Adding Semantics and Rigor to Association Rule Learning: the GenTree Approach

Yiheng Li Latanya Sweeney

January 2005
CMU-ISRI-05-101

Institute for Software Research International
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

Learning semantically useful association rules across all attributes of a relational table requires: (1) more rigorous mining than afforded by traditional approaches; and, (2) the invention of *knowledge ratings* for learned rules, not just statistical ratings. Traditional algorithms began by learning rules over one attribute expressed in the domain values of that attribute (Srikant and Agrawal, 1995). "People who buy diapers tend to buy beer" is an example from grocery purchases. In the second generation, Srikant and Agrawal, (1996) introduced a *hierarchy* whose base values are those originally represented in the data, and values appearing at higher levels in the hierarchy represent increasingly more general concepts of base values. Rules learned over the attribute using the hierarchy are termed *generalized association rules* (or *cross-level* rules). "People who buy baby products tend to buy controlled substances" is an example of a generalized association rule. Rules that mix the level of expressions for the same item were still not able to be learned. "People who buy beverages tend to buy beer" is an example of a rule that could not be properly learned if most beverage purchases were of beer. The work reported herein continues the evolution in the expressiveness of association rules to its broadest application – semantically rated rules learned from a large relational table having many attributes. Associated with each attribute is a hierarchy. We find the rules that are formed by combining mixed levels of generalizations across all attributes and that convey the maximum expression of information supported by attribute hierarchies, parameter settings and data tuples. We term these *robust rules*. An example of a robust rule is "People who buy baby products tend to buy controlled substances, use credit cards and make purchases on Saturdays." Multiple attributes from the transaction are included and cross-level and mix-level rules are expressed. Because of the improved expressiveness, many more statistically relevant rules are learned. To identify semantically useful rules among them, we compare rule features appearing on the left side (before "tend to") to those appearing on the right side (after "tend to") to identify *learning in breadth* (more attributes appearing on the right-side than on the left) and *learning in depth* (values on the right side appearing at lower levels in the corresponding hierarchies than those on the left). For a set of learned rules surpassing statistical thresholds of relevance, the most semantically relevant rules are those expressing more depth and/or breadth in the terms of their expression. The other rules are merely subset expressions. We also introduce a *GenTree* algorithm as an efficient means to learn robust rules from a table. Experiments using GenTree with two real-world datasets, containing 10,000 six-attributed tuples and over 4,000 eight-attributed tuples each, show that learned rules convey more comprehensive information than possible with traditional association rule mining algorithms.

This research was supported in part by the Data Privacy Laboratory, the Center for Automated Learning and Discovery, and the NSF Aladdin Center, all in the School of Computer Science at Carnegie Mellon University.

Keywords: *Association rules, classification problems, data mining, knowledge acquisition, rule learning, hierarchical learning.*

1. Introduction

The problem of discovering meaningful associations across all data elements in a large relational table requires: (1) more rigorous mining than afforded by traditional approaches to association rule learning; and, (2) the invention of knowledge ratings for learned rules, not just statistical ratings. Learned associations, expressed as rules, are the basis of association rule learning, first introduced by Srikant and Agrawal in 1993 [1]. Given sets of items appearing in transactions in a dataset, an association rule is an expression of the form *Body* ⇒ *Head*, where *Body* and *Head* are sets of items. An association rule can be stated as "*Body* tends to *Head*." An example of an association rule is: "People living near MIT (ZIP[1] 02139) tend to be Democrats[2]." An association rule is corroborated by the data from which it is observed by two measures – support and confidence. *Support* is the probability that both *Body* and *Head* occur in a transaction: P(*Body* & *Head*). *Confidence* is the conditional probability that given *Body* appears in a transaction, *Head* also appears: P(*Head* | *Body*). In a list of registered voters, the association rule "People living near MIT (ZIP 02139) tend to be Democrats" has 19% support and 57.9% confidence.

Finding the most "meaningful" rules for decision makers not only requires quantifiable corroboration using support and confidence, but also requires the ability to express rules in semantic terms useful to decision makers. Association rule learning can be formulated as a problem of searching through a predefined space of potential rules for the rules that best fit the examples quantifiably and semantically.

By selecting a rule representation, the human designer implicitly defines the space of all rules that an association rule learner (or "learner") can ever represent and therefore can ever learn. Imielinski and Mannila [3] were among the first to recognize that "there is no such thing as real discovery [in data mining], just a matter of the expressive power of the query language [used to learn]." We use the symbol R to denote the set of all possible rules that a learner may consider regarding a relational table $\mathcal{D}$. R is determined by the human designer's choice of representations to express the semantics associated with data values in $\mathcal{D}$. The simplest choice is to rely on values in $\mathcal{D}$ alone, as was done when association rule learning was first introduced [1]. In this case, learned rules can only be expressed in the base values of $\mathcal{D}$. The association rule "People who buy diapers tend to buy beer" is an example of a traditional association rule learned from a database of grocery purchases. The initial approach to learning association rules is based on identifying "Frequent Itemsets" using the *a priori* algorithm – i.e., find all combinations of values appearing in transactions and then partition the values within the combinations into eligible bodies and heads of rules and compute statistical relevance. Even though some values in $\mathcal{D}$ may be semantically related to other values (e.g., both milk and juice are beverages) there is no representation provided by which the learner can consider joining related values (milk and juice) to attain more generalized rules (about beverages) that may be more corroborated or more generally expressed.

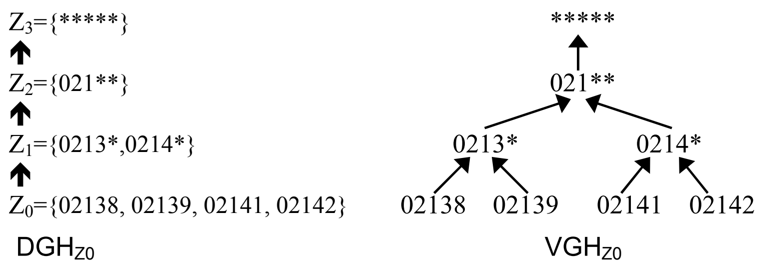


Figure 1. Attribute hierarchy for ZIP codes in Cambridge, Massachusetts, shown as domain generalization hierarchy on left and as value generalization hierarchy on right.

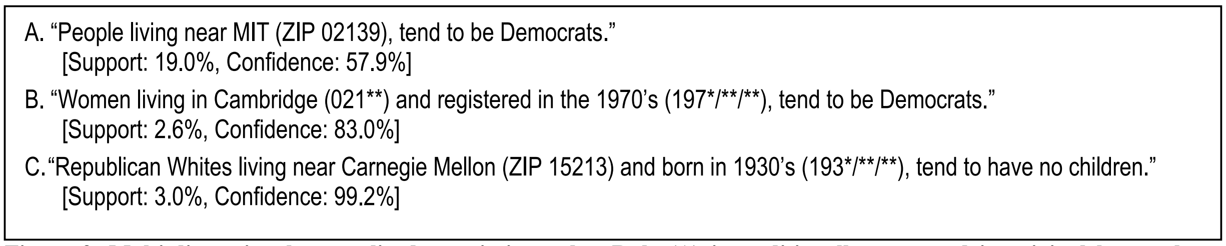
[1] Five-digit ZIP codes are postal codes in the United States. Including a digit, left to right, identifies an increasingly smaller geographic area. MIT is in ZIP 02139, Harvard in 02138 and Carnegie Mellon in 15213.
[2] In the United States, there are two primary political affiliations – Democrats and Republicans.

In 1996, the notion of associating a concept hierarchy with base values in transactional data was introduced [2, 10]. Rules could then be learned not only over the base values but also over generalized concepts that semantically related base values. For example, given transactions of grocery purchases and an "is-a" hierarchy that related milk and juice as beverages, rules involving milk, juice and beverages could then be learned. Such rules are called *generalized association rules* (or "cross-level" rules). Generalized association rule learners followed the traditional a proiri algorithm approach: find all Frequent Itemsets across all possible combinations of different levels in the hierarchy, and then treat partitions (subsets and complementary subsets) as eligible bodies and heads of rules. "People who buy diapers tend to buy beverages" is an example of a generalized association rule learned from grocery purchases using an "is-a" hierarchy over purchased items.

While generalized association rule learners are an improvement, these learners still cannot express rules that mix the level of expression for the same values because they partition values within transactions into subsets. "People who buy beverages tend to buy beer" is an example of a rule that cannot be learned (with the proper statistical corroboration) if most beverage purchases were of beer. This shortcoming can be easily overcome, but more inherently serious shortcomings include semantic redundancy and imprecise rule statements. We will further discuss these limitations after introducing the concept of *robust rules*.

The work reported herein continues the evolution in the expressiveness of association rules to its broadest application – semantically rated rules learned from a large relational table having many attributes. We allow the human designer to express a much larger and semantically extended rule space by associating membership hierarchies (e.g., "is-a") with each attribute in a large relational table $\mathcal{D}$. The base values appearing in an attribute of $\mathcal{D}$ appear as base values in the attribute's hierarchy. Values appearing at higher levels in the hierarchy represent more general concepts that remain semantically consistent with the base values. Therefore, attribute hierarchies impose a "more-general-than" relation on their values. Figure 1 has an example of an attribute hierarchy for postal codes (ZIP) in Cambridge, Massachusetts. Attribute hierarchies provide meaningful levels of concept aggregation, which could be the encoding of ZIP codes or dates, or the taxonomy ("is-a") of products. Even quantitative data such as age or income, can have meaningful attribute hierarchies achieved by aggregating base values into increasing ranges. For example, dates of birth can be generalized to month and year of birth, year of birth, 5-year age ranges, and so forth.

Assigning a hierarchy to each attribute greatly expands the size of the space of potential rules (R) a learner must consider. These rules can now express mixed levels of concept aggregation realized by combinations of values across different hierarchy tiers. We now introduce this as a natural extension to association rule learning and term these *multi-dimensional generalized association rules*. An example is: "People who buy baby products tend to buy controlled substances, use credit cards and make purchases on Saturdays." Figure 2 provides additional examples. This paper addresses new benefits realized and methods to overcome shortcomings related to this extension.


A. "People living near MIT (ZIP 02139), tend to be Democrats."
[Support: 19.0%, Confidence: 57.9%]

B. "Women living in Cambridge (021**) and registered in the 1970's (197*/**/**), tend to be Democrats."
[Support: 2.6%, Confidence: 83.0%]

C. "Republican Whites living near Carnegie Mellon (ZIP 15213) and born in 1930's (193*/**/**), tend to have no children."
[Support: 3.0%, Confidence: 99.2%]

Figure 2. Multi-dimensional generalized association rules. Rule (A) is traditionally expressed in original base values. Rules (B) and (C) have terms from higher levels in their hierarchies. The source table for (A) and (B) has attributes {*5-digit ZIP*, *gender*, *registration date* (*year/month/day*), *party*}. The table for (C) additionally includes {*race/ethnicity*, *own home*, *number of children*}.

Increasing the size of R yields additional rules to consider, but the learner's task is to find "meaningful" rules, which are based on both quantifiable corroboration and semantic expressiveness. Support and confidence quantify corroboration without additional human input. The idea is to assign "knowledge ratings" to rules requires the automated learner to know which characterizations of information are semantically relevant to the human decision maker. We achieve this using attribute hierarchies, in part. Predefined attribute hierarchies embed a priori groupings of concepts considered useful to the human decision-makers who constructed them and who interpret learned rules based on them. We introduce an automated learner capable of exploiting the semantics of the "more-general-than" relation imposed by attribute hierarchies. Doing so improves the expressive power of rules presented to humans.

Here is an example. Suppose each multi-dimensional generalized association rule in Figure 3 has the same quantified corroboration in data (e.g., the same support and confidence for the same set of tuples). Yet, to human decision-makers these statements do not equally convey the same expressive power.

A. "People living in ZIP 1521*, tend to be registered in 1965 (1965/**/**)."
B. "People living in ZIP 152**, tend to be registered in August 1965 (1965/08/**)."
C. People living in ZIP 152**, tend to be living in ZIP 1521* and registered in August 1965 (1965/08/**)."

Figure 3. Multi-dimensional generalized association rules learned from a table having attributes {*ZIP*, *registration date*} and having equal support and confidence for the same corroborating tuples. Rule (C) is semantically most robust, having the most general expression in the Body and the most specific expression in the Head.

To identify semantically useful rules among statistically relevant rules, we compare rule features appearing on the left side (before "tend to") to those appearing on the right side (after "tend to") to identify *learning in breadth* (based on the number of additional attributes appearing on the right-side than on the left) and *learning in depth* (based on the number and level difference of values on the right side appearing at lower levels in the corresponding hierarchies than those on the left). For a set of learned rules surpassing statistical thresholds of relevance, we postulate that the most semantically relevant rules are those expressing more depth and/or breadth in the terms of their expression. We term these *robust rules*. The other rules are merely subset expressions of the robust rules. Given the rules in Figure 3, rule C is the robust rule. More discussion appears on the semantic utility of robust rules in the next section (Section 2), but now we discuss the limitations of traditional approaches of mining generalized association rules and we show the benefits of mining robust rules.

Assume traditional approaches also take into account hierarchies of different attributes (not just one attribute as described in the original literature [1, 10]). Consider the dataset in Figure 4 that contains voter registration information for Cambridge, Massachusetts in 1997. We will assume all tuples in the dataset are only those shown in Figure 4 though the complete dataset will be used in the Results section. For hierarchies, consider the encoding of ZIP codes shown in Figure 1, noting that MIT is located in ZIP 02139 and Harvard in ZIP 02138. For registration date, consider a date hierarchy which aggregates dates from {month, day, year} to {month, year}, to {year}, and then to 10-year ranges.

ZIP	Registration Date
02139	1996/08/07
02141	1992/08/19
02139	1992/07/17
02139	1996/08/12
02140	1996/08/23

"Frequent Itemsets" corroborated by tuples highlighted	Corresponding subset and complementary
{02139, 1996/08/**}	{02139} and {1996/08/**}
{0213*, 1996/08/**}	{0213*} and {1996/08/**}
{02139, 1996/**/**}	{02139} and {1996/**/**}
{0213*, 1996/**/**}	{0213*} and {1996/**/**}

Figure 4. Dataset and Frequent Itemset Example.

	Traditional Rule	Statistical Ratings	English Interpretation
1	{02139} $\Rightarrow$ {1996/08/**}	support = 40%, confidence = 66.7%	"Voters living near MIT tend to have registered in August 1996."
2	{0213*} $\Rightarrow$ {1996/08/**}	support = 40%, confidence = 66.7%	"Voters living near MIT or Harvard tend to have registered in August 1996."
3	{02139} $\Rightarrow$ {1996/**/**}	support = 40%, confidence = 66.7%	"Voters living near MIT tend to have registered in 1996."
4	{0213*} $\Rightarrow$ {1996/**/**}	support = 40%, confidence = 66.7%	"Voters living near MIT or Harvard tend to have registered in 1996."

Robust Rule	Statistical Ratings	English Interpretation
{0213*} $\Rightarrow$ {02139, 1996/08/**}	support = 40%, confidence = 66.7%	"Voters living near MIT or Harvard tend to live near MIT and to have registered in August 1996."

Figure 5. Traditional association rules (above) [minimum support 40%, confidence 60%] with corresponding robust rule (below), as derived from dataset shown in Figure 4.

If the minimum support *minsup* = 40%, then traditional learners, such as the a priori algorithm, identify {02139, 1996/08/**}, {0213*, 1996/08/**}, {02139, 1996/**/**} and {0213*, 1996/**/**} as all legal "Frequent Itemsets" corroborated by the same two tuples highlighted in the Figure 4. The learners then partition these "Frequent Itemsets" into subsets and complementary sets respectively, and test to see if each pair of them satisfies the minimum confidence requirement. For simplicity, we assume the minimum confidence *minconf* = 60%, yielding 4 legal rules out of 8 candidates as shown in the top part of Figure 5.

Intuitively, the four traditionally derived rules appearing in the top of Figure 5 jointly convey a single piece of knowledge, but the traditional learner fails to succinctly identify this knowledge due to expressivity limitations inherent in traditional approaches. A *robust rule* expressing the overarching knowledge appears in the bottom of Figure 5. The *robust rule* semantically reflects the most accurate knowledge that is corroborated by supporting data tuples. Traditional learners produce semantic redundancies related to the robust rule, but failed to provide a single semantic expression of the rule itself.

The work described herein uses the structure imposed by attribute hierarchies and observed examples in $\mathcal{D}$ to organize the search of R for *robust rules*. Before we describe how this search is efficiently achieved in technical terms (in Section 3), we discuss prior work (in Section 2) related to the semantic utility of robust rules. After the GenTree structure and algorithm are presented (in Section 3), experimental results on real-world datasets are provided (in Section 4) that show the nature and number of *robust rules* learned. This paper ends with a discussion of future work on *robust rules* and other applications of GenTree.

2. Background

2.1 Robust Rules and Concept Learning

Our notion of robust rules expresses *concepts* learned from subsets of statistically corroborated association rules. Artificial intelligence literature contains a rich history of methods aimed at learning "concepts" (or general descriptions) drawn from observations [7]. Given training examples, a concept learner recognizes similar instances not appearing in the training examples. Plotkin (1971) provided an early formalization of the more-general-than relation [5]. Simon and Lea (1973) gave an early account of learning as search through a hypothesis space [9]. Version spaces and the Candidate-Elimination algorithm were introduced by Mitchell (1982) in which a more-general-than partial ordering is used to maintain a set of consistent hypotheses and to incrementally refine this representation as each new positive training example is encountered [4]. Sebag (1996) presents a disjunctive version space approach to learn from noisy data [8]. A separate version space is learned for each positive training example, then new instances are classified by combining the votes of different version spaces.

In the version space approach, two sets of consistent concept descriptions are maintained as observations are encountered. *Most specific* descriptions are those that cover all positive observations and exclude negative ones narrowly. *Most general* descriptions cover positive observations broadly while excluding negative examples. So, a positive observation may make the most specific description more general. Conversely, a negative observation may make the most general description more specific. For example, in Figure 5, traditional rule 1 is the most specific description of the corroborating tuples; and likewise, rule 4 is the most general description of those tuples.

We combine the notion of most specific and most general descriptions in robust rules by exploiting the view of an association rule as an implication. The interpretation of association rules by humans is like an *if-then* implication. In the mathematical implication $p \Rightarrow q$, p is the hypothesis, q is the conclusion, and $p \Rightarrow q$ is translated in English as "if p then q" [5]. Consider an association rule (written *Body* $\Rightarrow$ *Head*) having a hypothesis (*Body*) with a large number of potential elements and a conclusion (*Head*) narrowly specific. The broadly stated hypothesis expands the scope of items subject to the hypothesis, and the specific conclusion provides exacting information about the subjects. For a set of rules corroborated by the same tuples, rules having the most general expression in the *Body* and the most specific expression in the *Head* are robust rules.

To construct a robust rule from a set of rules covering a set of corroborating tuples, the most general expression appearing on the left sideof a rule is the body of the robust rule. The most specific expression appearing on the right side of a rule is the head of the robust rule. Given the example of traditional rules in Figure 5, the most general expression is {0213*} from rules 2 and 4. The most specific expression is rule 1, {02139, 1996/08/**}. So, the resulting robust rule is {0213*} $\Rightarrow$ {02139, 1996/08/**}, as shown in Figure 5.

More than one robust rule may exist for the same set of tuples; If the most general expression and the most specific expression are the same, then the robust rule is exactly one of the derived rules. By our definition, the two can never cross, thereby guaranteeing that there always exists a robust rule. If we had defined robust rules to be the opposite – i.e., the most specific expression appearing as a body in a derived rule becomes the body of the construction and the most general rule becomess the head of the construction, then the resulting construction would either be a redundant rule or uncorroborated. For example, of the rules in Figure 5, the oppositely defined construction would be the same as rule 3.

2.2 Attribute Hierarchies and Semantics

Recently, Sweeney (2002) introduced more-general-than attribute hierarchies as a basis for generalizing data to achieve semantically useful, yet provably anonymous data [12]. In the work presented herein, we similarly exploit the semantics of predefined more-general-than attribute hierarchies not only to improve human interpretation of results, but also to enrich rule representations.

2.3 Association Rule Learning

As discussed in Section 1, Agrawal et al. (1993) introduced the notion of learning single concept associations, expressed as rules, from base values in transactional data [1]. Srikant and Agrawal (1995) added a pre-defined "is-a" concept hierarchy over all the items in a transactional database to learn generalized association rules [10]. Learned rules involved terms at cross-levels of the hierarchy. Somewhat simultaneously, Han and Fu (1995) also introduced the use of a pre-defined "is-a" hierarchy to learn cross-level rules from transactional data, but the characterization was not one of learning more general rules by involving terms appearing increasingly higher in the hierarchy as was done in [10], but of "drilling down" to learn more specific rules by involving terms appearing increasingly lower in the hierarchy [2]. Srikant and Agrawal (1996) mined association rules from quantitative data in relational tables, thereby providing an early account of multi-dimensional rules [11]. Quantitative values appearing in the table were partitioned into intervals whose ranges were determined from the distribution of values appearing in the table. Learned rules were expressed in terms of base values and intervals. However, different tables having the same attributes may have rules expressed in different intervals because intervals are not pre-defined. This can confound rule comparison across tables.

In the work presented herein, we associate pre-defined "more-greater-than" hierarchies with each attribute, categorical or quantitative, in a large relational table. We then provide an efficient means to learn *robust rules* from the table using the hierarchies. Recognizing the large quantity of rules that can be learned, we provide a semantic measure, a "knowledge rating", for determining the most useful rules.

3. Methods

Prior to presenting methods to learn robust rules efficiently, we precisely define our terms and introduce the notion of a generalization tree.

3.1 Technical Details of Attribute Hierarchies

Given an attribute A of a table $\mathcal{D}$, we define a **domain generalization hierarchy** DGH_A for A as a set of functions $f_h : h=0,\ldots,k\text{-}1$ such that:

$$A_o \xrightarrow{f_0} A_1 \xrightarrow{f_1} \ldots \xrightarrow{f_{k-1}} A_k$$

$A=A_0$ and $|A_k| = 1$. DGH_A is over: $\bigcup_{h=0}^{k} A_h$

Clearly, the f_h's impose a linear ordering on the A_h's where the minimal element is the base domain A_0 and the maximal element is A_k. The singleton requirement on A_k ensures that all values associated with an attribute can eventually be generalized to a single value. Because generalized values are used in place of more specific ones, it is important that all domains in the hierarchy be semantically compatible.

Given a domain generalization hierarchy DGH_A for an attribute A, if $v_i \in A_i$ and $v_j \in A_j$ then we say $v_i \leq v_j$ if and only if $i \leq j$ and:

$$f_{j-1}(\ldots f_i(v_i)\ldots) = v_j$$

This defines a *partial ordering* $\leq$ on: $\bigcup_{h=0}^{k} A_h$

Such a relationship implies the existence of a **value generalization hierarchy** VGH_A for attribute A. Figure 1 illustrates a domain and value generalization hierarchy for domain Z_0, representing ZIP codes for Cambridge, Massachusetts.

We recognize situations in which it may be advantageous to consider several different DGH's for the same attribute. However, as long as such DGH's have the same singleton as their maximal elements, we may represent all of them in one corresponding VGH. In this case, the VGH is a general Directed Acyclic Graph (DAG), instead of a simple tree structure.

The search for meaningful rules realized from a table can be efficiently organized by taking advantage of a naturally occurring structure over the rule space – a general-to-specific ordering of rules imposed by the domain generalization hierarchies.

For simplicity, let us assume only one DGH is associated with each attribute. Given table $\mathcal{D}(A_1, \ldots, A_m)$ and DGH_{A_i}'s, $1 \leq i \leq m$, the set of all possible generalizations of values in $\mathcal{D}$ comprise a **generalization hierarchy**, $\text{GH}_{\mathcal{D}} = \text{DGH}_{A1} \times \ldots \times \text{DGH}_{Am}$, assuming the Cartesian product is ordered by imposing coordinate-wise order. $\text{GH}_{\mathcal{D}}$ defines a lattice whose minimal element has values in the same domains as $\mathcal{D}$. Each node in the lattice represents a unique combination of generalization concepts for expressing rules in $\mathcal{D}$. We know that the number of nodes in the lattice is the product of sizes of the attribute hierarchies: $\prod_{i=1}^{m} |DGH_i|$.

We define a **pair** to be any two such nodes that can be connected through a set of function links ($l_1, \ldots, l_m$), where

$$l_i = \prod_{h=h1}^{h2} f_{Aih} \quad \text{or} \quad \equiv \text{(equal to self).}$$

The total number of pairs in the lattice is the number of possible *multi-dimensional generalized rules* that can be represented. (The number of possible rules is greater, being instantiated by values appearing in the table.)---What does this sentence mean? For what purpose?

Because we view learning as a search, it is natural to consider exhaustive search through the entire rule space. But even for a table having a few attributes, each with short hierarchies, these equations show exhaustive search prohibitive. So, we are interested in efficient algorithms that search very larges rule spaces for *robust rules*.

3.2 Notations in DAG

Because the concept of a Directed Acyclic Graph (DAG) is heavily involved in our methodology, we clarify the basic notations we use. We use lower case letters, such as *x*, to denote *nodes* (or vertices) in a DAG. We say "*x* is connected to *y*" or "there is a connection between *x* and *y*", if there is an edge between *x* and *y*, regardless of direction. We call *p* a *parent* of *c* (or *c* a *child* of *p*), if there is an edge starting at *p* and ending at *c*. We call *p'* an ancestor of *c* (or *c* a descendant of *p'*), if there is an edge from *p'* to *c* in the transitive-closure of the corresponding DAG.

3.3 Rule Definition

A *generalized association rule* in transaction databases is defined in [2]. We basically follow that definition and define the *multi-dimensional generalized association rule* in relational databases.

Let $\mathcal{A} = \{A_1, \ldots, A_m\}$ be a set of attributes in a relational dataset $\mathcal{D}$, where each A_i has an associated hierarchy, the VGH of which is a DAG. A ***multi-dimensional generalized association rule*** is an implication of the form $X \Rightarrow Y$, where $X = \{v_1^x, v_2^x, \ldots, v_m^x\}$, $Y = \{v_1^y, v_2^y, \ldots, v_m^y\}$, and v_i^x, v_i^y are values in A_i's hierarchy. It is required that no v_i^y be an ancestor of v_i^x, i.e. $v_i^y \leq v_i^x$, for all $1 \leq i \leq m$. We call *X* the Body of a rule and *Y* the Head of a rule. We follow the traditional definition of confidence and support for association rules. *Confidence* equals *c* if *c*% of the data tuples in $\mathcal{D}$ that support Body also support Head; *Support* equals *s* if *s*% of data tuples in $\mathcal{D}$ support both Body and Head. Given our definition, clearly, any data tuple that supports Head must also support Body, because each attribute value in Body is more-general-than (or at least equal to) the corresponding value in Head. Hence, support for a multi-dimensional generalized association rule is simply the percentage of data tuples that support Head.

Traditional association rules are special cases of our definition. For example, a traditional rule of the form $\{v_1^x\} \Rightarrow \{v_2^y\}$, is equivalent to our form of $\{v_1^x, *_2, *_3, \ldots, *_m\} \Rightarrow \{v_1^y, v_2^y, *_3, \ldots, *_m\}$, where $v_1^x = v_1^y$

and each $*_i$ represents the highest level of generalization in A_i's hierarchy (root in corresponding VGH_{Ai}). Of course, by skipping those *'s in English interpretation and in "standard forms", we convert them to the more traditional appearing "simplified forms", as desired.

The benefit of our definition is that it enriches the expressivity of rules. Consider the following example. Let $\mathcal{A}$ = {*5-digit ZIP*, *registration date* (*year/month/day*), *status*}, we may have {021**, 1996/**/**, *} $\Rightarrow$ {021**, 1996/08/**, active}, with 89.8% confidence and 4.5% support. This rule learns both in "width", i.e., from knowing nothing to learning something about the *status* attribute (e.g., "*$\Rightarrow$active"); and in "depth", i.e., from knowing something general to learning something more specific about the *registration date* attribute ("1996/**/** $\Rightarrow$ 1996/08/**"). The latter type of learning ("in depth") cannot be achieved traditionally. Recall the example we used in the Introduction (Section 1) concerning the limitation of traditional rule mining approaches. This new rule definition enables us to express semantically robust rules directly.

A *robust rule* has quantifiable measures of support and confidence as well as semantic constraints. A Body X is more generally expressive in one rule than X' in another (written as $\boldsymbol{X > X'}$), if its terms appear at the same or higher levels in attribute hierarchies and/or if it has fewer terms. Similarly, a Head Y is more specifically expressive in one rule than Y' in another (written as $\boldsymbol{Y < Y'}$) if its terms appear at the same or lower levels in attribute hierarchies and/or it has more terms. Rules having the most generally expressive Body and the most specifically expressive Head for the same corroborating tuples are the *robust rules* for those tuples. A set of tuples will have at least one *robust rule,* which comes from corresponding *multi-dimensional generalized association rules* that can be derived. It may have more than one *robust rule*.

Now we give the formal definition of *robust rule*. Given a relational dataset $\mathcal{D}$, attribute set $\mathcal{A} = \{A_1, \ldots, A_m\}$, associated hierarchies VGH_{Ai}'s and minimum support and confidence thresholds *minsup* and *minconf*, a *multi-dimensional generalized association rule* $X \Rightarrow Y$, is a ***robust rule*** if the following conditions are satisfied:

1. $|Tuples(Y)|/|\mathcal{D}| \geq minsup$ and $|Tuples(Y)|/|Tuples(X)| \geq minconf$;
2. $\nexists\, Y'$, s.t. $Tuples(Y') = Tuples(Y)$ and $Y' < Y$;
3. $\nexists\, X'$, s.t. $Tuples(X') = Tuples(X)$ and $X' > X$.

where $Tuples(A)$ denotes the complete set of tuples that can be expressed by A, and $|\tau|$ denote the size of τ.

We also propose a semantic measure, ***knowledge rating*** (***KR***), to evaluate the insightfulness of a *robust rule* $X \Rightarrow Y$:

$$KR(X \Rightarrow Y) = \sum_{i=1}^{m} LevDiff_i(X, Y),$$

where $LevDiff_i(\alpha, \beta)$ is the level difference between α and β in the i'th attribute's hierarchy. Clearly, rules that learn deeper (having a general value in Body and a more specific value in Head on the same attribute), and/or wider (having a * in Body and a specific value in Head on the same attribute), would gain a high *KR*. We believe that high *KR* rules are desirable because they convey more knowledge semantically.

3.4 GenTree Definition

Starting in this section, we explain the data structure and rule mining algorithm we developed to accomplish the task of learning *robust rule*s.

A generalization tree (***GenTree***) is a DAG, which represents the multi-dimensional generalization relations among all data tuples in a relational dataset over a set of hierarchical attributes, and satisfies the properties of *completeness* and *conciseness*.

There are two types of nodes in GenTree: leaves and non-leaves. Each ***leaf*** node represents a corresponding data tuple. Each ***non-leaf*** node represents a multi-dimensional generalization form and the set of data tuples that can be generalized to that form. ***Root*** is a special non-leaf node, which represents the overall generalization of all attributes (total suppression of all attributes) and correspondingly the entire set of all data tuples.

3.4.1 Notations in GenTree

We use:

Form(x) to denote the corresponding multi-dimensional generalization form (or expression form if x is a leaf) which x represents. An example is *Form*(x) = (ab*, 1*) in Figure 6.

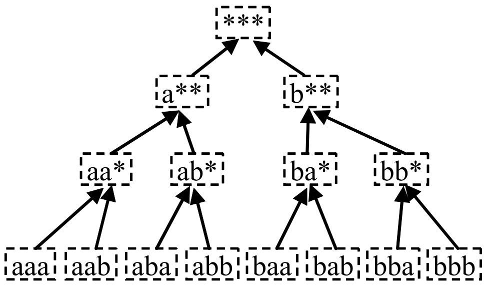
Form(x)$_i$ to denote the value of the i'th attribute in *Form*(x); and,

Tuples(x) to denote the set of tuples that can be expressed by, or can be generalized to *Form*(x).

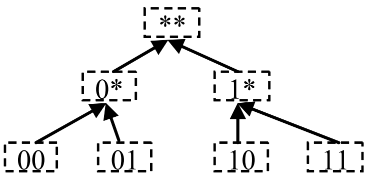
Table $\mathcal{D}$

attribute 1	attribute 2
aba	11
aab	11
abb	10
abb	11
aaa	11

Hierarchy (VGH) of an attribute 1



Hierarchy (VGH) of an attribute 2



GenTree based on $\mathcal{D}$

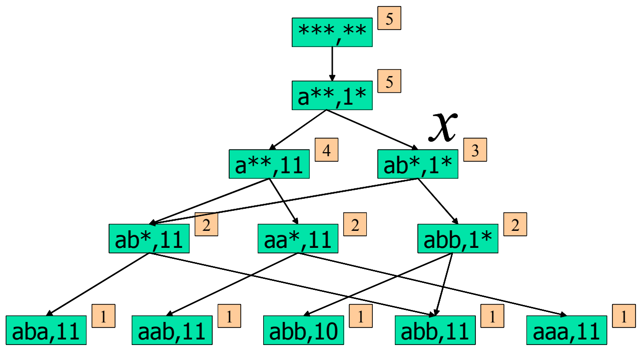


Figure 6. A GenTree using the table and VGH's shown. Tuples in the table appear as leaves. Parents are generalizations. The number associated with any *node* shows how many tuples are represented by it, i.e., |*Tuples*(*node*)|.

3.4.2 Definitions in a Generalization Tree

We define $Form(x)_i < Form(y)_i$ and $Form(y)_i > Form(x)_i$ iff. $Form(y)_i$ is "more-general-than" $Form(x)_i$ in VGH_i. That is, the path from $Form(x)_i$ to $Form(y)_i$ in VGH_i exists;

We define $Form(x) = Form(y)$ iff. $Form(x)_i = Form(y)_i$ for all $1 \le i \le m$;

We define $Form(x) < Form(y)$ (and $Form(y) > Form(x)$) iff. $Form(x)_i \le Form(y)_i$ for all $1 \le i \le m$, and $Form(x)_j < Form(y)_j$ for at least one j, $1 \le j \le m$;

We define x an ***ancestor*** of y (and y a ***descendant*** of x) if $Form(x) > Form(y)$, and also define x a ***parent*** of y (y a ***child*** of x) if x and y are directly connected. Clearly, *root* is an ancestor of all other nodes and a leaf is never an ancestor.

Given a node y, one of y's ancestor a, and a', a parent of a, the rule $GeneralForm \Rightarrow Form(y)$ is a **robust rule** iff: $Form(a) \le GeneralForm < Form(a')$ and $\nexists$ $OtherForm$, s.t. $GeneralForm < OtherForm < Form(a')$ over the same set of tuples.

3.4.3 Properties of a Generalization Tree

Completeness: Let Q be the set of all data tuple sets $Tuples(z)$'s that can be represented by each node, where $z \in$ {all nodes in a GenTree}, $\nexists$ $Form(z')$, s.t. $Tuples(z') \notin Q$.

That is, for any data tuple set that is not found in Q, you will not be able to use a generalization form based on attributes' hierarchies to represent it accurately.

Conciseness: For $\forall$ node $z \in$ {all nodes in a GenTree} (except *root*), $\nexists$ another form $Form(z')$, s.t. $Tuples(z') = Tuples(z)$ and $Form(z') < Form(z)$.

That is, each node's form is the most concise (or specific) one with regard to the tuples represented by it. *Root* is excluded because it has the most general generalization form, and merely serves as a foundation for tree construction.

3.4.4 Multiple Parents and Children

With regard to parent and child, since GenTree is a DAG, each node may have multiple parents as well as multiple children. However, it is easy to prove that a node x's possible maximal number of parents is the number of unsuppressed attributes in $Form(x)$; and the possible maximal number of children is

$$\sum_{v \in non\_base(Form(x))} Fanout(v)$$

where $non\_base(Form(x))$ is the set of $Form(x)$'s non-base attribute values (in corresponding hierarchies), $Fanout(v)$ is the number values that can be generalized to v directly (in corresponding hierarchy). For example, let's look at the node of form (aa*, 11) in Figure 6. The value "aa*" of attribute 1, is an unsuppressed and non-base value with $Fanout$("aa*") = 2 in the attribute 1's hierarchy; the value "11" of attribute 2, is an unsuppressed but base value in attribute 2's hierarchy. So the node could have at most 2 parents and 2 children, while in the actual tree it has 1 parent and 2 children, which is determined by the property of conciseness.

3.5 GenTree Construction

The basic steps of GenTree construction algorithm are shown in Figure 7. The main steps of this algorithm are step 2.6 and 2.7, which are shown in more detail in Figure 8 and 9, respectively.

In step 2.6, the algorithm searches the current tree in a top-down direction for possible new non-leaf nodes that could be created due to t. It starts by setting a cursor x at *root* and comparing x with t: 1) If t will be a descendant of x, the algorithm moves x down to each of its children; 2) If $Form(x)_i \ge Form(t)_i$ for some attribute i, it determines whether each most concise common ancestor g of x and t is already created, if not, g is created and is inserted in step 2.7, then x is moved down to its children; 3) In other situations, same actions as in 2) except that x will not move further down. Step 2.6 guarantees the properties of completeness and conciseness.

```
Algorithm GenTree Construction
Input: Table 𝒟 and a VGH for each attribute in 𝒟
Output: the root of the GenTree
1. create node root by generalizing all attributes to most general values, i.e., roots in hierarchies of all
attributes;
2. do the following until all tuples in database 𝒟 are treated:
    2.1. pick a new tuple from 𝒟;
    2.2. create leaf node t corresponding to the tuple picked;
    2.3. if (∃ a non-leaf node s, s.t. Form(s) = Form(t)) then do:
            2.3.1. connect t to s as its child;
            2.3.2. go to 2.1;
    2.4. create set To_Insert;
    2.5. add t to To_Insert;
    2.6. use the current tree to find all of the most concise new generalizations involving t and any
         eligible subset of current leaves, create corresponding non-leaf nodes and add them to
         To_Insert; See Figure for details.
    2.7. insert each node in To_Insert to current tree, as a descendant of root; See Figure for details.
    2.8. delete set To-Insert;
    2.9. go to 2.1;
3. return root
```

Figure 7. Basic Steps of GenTree Construction.

In Step 2.7, the algorithm's task is to find the proper parents and children for the new node *x*. The basic idea is starting from *root*, search downward for eligible parents of *x*. The tricky part is to find *x*'s children, details are shown in Figure 9.

3.5.1 Size of GenTree

For simplicity, consider full tree-like hierarchies. The maximum number of more-general values of a base value equals "the height of the hierarchy – 1".

For a combination of hierarchical attributes, the maximum number of cross-level ancestors of a leaf representing a base-valued tuple, is related to the product of the heights of the hierarchies:

$$(\prod_i H_i) - 1$$

where H_i is the height of hierarchy associated with attribute *i*.

An extreme upper bound of GenTree size is:

$$n \cdot \prod_i H_i$$

where *n* is the number of data tuples.

This is huge! BUT leaves have ancestors in common, and not all ancestors are necessary for a practical dataset, so actual GenTree size is much, much smaller than this upper bound!

3.6 Rule Mining Algorithm

In order to learn robust rules, we start by finding Heads. The property of conciseness of GenTree assures that the generalization form of each node (except *root*) is most explicit with regard to the tuple set it represents. The property of completeness assures that the set of node forms (except that of *root*) in GenTree may serve as a complete candidate set of Heads. Given *minsup* as a user-specified minimum support, we may prune the candidate set by removing $\forall Form(p)$, s.t. $|Tuples(p)| < |\mathcal{D}| \bullet minsup$.

For each candidate Head *Y* (*Form*(*y*) = *Y*), how do we find a good corresponding Body *X* in order to complete a rule $X \Rightarrow Y$?

```
Algorithm findNewNonLeafNodes
Input: new leaf node $t$, node $root$.
Output: set NewNonLeaves, containing non-leaf nodes representing the most concise new
generalizations involving $t$ and any eligible subset of current leaves rooted at $root$.
1. let NewNonLeaves $\leftarrow \Phi$
2. exploreNodePair($t$, $root$, NewNonLeaves)
3. return NewNonLeaves

Algorithm exploreNodePair
Input: new leaf node $t$, current node $x$, node set NewNonLeaves.
Output: void.
1. if ($x$ has been called by another exploreNodePair($t$, $x$, NewNonLeaves) then return
2. compare $Form(t)$ with $Form(x)$
  2.1. case ($Form(t) < Form(x)$):
    2.1.1. for each child $y$ of $x$ do exploreNodePair($t$, $y$, NewNonLeaves)
  2.2. case ($Form(t)_i \le Form(x)_i$ for some $i$):
    2.2.1. G $\leftarrow$ generalize($t$, $x$)
    2.2.2. for each node $g$ in G, if ($\nexists$ $h$ in current tree, s.t. $Form(h) = Form(g)$) then add $g$ to
         NewNonLeaves
    2.2.3. for each child $y$ of $x$ do exploreNodePair($t$, $y$, NewNonLeaves)
  2.3. case (else):
    2.3.1. G $\leftarrow$ generalize($t$, $x$)
    2.3.2. for each $g$ in G, if ($\nexists$ $h$ in current tree, s.t. $Form(h) = Form(g)$) then add $g$ to
         NewNonLeaves

Algorithm generalize
Input: node $x$, node $y$.
Output: node Set G, containing all of such node $g$, that is one of the most concise generalizations of $x$
and $y$, i.e., s.t. $Form(g) \ge Form(x)$, $Form(g) \ge Form(y)$, with a guarantee that $\nexists$ $g'$ s.t. $Form(g') \ge$
$Form(x)$, $Form(g') \ge Form(y)$ and $Form(g') < Form(g)$.
1. for each $i$, $1 \le i \le m$, do:
  1.1. in the hierarchy of the $i$'th attribute, find the closest "more-general" values shared by $Form(x)_i$
       and $Form(y)_i$
  1.2. add all found values in set $V_i$
2. let G $\leftarrow \Phi$
3. for each different generalization form, i.e., a combination of values chosen across $V_i$'s for all $i$, $1 \le i$
   $\le m$, create a corresponding node $g$ and add $g$ to G
4. return G
```

Figure 8. Algorithms that perform Step 2.6 in GenTree construction.

From the ancestors that are Body bases, how do we find most generalized form(s) for each of them? For each node c of them, we examine each and every parent c'. Usually, $|Tuples(c')| > |Tuples(c)|$ because of the superset-set relation, and the pair (c, c') bound the expression of a Body X such that X is most generalized, i.e. $Form(c) \le X < Form(c')$ and $\nexists\, Form(r)$ where $Form(x) < Form(r) < Form(c')$. Such a Body X can be composed by the following means: Let $\mathcal{S} = \{S, Form(c) \le S \le Form(c')\}$ be the set of all possible generalization forms between c and c', inclusive. Remove all generalizations from $\mathcal{S}$ who represents a tuple set larger than $Tuples(c)$. Remove each generalization G_1 from S if there exists a G_2 in S such that $G_2 > G_1$. Each form remaining in $\mathcal{S}$ is an expression for X.

```
Algorithm insertToTree
Input: node $x$, node $r$.
Output: void.
Assume: $Form(x) < Form(r)$, or $Form(x) = Form(r)$ only if $x$ is a leaf node and $r$ is a non-leaf node
1. if ($r$ has been called by another insertToTree($x$, $r$)) then return
2. let ConnectAsChild = true  //show whether $x$ needs to be connected to $r$ as a child
3. for each child $s$ of $r$ do:
    3.1. compare $Form(x)$ with $Form(s)$
    3.1.1. case (($Form(x) < Form(s)$) or ($Form(x) = Form(s)$, $x$ is a leaf and $s$ is a non-leaf)):
    3.1.1.1. insertToTree($x$, $s$)
    3.1.1.2. let ConnectAsChild = false
    3.1.2. case ($x$ is a non-leaf, $Form(x) > Form(s)$):
    3.1.2.1. insertBetween($s$, $x$, $r$)
    3.1.2.2. let ConnectAsChild = false
    3.1.3. case ($x$ and $s$ are a non-leaves, $Form(x)_i \le Form(s)_i$ or $Form(x)_i \ge Form(s)_i$ for each $i$, $1 \le i \le m$):
    3.1.3.1. addPossibleSubTree($s$, $x$)
4. if (ConnectAsChild = true) then connect $x$ to $r$ as its child

Algorithm insertBetween
Input: node $s$, node $x$, node $r$.
Output: void.
1. connect $x$ to $r$ as its child
2. if ($s$ is not a child of $x$ yet) then connect $s$ to $x$ as its child
3. disconnect $s$, a former child, from $r$

Algorithm addPossibleSubTree
Input: node $s$, node $x$.
Output: void.
1. if ($s$ has been called by another addPossibleSubTree($s$, $x$)) then return
2. for each child $u$ of $s$ do:
    2.1. if ($Form(u) < Form(x)$ and $u$ is not a child of $x$) then connect $u$ to $x$ as its child
    2.2. else if ($u$ is a non-leaf node, $Form(x)_i \le Form(u)_i$ or $Form(x)_i \ge Form(u)_i$ for each $i$, $1 \le i \le m$)
        then addPossibleSubTree($u$, $x$)
```

Figure 9. Algorithms that perform step 2.7 in GenTree construction.

The above approach is a conceptual prototype algorithm for finding robust rules in a GenTree. Figure 10 provides a more efficient modified algorithm. Starting at each child of *root*, we traverse the GenTree in a top-down direction and treat the generalization form of each node a candidate of Head Y ($Form(y) = Y$). If the number of tuples which a node represents does not satisfy *minsup*, none of its descendants will be visited. For each candidate node y, all eligible ancestor nodes (with regard to *minconf*) will be visited as X's bases. We create actual X's by further generalizing them to most generalized forms that represent the same set of tuples.

For some GenTree, there is a special situation: *root* has only one child q. In this case, $Tuples(q) = Tuples(root)$, representing all data tuples. A rule "$Form(root) \Rightarrow Form(q)$ with 100% support and 100% confidence" must be true. This means the most specific generalization form for all tuples in the corresponding dataset is $Form(q)$, i.e., the value of the i'th attribute for each tuple can be generalized to $Form(q)_i$, for all i. So, if any rule's Body includes $Form(q)_i$, then in order to make it most general, a special operation is needed: replace $Form(q)_i$ with $Form(root)_i$, i.e., the i'th attribute should be suppressed in Body. See Figure 11 for a rule mining example.

Theorem1: Given table $\mathcal{D}$, attribute set $\mathcal{A} = \{A_1, \ldots, A_m\}$, associated hierarchies and minimum support and confidence thresholds *minsup* and *minconf*, a rule generated by the GenTree algorithm is a robust rule.

Proof sketch: Condition 1 is satisfied as stated above; the property of conciseness guarantees condition 2 and the algorithm "generalize" in Figure 7 ensures that the GenTree constructed is consistent with the property of conciseness; condition 3 is satisfied by algorithm "findGoodPairs" in Figure 9.

Theorem2: The set of rules generated by the GenTree algorithm is complete with regard to *minsup* and *minconf*.

Proof sketch: The theorem is guaranteed by the property of completeness, the algorithm "exploreNodePair" in Figure 7 ensures that the GenTree constructed is consistent with the property of completeness and the rule mining algorithm ensures that all possible Body-Head combinations are considered.

```
Algorithm mineRobustRules
Input: node root, rule parameters minsup and minconf.
Output: association rule set R, which satisfies minsup and minconf.
1. let R ← Φ
2. for each child s of root do findRulesFor(s, minsup, minconf, R)
3. return R

Algorithm findRulesFor
Input: node s, rule parameters minsup and minconf, rule set R.
Output: void.
1. if (s has been called by another findRulesFor(s, minsup, minconf, R)) then return
2. if (|Tuples(s)| does not satisfy minsup) then return
3. findGoodPairs(s, s, minconf, R)  //rules with 100% confidence also considered
4. for each child q of s do findRulesFor(q, minsup, minconf, R)

Algorithm findGoodPairs
Input: node s, node s', rule parameter minconf, rule set R.
Output: void.
Assume: $Form(s') \geq Form(s)$
1. if (s and s' has been called by another findGoodPairs(s, s', minconf, R)) then return
2. if (|Tuples(s)|/|Tuples(s')| does not satisfy minconf) then return
3. let NewRules ← Φ
4. for each parent p of s' do:
    3.1. for each pair of attribute values ($Form(p)_i$, $Form(s')_i$) do:
        3.1.1. if ($Form(p)_i > Form(s')_i$) then for each value v which can be generalized directly to
                $Form(p)_i$ and $v \geq Form(s')_i$ in the hierarchy of the i'th attribute do:
            3.1.1.1. let FORM = Form(p)
            3.1.1.2. replace $FORM_i$ with v
            3.1.1.3. let new_rule be "$FORM \Rightarrow Form(s)$" with support = $|Tuples(s)| / |\mathcal{D}|$ and confidence =
                    $|Tuples(s)|/|Tuples(s')|$
            3.1.1.4. if (new_rule $\notin$ NewRules) then add new_rule to NewRules
4. add all rules contained in NewRules to R
5. for each parent p of s' do findGoodPairs(s, p, minconf, R)
```

Figure 10. Algorithms for mining robust rules from a GenTree.

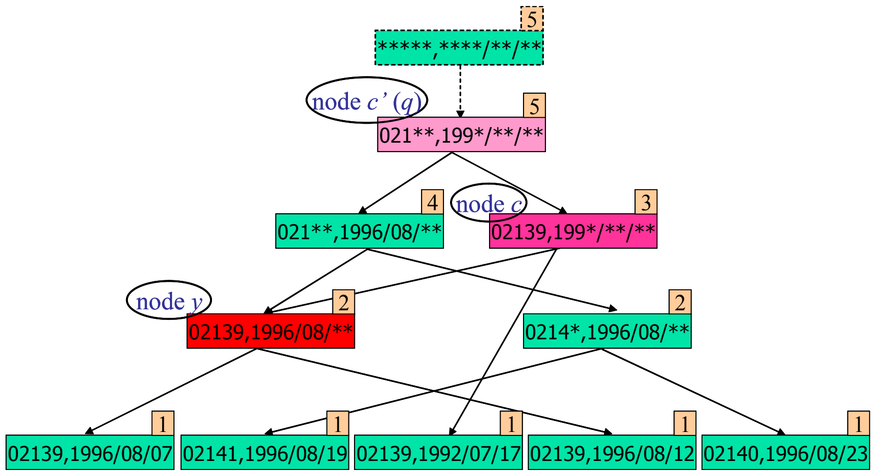


Figure 11. Rule mining example: Head Y = Form(*y*), a possible Body X is bounded by *c* and *c'*. Under usual GenTree mining conditions, Body X = (0213*, 199*//**) based on *c* and *c'*. However, in the above GenTree, *c'* is the only child *q* of root, which is a special GenTree mining case in which we replace "199*/**/**" with "****/**/**" (supression). So finally, Body X = (0213*, ****/**/**), and the corresponding robust rule is (0213*, ****/**/**) $\Rightarrow$ (02139, 1996/08/**).**

4. Experimental Results

In this work, we have: (1) added more rigor to association rule mining and, (2) introduced a knowledge rating to identify the semantic utility of rules. We achieved these goals by extending the language in which rules can be expressed, and by providing a GenTree data structure and associated algorithm to learn robust rules in the extended language using attribute hierarchies, parameter settings, and data tuples.

In this section, we demonstrate the performance of GenTree using two real-world voter lists[3]. Section 4.1 reports on the findings using voter information from Pittsburgh, Pennsylvania, and Section 4.2 reports on findings using voter information from Cambridge, Massachusetts. Learning robust rules from two different datasets allowed us to compare the growth and behavior of GenTree (see Section 4.3). Using the Cambridge voter list, we compared the learning of robust rules to the traditional approach to determine the number and nature of additional rules learned (see Section 4.4). Finally, in Section 4.5, we report on the utility of rule recommendations based on combinations of statistical ratings (support and confidence) and our knowledge rating. Here are the details of our findings.

4.1 Voter List for Pittsburgh, Pennsylvania

We conducted an experiment on dataset $\mathcal{D}_1$, the 2001 voter list for ZIP 15213 in Pittsburgh, Pennsylvania having a total of 4,316 records (or tuples) [14]. Each tuple had 8 attributes, i.e. {*sex*, *birthdate* (*year*/*month*/*day*), *registration_date* (*year*/*month*), *party*, *ethnicity*, *income*, *home_owner*, *havechild*}. With rule parameters *minsup* = 2% and *minconf* = 50%, 8160 robust rules were learned after applying the GenTree algorithm. The following are rule examples:

[3] We elected to use voter lists to demonstrate our work primarily for two reasons. First, they are publicly available (subject to costs and local data sharing agreements) – allowing our findings to be replicated. And second, the data elements concern basic demographics – making values easily understood to readers and results directly generalizable to other datasets.

{sex, birthdate, registration_date, party_code, ethnicity, income, home_owner, havechild}

{*, 196*/**/**, 198*/**, D, *, *, *, *} ⇒ {F, 196*/**/**, 198*/**, D, *, *, *, *}
"Democrats born in 1960's and registered in 1980's tend to be Female."
[Support: 2.2%, Confidence: 52.2%, KR: 1]

{*, ****/**/**, 19**/**, R, W, *, D, *} ⇒ {F, 19**/**/**, 19**/**, R, W, *, D, F}
"Republican Whites owning home tend to be females with no children."
[Support: 2.1%, Confidence: 55.4%, KR: 3]

Of the 8160 robust rules learned, 167 of them learned information in "depth". The ratio is not high because only 2 attributes, i.e. *birthdate* and *registration_date*, have multi-level hierarchies; the remaining 6 attributes are categorical, i.e. 2-level hierarchies with base values as the lower level tier and a total suppression as the higher level tier.

Figure 12 shows the growth of GenTree Size for $\mathcal{D}_1$. The Upper Bound estimates a maximum growth rate of 1920 nodes/tuple, while the GenTree actually grows at about 17.8 nodes/tuple. It is clear that the actual GenTree size is much, much smaller than the "extreme upper bound" and grows linearly with the number of tuples.

4.2 Voter List for Cambridge, Massachusetts

We conducted an experiment on the 1997 voter list for Cambridge, Massachusetts [13]. We randomly sampled 10,000 records as dataset $\mathcal{D}_2$, from the original 54,805 records. Each record had 6 attributes, i.e. {*ZIP* (*9 digits*), *sex*, *birthdate* (*year*/*month*/*day*), *registration_date* (*year*/*month*/*day*), *party*, *status*}. With rule parameters *minsup* = 2% and *minconf* = 50%, 4140 robust rules were learned after applying the GenTree algorithm. Figure 14 lists a sample of the rules that were learned.

Of the 4140 robust rules learned, 1117 of them learned information in "depth". Only 3 attributes in $\mathcal{D}_2$, i.e. *5-digit ZIP*, *birthdate* and *registration_date*, are associated with multi-level hierarchies.

Figure 13 shows the growth of GenTree Size for $\mathcal{D}_2$. The Upper Bound gives a slope of 1440 nodes/tuple, but the linear regression fitting line of actual growth only shows 17.0 nodes/tuple slope. As was the case with $\mathcal{D}_1$, the plot indicates that the actual GenTree size is much, much smaller than the "extreme upper bound" and grows linearly with the number of tuples.

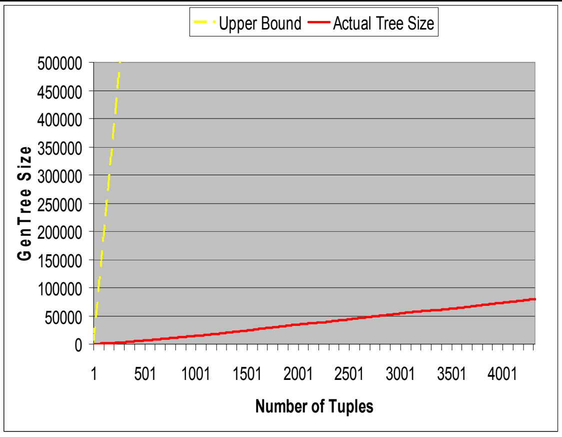


Figure 12. GenTree Size versus Number of Tuples for dataset $\mathcal{D}_1$ "Voter List for Pittsburgh 15213". Upper Bound line slope = 1920; Actual Tree Size's linear regression fitting line slope = 17.8, R^2 = 0.9932

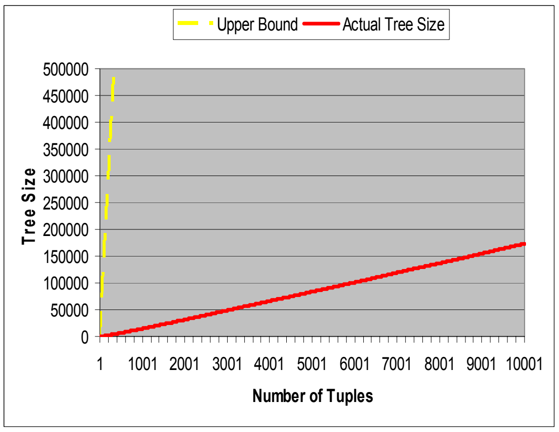


Figure 13. GenTree Size versus Number of Tuples for dataset $\mathcal{D}_2$ "Cambridge Voter List, 1997". Upper Bound line slope = 1440; Actual Tree Size's linear regression fitting line slope = 17.0, R^2 = 0.9987

4.3 Robust Rule Computation

Results from Section 4.1 and Section 4.2 are summarized in Figure 15. The number of rules learned through $\mathcal{D}_1$ (Pittsburgh data) is almost twice that of $\mathcal{D}_2$ (Cambridge data), although $|\mathcal{D}_1|$ is less than half of $|\mathcal{D}_2|$. This is not surprising because $\mathcal{D}_1$ has 8 attributes while $\mathcal{D}_2$ has 6, and the number of possible rules is exponential in the number of attributes, as discussed in Section 3.5.1. The ratio of rules that learned information in depth is much higher for $\mathcal{D}_2$ (1117/4140=27%) than for $\mathcal{D}_1$ (167/8160=2%). The reason is, $\mathcal{D}_2$ has 3 attributes with multi-level hierarchies (*ZIP*, *birthdate*, *registration_date*), but $\mathcal{D}_1$ has only 2 (*birthdate*, *regitration_date*). A larger number of rules would learn in depth if more attributes had hierarchies with levels greater than 2. The experiments with both datasets demonstrated that the actual tree sizes were much, much smaller than the "extreme upper bound," see Figure 12 and Figure 13.

	ZIP	Party	Sex	Birthdate	Regdate	Status	Tends	Zipcode	Party	Sex	Birthdate	Regdate	Status	Support	Confidence	KR
1	*********	*	*	****/**/**	****/**/**	*	==>	021******	*	*	19**/**/**	19**/**/**	*	99.86%	99.86%	3
2	*********	*	F	19**/**/**	19**/**/**	A	==>	021******	D	F	19**/**/**	19**/**/**	A	29.28%	65.42%	2
3	*********	*	M	19**/**/**	19**/**/**	A	==>	021******	D	M	19**/**/**	19**/**/**	A	20.53%	56.62%	2
4	*********	*	*	****/**/**	****/**/**	*	==>	021******	D	*	19**/**/**	19**/**/**	A	52.47%	52.47%	5
5	*********	*	F	19**/**/**	197*/**/**	*	==>	0213*****	*	F	19**/**/**	197*/**/**	A	2.19%	65.37%	3
6	*********	*	F	197*/**/**	1996/**/*	*	==>	021******	*	F	197*/**/**	1996/**/**	A	3.27%	99.70%	2

1	"All voters (tend to) live in Cambridge (021**), were born in the 1900's (19**/**/**) and registered to vote in the 1900's (19**/**/**)." *Commentary: The voter list for Cambridge, Massachusetts contains people who live in Cambridge and were registered to vote in the 1900's. A few people were born in the 1800's, but as shown by the rule above, almost all voters were born in the 1900's.*
2 3 4	"Female voters born in the 1900's, registered in the 1900's, and who are active voters tend to live in Cambridge and be Democrats." "Male voters born in the 1900's, registered in the 1900's, and who are active voters tend to live in Cambridge and be Democrats." "Voters tend to live in Cambridge, be Democrats, been born in the 1900's, registered in the 1900's, and are active voters." *Commentary: About half the voters in the Cambridge voter list are registered Democrats; the other half of the voters have no party affiliation or are registered as Republicans. This is confirmed by the support for rules 2 and 3 which together total 49.81%. In comparison, rule 4 alone, which has one of the largest knowledge rating (KR) values awarded to the dataset, states this concept succinctly.*
5	"Female voters born in the 1900's and registered in the 1970's tend to live near MIT and Harvard (0213**) and be active voters." *Commentary: The women's community in Cambridge is legendary, having many area firsts located around the Universities – a women's bookstore, a national women's newspaper, women-oriented community centers, and numerous women-centered programs. Many of these began in the 1970's with the "Women's Movement" and over time these accomplishments have been fading away. But as described by the rule, those women who arrived in Cambridge at that time and who have remained in Cambridge tend to continue to be active voters.*
6	"Female voters born in the 1970's and registered in 1996 tend to live in Cambridge and be active voters." *Commentary: These data are drawn from the 1997 voter list. The women who are the subject of this rule are therefore in their 20's (and most likely students). 1996 was a presidential election year, so the rule gives evidence that political awareness remains strong among women in Cambridge.*

Figure 14. Sample from 4140 robust rules learned from Cambridge, Massachusetts voter data. The topmost table shows the rules as processed by GenTree. The shaded values may be ignored in the English translation of the rule. The bottom table includes English translations and commentaries.

	Pittsburgh ($\mathcal{D}_1$)	Cambridge ($\mathcal{D}_2$)
Number of tuples	4316	10,000
Number of attributes	8	6
Number of hierarchies having height > 2	2	3
Number of robust rules learned	8160	4140
Number of rules learning in depth	167	1117

Figure 15. Summary results of applying GenTree to two voter lists.

4.4 Comparison of Robust Rule Learning to Fixed-Level Mining

Because mining for *robust rules* across a relational table having multiple attributes is novel, we are unable to find other currently existing approaches against which to compare. However, theoretical comparisons between our approach and an adapted traditional *a priori* algorithm can be done. One way to adapt the traditional approach is to replace base attribute values with generalized equivalents and then have the traditional learner work on the generalized values. We term this "fixed-level" mining. In this section, we report on an experiment that compares fixed-level mining to GenTree in order to determine the number and nature of additional rules learned by GenTree over traditional approaches.

We experimented on $\mathcal{D}_2$, the Cambridge voter data, based on different settings for attributes with multi-level hierarchies. The 3 attributes used were {*ZIP*, *birthdate*, *registration_date*}. We then (1) executed GenTree to learn robust rules; and, performed fixed-level mining on data having: (2) 5-digit ZIP, year of birth, year of registration; (3) 3-digit ZIP, year of birth, year of registration; (4) 5-digit ZIP, decade of birth, year of registration; (5) 5-digit ZIP, year of birth, decade of registration; (6) 3-digit ZIP, decade of birth, year of registration; (7) 3-digit ZIP, year of birth, decade of registration; (8) 5-digit ZIP, decade of birth, decade of registration; and, (9) 3-digit ZIP, decade of birth, decade of registration. Figure 16 provides the comparison of the number of rules that can be learned based on each of these settings. We are not surprised to see that the number of rules learned from any fixed-level setting is far fewer than the number of robust rules learned from GenTree.

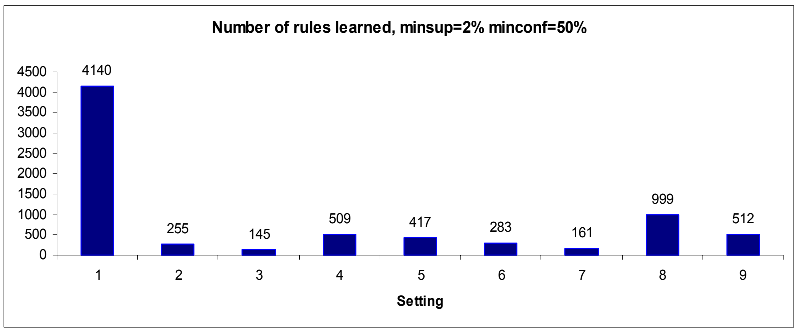


Figure 16. Number of rules from GenTree versus fixed-level mining, having minimum support of 2% and minimum confidence of 50% with attributes {*ZIP*, *birthdate*, *registration_date*}. Legend: (1) GenTree robust rules; (2) 5-digit, year, year; (3) 3-digit, year, year; (4) 5-digit, decade, year; (5) 5-digit, year, decade; (6) 3-digit, decade, year; (7) 3-digit, year, decade; (8) 5-digit, decade, decade; (9) 3-digit, decade, decade.

In order to analyze the distribution of rules learned by GenTree and by fixed-level mining, we projected learned rules on the support-confidence plane for each setting (see Figure 17). We identified those robust rules learned by GenTree in (1) which were also learned by fixed-level mining using settings (2) through (9), respecting data corroboration, i.e. making sure rules considered identical were not only the same in form but also had exactly the same set of supporting data tuples for Body and Head respectively. There were 2931 robust rules learned that were unmatched by any rules learned in any of the settings of fixed-level mining (see Figure 18). Notice the large numbers of high confidence and of low support rules learned only by our GenTree method.

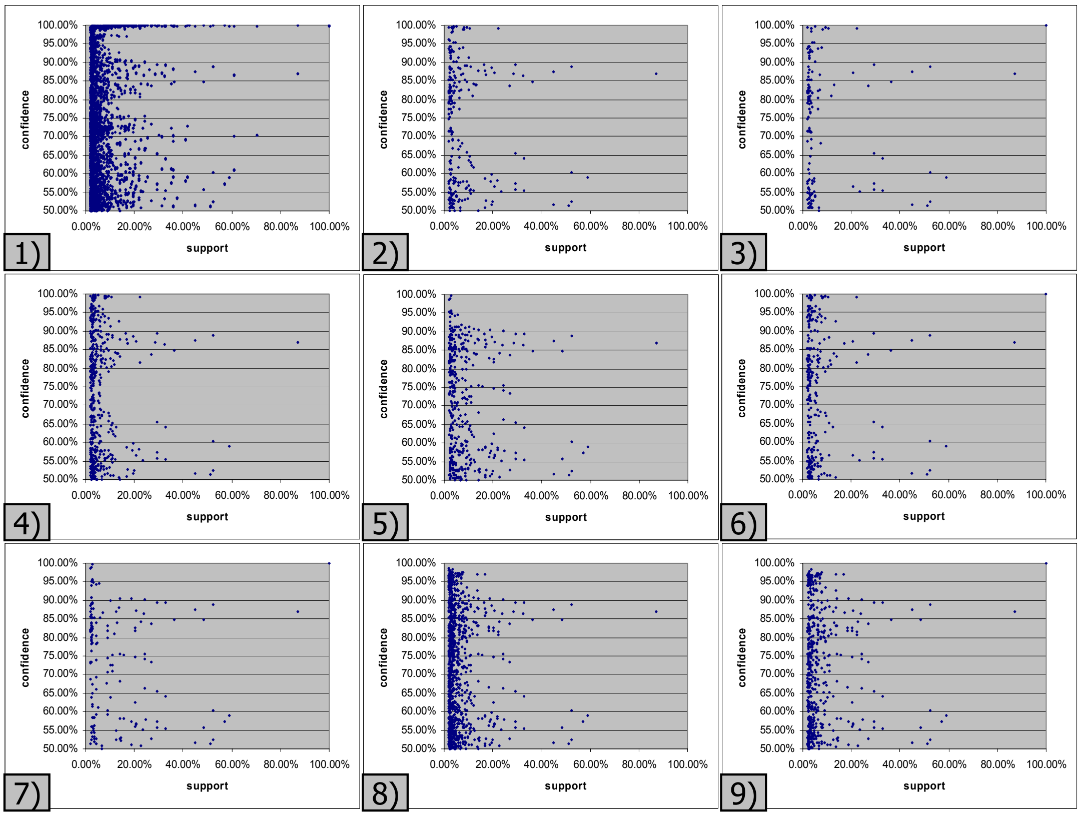


Figure 17. Rules projected on support-confidence planes for settings (1) through (9) described in Figure 16.

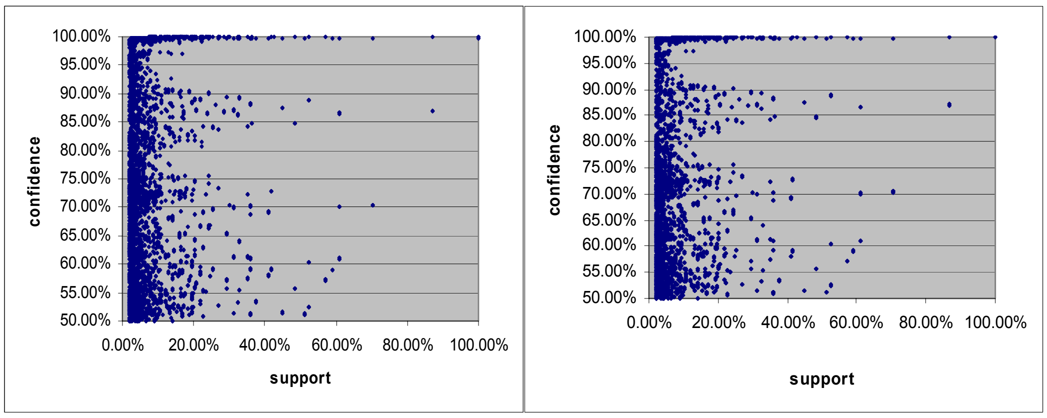


Figure 18. Support-confidence plan for 4140 robust rules (on left) and with rules that were also learned by fixed-level mining removed (on right). There are 2931 remaining rules shown on the right.

In summary, the total number of rules generated by the fixed-level settings is (255 + 145 + … + 512) = 3281, but in terms of data corroboration they only account for (4140 – 2931) = 1209 robust rules. This is due to the limited expressivity of traditional rule forms, as was discussed earlier in Sections 1 and 2.

4.5 Comparing Statistical Ratings and Knowledge Ratings

We experimented on the 4140 robust rules learned by GenTree from $\mathcal{D}_2$, the Cambridge voter data, to identify the nature of rules "recommended" by statistical and knowledge ratings. The notion of "recommending a rule" relates to the ability of numerical measures to draw the attention of human decision-makers to rules believed to be most interesting.

Figure 19 shows the distributions of knowledge and statistical ratings over the 4140 rules. The maximum knowledge rating awarded was 5, the minimum was 1 with an average of 1.8 and a standard deviation of 0.8. Only 3 rules received the rating of 5. These appear in Figure 21 and will be discussed subsequently. As for statistical measures, there is rapid decay in support ratings as the probability increases. Only 10 rules had support of 70% or better. (These appear in Figure 20.) On the other hand, more than 700 rules had a confidence rating of 90% or better. Only one rule had a confidence and support of 100%. It appears in Figure 20.

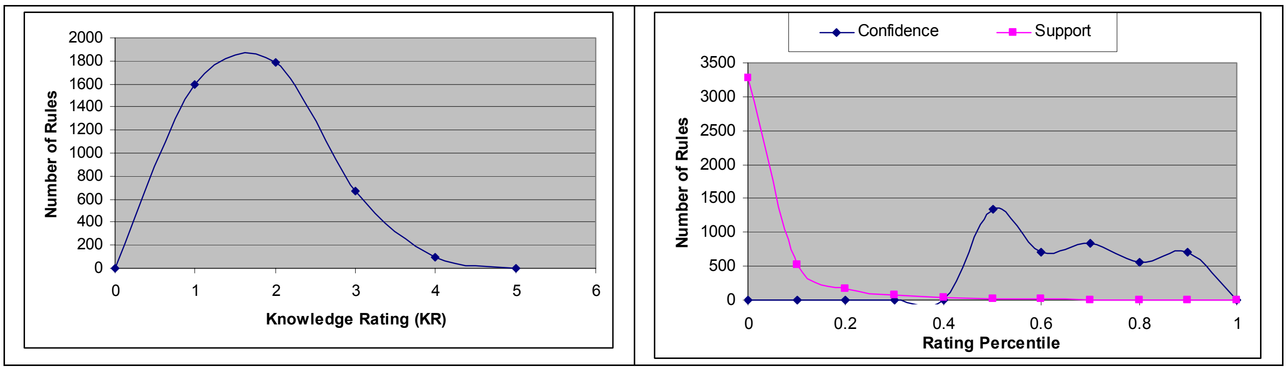


Figure 19. Distributions of knowledge rating (on left) and confidence and support ratings (on right) for the 4140 robust rules learned from the Cambridge Voter list ($\mathcal{D}_2$).

Support - Confidence Top 10 Rules

	ZIP	PARTY	SEX	BIRTHDATE	REG_DATE	STATUS	***tends***	ZIP	PARTY	SEX	BIRTHDATE	REG_DATE	STATUS	support	confidence	KR
	(9-digit)	(code1)	(M/F)	(yyyy/mm/dd)	(yyyy/mm/dd)	(A/I)		(9-digit)	(code1)	(M/F)	(yyyy/mm/dd)	(yyyy/mm/dd)	(A/I)			
1	*********	*	*	****/**/**	****/**/**	*	==>	021******	*	*	****/**/**	19**/**/**	*	100.00%	100.00%	2
2	*********	*	*	****/**/**	****/**/**	*	==>	021******	*	*	19**/**/**	19**/**/**	*	99.86%	99.86%	3
3	*********	*	*	****/**/**	****/**/**	*	==>	021******	*	*	****/**/**	19**/**/**	A	87.04%	87.04%	3
4	*********	*	*	****/**/**	19**/**/**	A	==>	021******	*	*	19**/**/**	19**/**/**	A	86.94%	99.89%	2
5	*********	*	*	19**/**/**	19**/**/**	*	==>	021******	*	*	19**/**/**	19**/**/**	A	86.94%	87.06%	2
6	*********	*	*	****/**/**	****/**/**	*	==>	021******	*	*	19**/**/**	19**/**/**	A	86.94%	86.94%	4
7	*********	*	*	****/**/**	****/**/**	*	==>	0213*****	*	*	****/**/**	19**/**/**	*	70.46%	70.46%	3
8	0213*****	*	*	****/**/**	19**/**/**	*	==>	0213*****	*	*	19**/**/**	19**/**/**	*	70.33%	99.82%	1
9	*********	*	*	19**/**/**	19**/**/**	*	==>	0213*****	*	*	19**/**/**	19**/**/**	*	70.33%	70.43%	2
10	*********	*	*	****/**/**	****/**/**	*	==>	0213*****	*	*	19**/**/**	19**/**/**	*	70.33%	70.33%	4

Confidence-Support Top 10 Rules

	ZIP	PARTY	SEX	BIRTHDATE	REG_DATE	STATUS	***tends***	ZIP	PARTY	SEX	BIRTHDATE	REG_DATE	STATUS	support	confidence	KR
	(9-digit)	(code1)	(M/F)	(yyyy/mm/dd)	(yyyy/mm/dd)	(A/I)		(9-digit)	(code1)	(M/F)	(yyyy/mm/dd)	(yyyy/mm/dd)	(A/I)			
1	*********	*	*	****/**/**	****/**/**	*	==>	021******	*	*	****/**/**	19**/**/**	*	100.00%	100.00%	2
2	0214*****	*	*	****/**/**	19**/**/**	*	==>	0214*****	*	*	19**/**/**	19**/**/**	*	29.53%	99.97%	1
3	*********	*	*	****/**/**	199*/**/**	A	==>	021******	*	*	19**/**/**	199*/**/**	A	48.44%	99.96%	2
4	*********	*	M	****/**/**	199*/**/**	*	==>	021******	*	M	19**/**/**	199*/**/**	*	24.78%	99.96%	2
5	*********	U	*	****/**/**	199*/**/**	*	==>	021******	U	*	19**/**/**	199*/**/**	*	24.36%	99.96%	2
6	*********	D	*	****/**/**	199*/**/**	A	==>	021******	D	*	19**/**/**	199*/**/**	A	23.72%	99.96%	2
7	*********	*	F	****/**/**	199*/**/**	A	==>	021******	*	F	19**/**/**	199*/**/**	A	22.74%	99.96%	2
8	*********	*	*	****/**/**	199*/**/**	*	==>	021******	*	*	19**/**/**	199*/**/**	*	57.19%	99.95%	2
9	*********	*	M	****/**/**	199*/**/**	A	==>	021******	*	M	19**/**/**	199*/**/**	A	20.29%	99.95%	2
10	*********	U	*	****/**/**	199*/**/**	A	==>	021******	U	*	19**/**/**	199*/**/**	A	20.12%	99.95%	2

Figure 20. The top 10 of the 4140 rules learned from the Cambridge Voter list ($\mathcal{D}_2$) based on traditional statistical ratings, sorted by support then confidence (top) and by confidence then support (bottom). Shaded values may be ignored in the English translation of the rule.

Figure 20 shows the top 10 rules recommended using traditional statistical measures. The topmost table in Figure 20 shows the first 10 rules after sorting all 4140 rules by support then confidence.

Because support is the probability of the head appearing, rules with a high support rating are descriptive of the overall dataset. The first 6 rules in Figure 20 are general descriptions of the population – these rules describe voters as living in Cambridge, being born in the 1900's and having registered in the 1900's. Notice that Rule 6 succinctly addresses these features for active voters and Rule 10 does so regardless of recent voting activity. Of the top 10 support rules, only these rules have a high KR rating, and they were the most concise at summarizing the key concepts addressed by all the topmost support rules.

The 10 rules having the most confidence appear in the bottom table of Figure 20. These rules result from sorting all 4140 rules by confidence then by support. All the rules (except the first) address the concept that voters who registered in the 1900's, live in Cambridge and were born in the 1900's. While confidence is high, the gained knowledge is not very interesting. Notice that none of these rules have a high KR rating. The first rule in each of the tables in Figure 20 is the same. With 100% confidence and 100% support, all the voters in the voting list live in Cambridge and registered in 1900's.

KR-Confidence Top 10 Rules

	ZIP (9-digit)	PARTY (code1)	SEX (M/F)	BIRTHDATE (yyyy/mm/dd)	REG_DATE (yyyy/mm/dd)	STATUS (A/I)	***tends***	ZIP (9-digit)	PARTY (code1)	SEX (M/F)	BIRTHDATE (yyyy/mm/dd)	REG_DATE (yyyy/mm/dd)	STATUS (A/I)	support	confidence	KR
1	*********	*	*	****/**/**	****/**/**	*	==>	0213*****	*	*	19**/**/**	19**/**/**	A	60.92%	60.92%	5
2	*********	*	*	****/**/**	****/**/**	*	==>	021******	D	*	19**/**/**	19**/**/**	A	52.47%	52.47%	5
3	*********	*	*	****/**/**	197*/**/**	*	==>	0213*****	D	*	19**/**/**	197*/**/**	A	2.92%	50.69%	5
4	*********	*	*	****/**/**	****/**/**	*	==>	021******	*	*	19**/**/**	19**/**/**	A	86.94%	86.94%	4
5	*********	*	F	****/**/**	197*/**/**	*	==>	021******	D	F	19**/**/**	197*/**/**	A	2.64%	78.57%	4
6	*********	*	F	****/**/**	1983/**/**	*	==>	021******	D	F	19**/**/**	1983/**/**	A	3.75%	77.96%	4
7	*********	*	*	****/**/**	1983/**/**	*	==>	021******	D	*	19**/**/**	1983/**/**	A	6.44%	77.50%	4
8	*********	*	*	****/**/**	197*/**/**	*	==>	021******	D	*	19**/**/**	197*/**/**	A	4.26%	73.96%	4
9	*********	*	M	****/**/**	1996/**/**	*	==>	0213*****	*	M	19**/**/**	1996/**/**	A	5.97%	73.07%	4
10	*********	*	*	****/**/**	1996/**/**	*	==>	0213*****	*	*	19**/**/**	1996/**/**	A	16.13%	72.04%	4

KR-Support Top 10 Rules

	ZIP (9-digit)	PARTY (code1)	SEX (M/F)	BIRTHDATE (yyyy/mm/dd)	REG_DATE (yyyy/mm/dd)	STATUS (A/I)	***tends***	ZIP (9-digit)	PARTY (code1)	SEX (M/F)	BIRTHDATE (yyyy/mm/dd)	REG_DATE (yyyy/mm/dd)	STATUS (A/I)	support	confidence	KR
1	*********	*	*	****/**/**	****/**/**	*	==>	0213*****	*	*	19**/**/**	19**/**/**	A	60.92%	60.92%	5
2	*********	*	*	****/**/**	****/**/**	*	==>	021******	D	*	19**/**/**	19**/**/**	A	52.47%	52.47%	5
3	*********	*	*	****/**/**	197*/**/**	*	==>	0213*****	D	*	19**/**/**	197*/**/**	A	2.92%	50.69%	5
4	*********	*	*	****/**/**	****/**/**	*	==>	021******	*	*	19**/**/**	19**/**/**	A	86.94%	86.94%	4
5	*********	*	*	****/**/**	****/**/**	*	==>	0213*****	*	*	19**/**/**	19**/**/**	*	70.33%	70.33%	4
6	*********	*	*	****/**/**	****/**/**	*	==>	0213*****	*	*	****/**/**	19**/**/**	A	61.02%	61.02%	4
7	*********	*	*	****/**/**	****/**/**	*	==>	021******	D	*	19**/**/**	19**/**/**	*	59.01%	59.01%	4
8	*********	*	*	****/**/**	****/**/**	*	==>	021******	*	*	19**/**/**	199*/**/**	*	57.19%	57.19%	4
9	*********	*	*	****/**/**	****/**/**	*	==>	021******	D	*	****/**/**	19**/**/**	A	52.53%	52.53%	4
10	*********	*	*	****/**/**	****/**/**	*	==>	021******	*	F	19**/**/**	19**/**/**	*	51.13%	51.13%	4

Figure 21. The top 10 of the 4140 rules learned from the Cambridge Voter list ($\mathcal{D}_2$) based on a combination of our knowledge rating (KR) and statistical ratings, sorted by KR then confidence (top) and by KR then support (bottom). Shaded values may be ignored in the English translation of the rule.

Figure 21 shows the top 10 rules recommended using our knowledge rating (KR). The topmost table ("KR-Confidence") in Figure 21 shows the first 10 rules resulting from sorting all 4140 rules by KR then confidence. The bottom table ("KR-Support) shows the first 10 rules after sorting by KR then support. The first 4 rules are the same in each table. The first 3 rules have the highest KR rating given to the dataset. Rule 1 describes voters living near MIT and Harvard (who were born in the 1900's, and registered in the 1900's) as being active voters. Rule 2 describes voters as being active Democrats. And, rule 3 describes voters who registered in 1970's as being active Democrats. Notice that each of these rules provides comprehensive descriptions of sub-populations[4].

In general, the rules in KR-Support describe the overall population, whereas the rules in KR-Confidence additionally describe interesting sub-populations. For example, Rule 5 in KR-Confidence describes female voters who registered in the 1970's as being active Democrats, and Rule 6 describes female voters who registered in 1983 as being active Democrats.

[4] In this case, these rules could be practically useful to a political campaign analyst.

5. Discussion

In this paper, we have continued the evolution in the expressiveness of association rule learning to its broadest application – learning semantically rated rules from a large relational table having many attributes. We used the partial ordering imposed by "more-general-than" attribute hierarchies to extend the rule space. A GenTree data structure and algorithm were introduced that organizes search through the extended rule space for robust rules, which are mix-level generalizations across all attributes based on quantifiable corroboration (support and confidence) and on semantic expressiveness (using our knowledge rating metric introduced). Using GenTree with real-world voting data, we were able to demonstrate significant increases in the number of rules learned and in the semantic merit of those rules. In conclusion, we examine future work.

5.1 Multiple Hierarchies for an Attribute

Rather than using a single VGH for an attribute, there are situations in which multiple VGH's seem natural because they capture different criteria for describing a value. Even though these descriptions can be encoded into a single hierarchy and GenTree used to mine robust rules, different rules result depending on the level at which the encoding takes place. Here is a simple example. Consider a relational table of customer demographics and purchased beverages. An "is-a" hierarchy associates both milk and juice as beverages in the top levels, and brands of juices and milks in the lower levels. But some juices and milks have the same brands. We could alternatively have the hierarchy organized first by brand and then by type of beverage. Notice how the same rules learned from these two different hierarchies may have different values for support and confidence! This happens because the ordering of characteristics within a hierarchy matters not only semantically, but also quantifiably. Replacing this imposed ordering of characteristics with multiple hierarchies (e.g., separate beverage and brand hierarchies) for the same attribute poses another new exploration in rule mining. An extension to GenTree construction seems particularly well suited to accommodate attributes having multiple hierarchies. Additional nodes representing the various forms of cross-level generalizations using the hierarchies would be added. Mining robust rules from the revised GenTree would then use the same algorithm as presented herein.

5.2 Size of a GenTree

The extreme upper bound of a GenTree is much, much larger than the size of a GenTree in practice because of the ways in which tuples combine. In the real-world, tuples tend not to represent every possible combination of values. Also, in real-world data the number of tuples tends to be much larger than the number of attributes, thereby increasing the number of tuples likely to share common values. The actual size of a GenTree is related to the fractal dimension of the data. Exploring this relationship further allows us to determine a tighter upper bound based on the fractal dimension. There will be two key components here: 1) how to estimate the fractal dimension of a dataset with attribute hierarchies in a fast yet accurate way, and 2) how to incorporate the fractal dimension in the upper bound formula.

5.3 Privacy and *k*-anonymized Data

The notion of *k*-anonymized data is simple. Given a table in which each tuple contains information about a single person and no two tuples relate to the same person, a table is *k*-anonymized if for every tuple there are at least *k*-1 other tuples in the table having the same values over the attributes considered sensitive for re-identification [12]. A *k*-anonymized table provably provides privacy protection by guaranteeing that for each tuple there are at least *k* individuals to whom the tuple could refer. The use of *k*-anonymity fits nicely into stated laws and regulations because these expressions need only to identify a population, attributes, and a value for *k*, all of which are easy to formulate in human language. Despite its popularity, however, the use of *k*-anonymized data has been limited due to a lack of methods to learn from the data. Because GenTree stores the number of tuples at each node that account for the generalized form of the node, it is easy to see that GenTree could be used to produce *k*-anonymized data. More importantly, GenTree would be the first to mine association rules from *k*-anonymized data. These are natural by-products of this work.

6. Acknowledgements

We thank Bradley Malin for comments and members of the Data Privacy Lab for a rich work environment. This work was supported in part by the Data Privacy Laboratory, the Center for Automated Learning and Discovery, and the NSF Aladdin Center, all in the School of Computer Science at Carnegie Mellon University.

References

[1] Agrawal, R. Imielinski, T. and Swami, A. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, Washington, DC, May 1993.

[2] Han, J. and Fu, Y. Discovery of multiple-level association rules from large databases. *In Proc. of the 21st Int'l Conference on Very Large Databases*, Zurich, Switzerland, September 1995.

[3] Imielinski, T and Mannila, H: A Database Perspective on Knowledge Discovery. *Communications of the ACM* 39(11), 1996, 58-64.

[4] Mitchell, T. Generalization as search. *Artificial Intelligence, 18,* 2, 1982, 203-226.

[5] Plotkin, G. A note on inductive generalization. In Meltzer & Michie (Eds.), *Machine Intelligence,* 5, 1970 (pp.153-163). Edinburgh University Press.

[6] Rosen, K. *Discrete Mathematics and Its Applications,* (pp.1-9). New York: McGraw-Hill 1991.

[7] Russell, S. and Norvig, P. Artificial Intelligence: a modern approach, (Chapter 18). Englewood Cliffs, New Jersey: Prentice Hall, 1995.

[8] Sebag, M. Delaying the choice of bias: A disjunctive version space approach. In *Proc. of the 13th Int'l Conference on Machine Learning*, San Francisco: Morgan Kaufman1996.

[9] Simon, H. and Lea, G. Problem solving and rule induction: a unified view. In Gregg (Ed.), *Knowledge and Cognition,* (pp.105-127). New Jersey: Lawrence Erlbaum Assoc 1973.

[10] Srikant, R. and Agrawal, R. Mining Generalized Association Rules. In *Proc. of the 21st Int'l Conference on Very Large Databases*, Zurich, Switzerland, September 1995.

[11] Srikant, R. and Agrawal, R. Mining quantitative association rules in large relational tables. In *Proc. of the ACM SIGMOD Conference on Mgt of Data*, Montreal,Canada, June 1996.

[12] Sweeney, L. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10 (5), 2002; 571-588.

[13] Voter List for Cambridge, Massachusetts, Registry of Voters, Massachusetts, 1997.

[14] Voter List for Pittsburgh, Pennsylvania ZIP15213, Registry of Voters, Pennsylvania, 2001.