

An Object-Oriented Approach to Multi-Level Association Rule Mining

Scott Fortin

University of Alberta

fortin@cs.ualberta.ca

<http://www.cs.ualberta.ca/~fortin>

Ling Liu

University of Alberta

lingliu@cs.ualberta.ca

<http://www.cs.ualberta.ca/~lingliu>

Abstract

Previous studies in data mining have yielded efficient algorithms for discovering association rules. To increase the expressiveness and relevance of the discovered rules, researchers typically augment the thresholds with domain-specific knowledge. Most existing proposals use the relational approach to organize and maintain these multi-level concept hierarchies. We argue that an object-oriented approach is better suited for focusing the search and regulating the mining of association rules both at multi-levels within one concept hierarchy, and across multiple concept hierarchies. We propose an adaptive encoding scheme for focusing the mining on semantically deeper and more informative knowledge. We demonstrate that the application of an object-oriented approach provides us with the benefits of a flexible combination of multiple multi-level concept hierarchies for focusing the mining on more informative and refined knowledge, and also allows us to enjoy seamless integration of multi-level concept hierarchies with legacy databases. In addition, by using an adaptive encoding scheme, efficient algorithms developed for discovering association rules can be integrated into the object-oriented framework with no or little extra cost.

1 Introduction

Data mining (knowledge discovery in databases) is motivated by decision support problems faced by most business organizations. It is a field of increasing interest combining databases, artificial intelligence, and machine learning. Mining for association rules between items in a large database of sales transactions has been described as an important database mining problem in decision making, risk management, and development of customized marketing programs and strategies.

Association rules, introduced by Agrawal, Imielinski, and Swami [AIS93], are a class of simple but powerful means to facilitate the understanding of large amount of data by discovering interesting regularities [Han95]. The problem can

be stated as follows: given a database with many transactions, where each transaction consists of a number of items, find rules which associate the presence of one set of items with the presence of another set. Quite often, we wish to find all rules of form $X \rightarrow Y$ satisfying some given support and confidence threshold parameters, where X and Y are disjoint sets of items, the **support** is defined as the probability that a transaction contains X , and the **confidence** is the probability that a transaction containing X also contains Y .

Several fast algorithms have been proposed for efficiently discovering association rules, satisfying given thresholds for support and confidence [AIS93, AS94, KMR⁺94, PCY95, HF95, HKMT95, SON95]. However, these algorithms often produce a large number of rules, creating an additional knowledge management problem. This arises from the fact that in large databases, the rule sets discovered by using a flat relational table and minimum values for support and confidence are often very large and contain many uninteresting rules [PSF91, KMR⁺94]. In addition, the large sets of discovered association rules are mostly managed in relational systems as internal system data, which makes it difficult to be queried and further utilize the rules to find more interesting information with respect to particular business objectives.

In this paper we propose an object-oriented framework for finding interesting association rules in the context of multi-level concept hierarchies. The discovered rules can span a single level in a single hierarchy (as was done in [HF95]), many levels within a single hierarchy, or many levels within multiple hierarchies. A typical example of a single level association rule is “80% of customers who purchase milk also purchase bread”. Not surprisingly, level and hierarchy spanning association rules, such as “50% customers buy whole-wheat bread if they buy Dairyland 1% milk”, or “45% of customers buy Dairyland Cheese if they buy Dairyland milk and a Coupon item”, are often more informative and useful. We also extend the semantics of association rules to allow *conjunctive items*; for example “12% of customers buy Lucern Cheese if they buy an item which is *both* whole-wheat bread *and* On Sale”. The concept of multi-level association rule mining was originally introduced by Han and Fu [HF95] as an extension to single concept level association rule mining, with the objective of exploring more interesting rules from

Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee.

CIKM 96, Rockville MD USA

© 1996 ACM 0-89791-873-8/96/11 ..\$3.50

large transaction databases. However, their use of flat relational data structures to construct the base relations for building multi-level concept hierarchies gives rise to a fixed encoding scheme and the difficulty of dynamically restructuring of multiple concept hierarchies to meet users' diverse customization requests. We demonstrate that the application of an object-oriented approach to multi-level association rule mining not only provides flexible utilization of existing data mining algorithms, better integration with the query language, and seamless evolution of concept hierarchies in the presence of database updates, but also allows the specification and application of user-defined domain knowledge to focus the mining on more interesting and informative rules and to actively monitor the mining of interesting association rules in the event of database updates and changes in users' customization requirements. More interestingly, with the adaptive construction of a dynamically encoded transaction table all these benefits come with little loss in performance, since the same mining algorithms used with the relational data can be seamlessly adapted to the object-oriented framework.

The rest of the paper is organized as follows. In Section 2 we analyze the limitations of using the relational approach to managing and mining association rules. In Section 3 we describe an object-oriented solution by examining a large database of sales transactions in retail organizations. The adaptive encoding scheme introduced can be seen as an extension to the fixed encoding scheme proposed in [HF95]. We discuss the benefits of using an object-oriented approach to focusing association rule mining in Section 4, compare our work with related research in Section 5, and conclude the paper in Section 6. An expanded version of this paper, which introduces a query language and sketches a preliminary implementation, can be found in [FLG96].

2 Limitations of Relational Approaches

Previous work on solving the problem of managing the potentially large rule sets discovered through known algorithms [AIS93, AS94, PCY95, SON95] includes techniques such as templates [KMR⁺94], metaqueries [SOMZ95, FH95] and mining rules at multiple concept levels [HF95]. All these techniques are confined to the relational domain, in the sense that the concept hierarchies and/or domain-dependent knowledge are captured and used in terms of their relational mappings. While they more selectively filter the rules which are presented to the user, the use of relational mapping approach has several undesired side-effects. These limitations are not a result of the previous data mining algorithms *per se*, but are due to the inability of the relational model to fully capture the semantics of multi-level concept hierarchies and domain-specific knowledge, which is required by the database mining software for focused association rule mining.

(1) Weak Item Representation

One obvious disadvantage of a relational approach is the semantic straight-jacket the database designer has to wear when

barcode	category	brand	content	size
14998	milk	Diaryland	skim	2L
12998	mechanical	MotorCraft	valve 23a	12 in
...

Figure 1: An example sales_items relation

deciding how to represent information about the database items. Using the relational approach, every item must fit into a single flat relation to help it be classified by the rule mining software. For example, Figure 1 shows a possible relation called sales_items which might be used as a base relation for association rule mining in a retail organization. By placing each item into such a relation, every item is required to have exactly the same attributes. If we wanted to add an attribute "expiry_date" for milk items, we would have to include that attribute for every item in the database — even though for certain items (such as valves) the expiry date has no real meaning.

(2) Anomalies of Relational Mappings of Concept Hierarchies

As pointed out in [HF95], relational tables such as sale_items in Figure 1 can only represent the lower levels in the concept hierarchy. The higher level concepts, such as "milk is a type of food", "apples and pears are a type of fruit", "peanut, pistachios, ..., walnuts are a type of nuts", etc., may not be explicitly or uniformly present in the form of relational tables; these often have to be manually specified by experts as domain-specific knowledge. A table or set of tables is typically created [HF95] to map high level concepts to actually tuples in a relation like sale_items. These tables are essentially internal data structures created and maintained by the the mining software.

This creates an immediate problem for user queries. Suppose that we want to capture the knowledge that peanut, pistachios, ..., walnuts \subset nuts, and there is no place in the relational tables of the database where the concept of "nuts" is explicitly mentioned. We would thus create a node for "nuts" in the mapping tables so data mining could proceed on this concept. After discovering an interesting rule about "nuts," a user may wish to query the database to get a report on the average price of nuts for the previous year. This operation would be quite difficult to perform. We would have to modify the relevant database programs to use the internal mapping tables of the data mining program. Such modification would almost certainly have to be re-written for different mining software, and maybe even different versions of the same software, due to varying choices in implementation of the internal data structures for mapping tables. The underlying problem is that the domain-specific knowledge, which has to be added for the mining software to work, is not natively supported by the relational architecture.

Update 1	Update 2
Move all items in category c_1 to category c_2	Add item i to category c_1 ...
Delete category c_1	
...	

Figure 2: An undesired concurrent update: Item i is lost

Further, the insertion of new items is problematic. When the data mining software is first installed, it is relatively easy to examine all items and make sure each one is classified somewhere in the concept hierarchy. However, at some point in the future a new item might be added by a user who has no interest in the data mining portion of the database, and thus is not responsible for including the new item in the hierarchy. For example, suppose an executive involved in a data mining task discovers that valve 23a has not been placed in the hierarchy. To examine buying patterns for mechanical items, it is desirable to include all the relevant items that are newly added into the database, such as valve 23a. Unfortunately, it could be very difficult to figure out if the item belongs to the car making or dishwasher products. The creator of the item should have been forced to classify it when it was created, but this constraint is difficult to enforce in a relational setting, where the concept hierarchy is laid on top of relational tuples.

(3) Problems with Concurrent Updates of Concept Hierarchies

In relational systems, there is no support at the system level for taking a set of relations as a unit of concurrency control, in the event of updating concept hierarchies or adding new items. It is user's responsibility to ensure that the concurrent updates to the concept hierarchies and to the database are consistent and correct. For example, when the hierarchy information is being changed, updates to the item information should not be allowed. Figure 2 shows a scenario that could occur when this rule is not followed. **Update 1** is removing a category (where a category corresponds to a high level concept which spans multiple relational tables) and **Update 2** is adding a new item into a hierarchy, time goes downward. When a concept hierarchy is multi-level and thus mapped to multiple relational tables, concurrent updates may result in the loss of a new item because a multi-level concept hierarchy can not be treated as a unit of concurrency control in relational systems. It is left to the users to enforce the complex object level locking protocol that is necessary to guarantee the correctness of concurrent updates of concept hierarchies. This is another exposition that using the relational approach to create and maintain the concept classification hierarchies is more prone to error in a multi-user environment.

3 An Object-Oriented Solution

3.1 Problem Description

Let I be a set of all literals, called items, and S be a set of transactions over I , where each transaction is a subset of I and uniquely identified by its transaction identifier. Items in I may be related differently with respect to different domain requirements and business rules. We describe the relationships between items with respect to some pre-defined domain knowledge in a hierarchical structure, called a concept hierarchy. A concept hierarchy C may have m number of layers. When $m = 1$, we call C a single-level concept hierarchy. Otherwise C is called multi-level concept hierarchy.

An association rule is an expression of form $X \rightarrow Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$. Given an itemset $X \subseteq I$, the cardinality of X is called the *size* of X . Itemsets of size k are called k -itemsets. Two parameters, namely *support* and *confidence*, have been used to quantify the mining of interesting association rules [AIS93]. We define the **support** of X in S , $\sigma(X, S)$, to be the ratio of the number of transactions in S containing X versus the total size of S . The **confidence** of a rule $X \rightarrow Y$ in S , $\varphi(X \rightarrow Y, S)$, is the ratio of $\sigma(X \cup Y, S)$ versus $\sigma(X, S)$. Intuitively, the support is a measure of the number of transactions in S where a rule possibly could have been satisfied, and the confidence is a measure of how many of the transactions in S actually satisfied the rule.

Given a database $\langle I, S \rangle$, the process for discovering association rules can be broken into two broad steps, as noted in [AS94]:

1. Find all sets of items (i.e. subsets of I) which have support greater than the minimum support. These itemsets are called large itemsets.
2. For each large itemset, find all the rules which have confidence greater than the minimum confidence.

Han and Fu [HF95] developed a fast algorithm for mining multi-level association rules that associate concepts at the same level of a given concept hierarchy. In order to facilitate the mining of multi-level association rules that associate items spanning over several levels of a concept hierarchy, we replace the fixed encoding scheme used in Han and Fu's algorithm by an adaptive encoding scheme (see Section 3.3) which dynamically massage the items being associated to "lie" on the same level. As a result, the problem of mining association rules of form $X \rightarrow Y$ where X, Y are at any level of a concept hierarchy can be reduced to the problem of discovering all rules $X \rightarrow Y$ where X, Y are at the k -th level of the concept hierarchy, and

- all items in X and Y are large at level k
- $\varphi(X \rightarrow Y, S)$ is greater than the minimum confidence for level k
- $\sigma(X \cup Y, S)$ is greater than the minimum support for level k .

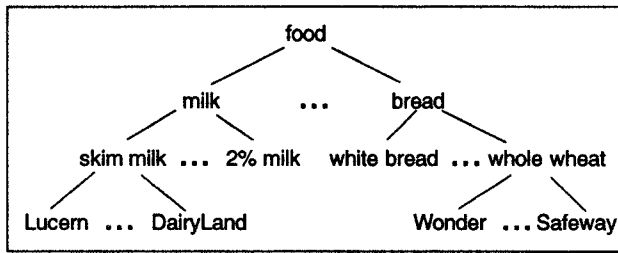


Figure 3: The example concept hierarchy

3.2 Motivating Examples

Assume we are a senior executive working at a large chain of grocery stores, and have access to the database of customer sales transactions. Figure 3 represents a portion of the concept hierarchy for the grocery items. We want to discount the price of a single milk product so that when this milk product sales increase (presumably because of the discount) the sale of bread products will also increase.

Data mining at a single concept level (i.e., the lowest level in the hierarchy only) will not be very useful in this case. This method, used for association rule mining at single concept level, would return rules of the form “Diaryland skim milk → Wonder whole wheat bread”. Obviously the discovered rules of this form would only give us the milk product which is most likely to cause an increase in *a specific bread product*. We are looking for a milk product which gives us the maximal increase over all types of bread.

Consider mining at multiple concept levels [HF95]. If each rule is restricted to associate only items at the same level of a concept hierarchy, even though the mining algorithm [HF95] would search over each level of the entire hierarchy, we would only be able to obtain rules such as “milk → bread” or “skim milk → whole-wheat bread”. However, what we want is to be able to find a specific milk product to discount, which would boost the sales of all types of bread the most.

Clearly we need to mine association rules which span multiple levels for this type of knowledge discovery. More concretely, we wish to find rules that are strong in association of items between the bottom level concepts in the subtree rooted at milk and the top level concept of the subtree rooted at bread. This would lead to find rules of the form “Diaryland skim milk → bread”, and then we could choose the product with the highest probability of causing an increase in bread sales.

Another incentive is to mine rules across many multi-level concept hierarchies. For example, we may realize that we also have access to promotional data from the previous month. That is to say, we can organize the items into *another* hierarchy, not based on the category of product contents but instead on their promotional status. Using both hierarchies we are able to discover rules like “if a customer buys a milk product that is on sale, then the customer will usually also

buy Wonder whole wheat bread”. These rules would help us more closely predict the outcome of putting a milk product on sale. The executive may even decide to increase the price of “Wonder whole wheat bread” to help recoup the losses.

3.3 Algorithms Revisited: The Use of Adaptive Encoding Scheme

As addressed in the previous section, there are cases where mining association rules which cross levels and which involve multiple hierarchies would be advantageous. In this section we discuss ways of extending previous algorithms proposed by Han and Fu [HF95] to handle these cases. These algorithms do not need to operate directly on the relational transaction table, but instead first encode each item based on its position in the concept hierarchy. The original proposal of Han and Fu uses a fixed encoding scheme to transform these items into encoded strings, however moving into the object-oriented paradigm it seems natural to instead use an adaptive encoding scheme which transforms the encoded items to allow us to extend the expressiveness of the discovered rules. This allows existing algorithms [HF95], for mining rules that associate items at the same level of a concept hierarchy, to be reused for discovering interesting rules that associate items at different levels of a concept hierarchy or across multiple concept hierarchies.

The design of our adaptive encoding scheme consists of two major steps. In the first step, we only consider how to generate the encoded item by scanning through the transaction table in the database, and replacing each item in a transaction with its encoded form. The encoded form for an item is a string which describes its location in the concept hierarchy. For example Food might have an encoding of “1”, Bread might have an encoding of “1,2” (the 1 stands for the fact that bread is a food, and the 2 represents that bread is a particular type of food) and whole wheat bread might have an encoding of “1,2,1”. The second step transforms the encoding for an item, based upon a user request. We present three different transformations, to support rules which has items from different levels in a hierarchy, from different hierarchies altogether, and from a conjunction of two or more hierarchies.

The first transformation we present support the discover of rules which span different levels in a single hierarchy.

Input: the encoded item E , the selecting item S ,
a target depth $targ$, and the maximum depth max
Output: a new encoded item, or NULL
Procedure:
if E is not a descendent of S then
return NULL;
 $E' \leftarrow$ the first $targ$ characters of E
insert $(max - targ)$ dummy characters into E'
starting at position $(targ - 1)$
return E'
end

To illustrate the use of this transformation algorithm, consider this example: Suppose we want to find a rule relating

specific milk products to bread. For each item in the hierarchy of Figure 3 we would call this transformation twice. The first call uses milk as the selecting item with the target depth of 4 and the maximum depth of 4. This would return non-NULL only for items which were specific milk products. A second call would be made with Bread as the selecting item and a target depth of 2 and a maximum depth of 4. This would be non-NULL only for items which descended from Bread, and would return the encoding “1,#,#,2” where # is the dummy character. Thus the encoded table would consist of specific milk products or an item which represents bread. Note that after this transformation, the milk products and bread are on the same level. Thus we can modify the existing algorithms [HF95] by simply using the encoded transaction table generated by the adaptive encoding scheme. Obviously this slightly modified algorithm will be able to return rules of the form “Diaryland skim milk → bread”.

Although the research reported in [HF95] suggests alternative ways of handling level crossing rules, the discussion has not led to an effective and well-defined algorithm; for example, the minimum support threshold of a rule which spans levels does not clearly follow from the single level thresholds which are supplied by the user. In contrast, by using our adaptive encoding scheme, no change to the underlying semantics or algorithms is required.

There are several interesting points to note about the adaptive transformation technique we propose. We see that bread must satisfy not just the minimum support threshold for level 2, but also the thresholds for levels 3 and 4. If the thresholds decrease as we go down levels, then if bread survives on level 2 it will always survive to level 4 as well. However, the reverse case is not necessarily true. An item which is large at a lower level may and need not be large if the thresholds at a higher level are used. Interesting to note is that the adaptive transformation has some pruning effect on the generated rules. The level 4 rules generated will involve only specific milk products and possibly bread. The discovered rules which do not include bread, i.e. are all milk products, can be filtered out before presenting the set of interesting rules discovered to the user. Furthermore, the adaptive encoding transformation scheme can be seen as a means for mining rules that involve several types of items, and be applied for many subtrees. For example a user may say that she wanted all rules involving milk at level 3, bread at level 4, cheese at level 4, and snacks at level 2.

Using this idea of transformations, it is relatively straightforward to support multiple ways of organizing the transaction data. For example we may want to organize the item sales not only by their content categories but also by, for example, their locations in the store and/or their promotion categories. These additional organization structures are shown in Figure 4. The second transformation we introduce is very simple, and allows us to integrate the hierarchies. Assume that for every item we get n encodings, e_1, \dots, e_n , corresponding to the item’s position in the n different concept hierarchies. Then the transformation to find rules with items

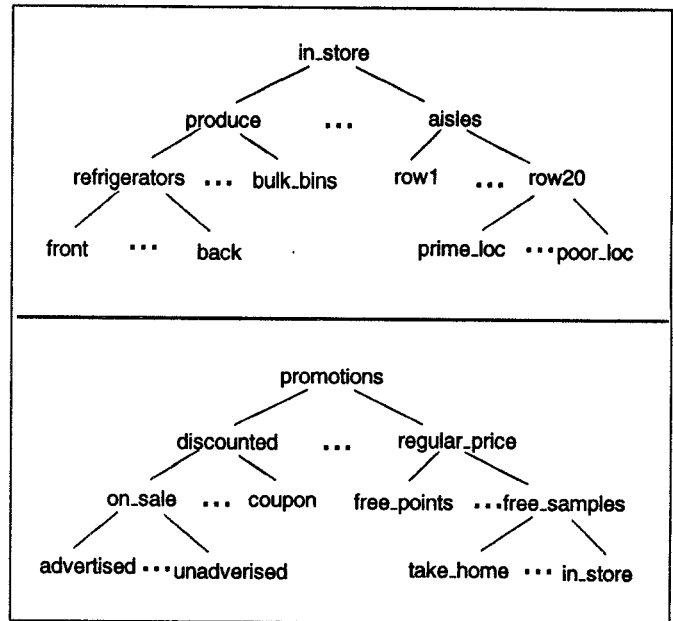


Figure 4: Hierarchies for location in store and promotional offers

from any of the hierarchies is simply stated: prepend an i to every encoding e_i . This (along with the previous transformations) would let us find rules like “customers who buy milk and some item that is on sale usually buy whole wheat bread”. These rules, while useful, would not allow us to use the relationships between the items in different hierarchies. In essence, we are finding rules like:

$$\forall x, y, z \in I \\ \text{type}(x, \text{Bread}) \wedge \text{promotion}(y, \text{OnSale}) \rightarrow \text{type}(z, \text{Milk})$$

Thus this rule would gain support if a bread item and *any* item on sale were together with a milk item. We may to constrain this further, and only increment the support if there is an item which is *both* bread and on sale. Thus we would get rules like:

$$\forall x, y \in I \\ \text{type}(x, \text{Bread}) \wedge \text{promotion}(x, \text{OnSale}) \rightarrow \text{type}(y, \text{Milk})$$

The third transformation we present allows us to discover rules of this form, by taking two different encodings of an item and returning a single *conjunctive item* if the item belongs to the two subtrees we are interested in. This transformation for the conjunction of two hierarchies is given below, it generalizes in clear way to a conjunction of n hierarchies.

Input: two different encoding as E_1 and E_2 , and selector items S_1 and S_2

Output: a new encoded item, or NULL

Procedure:

if E_1 is not a descendent of S_1 then
return NULL;

if E_2 is not a descendent of S_2 then

```

    return NULL;
  for  $i \leftarrow 1 \dots \text{size}(E_1)$ 
     $E'[i] \leftarrow f(E_1[i], E_2[i])$ 
    (where  $f(x, y)$  is some function like:  $x + 10y$ )
  return  $E'$ 
end

```

For example, suppose we have an item encoded as “1,2,1,3” and “1,4,5,1”, milk is encoded as “1,2” and on sale was encoded as “1,4”. Then the transformation would return the conjunctive item “11,24,15,31”, where 11 represents food \wedge promotion, 24 represents milk \wedge on sale, and so forth. Users could use this to specify another type of item to be included in their output, which is a conjunction between two or more hierarchies. Thus a user query might look like “find me all rules involving milk items at level 4, bread items and level 2, and items which are both cheese and on sale”.

The technique of transformations does not depend directly on any underlying model, it could be implemented on either a relational or object-oriented system. However, the object-oriented model allows the encoding and decoding of items to be encapsulated within the class that contains the item data, making the system easier to implement and maintain.

4 Benefits of the Object-oriented Approach

4.1 Eliminating Anomalies of Using Relational Mappings

Placing the taxonomy in an object-oriented class hierarchy has many advantages over the *ad-hoc* relational approaches. Probably most significant is that it encourages the uniform treatment of class hierarchies and relational data through an object-oriented front-end user-friendly query language. Any item in an association rule discovered by the mining software system will either be a class, or an object of a certain class. For example, in the rule $\text{nuts} \rightarrow \text{PartyTime unsalted pretzels}$, “nuts” would correspond to a class in the class hierarchy of the database, while “PartyTime unsalted pretzels” would correspond to all the Pretzel objects where $\text{salt_content} = \text{'none'}$ and $\text{manufacturer} = \text{'PartyTime'}$. In an object-oriented framework, classes and instances of a class are all treated as first-class objects. Through an object-oriented query interface, such as those provided in OODBMS systems, users may query over classes and instance objects of the classes in the same manner [ea92, MS90, CGJD91]. Any clause included in a rule is also directly queryable.

Another advantage of implementing the taxonomy with the object-oriented class system is the potential of using object-based concurrency control mechanisms to support concurrent updates of concept hierarchies. For example the situation in Figure 2 could never occur, when the concept hierarchies are created and maintained through an OODBMS such as ObjectStore.

Furthermore, in the object-oriented framework, items can be viewed by multiple representations through the class hierarchy. Put differently, instead of collecting items of different representations into a single relational table, items can

be organized and represented in terms of classes of different number of properties. For example, Milk might have a property named “butterfat_content”, but Bread would probably not. Instead of forcing Milk items and Bread into the same `sale_items` relation, we may simply organize Milk and Bread as two different subclasses in the Food class hierarchy. In the object-oriented paradigm, the only way to create objects is through their classes. Thus every item would have to be assigned to a node in the classification hierarchy when it was created. Moreover, the class hierarchy provides a clear concept taxonomy that helps the creator of new objects to decide to which classification the new items belong.

4.2 Dynamic Adjustment of Concept Hierarchies

Most of mining algorithms represent predefined concept hierarchies in the form of flat relational tables (e.g., `sale_items` in Figure 1). Unfortunately some portions of the user-defined and domain-specific concept hierarchies, such as $\text{milk}, \dots, \text{bread} \subset \text{food}$, or $\text{peanut, pistachios}, \dots, \text{walnuts} \subset \text{nuts}$, etc., may not be explicitly and uniformly presented in the form of relational tables.

Using the object-oriented framework, the high levels of concept hierarchies are captured explicitly and uniformly in the form of classes or objects in classes, and the lower levels of concept hierarchies can be adjusted dynamically upon request. This is because, quite often, the upper levels of the concept hierarchy, such as `food`, are fixed in their order since other orderings like $\text{beverages, fruits, grains} \subset \text{milk} \subset \text{food}$ make little sense. However lower levels of the hierarchy may be re-ordered. For example both $\text{Diaryland skim milk} \subset \text{Diaryland milk} \subset \text{skim milk} \subset \text{milk}$ and $\text{Diaryland skim milk} \subset \text{skim milk} \subset \text{Diaryland milk} \subset \text{milk}$ are valid orderings. By having the encoding of the lower levels of the hierarchy handled by a method, it is easy to make the mining algorithm dynamic in nature. The translation to encoded form is determined at run-time, so it is not too difficult to imagine a scheme where the user specifies which properties he or she is interested in, in which preferred order these interested attributes may rank, and the encoding methods then use those information to determine the relevant encoding.

4.3 Supporting the Adaptive Encoding Scheme

There are many advantages with an adaptive object-oriented approach to encoding items. One key benefit is that it enables the encoding algorithm to be closely associated with the item being encoded, in contrast to the relational case where the encoding algorithm is one of the decisions that the mining software makes. As a consequence, it allows other programs to make use of the encoding information being available. Further the kind of dependencies that the encoding scheme follows can be handled in a structured way — if the mining application wishes to change the encoding format, then that change could be handled as a schema update in object-oriented systems [Liu95].

More importantly, our adaptive encoding scheme accepts objects that may have different numbers of properties, and

rules that associate attributes at different levels of a concept hierarchy. For example, Milk objects might have 5 properties and Bread objects might only have three. This can be handled in the object-oriented solution by simply inserting dummy values into the bread objects' encoded representation so that both objects have the same number of properties in the encoded table.

Finally, the adaptive encoding mechanism provides adequate support for mining association rules across multiple concept hierarchies. As mentioned previously, to support a different hierarchy all we need to do is provide routines to encode and decode the items based on that concept organization, following the transformation outlined in Section 3.3. Items in each transaction would then be encoded k times, where k is the number of hierarchies desired in the final rule, and the union of these encodings would become the transaction in the encoded table. Searching for association rules would then proceed as normal; however since more than one hierarchy is being used, the lowest level (i.e. the actual items) could give rise to trivial rules where a single item is associated with itself. These kinds of rules would only arise at the lowest levels and could be simply pruned there. This technique could also be used in relational systems, but is more naturally supported in an object-oriented framework.

5 Related Work

Mining association rules in a transaction database was a problem introduced by Agrawal, Imielinski, and Swami [AIS93]. Subsequent research has resulted in fast algorithms for the generation of these types of rules [AS94, PCY95, SON95]. This research confines itself to finding association rules in a relational database system, at what is known as a single concept level. Mining at a single concept level means that the items in a rule are always concrete; thus we can find rules like *85% of people who buy Dairyland skim milk may also buy Wonder Whole Wheat Bread*, but not conceptually higher rules like *90% of people who buy milk may also buy bread*. One primary concern with mining at single concept level is that the number of association rules generated are either so large as to overwhelm the user, or is too small to contain enough "interesting" rules [KMR⁺94] if the thresholds of support and confidence are tightened. It seems that using the two thresholds of support and confidence is not sufficient in many cases to filter out the "weak" or "uninteresting" rules.

Several researchers proposed using domain-specific classification taxonomies to focus the search of interesting association rules from the large set of discovered ones. Klemettien et al [KMR⁺94] propose to reduce the number of generated rules by having the user specify a template, which details the form of the rule (i.e., the number of items in the antecedent and the consequent) and restricts the items that can be used in the rule. This is accomplished by organizing the items into a classification hierarchy (within a relational database), and using this information to guide the rule selection process. However, this technique only deals with mining

rules at one concept level — templates are merely used to select a subset of the rules from those that would have been returned by the previous algorithms.

Another approach, proposed by Han and Fu [HF95], deals with the problem of focusing the search for interesting association rules by mining multi-level association rules. They also use a concept classification hierarchy to capture taxonomy knowledge of the relational data. The novelty of their approach is that they do not use the classification hierarchy simply for filtering of low-level rules, but instead are able to find associations among higher level concepts. Thus their approach allows not only to find rules generated using the basic mining algorithms, but also to discover rules which associate more generalized items. However, their approach suffers from the limitation of presenting concept hierarchies in the form of relational tables and of the fixed encoding scheme.

Holsheimer et al. [HKMT95] report on integrating the data mining process more closely into the database paradigm. They present an algorithm which finds association rules at a single concept level using only union and intersection operations, and show how it can be extended to mine at multiple concept levels. They succeed in incorporating the data structures and operations used in database mining into the relational paradigm, at the loss of some efficiency. To mine at multiple concept levels, the classification hierarchy is utilized although this type of information is not fully integrated into the relational database system.

Shen et. al. [SOMZ95] introduce the notion of metaqueries for guiding the data mining process. Similar to templates, metaqueries work by specifying an abstract form that the rule must satisfy. However, metaqueries in their most general form are most useful for mining data from different relations in a database, which is not the case for transaction data. Fu and Han [FH95] show how metaqueries can be used in mining association rules at multiple concept levels, however the type of data they examine is quite different from those used in [SOMZ95]. Moreover, both papers consider metaqueries solely, and do not specify a complete architecture for data mining.

Knowledge discovery in object oriented databases (OODBs) is examined by Han, Nisho, and Kwanono in [HMK93]. Their focus is mainly on the ways of discovering knowledge *contained* in an arbitrary OODB, rather than using an object-oriented approach to organize the concept hierarchies for focusing the mining of interesting and strong association rules from large relational databases.

6 Conclusion and Future Work

We have presented an object-oriented approach to mining multi-level association rules. The main contributions of this paper are the following. First, we identify a number of limitations that the relational approach presents and demonstrate the benefits of using an object-oriented approach for organizing the concept hierarchies for focusing the mining on strong and interesting multi-level association rules. Second, we pro-

pose an adaptive encoding scheme to support dynamic adjustment of concept hierarchies in order to find more informative and semantically deeper association rules. Our encoding scheme can be seen as an adaptive extension to the fixed encoding scheme used in [HF95]. Thirdly, we demonstrate that the application of an object-oriented approach suits better for focusing the search and regulating the mining of multi-level association rules that span over many levels of a concept hierarchy or multiple concept hierarchies.

By using the adaptive encoding scheme, we have shown that the problem of mining association rules of form $X \rightarrow Y$ where X, Y are at any (possibly different) levels of a concept hierarchy can be reduced to the problem of discovering all rules $X \rightarrow Y$ where X, Y are at the same level of the concept hierarchy. Note that we have advanced the notion of directly reusing existing algorithms for expositional clarity. In an actual implementation, optimizations could be made to the base algorithms — for example if we “push” an item down from level 2 to 4, there is no need to calculate its support at level 3 or 4 because it will not have changed from level 2.

Our work on using an object-oriented approach to multi-level association rule mining continues. We are interested in building a prototype for experimenting with an object-oriented approach to association rule mining, and studying the performance of combining our adaptive encoding scheme with existing algorithms for multi-level association rule mining. We are also interested in applying data mining techniques to resource discovery in global and interoperable information systems [LP95, LPL96].

References

- [AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM-SIGMOD International Conference on the Management of Data*, pages 207–216, Washington, D.C., May 1993.
- [AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.
- [CGJD91] C.Lamb, G.Landis, J.Orenstein, and D.Weinreb. The object-store database system. *Communications of the ACM*, 34(10):51–63, October 1991.
- [ea92] E. Bertino et. al. Object-oriented query languages: the notion and the issues. *IEEE Transactions on Knowledge and Data Engineering*, 4(3):223–237, 1992.
- [FH95] Y. Fu and J. Han. Meta-rule-guided mining of association rules in relational databases. In *Proceedings of the 1995 International Workshop on Knowledge Discovery and Deductive and Object-Oriented Databases*, Singapore, December 1995.
- [FLG96] S. Fortin, L. Liu, and R. Goebel. Enhancing multi-level association rule mining with dynamic hierarchies in an object-oriented framework. Technical Report TR 96–15, University of Alberta, 1996.
- [Han95] Jiawei Han. Mining knowledge at multiple concept levels. In *ACM International Conference on Information and Knowledge Management (CIKM'95)*, Baltimore, Maryland, USA, November 1995.
- [HF95] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of the 21st VLDB Conference*, Zurich, Switzerland, 1995.
- [HKMT95] M. Holsheimer, M. Kersten, H. Mannila, and H. Toivonen. A perspective on databases and data mining. Technical Report CS-R9531, University of Helsinki, 1995.
- [HNK93] J. Han, S. Nishio, and H. Kawan. Knowledge discovery in object-oriented and active databases. In *Proceedings of the 1993 International Conference on Building and Sharing of Very Large Scale Knowledge Bases*, pages 205–214, Tokyo, Japan, December 1993.
- [KMR⁺94] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding interesting rules from large sets of discovered rules. In *Proceedings of the 3rd International Conference on Information and Knowledge Management*, pages 401–408, Gaithersburg, Maryland, Nov 1994.
- [Liu95] L. Liu. Database consistency in the presence of schema evolution. In *Proceedings of IFIP Data Semantics conference*, Atlanta, 1995.
- [LP95] Ling Liu and Calton Pu. The distributed interoperable object model and its application to large-scale interoperable database systems. In *ACM International Conference on Information and Knowledge Management (CIKM'95)*, Baltimore, Maryland, USA, November 1995.
- [LPL96] L. Liu, C. Pu, and Y. Lee. An adaptive approach to query mediation across heterogeneous databases. In *Proceedings of the International Conference on Cooperative Information Systems*, Brussels, June 19–21 1996.
- [MS90] D. Maier and J. Stein. Development and implementation of an object-oriented DBMS. In S.Zdonik and D.Maier, editors, *Readings in OODB Systems*. 1990.
- [PCY95] J.S. Park, M.S. Chen, and P.S. Yu. An effective hash based algorithm for mining association rules. In *Proceedings of the 1995 ACM-SIGMOD International Conference on the Management of Data*, San Jose, CA, May 1995.
- [PSF91] G. Piatetsky-Shapiro and W. J. Frawley. *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.
- [SOMZ95] W. Shen, K. Ong, B. Mitbender, and C. Zaniolo. Metaqueries for data mining. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1995.
- [SON95] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *the 22nd International Conference on Very Large Databases*, Zurich, Switzerland, November 1995.