levels of the cluster tree, the number of clusters at the lowest non-leaf level† (shown in the table as 'No. of clusters'), the number of lowest non-leaf level clusters that are completely the same as a cluster from the expert (No. of compl. cor. clusters)‡ and the grouping accuracy.

The grouping accuracy is calculated as follows: for each pair of rules, we know whether they should belong to the same group based on the grouping result from the domain expert. If two rules that should belong to the same group are clustered into the same group or if two rules that should not belong to the same group are clustered into different groups, we call it a 'match'; otherwise, it is a mismatch. The grouping accuracy is defined as

$$\text{accuracy} = \frac{\text{Number of matches}}{\text{Total number of rule pairs}}.$$

The number of rule pairs can be easily calculated based on the number of rules. For example, if we have 100 rules, the number of possible pairs of rules is $(100 \times 100 - 100)/2 = 4500$. The number of matches can also be calculated by a program using

†The lowest non-leaf level is the level right above leaves. The leaf level contains the rules. For example, the number of lowest non-leaf level clusters in the cluster tree in figure 1 is 4.

‡Note that a cluster is not considered to be 'completely correct' even though a single rule is misgrouped or it misses a single rule. This may be a too strict measure.

Table 1. Size distribution of the groups (from experts).

Group size	1	2	3	4	5	20	25	26
No. of groups	4	4	1	1	2	1	1	1

a triangle matrix. As an example, the number of matches for the OG-1 run is 3655, and thus its grouping accuracy is 81.2%, as shown in the table.

We observe that the two runs of the OG algorithm have different performance in terms of the measures in table 2. This is because the performance is evaluated at the lowest non-leaf level. By setting the threshold (which is the maximum of rules allowed for the lowest non-leaf level cluster) to be 26, we do not break the two largest clusters specified by the expert. However, if the threshold is 20, the two largest clusters are broken into smaller ones. This causes the grouping accuracy to decrease. If we look at the small clusters, both runs give the same results. Since the OG algorithm produces a tree of clusters, the user can always look at higher levels if they think the lower level is in too much detail due to the low setting of the threshold.

5.2.3 The results of subjective grouping. The information objects in the Livelink environment are organized into a structure that contains over 2000 trees. The leaf level objects are files or documents, and the objects at non-leaf levels are 'folders' that contain some information and links to its children objects. The association rules can contain objects at both the leaf and non-leaf levels. To use the SG algorithm, we first tag the trees with RSP values using our automatic algorithm for assigning RSPs. To distinguish the objects in different trees, we use different RSP value ranges for different trees.

After the trees are tagged, we run our SG program on the 100 rules used in our experiments. The results are shown in table 2. We use a hierarchical agglomerative clustering algorithm to cluster the line segments within the SG program. The program generates a dendrogram that shows the levels of nested merging. When we evaluate the performance, we cut the dendrogram at the level where there are 15 clusters, and based on these clusters we calculate the grouping accuracy. Due to the use of hierarchical clustering, the result of grouping is also a tree of clusters. The number of levels shown in table 2 for the SG run is the level of the tree produced on top of the 15 clusters. In general, the program can do k-way clustering. Here we set k to be 15, which is the actual number of groups produced by the expert. From table 2, we can see that the SG algorithm produces more accurate results than the OG algorithm.

Table 2. Results of the OG and SG algorithms.

Run	Threshold	No. of levels	No. of clusters	No. of compl. cor. clusters.	Accuracy
OG-1	26	3	11	6	81.2%
OG-2	20	4	13	4	71.5%
SG	15	9	15	12	93.7%

5.3 *Scaleup experiments*

To evaluate the scalability of the two algorithms, we run our programs on several sets of rules with increasing rule numbers. Figure 11 shows that the running time of the two programs increases almost linearly with the size of the rule set. The time for the SG program does not include the time for tagging the trees because this can be done off-line. Between the two programs, the OG program is faster. We notice that the reason for the SG program to be slower is mainly due to the time taken for loading the tagged trees. Since there are over 2000 trees in our application and the total number of objects is over 300,000, the total size of the trees is large. Based on the results, we can say that both algorithms are scalable.

6. Conclusions

We have presented two new algorithms for grouping a large number of (interesting) association rules. The first algorithm, the objective grouping (OG) algorithm, groups the rules by recursively selecting seed rules with largest cover size. It groups the rules according to only the structures of the rules and generates a tree of clusters as a result. The tree of clusters provides the user with the capability of conducting a top–down analysis of rules. The second algorithm, the subjective grouping (SG) algorithm, groups the rules according to the semantic distance between the rules. It is an offline interactive (or subjective) process that takes input from domain experts by asking them to specify a semantic network. Specifying a semantic network is an easy process for domain experts, since it represents the general knowledge or common sense about the domain. The novel feature of the SG algorithm is that we automatically tag the nodes of the semantic network (which is a tree or a forest) with RSPs, represent a rule with its antecedent and consequent mean RSPs, which leads to a graphical representation of rules with directed

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

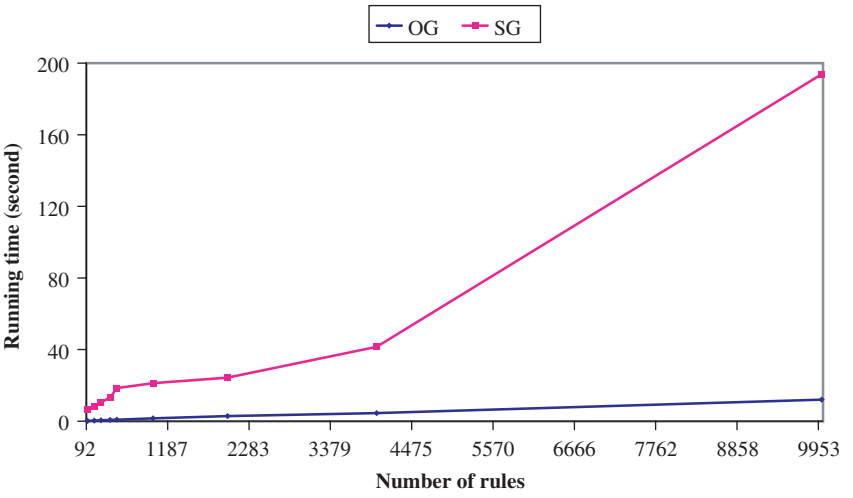


Figure 11. Results of the scaleup experiments.

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008

line segments, and finally, group the line segments with a special designed distance function. In addition, both OG and SG algorithms provide a summary description for each group. With the OG algorithm, we can use the seed rule of each group to describe the rules, while with the SG algorithm we can use the 'mean rule' which is described by two RSP means (one for the antecedent and the other for the consequent).

We also presented an application in which we evaluated the two algorithms by grouping a set of interesting association rules discovered from the Livelink log data. Our experiment shows that both methods are effective and produce good grouping result with respect to the expert grouping result. Between the OG and SG algorithms, we showed that the SG algorithm produces more accurate results than the OG algorithm. This is because it groups the rules according to the semantic 'positions' of the rules, which is closer to the way of grouping by the expert. Our experiment results also showed that both algorithms scale well with an increasing number of rules.

We also found from our experiments that some rules that were grouped together by our expert (because the expert thinks the two rules are semantically related) were clustered into different groups by the SG algorithm. This is because the objects in the two rules are far away from each other in the trees. This indicates that the semantic trees that we have for this application do not exactly reflect the semantic relationship of the objects. Some type of network structures which are not trees would be more accurate to describe the domain knowledge. For example, consider three items, 'books', 'glasses' and 'sunglasses'. Usually, 'books' is under a different category from 'glasses' and 'sunglasses'. However, semantically, 'books' and 'glasses' are close to each other, 'glasses' and 'sunglasses' are also close to each other, but 'books' and 'sunglasses' are far way. Thus, similarity is not transitive. The tree structure cannot represent this intransitivity.

In the future, we will look into solutions to this problem. We will also conduct experiments on more data sets to test the two algorithms.

Acknowledgments

The authors would like to thank Open Text Corporation and CITO (Communications and Information Technology Ontario) for supporting this research. In particular, they would like to thank Gary Promhouse of Open Text for spending time on grouping the association rules generated in the experiments. Their thanks also go to Mr Miao Wen for his help in implementing the algorithms presented in the article.

References

G. Adomavicius and A. Tuzhilin, "Expert-driven validation of rule-based user models in personalization applications", *Data Mining and Knowledge Discovery*, 5, January/April 2001.

R. Agrawal, T. Imielinski and A. Swami, "Mining associations between sets of items in massive databases", in *Proceedings of the ACM-SIGMOD 1993 International Conference on Management of Data*, Washington, DC, 1993. pp. 207–216.

R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", in *Proc. of the 20th International Conference on Very Large Databases*, Santiago, Chile, September 1994.

J. Blanchard, F. Guillet, R. Gras and H. Briand, "Using information-theoretic measures to assess association rule interestingness",

to appear in the *Proceedings of the 2005 IEEE International Conference on Data Mining (ICDM'05)*, New Orleans, Louisiana, USA, 2005.

L. Cristofor and D. Simovici, "Generating an informative cover for association rules", *Technical Report TR-02-01*, Department of Computer Science, University of Massachusetts at Boston, Boston, MA, 2002.

J. Han, J. Pei and Y. Yin, "Mining frequent patterns without candidate generation", in *Proc. of the 2000 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'00)*, Dallas, TX, 2000.

R.J. Hilderman and H.J. Hamilton, "Knowledge discovery and interestingness measures: a survey", *Tech. Report 99-4*, Department of Computer Science, University of Regina, p. 27, 1999.

X. Huang, A. An, N. Cercone and G. Promhouse, "Discovery of interesting association rules from livelink web log data", in *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, Maebashi City, Japan, 2002.

B. Lent, A.N Swami and J. Widom, "Clustering association rules", in *Proceedings of International Conference on Data Engineering*, 1997.

B. Liu, W. Hsu and Y. Ma, "Pruning and summarizing the discovered associations", in *Proceedings of the fifth ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, San Diego, CA, pp. 125–134, 1999.

J. Pei, J. Han and L.V.S Lakshmanan, "Mining frequent itemsets with convertible constraints", in *Proceedings of the IEEE Int. Conf. on Data Eng. (ICDE'01)*, Heidelberg, Germany, February 2001.

Piatetsky-Shapiro, G., "Discovery, analysis and presentation of strong rules". Knowledge Discovery in Databases, AAAI, 1991, pp. 229–248.

S. Sahar, "Interestingness via what is not interesting", in *Proceedings of KDD'99*, 1999, pp. 332–336.

A. Savasere, E. Omiecinski and S. Navathe, "An efficient algorithm for mining association rules in large databases", in *Proceedings of the 21st International Conference on Very Large Databases (VLDB'95)*, Zurich, Swizerland, pp. 407–419, 1995.

D. Shah, L.V.S Lakshmanan, K. Ramamritham and S. Sudarshan, "Interestingness and pruning of mined patterns", *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, May 1999.

R. Srikant, Q. Vu and R. Agrawal, "Mining association rules with item constraints", in *Proceedings of the Third International Conference on Knowledge Discovery in Databases*, Newport Beach, CA, pp. 67–73, 1997.

P. Tan and V. Kumar, "Interestingness measures for association patterns: a perspective", *Technical Report TR00-036*, Department of Computer Science, University of Minnestota, 2000.

H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen and H. Mannila, "Pruning and grouping discovered association rules", in *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, Montreal, Canada, AAAI Press, 1995.

K. Wang, S.H.W Tay and B. Liu, "Interestingness-based interval merger for numeric association rules", in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98)*, 1998.

Downloaded By: [Canadian Research Knowledge Network] At: 17:59 1 October 2008