Chapter IX
Variations on Associative Classifiers and Classification Results Analyses

Maria-Luiza Antonie
University of Alberta, Canada

David Chodos
University of Alberta, Canada

Osmar Zaïane
University of Alberta, Canada

ABSTRACT

The chapter introduces the ***associative classifier****, a classification model based on* ***association rules****, and describes the three phases of the model building process:* ***rule generation****, pruning, and selection. In the first part of the chapter, these phases are described in detail, and several variations on the* ***associative classifier*** *model are presented within the context of the relevant phase. These variations are: mining data sets with* ***re-occurring items****, using* ***negative association rules****, and pruning rules using* ***graph-based techniques****. Each of these departs from the standard model in a crucial way, and thus expands the classification potential. The second part of the chapter describes a system, ARC-UI that allows a user to analyze the results of classifying an item using an* ***associative classifier****. This system uses an intuitive, Web-based interface and, with this system, the user is able to see the rules that were used to classify an item, modify either the item being classified or the rule set that was used, view the relationship between attributes, rules and classes in the rule set, and analyze the training data set with respect to the item being classified.*

INTRODUCTION

The process of creating an **associative classifier** from a training data set has three main phases: (1) mining the training data for **association rules** and keeping only those that can classify instances, (2) pruning the mined rules to weed out irrelevant or noisy rules, and (3) selecting and combining the rules to classify unknown items. Within each of these steps, there is a great deal of potential for variation and improvement. The first three sections describe each of the three phases of the associative classification process in detail. In addition, three variations on this process are described, each within the context of the relevant classification phase. Each of these variations are outlined briefly in the following paragraphs and described using a running example of a department store sales dataset. To put the preceding paragraph into this context, we can imagine a store with data from previous months on the sales of various items in the store, and an assessment from a manager of whether the items were worth stocking. An **associative classifier** for this context would create a set of rules relating items' sales figures to their overall profitability, and allow the manager to assess the current month's stock based on data accumulated from previous months.

The first variation considers data sets with **re-occurring items**. **Associative classifiers** are typically concerned only with the presence of an attribute, which ignores potentially valuable information about the number of occurrences of that attribute. For example, in a text classification context, the number of occurrences of a word in a document and in a collection are crucial indicators of its importance. Or, to use the department store example, knowing *how many* shirts were sold might be more important than knowing whether or not any shirts were sold. A classification model by Rak et al (Rak, 2005) considers the number of occurrences of an attribute both in generating rules and in classifying items according to those rules. In the **rule generation** phase, the model increments a rule's support by an amount proportional to the number of attribute occurrences. In the item classification phase, the model uses the Cosine Measure to measure the similarity between an item and rules which have re-occurring attributes.

The second variation presents a classifier which works with both positive and negative rules. Negative rules either use attribute/negated value pairs, or imply a negative classification, and can capture patterns and relationships that would be missed by positive only rule-based **associative classifiers**. In the department store context, knowing that a store did not sell any shirts of a certain brand could help a manager decide not to stock more shirts of that brand. Generating a complete set of **negative association rules** from a set of positive rules is a very difficult task, and can result in an exponential growth in the number of **association rules**. A method developed by Antonie and Zaïane (Antonie, 2004c) deals with this issue in two ways. First, the negative rules generated are restricted to those where either the entire antecedent or consequent is negated. Thus, a rule that identifies one brand of shirt that *did not* sell and another that did would not be generated using this method. Second, the correlation coefficient between a pattern and a frequent itemset is used to guide the generation of **negative association rules**. This method also incorporates both negative and positive rules into the item classification process.

The third variation deals with the issue of pruning rules generated through frequent itemset mining. Since frequent itemset mining can generate hundreds or even thousands of rules, pruning this initial rule set is crucial for maintaining classification accuracy and comprehension. However, it is important not just to reduce the number of rules, but to do so in such a way that the accuracy of the classifier does not suffer. That is, one must reduce the number of rules while preserving the rules which do a good job of classification. To this end, a technique was developed that evaluates each rule by using the rule set to re-classify the training data set and measuring each rule's number of correct and incorrect classifications (Zaïane, 2005). These measurements are then plotted on a graph, and the rules are then categorized (e.g., frequently used and often inaccurate) and prioritized for pruning.

An important part of associative classification process that is often overlooked is the analysis of classification results by the user. Often, a user will be interested not only in the classification of an item, but in the reasons behind that classification, and how the classification might change with slightly different input. Returning to the department store example, a manager might want to know *why* a brand of shirt was classified as a "very profitable item"; is it because of customer demand, a supplier discount, or a beneficial profit margin? The second section of the chapter presents an analysis system for **association rule** classifiers, ARC-UI, which offers the user a variety of classification analysis and speculation tools.

The analysis tools offered by ARC-UI are based on research on analyzing the results of linear classifiers by Poulin *et al* (Poulin, 2006). However, these tools have been modified for use with **associative classifiers**, and deal with rule sets as opposed to weighted attributes. The decision speculation component offers robust speculation capabilities which allow the user to modify both the item being classified and the rules used to perform the classification. The decision evidence component shows the impact of all relevant rules on an item's classification, while the ranks of evidence component offers a concise graphical representation of the relationship between attributes, rules, and classes. Finally, the source of evidence component allows the user to analyze the relationship between the item being classified and the training data set used to generate the **associative classifier**.

ASSOCIATIVE CLASSIFIERS

The first reference to using **association rules** as classification rules is credited to Bayardo (Bayardo, 1997), while the first classifier using these **association rules** was CBA, introduced by Liu (Liu, 1998) and later improved in CMAR (Li, 2001), and ARC-AC and ARC-BC (Antonie, 2002a). Other **associative classifiers** that have been presented in the literature include CPAR (Yin, 2003), Harmony (J. Wang, 2005), and 2SARC (Antonie, 2006).

The idea behind these classifiers is relatively simple. Given a training set modeled with transactions, where each transaction contains all features of an object in addition to the class label of the object, we can constrain the mining process to generate **association rules** that always have a class label as their consequent. In other words, the problem consists of finding the subset of strong **association rules** of the form $X \rightarrow C$ where C is a class label and X is a conjunction of features.

The main steps in building an **associative classifier** when a training set is given are the following:

1. *Generating the set of **association rules** from the training set:* In this phase, **association rules** of the form *set_of_features* $\rightarrow$ *class_label* are discovered using a mining algorithm. This phase can be completed in two ways:
   - Using an **association rule** mining algorithm, generate all the strong **association rules**. Once these are generated, filter them so that only the rules of interest are kept (those that have a class label as the consequent and the antecedent composed of features other than class labels).
   - Modifying an **association rule** mining algorithm by imposing a constraint. The constraint is that the **association rules** must have a class label as the con-

sequent. This improves the algorithm's efficiency since less candidate items are generated. All of the candidate itemsets that are generated contain a class label on the right-hand side.

2. *Pruning the set of discovered rules:* The previous phase may generate a large set of **association rules**, especially when a low support is given. Thus, pruning techniques are used in order to discover the best set of rules that can cover the training set. In this phase, rules that may introduce errors or cause overfitting in the classification stage are weeded out.
3. *Classification phase:* At this level a system that can classify a new item is built. The challenge here is using the set of rules from the previous phase to classify new items effectively. In order to classify a new item effectively using these rules, we need a good way of selecting one or more rules to participate in the classification process. In addition, another challenge is dealing with conflicting rules – when multiple rules with the same antecedent point to different classes.

RULE GENERATION: FROM ASSOCIATION RULES TO CLASSIFICATION RULES

The first step in creating an **associative classifier** is to generate a complete set of **association rules** for the training data set. The key difference between **associative classifiers** and other rule-based classifiers is that **associative classifiers** seek to use a *complete* set of rules, rather than using heuristics to identify a small set of hopefully relevant rules. To start this process, the frequent itemsets are found using an established technique such as Apriori (Agrawal, 1993) or FP-Growth (Han, 2000). This set of frequent itemsets is converted into **association rules**, and these are pruned so that only those rules with the class label in the consequent are kept. Finally, from this list, all of the **association rules** which do not meet a certain minimum confidence level are discarded. Thus, the resulting rule set contains *all* the rules which a) have a sufficient level of support in the training data set and b) may be used to classify an unknown item.

While this approach works in many situations, it makes some assumptions which may not always work best. For one, the rules that are generated are assumed to be positive rules – that is, they associate the presence of an attribute with a particular class label. They do not, however, explicitly consider the *absence* of an attribute, nor do they consider rules where a class label is ruled out, rather than implied. Second, the rules are binary, in the sense that they are concerned only with the presence of an attribute. However, there are situations where the cardinality (number of occurrences) of an attribute is as important as its presence (e.g. image classification, text classification). Two variations on this approach are described in the sections that follow which address each of these concerns.

Data Sets with Recurrent Items

Associative rule classifiers typically use rules which are based on the presence of an attribute. For example, market-basket analysis focuses on questions such as "has the customer purchased both cereal and milk". However, this ignores potentially important information about the number of times an attribute occurs in a transaction. For instance, returning to market-basket analysis, one might want to know how many cans of pop a customer has purchased. Or, in a text classification context, the number of occurrences of a word in a document and in a collection are crucial indicators of its importance. This section presents a **rule generation** algorithm and classification scheme by Rak *et al* which makes use of attribute frequency (Rak, 2005).

The task in this case is to combine associative classification with the problem of recurrent items. Stated more formally, the original approach is modified such that transactions of the form $<\{o_1 i_1, o_2 i_2, \ldots o_n i_n\}, c>$ are used, where o_k is the number of the occurrences of the item i_k in the transaction and c is a class label.

Association rules have been recognized as a useful tool for finding interesting hidden patterns in transactional databases. However, less research has been done considering transactions with recurrent items. In (W. Wang, 2004), the authors assign weights to items in transactions and introduce the WAR algorithm to mine the rules. This method has two phases: first, frequent itemsets are generated without considering weights, and then weighted association rules (WARs) are derived from each of these itemsets. The MaxOccur algorithm (Zaïane, 2000) is an efficient Apriori-based method for discovering **association rules** with recurrent items. It reduces the search space by making effective use of joining and pruning techniques. The FP'-tree approach presented in (Lim, 2001) extends the FP-tree design (Han, 2000) by combining it with concepts from the MaxOccur algorithm. For every distinct number of occurrences of a given item, a separate node is created. In the case where a new transaction is inserted into the tree, it might increase support count for the different path(s) of the tree as well. This is based on the intersection between these two itemsets. Given the complete tree, the enumeration process to find frequent patterns is similar to that of the FP-tree approach.

Description of Rule Generation Process

The rule generator differs from traditional, Apriori-based algorithms in two important respects. First, the **rule generation** process is designed for finding all frequent rules in the form of $<\{o_1 i_1, o_2 i_2, \ldots o_n i_n\}, c>$ from a given set of transactions and is based on ARC-BC (Antonie, 2002a): transactions are divided by class, rules are generated from each subset and then the rule sets for each class are merged to create a rule set for the entire data set. Second, attribute frequency is taken into account when calculating the support for a particular pattern. A transaction may only add support to a pattern if its attributes occur at least as many times as in the pattern. Similarly, the amount of support a transaction adds to a pattern is proportional to the cardinality of its attributes, as compared to that of the pattern.

The rule generator for each class C_x takes into account recurrent items in a single transaction à la MaxOccur (Zaïane, 2000). To accomplish this, the support count is redefined. Typically, a support count is the number of transactions that contain an item. In our approach, the main difference is that a single transaction may increase the support of a given itemset by more than one. The formal definition of this approach is as follows: a transaction $T=<\{o_1 i_1, o_2 i_2, \ldots o_n i_n\}, c>$ supports itemset $I=\{l_1 i_1, l_2 i_2, \ldots l_n i_n\}$ if and only if $\forall\ i=1..n;\ l_1 \leq o_1 \wedge l_2 \leq o_2 \wedge \ldots \wedge l_n \leq o_n$. The number t by which T supports I is calculated according to the formula: $t=\min[o_i / l_i]\ \forall\ i=1..n,\ l_i \neq 0 \wedge o_i \neq 0$.

Description of Classifier

Because of the added complexity of attribute recurrence, the probability of obtaining an exact match between an item to be classified and an **association rule** in the rule set is quite low. Thus, the classifier must use some notion of similarity between an item and a non-matching rule. Several definitions were considered by Rak *et al* (Rak, 2005) and tested extensively. The measure which proved most effective was the Cosine Measure (CM), which measures the angle between the vector representations of an item and an **association rule**. A small angle between the vector representations indicates that the item and the rule are similar.

Negative Association Rules

Most **associative classifiers** are based on rules which are made up of attribute/value pairs and an implied classification. However, one may also consider negative rules–that is, rules which either use attribute/negated value pairs, or which imply a negative consequent. This section presents a classifier developed by Antonie and Zaïane (Antonie, 2004c) that is based on this type of rule.

Utility of Negative Association Rules

Negative association rules, in general, rely on or imply the absence of a particular attribute value or class, respectively. In other words, negated predicates can exist in the antecedent or consequent of the rule. Negative attribute value rules identify situations where the absence of an attribute value is an important indicator of an item's class.

Example 1. Let us consider an example from the context of market basket analysis. In this example we want to study the purchase of organic versus non-organic vegetables in a grocery store. Table 1 gives us the data collected from 100 baskets in the store. In Table 1 "organic" means the basket contains organic vegetables and "¬organic" means the basket does not contain organic vegetables. The same applies for the term "non-organic".

Using this data, let us find the positive **association rules** in the "support-confidence" framework. The **association rule** "non-organic → organic" has 20% support and 25% confidence (supp(non-organic ∧ organic)/supp(non-organic)). The **association rule** "organic → non-organic" has 20% support and 50% confidence (supp(non-organic ∧ organic)/supp(organic)). The support is considered fairly high for both rules. Although we may reject the first rule on the basis of confidence, the second rule appears to be valid and may be analyzed more closely. Now, let us compute the statistical correlation between the *non-organic* and *organic* items. For more details on the correlation measure, see (Antonie, 2004b). The correlation coefficient between these two items is -0.61. This means that the two items are negatively correlated. This measure sheds new light on the data analysis for these items. The rule "organic → non-organic" is misleading. The correlation, therefore, provides new information that can help in devising better marketing strategies.

The example above illustrates some weaknesses in the "support-confidence" framework and shows the need for the discovery of more interesting rules. The "interestingness" of an **association rule** can be defined in terms of the measure associated with it and in terms of the form of the association.

Brin *et al* (Brin, 1997) were the first in the literature to mention the idea of negative relationships. Their model is chi-square based. Specifically, they use the chi-square statistical test to verify the independence between two variables. To determine the nature (positive or negative) of the relationship, a correlation metric is used. In (Savasere, 1998) the authors present a new idea to mine strong negative rules. They combine positive frequent itemsets with domain knowledge, in the form of a taxonomy, to mine negative associations. However, their algorithm is hard to generalize since it is domain dependant and

Table 1. Example 1 data

	organic	¬organic	Σ_{row}
non-organic	20	60	80
¬non-organic	20	0	20
Σ_{col}	40	60	100

requires a predefined taxonomy. Wu *et al* (Wu, 2002) derived another algorithm for generating both positive and **negative association rules**. They add a measure called *mininterest* to the support-confidence framework in order to better prune the frequent itemsets that are generated. In (Teng, 2002) the authors use only negative associations of the type $X \rightarrow \neg Y$ to substitute items in market basket analysis.

In (Antonie, 2004b) the authors define a *generalized **negative association rule*** as a rule that contains a negation of an item – that is, a rule whose antecedent or consequent can be formed by a conjunction of presence or absence of terms. An example of such an **association rule** is: $A \wedge \neg B \wedge \neg C \wedge D \rightarrow E \wedge \neg F$. To the best of our knowledge, there is no algorithm that can determine this type of **association rule**. Deriving such an algorithm would be quite difficult, since it involves expanding the itemset generation phase, which is already a very expensive part of the association rule mining process. Such an algorithm would need not only to consider all items in a transaction, but also all possible items absent from the transaction. There could be a considerable exponential growth in the candidate generation phase. This is especially true in datasets with highly correlated attributes. Thus, it is not feasible to extend the attribute space by adding the negated attributes and continuing to use existing **association rule** algorithms. In (Antonie, 2004b), the authors generate and use a subset of the generalized **negative association rules**, referred to as *confined **negative association rules***, in the classification process. A confined **negative association rule** is defined as follows: $\neg X \rightarrow Y$ or $X \rightarrow \neg Y$, where the entire antecedent or consequent must be a conjunction of negated attributes or a conjunction of non-negated attributes.

Finding Positive and Negative Rules for Classification

Generating a complete set of **negative association rules** is a very difficult task, as it involves the identification of all possible patterns that do not exist in a data set. For a frequent pattern in a data set, there is an exponentially larger number of possible frequent negative patterns, since any attribute that is not in the frequent pattern may either be negated or be absent in the negative pattern. Antonie and Zaïane deal with this issue by restricting the rules that are generated to those that fit the definition of confined **negative association rules** (Antonie, 2004b), which was described previously. This reduces the problem to adding at most two negative rules for each positive rule, which means that the overall number of rules increases by a small constant factor.

When processing the training data set, the algorithm uses the correlation coefficient between an itemset in the data set and each class to efficiently generate both positive and negative rules. Specifically, for each itemset I in the data set, the correlation coefficient is calculated for each class c. If corr(I, c) is above a positive threshold, then a positive rule ($I \rightarrow c$) is generated and, provided the rule's confidence is high enough, that rule is added to the positive rule set. However, if corr(I, c) is below a negative threshold, then negative rules ($\neg I \rightarrow c$ and $I \rightarrow \neg c$) are generated and, if their confidence levels are high enough, added to the negative rule set. Finally, the positive and negative rule sets are combined to create the classifier's rule set.

Classification Using Positive and Negative Rules

In (Antonie, 2004a), the classification of an item using positive and negative rules occurs as in a positive rule-based classifier for the most part. The item is compared against each rule in the rule set, and a set of matching rules is generated. These matching rules are then divided by class, the average confidence for each class is calculated, and the item is assigned the class with the highest average confidence. Note that an item matches a rule with negative attributes if the item does not contain any of the attributes in the rule. The main variation comes when considering rules which imply a negative classification. When calculating the average confidence for a class, these rules subtract from the total confidence for the class, rather than adding to it.

The set of rules that are generated, as discussed in the previous section, make up the classifier's model. This model is used to assign classification labels to new objects. Given a new object, the classification process searches in this set of rules for those classes that are relevant to the object presented for classification. The set of positive and negative rules are ordered by confidence and support. This sorted set of rules forms the basis for ARC-PAN (**Association Rule** Classification with Positive And Negative) (Antonie, 2004a) **associative classifier**, which uses an average confidence per class as a score.

Association rules of the type $X \rightarrow C$ and $\neg X \rightarrow C$ can be treated in the same way. Both of them have an associated confidence value and class label. These types of rules can be considered together and their confidence can be added to the C class total. However, the rules of the type $X \rightarrow \neg C$ have to be treated differently. Their confidences are subtracted from the total confidence of their corresponding class since they strongly indicate that the object should not be assigned to this class.

RULE PRUNING: KEEPING THE ESSENTIAL

The completeness of the initial rule set is, in theory, an advantage of associative classification over other rule-based algorithms, in that it guarantees that the classifier will not miss any potentially relevant rules. That is, all the rules meeting our definition of "relevant" (i.e., above a certain confidence threshold) are generated and included in the initial rule set. Thus, by using the initial rule set, the classifier will be able to draw on the information from *all* the relevant rules. However, in practice, the size of the generated rule set is often prohibitively large, containing thousands of rules. Thus, **associative classifiers** use **rule pruning** techniques to maintain the comprehensiveness of the rule set, while reducing the rule set to a more manageable size.

There are several heuristics that are commonly used to prune rule sets, such as removing low-ranking specialized rules, removing conflicting rules, and using database coverage. Each of these will be discussed in the following paragraphs.

Specialized rules occur when two rules have different characteristics, but the same classification. Let us consider two rules, r_1 and r_2, where $r_1 = a_1 \ldots a_n \rightarrow c$ and $r_2 = b_1 \ldots b_m \rightarrow c$ $(a_1 \ldots a_n \subset b_1 \ldots b_m)$. If $n < m$, then r_2 is considered a *specialized* version of r_1 – that is, they both provide the same classification, but r_2 requires a larger number of characteristics, and is arguably less useful than the more generally-applicable r_1. Moreover, if r_2 also has a lower confidence, then it can be safely pruned, since it requires more information than r_1 to make a classification decision, and is less confident about that decision in those cases when it is actually applicable.

Conflicting rules, meanwhile, occur when two rules have the same characteristics, but different classifications. Let us again consider two rules, r_1 and r_2, where $r_1 = a_1 \ldots a_n \rightarrow c_1$ and $r_1 = a_1 \ldots a_n \rightarrow c_2$. Thus, if we are attempting to classify an item that matches attributes $a_1 \ldots a_n$, r_1 implies that the item should be given the class label c_1, while r_2 implies that the label should be c_2. In this case, the obvious solution to the problem is simply to prune the rule with a lower confidence value, as it provides a less certain answer.

Database coverage consists of going over all the rules and evaluating them against the training instances. Whenever a rule applies correctly on some instances, the rule is marked and the instances eliminated until all training instances are covered. Finally, the unmarked rules are simply pruned.

An alternative approach to pruning described below, uses **graph-based techniques** to categorize rules according to their utility and accuracy, and then prune those rules accordingly (Zaïane, 2005).

Pruning Using Graph-Based Techniques

Rule Categorization

One crucial difficulty in pruning rules is that it is important not just to reduce the number of rules, but to do so in such a way that the accuracy of the classifier does not suffer. That is, one must reduce the number of rules while preserving the rules which do a good job of classification. To this end, Zaïane and Antonie (Zaïane, 2005) evaluate each rule in the rule set by using it to re-classify the training data. In doing so, they keep track of the number of correct and incorrect classifications made by each rule, and then graph each rule as a point on a plane, as shown in Figure 1. The graph may then be divided using various thresholds, for instance shown in Figure 1 with thick lines. Rules above the horizontal line have a large number of false positives. Rules above the diagonal line classify more items incorrectly than they do correctly. Finally, rules to the right of the vertical line classify many items correctly.

Rule Pruning Strategies

Using the thresholds concept described previously, one may divide the graph into four quadrants or regions. Given the rule evaluation related to these quadrants, the authors propose a variety of **rule pruning** schemes based around eliminating rules from particular quadrants (illustrated in Figure 2):

1. **Eliminate the high offender rules:** By tracing a horizontal line at a given threshold, we can eliminate all the rules above the line. This is illustrated in the chart on the left in Figure 2. The authors suggest a line at 50% by default but a sliding line can also be possible aiming at a certain percentage of rules to eliminate.
2. **Eliminate the rules that misclassify more than they classify correctly:** By tracing a diagonal line such rules can be identified. This is illustrated in the chart in the middle of Figure 2. Notice that when the axes of the plot are normalized, the diagonal indicates the rules that correctly classify as many

Figure 1. Quadrants for pruning rules

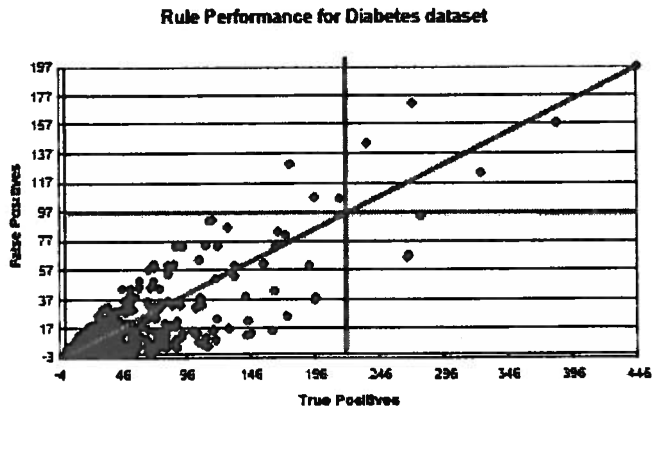
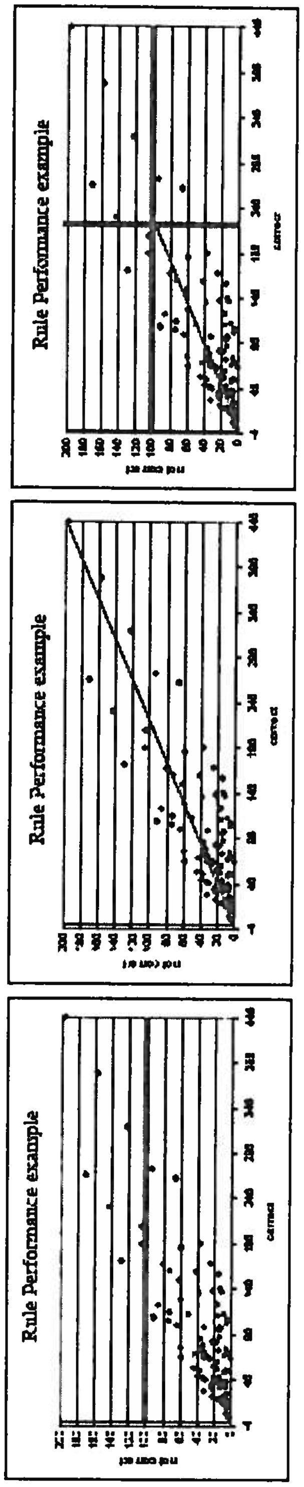


Figure 2. Filtering by quadrant and diagonal slicing



times as they misclassify. When the axes are not normalized, the diagonal indicates a relative ratio, which the authors advocate.

3. **Elimination by quadrant slicing:** The plot could be divided into four regions, as shown in the chart on the right of Figure 2. The top left (*Region A*) contains rules that are incorrect more than they are correct. The top right (*Region B*) contains rules that are frequently used but equally misclassify and correctly classify. The bottom left (*Region C*) has rules that are infrequently used but equally misclassify and correctly classify. Finally, the bottom right (*Region D*) contains the good rules which frequently classify correctly but seldom misclassify. The idea is to successively remove the rules that are in *Region A*, then *Region B*, then *Region C*.
4. **A combination of the above methods:** After removing regions *A* and *B*, eliminating the rules in *Region C* (bottom left) can be costly because many rules may be seldom used but have no replacements. Once removed, other rules are "forced" to play their role and can in consequence misclassify. The idea is to use a diagonal line to identify within *Region C* the rules that misclassify more than they are correct. This strategy strikes a good balance between removing a sufficient number of rules and retaining enough accurate rules to maintain effective classification.

Pruning classification rules is a delicate enterprise, because even if a rule misclassifies some objects, it has a role in correctly classifying other objects. When removed, there is no guarantee that the object the rule used to correctly classify will be correctly classified by the remaining rules. This is why we advocate progressive strategies depending upon the datasets at hand.

RULE SELECTION

Once the rule set has been pruned, all that remains is to classify unknown items using this optimized rule set. If for an unknown item only one relevant rule exists, then classifying the item would be quite straightforward: find the relevant rule for that item, and assign the class label implied by the relevant rule. However, even after pruning, rule sets for an **associative classifier** can contain a large number of rules. Thus, when classifying an item, it is very unlikely that only one rule will apply. Rather, there may be dozens or even hundreds of rules that apply for a particular item, meaning the potential application of a wide variety of class labels. How, then, does the classifier determine which class label to apply? Researchers have suggested a variety of answers to this problem. The following paragraphs will discuss four such techniques: CBA (Liu, 1998), CMAR (Li, 2001), ARC-AC and ARC-BC (Antonie, 2002a), and 2SARC (Antonie, 2006).

CBA may be viewed as the most straightforward of these techniques. It seeks to find a single "best" rule to classify the unknown item (Liu, 1998). Typically, the best rule is determined using Confidence, Support, size of Antecedent (CSA) ordering – that is, the rule with the highest confidence is used, unless there is a tie. In this case, the rule with the higher support is used. In the (highly unlikely) event that there is more than one rule with the same confidence and support, the size of the antecedent is used to break a tie. CBA has the advantage of being quite intuitive, and fairly straightforward to implement. However, relying on a single rule to classify an item can lead to incorrect classifications if that rule is not the best match for the item. For instance, a general rule with high confidence will invariably take precedence over a more specific rule with a slightly lower confidence. A way to solve this particular problem is to use ACS (antecedent, confidence, support) ordering, thus giving priority to more specific rules (Coenen, 2004). However, the reliance on a single rule remains a drawback to the CBA technique, in general.

CMAR, Classification based on Multiple Association Rules, is one technique that considers groups of rules in making a classification decision. Specifically, it groups the rules that apply according to their class labels, and uses a weighted χ^2 measure to "integrate both information of correlation and popularity" into the classification process (Li, 2001). The overall effect of each group of rules is calculated, and the group with the strongest effect is chosen to classify the unknown item (Coenen, 2004). Thus, by considering the correlation between rules, in addition to the characteristics of individual rules such as confidence and support, the technique makes a "collective and all-round" decision which helps "avoid bias, exceptions and over-fitting" (Li, 2001).

The ARC-AC (Antonie, 2002a) technique takes a different approach to multiple rule-based classification. Like CMAR, ARC-AC groups rules according to their classification label. Instead of the weighted χ^2 value, however, ARC-AC calculates the average confidence for each class, and uses the class with the highest average confidence to classify the unknown item or the top scores based on dominance factor analysis if the application requires multi-label classification.

ARC-BC (Antonie, 2002a), Association Rule based Classification By Category, takes a slightly more refined approach than ARC-AC. This technique recognizes that rules which classify rare events may be "drowned out" by more common occurrences. In some contexts, such as classifying tumors, we are interested in precisely those rare cases. In order to ensure that rules that are able to classify rare events are not ignored during the classification process, ARC-BC generates classification rules by separating the training data into disjoint sets for each class label. Thus, the rules for class label c_i are generated from those items in the training data set with the class label c_i. By taking this approach, rules that classify rare events will be given equal weight as those that classify common events, thus ensuring that all rules are equally relevant in classifying an unknown item. The **rule selection** is also based on confidence average.

The above strategies assign classes to new objects based on the best rule applied or on some predefined scoring of multiple rules. Moreover, CMAR and ARC trade part of their comprehensibility inherited from the **association rules** for improved performance. This trade-off is the result of using a weighting score on the rules.

In (Antonie, 2006) a weighted voting scheme is proposed to combine the class predictions of the selected rules to produce a final classification. Instead of pre-defining the way in which weights are computed, 2SARC (Antonie, 2006) uses a second learning algorithm to automatically determine the weights for the application at hand. Therefore, with 2SARC, the learning takes place in two stages.

First, an **associative classifier** is learned using standard methods as described in the above sections. Second, predefined features computed on the outputs of the rules in the learned associative classifier are used as the inputs to another learning system, which is trained (using a separate training set) to weigh the features appropriately to produce highly accurate classifications. The advantage is that the **rule selection** can adapt to the data at hand rather than being static as CBA, CMAR and ARC.

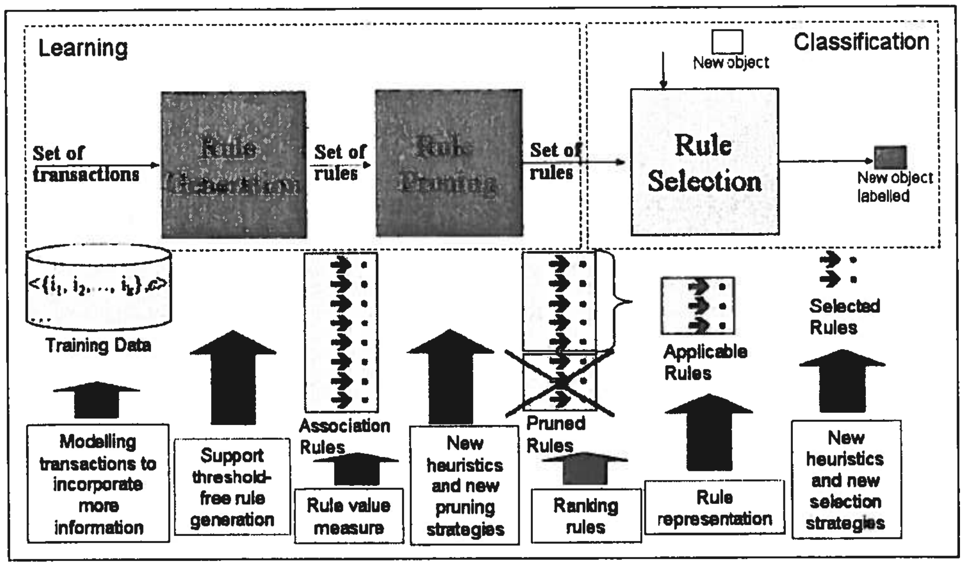
CHALLENGES AND OPEN PROBLEMS

In each stage of the associative classification process, various techniques and heuristics are used to find the best balance between various factors. When generating rules, the **rule generation** algorithm must be carefully selected and tuned so as to return a set of rules that accurately reflects the connections between item attributes and classification labels. After this rule set has been generated, it must be pruned to a manageable size, while maintaining the comprehensive quality that is one of the main advantages of associative classification over other rule-based classifiers. Once the rule set has been pruned appropriately, one must determine how to weigh the influence of numerous relevant rules in arriving at a single classification for an unknown item. At each of these stages, there is room for improvement in accuracy, flexibility and comprehensiveness. Figure 3 shows a chart of this process, with open problems clearly indicated.

In each of these open problem areas, researchers have been working to develop new solutions and techniques. In the area of **rule generation**, Wang, Do and Liu have recently developed a text classification scheme that uses a graph-based model in order to capture the relationships between terms (W. Wang, 2005). Thus, rules generated using this technique capture not only the frequency of items, but the connections between those items. This represents an important departure from conventional frequency-based rule sets, and thus requires novel approaches when determining large itemsets, significant rules, and so forth.

In the area of rule ranking, Arunaslam and Chawla have developed a new measure, the Complement Class Support (CCS) (Arunaslam, 2006). The CCS measure rules according to their strength in the class complement (that is, all classes except for the one matching the rule's class label), which results in stronger rules being assigned smaller CCS values. Using CCS results in rules which are guaranteed to be positively correlated, a desirable property when working with row enumeration algorithms, such as those used in analyzing microarray data. CCS is very well-suited for situations where there is an imbalanced distribution of classes, since it does not rely on class frequencies in the same manner as algorithms such as CBA and CMAR. Finally, CCS is relevant to **rule pruning** and threshold parameters, as it "allows the pruning of rules without the setting of any threshold parameter" (Arunaslam, 2006).

Figure 3. Open problems in associative classification



Coenen *et al* have done a great deal of work in automatically determining threshold values, thus contributing to the area of threshold-free algorithms (Coenen, 2007). In their work, they propose a hill-climbing method to determine the best threshold values when pruning rules, rather than using database coverage, a pruning technique discussed in the first part of Section 2.

Finally, in the area of rule representation, but not specifically for associative classifiers, Leung *et al* have developed a method of representing frequent itemsets based on wiring diagrams (Leung, 2007). This visualization technique follows established design principles such as minimizing edge crossings and using orthogonal coordinates in order to maximize readability and user comprehension. As well, the authors use a principle of overview and zoom in order to both manage the potentially large number of frequent itemsets and provide the user with meaningful analysis capabilities.

ANALYZING AND INTERPRETING ASSOCIATIVE CLASSIFIER RESULTS

The previous sections described the key concepts in the associative classification process, and presented several variations on this theme. Thus, the transition from raw data to classified information through the use of an **associative classifier** has been thoroughly described. However, this is not the whole story. Even the most rigorously trained and finely calibrated **associative classifier** is useless if the results are not properly interpreted by the person using it. Thus, a thorough discussion of the associative classification process is incomplete without exploring the analysis and interpretation of the classification results by the user. The following section presents the ARC-UI system, developed by Chodos and Zaïane which addresses this very issue.

Overview

The classification of items based on previously classified training data is an important area within data mining, and has many real-world applications. However, one drawback to many classification techniques, such as those based on neural networks or support vector machines (SVM), is that it is difficult for the user to understand the reasoning behind a classification result, or interpret the learned classification model. This is particularly important in a context where an expert user could make use of domain knowledge to either confirm or correct a dubious classification result.

Rule-based classifiers address this shortcoming by using a collection of simple rules to perform classification. Each rule is made up of one or more attribute/value pairs and a class, and is thus quite easy to understand. Most rule-based classifiers perform a heuristic search to discover classification rules, often missing important ones. **Associative classifiers**, on the other hand, use **association rule** mining to perform an exhaustive search to find classification rules. However, the set of rules generated by an associative classifier may contain hundreds of thousands of rules, and thus it is difficult for the user to ascertain which rules are relevant to the classification of an item, and to what extent the relevant rules influence a classification decision. The process of analyzing a set of generated **association rules**, referred to as post-mining, is a broad field and includes topics such as finding smaller, equally expressive, sets of rules and performing visual analysis on the set of rules. However, as the focus of this section is on the classification process, the discussion of post-mining will be focused on association rules used for classification.

This section presents ARC-UI, a tool that allows the user to understand the reasoning behind an associative classification result via a graphical, interactive interface. Furthermore, the user is able to modify the rules that are used and immediately see the results of this modification, thus allowing the user to improve the accuracy of the classifier through the application of domain expertise. This capability has the added benefit of increasing the user's confidence in the classifier.

The screenshots that follow were taken from the system's use in the context of classifying mushrooms. The well-known "mushroom" data set, downloaded from the UCI data repository, contains over 8,000 mushrooms that have been classified as either poisionous or edible (Asuncion, 2007). Each item in the data set contains twenty-two characteristics, such as gills-attached, colour and odor, that help determine the item's classification. The Weka data analysis tool (Witten, 2005) was used to generate classification rules (1,275 in total), which were then imported into the system. Thus, the screenshots show the system analyzing the classification of an unknown mushroom using these rules.

Classification Analysis Component

The classification component shows the result of classifying the item, as well as all other possible classifications, as shown in Figure 4. This allows the user to compare the result with other possibilities, and thus assess the likelihood of an alternative result. The classification possibilities are listed in decreasing order of likelihood, to facilitate comparison between the various possibilities.

Decision Evidence Analysis Component

The decision evidence component shows the rules that were used to classify an item. This gives the user an initial understanding of the reasoning used by the classifier. If the relevant rule set is small enough, these rules are shown in a bar graph, as in Figure 5.

However, if the rule set is too large for this to be feasible, the bar graph is compressed in order to present the rule set characteristics in a meaningful, visual manner, as shown in Figure 6. By moving the mouse over the bars, details of individual rule is displayed. In either case, compressed or non compressed, the bar graph is colour-coded according to the class labels, to facilitate comparison among the rules shown. As well, the component presents a summary of the rules influencing each classification possibility. This summary includes the rules' confidence values and the overall confidence for each class, which is calculated by combining rules using different methods, as specified by the user.

Decision Speculation Component

The decision speculation component allows the user to modify the item being classified, the method used to calculate the confidence for each class, and the rules used in classification. After performing the desired modifications, the user is immediately shown the results of this modification. The decision speculation interface is shown in Figure 7. This allows the user to experiment with the classification engine, thus offering insight into the process behind item classification. In selecting the confidence calculation method, the user may choose between the best rule (Liu,

Figure 4. Classification component

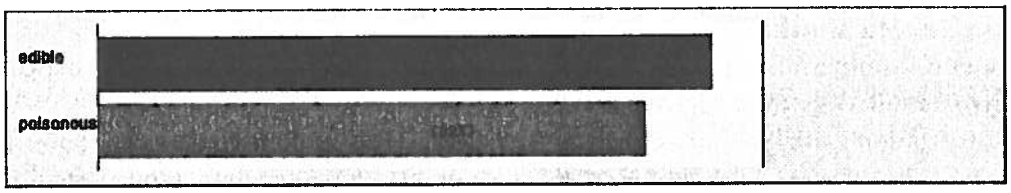
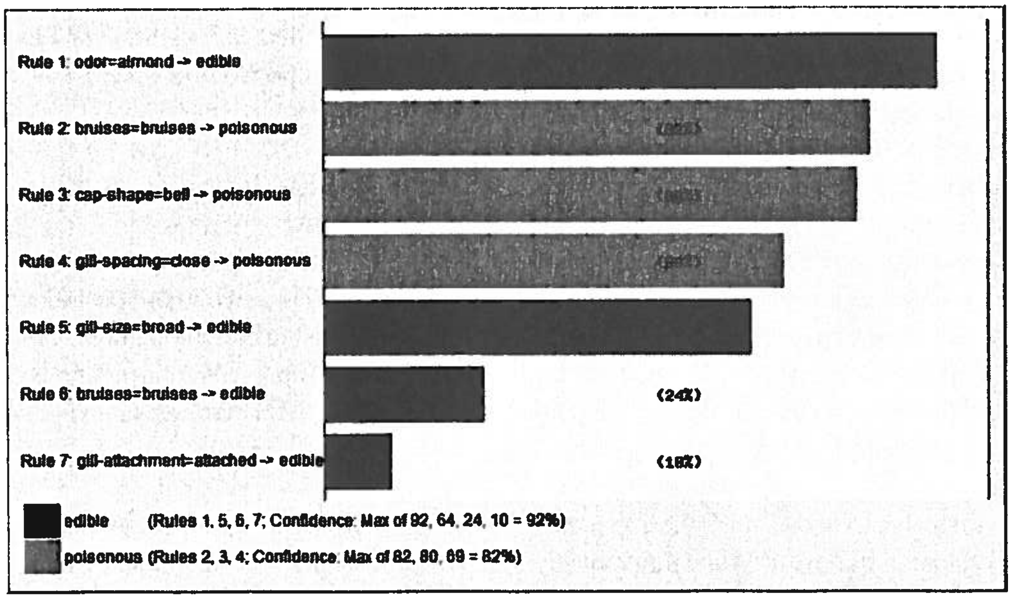


Figure 5. Decision evidence component


Rule 1: odor=almond -> edible

Rule 2: bruises=bruises -> poisonous

Rule 3: cap-shape=bell -> poisonous

Rule 4: gill-spacing=close -> poisonous

Rule 5: gill-size=broad -> edible

Rule 6: bruises=bruises -> edible (24%)

Rule 7: gill-attachment=attached -> edible (10%)

■ edible (Rules 1, 5, 6, 7; Confidence: Max of 92, 64, 24, 10 = 92%)

▨ poisonous (Rules 2, 3, 4; Confidence: Max of 82, 80, 69 = 82%)

1998) and average rule methods (Antonie, 2002b). When editing the rules used in classification, the user can:

- Edit the classification or confidence for a rule
- Add, modify or delete clauses within a rule
- Remove a rule entirely (causing it to be ignored)
- Create a new rule

This editing capability is provided via a simple interface, as shown in Figure 8. Thus, the user can draw on expert knowledge to edit the computationally-generated rule set. Moreover, the user is shown immediately whether this modification improved the accuracy of the classifier. It should be noted that the speculative changes made by the user are not immediately made permanent. However, the user has the option of making the speculative changes permanent in the classification model, once the results of these changes have been presented, and accepted by the user. Thus, the tool offers the ability to interactively analyze and improve the classifier.

Figure 6. Decision evidence in compressed format

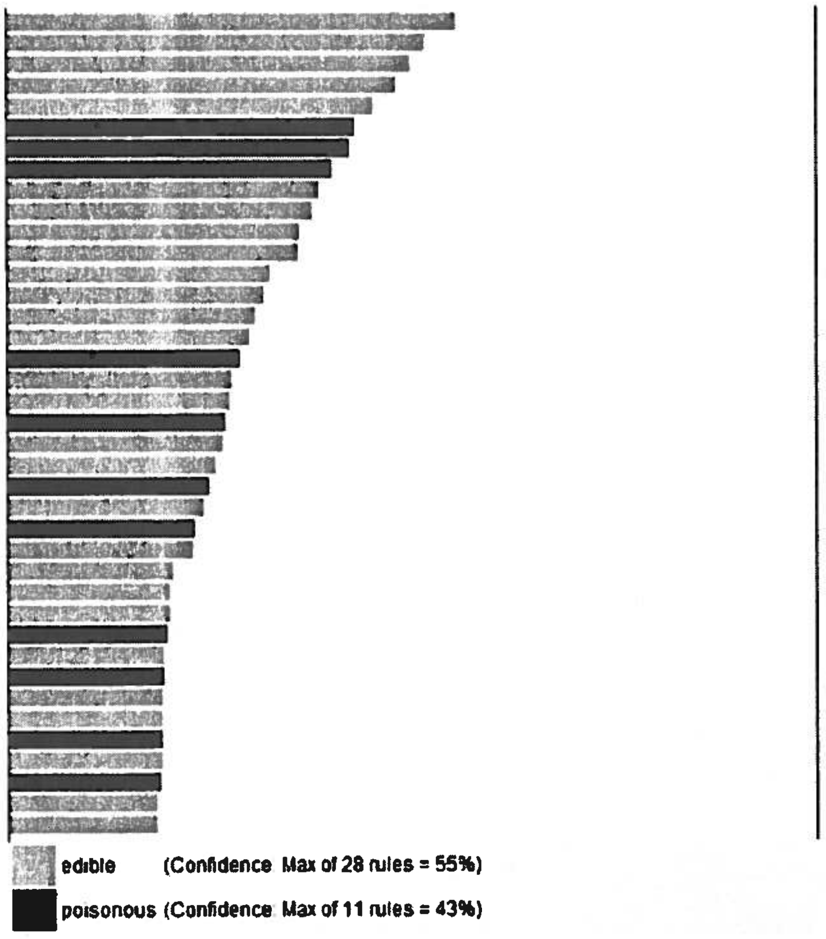
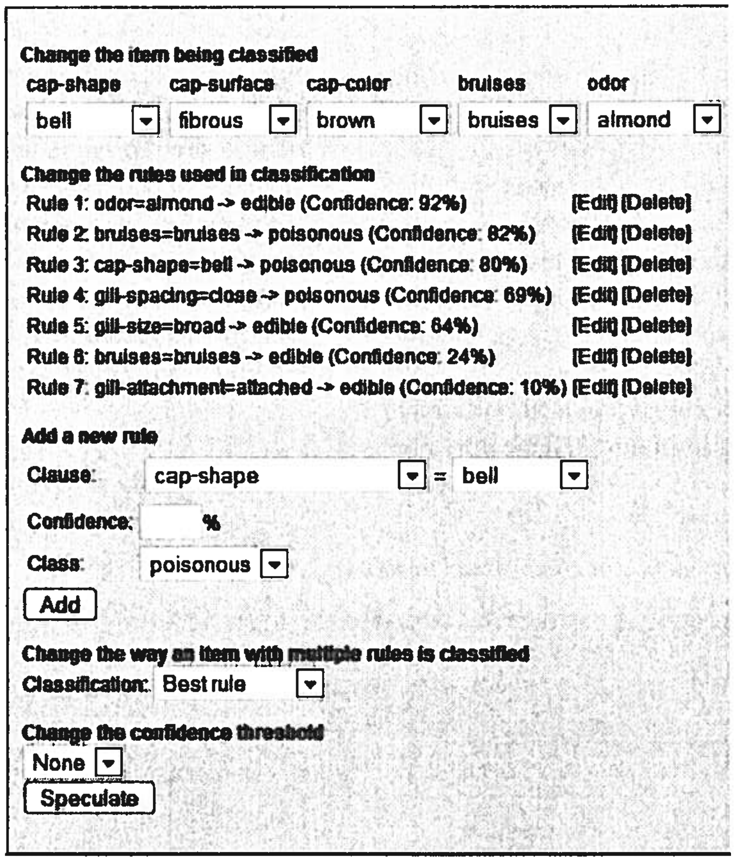


Figure 7. Decision speculation component



Change the item being classified

cap-shape	cap-surface	cap-color	bruises	odor
bell	fibrous	brown	bruises	almond

Change the rules used in classification

Rule 1: odor=almond -> edible (Confidence: 92%) [Edit] [Delete]
Rule 2: bruises=bruises -> poisonous (Confidence: 82%) [Edit] [Delete]
Rule 3: cap-shape=bell -> poisonous (Confidence: 80%) [Edit] [Delete]
Rule 4: gill-spacing=close -> poisonous (Confidence: 69%) [Edit] [Delete]
Rule 5: gill-size=broad -> edible (Confidence: 64%) [Edit] [Delete]
Rule 6: bruises=bruises -> edible (Confidence: 24%) [Edit] [Delete]
Rule 7: gill-attachment=attached -> edible (Confidence: 10%) [Edit] [Delete]

Add a new rule

Clause: cap-shape = bell
Confidence: %
Class: poisonous

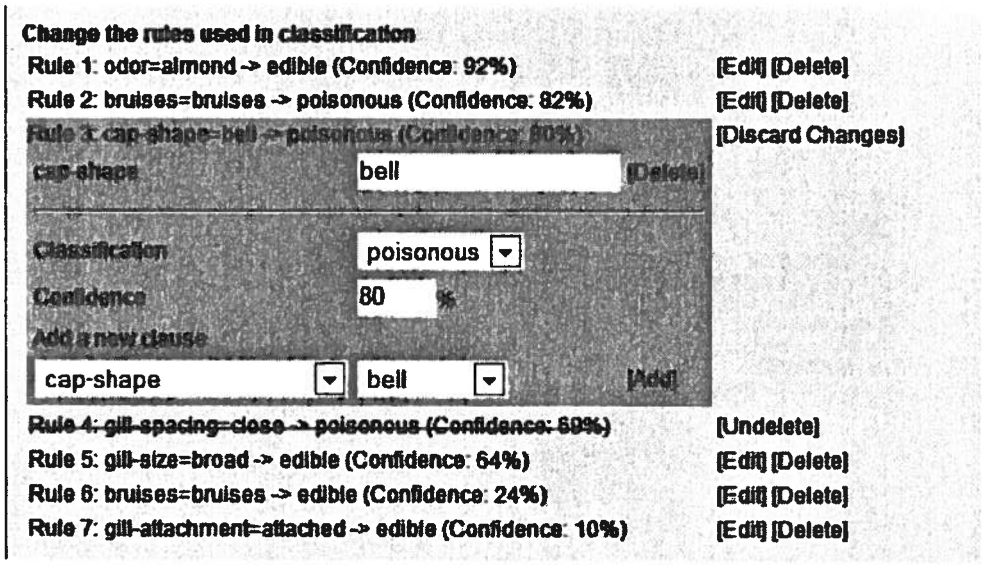
Change the way an item with multiple rules is classified

Classification: Best rule

Change the confidence threshold

None

Figure 8. Rule editing interface



Change the rules used in classification

Rule 1: odor=almond -> edible (Confidence: 92%) [Edit] [Delete]
Rule 2: bruises=bruises -> poisonous (Confidence: 82%) [Edit] [Delete]
Rule 3: cap-shape=bell -> poisonous (Confidence: 80%) [Discard Changes]
cap-shape bell [Delete]
Classification poisonous
Confidence 80 %
Add a new clause
cap-shape bell [Add]
~~Rule 4: gill-spacing=close -> poisonous (Confidence: 69%)~~ [Undelete]
Rule 5: gill-size=broad -> edible (Confidence: 64%) [Edit] [Delete]
Rule 6: bruises=bruises -> edible (Confidence: 24%) [Edit] [Delete]
Rule 7: gill-attachment=attached -> edible (Confidence: 10%) [Edit] [Delete]

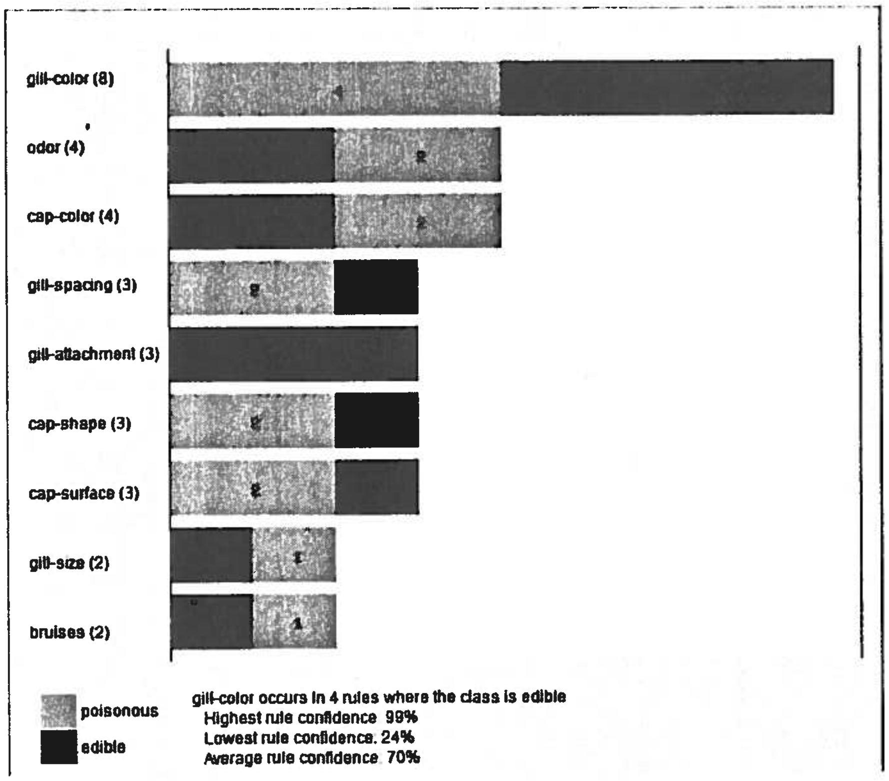
Ranks of Evidence Analysis Component

The ranks of evidence component shows the relationships between characteristics, **association rules** and classifications. This provides the user with further information about the way the classifier works, independent of any particular item or rule set. The system uses a colour-coded bar chart-based visualization scheme, as shown in Figure 9. The length of each bar indicates the total number of times a characteristic appears in the rule set. The colour-coded segments show the number of rules containing a given characteristic that result in a particular classification. By moving the mouse over each segment, the use is shown a more detailed summary of the rules that contain a given characteristic and result in the selected classification. This approach is both visually appealing and scalable, which is quite beneficial when dealing with very large rule sets. In Figure 10, we see that the "gill-spacing" characteristic appears in three rules, two of which have the class "poisonous", and one with the class "edible" (represented by green and red segments, respectively). By placing the mouse over the "poisonous" segment of the bar for the "gill-spacing" characteristic, we are shown more information about the rules containing the "gill-spacing" characteristic where the class is "poisonous", as shown in Figure 9.

Source of Evidence Analysis Component

Finally, the source of evidence component allows the user to make connections between the item

Figure 9. Ranks of evidence component – attribute/value pair information

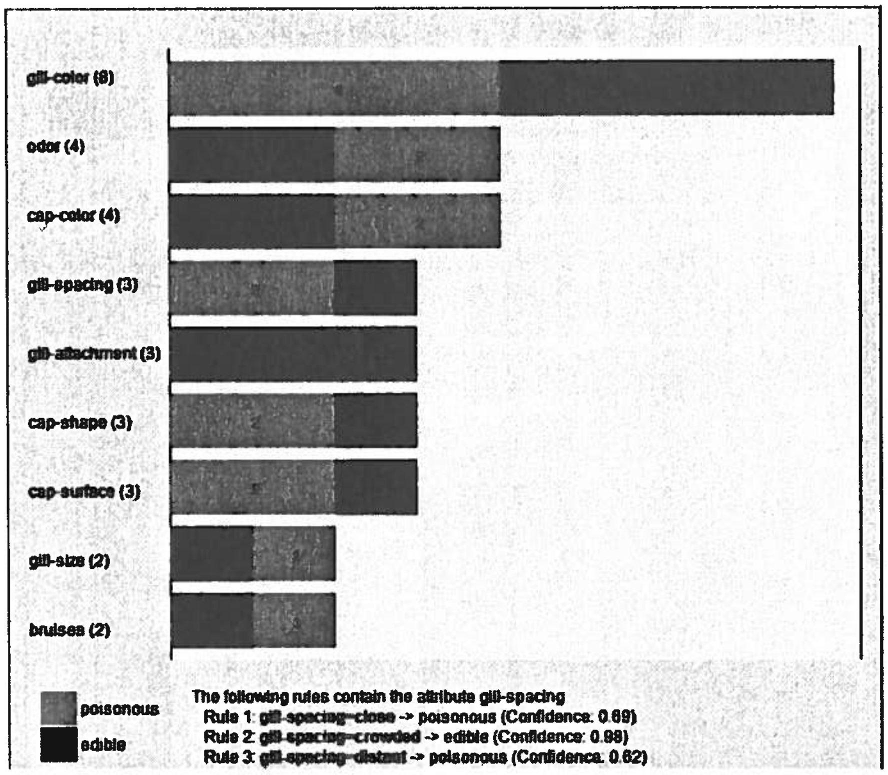


being classified and the entries in the data set that were used to generate the associative classification rules. This may be useful when investigating a dubious classification result -- the user can check the data used in training the classifier to see if there were any anomalies in that original data. Specifically, the component shows the entries in the training set in a colour-coded list using shades of red and green, as shown in Figure 11. A green entry indicates that the entry has the same class as the item being analyzed, while a red entry indicates that they have different classes. The intensity of the colour indicates the proximity of the entry to the current item, in terms of matching attribute-value pairs. Finally, the user is able to specify a variety of further analysis options to restrict the list of entries to those matching certain classification or characteristic criteria. In particular, when filtering by attribute, the user is shown a chart of the distribution of that attribute among the possible classifications, divided by the possible attribute values. Figure 12 shows the class breakdown for the possible values of the "cap-shape" attribute. For example, in 89% of the 402 items containing the "cap-shape=bell" attribute-value pair, the class was "edible". The table also shows that there were only two items which contained "cap-shape=conical" both of which poisonous, and thus this attribute-value pair had very little impact on the classification model.

CONCLUSION

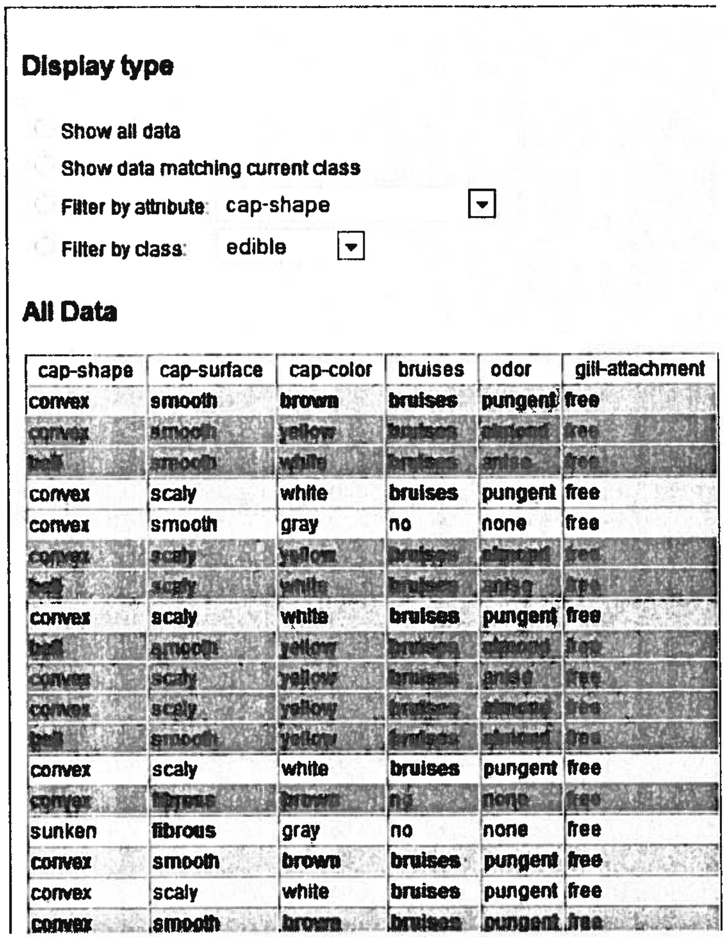
Association rules, originally introduced in the context of market basket analysis, have many more

Figure 10. Ranks of evidence component – overall attribute information



The following rules contain the attribute gill-spacing
Rule 1: gill-spacing=close -> poisonous (Confidence: 0.69)
Rule 2: gill-spacing=crowded -> edible (Confidence: 0.98)
Rule 3: gill-spacing=distant -> poisonous (Confidence: 0.62)

Figure 11. Source of evidence component



Display type

Show all data

Show data matching current class

Filter by attribute: cap-shape

Filter by class: edible

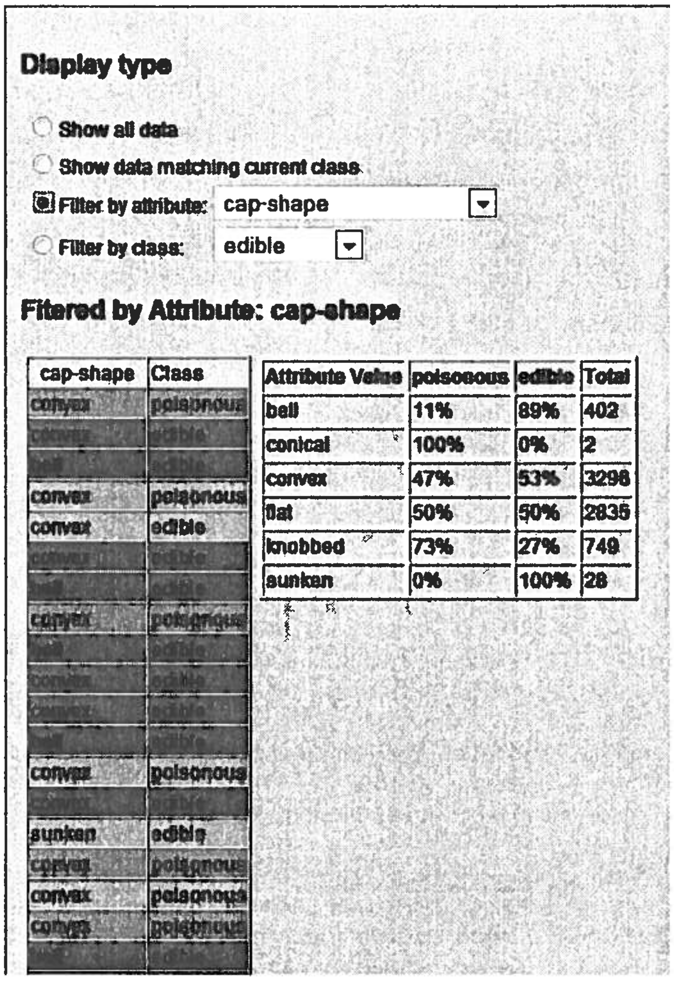
All Data

cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment
convex	smooth	brown	bruises	pungent	free
convex	smooth	yellow	bruises	almond	free
bell	smooth	white	bruises	anise	free
convex	scaly	white	bruises	pungent	free
convex	smooth	gray	no	none	free
convex	scaly	yellow	bruises	almond	free
bell	scaly	white	bruises	anise	free
convex	scaly	white	bruises	pungent	free
bell	smooth	yellow	bruises	almond	free
convex	scaly	yellow	bruises	anise	free
convex	scaly	yellow	bruises	almond	free
bell	smooth	yellow	bruises	almond	free
convex	scaly	white	bruises	pungent	free
convex	fibrous	brown	no	none	free
sunken	fibrous	gray	no	none	free
convex	smooth	brown	bruises	pungent	free
convex	scaly	white	bruises	pungent	free
convex	smooth	brown	bruises	pungent	free

applications. We discussed here the use of **association rules** in building a classification model. Classifiers that use **association rule** mining to learn a classification model from a training set are called **associative classifiers**. In the first part of this chapter, the process of associative classification is explained briefly, and the three phases of the process – **rule generation, rule pruning,** and **rule selection** – are explored in detail. For each phase, the principal ideas of the phase are explained, and the phase is placed within the overall context of the associative classification process. As well, variations on each phase are described; these variations expand and improve the classification possibilities in various ways. Finally, open problems in the field are identified, and work in these areas is briefly described.

In the second part, the issue of understanding associative classification results is addressed via ARC-UI, an analysis and speculation tool. Using the analysis tools, one can find out which rules were involved in the classification, the effect of different attribute values, and the connection between the item being classified and the entries in the training data set. Moreover, using the speculation tool, one can temporarily modify

Figure 12. Source of evidence chart



Display type

- ○ Show all data
- ○ Show data matching current class
- ● Filter by attribute: cap-shape
- ○ Filter by class: edible

Filtered by Attribute: cap-shape

cap-shape	Class
convex	poisonous
convex	edible
bell	edible
convex	poisonous
convex	edible
convex	edible
bell	edible
convex	poisonous
bell	edible
convex	edible
convex	edible
bell	edible
convex	poisonous
convex	edible
sunken	edible
convex	poisonous
convex	poisonous
convex	poisonous
[illegible]	[illegible]

Attribute Value	poisonous	edible	Total
bell	11%	89%	402
conical	100%	0%	2
convex	47%	53%	3298
flat	50%	50%	2835
knobbed	73%	27%	749
sunken	0%	100%	28

either the item being analyzed or the rules used in analysis, and see the classification that results from this modification. Thus, the user is given the ability to interactively test and improve the classifier, which increases both the classifier's accuracy and the user's trust in the system.

REFERENCES

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of SIGMOD*, 22(2), 207–216.

Antonie, M.-L., & Zaïane, O. R. (2002a). Text document categorization by term association. In *Proceedings of ICDM*, (pp. 19–26).

Antonie, M.-L., & Zaïane, O. R. (2004a). An associative classifier based on positive and negative rules. In *9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD-04)*, (pp. 64–69), Paris, France.

Antonie, M.-L., & Zaïane, O. R. (2004b). Mining positive and negative association rules: An approach for confined rules. In *8th European Conference on Principles and Practice of Knowl-*

edge Discovery in Databases (PKDD 04), (pp. 27–38), Pisa, Italy.

Antonie, M.-L., Zaïane, O. R., & Holte, R. (2006). Learning to use a learned model: A two-stage approach to classification. *In Proceedings of ICDM*, (pp. 33–42).

Antonie, M.-L., & Zaïane, O.R. (2002b). Text document categorization by term association. In *IEEE Data Mining (ICDM)*, (pp. 19–26).

Antonie, M.-L., & Zaïane, O.R. (2004c). An associative classifier based on positive and negative rules. In *DMKD '04: Proceedings of the ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, (pp. 64–69), New York, NY, USA.

Arunaslam, B., & Chawla, S. (2006). *CCCS: A top-down associative classifier for imbalanced class distribution* (Technical report). University of Sydney, School of IT.

Asuncion, A., & Newman, D.J. (2007). UCI machine learning repository.

Bayardo, R. (1997). Brute-force mining of high-confidence classification rules. In *SIGKDD 1997*, (pp. 123–126).

Brin, S., Motwani, R., & Silverstein, C. (1997). Beyond market basket: Generalizing association rules to correlations. *In Proceedings of the 1997 ACM-SIGMOD International Conference on the Management of Data*, (pp. 265–276), Tucson, Arizona.

Coenen, F., & Leng, P. (2007). The effect of threshold values on association rule based classification accuracy. In *Journal of Data and Knowledge Engineering, 60*(2), 345–360.

Coenen, F., & Leng, P. (2004). An evaluation of approaches to classification rule selection. In *ICDM 2004*, (pp. 359–362).

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of SIGMOD*, (pp. 1–12).

Leung, C. K., & Carmichael, C. (2007). *Frequent itemset visualization* (Technical report). University of Manitoba.

Li, W., Han, J., & Pei, J. (2001). CMAR: Accurate and efficient classifcation based on multiple class-association rules. In *ICDM 2001*, (pp. 369–376).

Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining. *Proceedings of SIGKDD*, (pp. 80–86).

Lim, W.-K., Ong E.-P., & Ng, K.-L (2001). Multi-level rules with recurrent items using fp'-tree. In *Proceedings of ICICS*.

Poulin, B., Eisner, R., Szafron, D., Lu, P., Greiner, R., Wishart, D.S., Fyshe, A., Pearcy, B., MacDonnell, C., & Anvik, J.. Visual explanation of evidence in additive classifiers. *In Proceedings of the Conference on Innovative Applications of Artificial Intelligence*, (pp. 1–8).

Rak, R., Stach, W., Zaïane, O.R., & Antonie, M. (2005). Considering re-occurring features in associative classifiers. In *PAKDD'05: Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, (pp. 240–248).

Savasere, A., Omiecinski, E., & Navathe, S. (1998). Mining for strong negative associations in a large database of customer transactions. In *Proceedings of ICDE*, (pp. 494–502).

Teng, W. G., Hsieh, M. J., & Chen, M. S. (2002). On the mining of substitution rules for statistically dependent items. In *Proceedings of ICDM*, (pp. 442–449).

Wang, J., & Karypis, G. (2005). Harmony: Efficiently mining the best rules for classification. In *Proceedings of SIAM*.

Wang, W., Bich Do, D., & Lin, X. (2005). Term graph model for text classification. In *Advanced Data Mining and Applications*, pages 19–30.

Wang, W., Yang, J., & Yu, P. (2004). War: Weighted association rules for item intensities. *Knowledge and Information Systems, 6*, 203–229.

Witten, I. H., & Frank, E. (2005). Data Mining: Practical machine learning tools and techniques. Morgan Kaufman, 2nd edition.

Wu, X., Zhang, C., & Zhang, S. (2002). Mining both positive and negative association rules. In *Proceedings of ICML*, (pp. 658–665).

Yin, X., & Han, J. (2003). CPAR: Classification based on predictive association rules. In *Proceedings of SDM.*

Zaïane, O. R., Han, J., & Zhu, H. (2000). Mining recurrent items in multimedia with progressive resolution refinement. In *Proceedings of ICDE*, (pp. 461–470).

Zaïane, O. R., & Antonie, M.-L. (2005) On pruning and tuning rules for associative classifiers. In *KES'05: Knowledge-Based Intelligence Information & Engineering Systems*, (pp. 966–973).