Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

Instance-Based Clustering for Databases[1]

Matthew Merzbacher
Wesley W. Chu[2]
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90025

We present a method for automatically clustering similar attribute values in a database system spanning multiple domains. The method constructs a value abstraction hierarchy for each attribute using rules that are derived from the database instance. The rules have a confidence and popularity that combine to express the "usefulness" of the rule.

Attribute values are clustered if they are used as the premise for rules with the same consequence. By iteratively applying the algorithm, a hierarchy of clusters can be found. The algorithm can be improved by allowing domain expert direction during the clustering process.

1. INTRODUCTION

In a conventional database system, queries are answered with absolute certainty. If a query has no exact answer, then the user's needs remain unsatisfied. A cooperative query answering (CQA) system behaves like a conventional system when the query can be answered normally, but tries to find an approximate answer when the original query condition cannot be matched exactly [4, 10].

Many information retrieval (IR) systems are like CQA systems because they return partially matching objects to their queries. Such systems use a set of keywords as an abstract representation of each underlying object. Queries are specified as a set of keywords which are then matched against the abstract representations. If an abstraction matches the query set closely enough, then the underlying document is retrieved.

Unlike IR systems, CQA systems can search the underlying data directly to determine the accuracy of a candidate answer. Early CQA systems found approximate answers by evaluating a nearness function for every tuple in the database. Although this approach works for small systems, it is too inefficient for large applications.

Our approach to CQA, query modification, finds approximate answers by relaxing the query conditions. When a query has no answer, its conditions are weakened to obtain a nearby, or approximate answer. For example, consider a query about transporting ground vehicles for a military application:

> Find a large vehicle that can be packed and easily airlifted to Bizerte

The four conditions on this query are:

---

1. This work supported in part by DARPA contract N00174-91-C-0107
2. The authors may be reached at {matthew,wwc}@cs.ucla.edu

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

C_1: VEHICLE SIZE = large
C_2: PACKABLE? = yes
C_3: AIRLIFT = easy
C_4: DESTINATION = Bizerte

If no such vehicle is available, the query modifier selects one or more attributes for relaxation and replaces the value of that attribute with a nearby approximation. For example, VEHICLE SIZE can be relaxed from "large" to allow medium-sized vehicles as well.

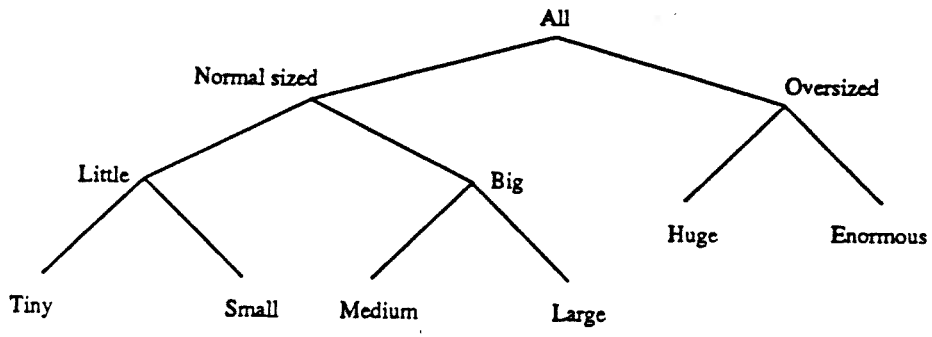

Figure 1: Abstraction Hierarchy for Vehicle Size

To restrict the search space after relaxation to a small set of possible answers, we employ an abstraction mechanism that can discriminate likely candidates efficiently. Our principal data structure, the value abstraction hierarchy groups like values for each attribute. Figure 1 shows the classification of vehicle size for the airlift example. "Medium" and "Large" are clustered into "Big" vehicles, while "Tiny" and "Small," "Huge" and "Enormous" vehicles are clustered into "Little" and "Oversized" respectively. The grouping can be recursive, as "Big" and "Little" are clustered into "Normal sized" at the next level of the hierarchy.

There is an abstraction hierarchy for each attribute domain. Attributes are relaxed by finding their value at the bottom of the appropriate hierarchy and replacing the value in the query with any of the values in the bottom-most cluster of the original value. If searching one cluster in the hierarchy is unsuccessful in yielding an answer, then the condition can be further relaxed by traversing up another level in the hierarchy. Eventually, the query will be sufficiently relaxed to generate an answer.

In this paper, we present a bottom up approach for constructing the abstraction hierarchy called *Pattern-Based Knowledge Induction* (PKI). PKI determines clusters based on rules derived from the instance of the current database. The rules are not 100% certain; instead, they are rules-of-thumb about the database, such as:

If VEHICLE SIZE = small then AIRLIFT = easy

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

Each rule has a popularity that measures how often the rule applies, and a confidence indicating how well the rule "fits" the database. In certain cases, a more sophisticated rule with high confidence can be derived by combining two simpler rules.

The PKI approach generates a set of useful rules which can then be used to construct the abstraction hierarchy by clustering the consequences of rules sharing a similar premise. For example, if the three rules:

If VEHICLE SIZE = tiny then AIRLIFT = easy
If VEHICLE SIZE = small then AIRLIFT = easy
If VEHICLE SIZE = medium then AIRLIFT = easy

each had high confidence, then these three sizes would be clustered closely together. Each invocation of the clustering algorithm adds a layer of abstraction to the hierarchy. By repeatedly applying the algorithm, we will construct an set of abstraction hierarchies.

Considerable past effort has gone into the field of automatic classification, particularly in the IR domain. Most of this has been concentrated on single-domain classifications, such as classifying the symptoms in a medical expert system or the keywords in a book. The classification techniques are usually a variant of the basic nearest-neighbor methods used in vision and pattern matching algorithms [6, 3, 5] These algorithms classify items into sets based on how well the properties of the items match the properties of the sets.

Unlike CQA systems where clustering is done for relaxation, the goal of clustering in information retrieval is to improve retrieval efficiency. Database relations typically span several domains — time, numeric information, names — and thus their tuples cannot be represented by a single set of keywords. Further, the size of a tuple is relatively small compared with the size of a typical document in an IR system. For these reasons, the clustering algorithms that work for single-domain IR systems are inappropriate for databases containing thousands of values spanning multiple domains.

This paper is organized as follows. We first introduce the example domain in detail. Then, after reviewing terms and definitions, we present the Pattern-Based Knowledge Induction (PKI) algorithm and show how it can be used to classify values into clusters. We conclude by presenting an example taken from the transportation planning domain.

2. A TRANSPORTATION EXAMPLE

Our examples for this paper come from a military transportation database used to simulate scenarios of troop and supply movement. The database is used by strategists to plan a time phased force deployment and contains specifications of planes, ships and the cargo they carry. For clarity, we have selected a portion of the database that includes the specifications of ground vehicles, ranging from motorcycles and jeeps to full-sized trucks, but limited to non-hazardous cargo. We focus on four attributes of the vehicle relation:

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

- airlift status — an indication of if the vehicle can be moved by air and takes three values:
  - — easy: the vehicle can be moved by standard bulk transport.
  - — difficult: to carry this, the transporting airplane must be specially adapted, or the cargo must be broken down for transport.
  - — impossible: the vehicle is too big for either the C-130 or C-141 cargo plane and must be transported by ship.
- weight — the weight of the vehicle, in tons.
- size — the rough size of the vehicle
- pack status — can the vehicle be packaged for transport easily?

There are 67 vehicle types specified in the database, shown in Table 1 with the ID numbers removed to save space.

AIRLIFT	WEIGHT	SIZE	PACK	AIRLIFT	WEIGHT	SIZE	PACK
impossible	11-30	Small	No	difficult	11-30	Large	No
easy	under 5	Tiny	No	impossible	11-30	Medium	No
difficult	11-30	Large	No	easy	under 5	Tiny	No
difficult	under 5	Medium	No	impossible	11-30	Small	No
difficult	5-10	Medium	No	impossible	31-50	Enormous	No
difficult	11-30	Huge	No	easy	under 5	Medium	No
impossible	11-30	Medium	No	difficult	11-30	Large	No
easy	under 5	Small	No	impossible	11-30	Small	No
difficult	under 5	Small	Yes	difficult	11-30	Medium	No
easy	under 5	Tiny	Yes	difficult	5-10	Medium	No
impossible	under 5	Tiny	No	impossible	11-30	Small	No
difficult	under 5	Medium	Yes	difficult	11-30	Large	No
easy	under 5	Medium	Yes	difficult	under 5	Medium	No
impossible	11-30	Medium	No	difficult	11-30	Small	No
difficult	under 5	Small	No	impossible	51-60	Large	No
difficult	11-30	Large	No	easy	under 5	Tiny	Yes
impossible	11-30	Medium	No	difficult	11-30	Large	No
easy	under 5	Tiny	No	impossible	51-60	Large	No
difficult	11-30	Large	No	easy	under 5	Tiny	Yes
difficult	5-10	Large	No	difficult	under 5	Medium	No
difficult	under 5	Small	No	impossible	11-30	Medium	No
difficult	11-30	Large	No	easy	under 5	Tiny	No
impossible	11-30	Small	No	difficult	11-30	Large	No
impossible	11-30	Small	No	difficult	11-30	Medium	No
easy	under 5	Small	No	impossible	11-30	Small	No
difficult	11-30	Medium	No	easy	under 5	Tiny	No
impossible	51-60	Large	No	difficult	11-30	Large	No
easy	under 5	Tiny	Yes	difficult	under 5	Tiny	No
impossible	51-60	Large	No	difficult	11-30	Large	No
easy	under 5	Tiny	Yes	impossible	31-50	Enormous	No
difficult	11-30	Huge	No	easy	under 5	Medium	No
impossible	under 5	Tiny	No	difficult	5-10	Medium	No
impossible	11-30	Medium	No	difficult	11-30	Large	No
easy	under 5	Tiny	No				

Table 1: The database table VEHICLES

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

3. PATTERN-BASED KNOWLEDGE INDUCTION (PKI)

In this section, we show how pattern-based knowledge induction systematically acquires rules from database instances. The induced knowledge is then used to cluster semantically similar values.

3.1. Definitions

In order to acquire generalized knowledge such as:

If VEHICLE SIZE = tiny then AIRLIFT = easy

rather than specific facts, such as:

Vehicle #XJ154 is small

we introduce the notion of a *pattern* [1] as the abstract representation for a group of database instances with specified properties. We then represent relationships between data via the relationships between patterns.

A pattern is defined by a *query condition*, such as:

VEHICLE SIZE = tiny *or* WEIGHT = 1130 tons

and the objects that satisfy the pattern condition are said to match the pattern.

Abstract Pattern Class: The set of objects matching a pattern form the Abstract Pattern Class (APC) of that pattern.

An APC is denoted by $P(C)$ where P is its name and C is the pattern condition, and can be abbreviated as P_C.

Rules relate two pattern classes — a premise and a consequence. A rule *applies* to the database if the consequence usually follows when the premise is true. The usefulness of a rule can be measured in terms of how often and how well the rule applies to the database. To provide quantitative values for these properties, we introduce the notions of confidence and popularity, each defined in terms of cardinality.

Cardinality: The cardinality of an APC named P is the number of distinct objects in P and is denoted by $|P|$. For example, the pattern:

AIRLIFT = impossible

has a cardinality of twenty-one, since there are twenty-one tuples in the database that satisfy the pattern condition.

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

A *rule* is an inferential relationship, represented by $A \rightarrow B$, where A and B are patterns. A is the premise and B is the consequence of a rule. The rules that hold in most cases are said to have high *confidence.*

Inferential Confidence: Let A -> B be a rule. The confidence of the rule, denoted by $\xi(A \rightarrow B)$, is defined as:

$$\xi(A \rightarrow B) = \frac{|P_A \cap P_B|}{|P_A|}$$

and is in the range from 0 to 1. If the confidence of a rule is 1 then the rule is deterministic.

For example, assuming $|P_A| = 100$, $|P_B| = 320$, and $|P_A \cap P_B| = 80$, then $\xi(A \rightarrow B) = 80\%$, however, $\xi(B \rightarrow A) = 25\%$. Note that the confidences of rules for two opposite inference directions are neither symmetric nor complementary, a major difference from traditional IR measures that compare set overlap [7, 8, 9]

3.2. Combined Rules

Given rules A → B and B → C with certain confidences, the confidence of A → C depends on the instances of the APCs defined on these conditions. Two rules may be composed using the following formula:

Given rules A → B and B → C, then A → C has confidence

$$\xi(A \rightarrow C) = \frac{|P_A \cap P_C|}{|P_A|}$$

In addition to composition, forms for others compound rules (disjunction, conjunction, negation and ranging) may be found in [1].

3.3. Popularity

Popularity is another important measure of a rule, indicating "how common" the rule is. Consider the rule

ID= *#XJ*154 → SIZE = *small*

that has confidence of 100% but very little coverage relative to the total cardinality of VEHICLES. Such a rule is not sufficiently popular for use in the knowledge base.

The coverage of a rule is measured by its *popularity*; that is, the percentage of its covered cardinality in the total cardinality of the given universe. Popularity is a good indication of how well a rule reduces the size of the database. 100% confident rules about one or two tuples are not necessarily good, since the original tuples could have been used directly without the added computational overhead of rule lookup.

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

A rule belongs to several universes. It applies to one or more relations in the database. For each of those relations, the rule has a *local popularity*, indicating the percentage of tuples in the relation where the premise applies. Further, the rule applies to the database as a whole, so it has a second measure, the *referential popularity*, specifying the percentage of tuples in the entire database where the premise applies.

Formally, let P_A and P_B be APCs used in the rule A → B that applies to a relation R_k, and $|R_k|$ be the number of tuples in R_k (cardinality). The local popularity of the rule over R_k is defined as

$$\eta_{R_k}(A \rightarrow B) = \frac{|P_A|}{|R_k|}$$

Further, if the cardinality of the entire database is |D| and there are r relations in the database, then the referential popularity of the rule over the entire database is

$$\eta_D(A \rightarrow B) = \frac{1}{|D|} \sum_{k=1}^{r} \eta_{R_k}(A \rightarrow B)$$

The popularity of a rule is solely dependent on the premise of the rule and independent of the consequence.

In the example transportation database, there is only one relation, so $\eta_R = \eta_D$ for the rules. Since they are the same in our examples, for the remainder of this paper, we will refer to this value simply as η.

3.4. Knowledge Induction *via* Atomic Patterns

Atomic Patterns: Atomic patterns are the patterns whose *conditions* are on a single attribute, such as "SIZE = tiny" or "WEIGHT under 5 tons".

For example, the atomic patterns of the AIRLIFT attribute are:

P(AIRLIFT = easy)
P(AIRLIFT = difficult)
P(AIRLIFT = impossible)

The algorithm to induce knowledge from a relation via atomic patterns follows directly:

Derive atomic patterns for each attribute in the relation
For every possible pair of atomic patterns (I,J)
    consider I → J as a candidate rule
    add rule if its popularity and confidence exceed threshold values

The popularity and confidence thresholds can be determined based on the database size and the desired size of the final knowledge base. Typically values might require the local popularity to exceed 1% and the confidence to exceed 25% for the rule to be retained. Discarding low-confidence

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

rules is dangerous, however, since those rules might be candidates for composition into more sophisticated rules with higher confidence.

3.5. An Implementation Example

We now demonstrate our rule generation approach using the *VEHICLES* relation of Table 1. For this example, we compare the confidences of the rules with premise patterns

P_A(WEIGHT = 11-30 tons)
P_B(SIZE = large)

and consequent pattern

P_Z(AIRLIFT) = difficult)

From the relation *VEHICLES*

$$|P_A| = 31 \; |P_B| = 17 \quad |P_A \cap P_Z| = 18 \quad |P_B \cap P_Z| = 13$$

From this, and the formula for confidence,

$\xi = 0.58, \eta = .46$: WEIGHT = 11-30 tons → AIRLIFT = difficult
$\xi = 0.76, \eta = .25$: SIZE = large → AIRLIFT = difficult

Based on the high confidence of these rules, and their shared consequence pattern, we consider the more sophisticated rule $(A \wedge B) \rightarrow Z$ and evaluate:

(WEIGHT = 11-30 tons *and* SIZE = large) → AIRLFIT = difficult

Both enumeration from the database directly and the composition formula of section 3.2 yield ξ = 1.00 for the proposed rule. Further, the cardinality of this rule is still high, since it applies to twelve of the tuples in the database, so η = 12/67 = .18

For the entire *VEHICLES* relation, there are 110 basic rules relating the four attributes. Almost half of these rules have a confidence over 50% and twenty-two of the rules are deterministic (have 100% confidence). Only twelve of the rules have a cardinality of one - the remainder apply to at least two tuples in the relation. Thus, the popularity of the rules is sufficient to justify their inclusion in the knowledge base.

4. THE CLUSTERING ALGORITHM,

In this section, we show how to use the rules generated by PKI to construct an initial clustering. The cluster algorithm groups attribute values which appear as premises in rules with the same consequence based on the following rule:

Rule of Shared Consequence If two rules share a consequence and have the same attribute as a premise (but different values), then those values are candidates for clustering.

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

For example, the rules

$$A = a_1 \rightarrow B = b \text{ with confidence} = \xi_1$$
$$A = a_2 \rightarrow B = b \text{ with confidence} = \xi_2$$

share a consequence and have the same attribute (A) as a premise with different values, a_1 and a_2.

4.1. The Clustering Correlation

Using the rule of shared consequence, the *clustering correlation* between two values is the product of the confidences of the two rules, $\xi_1 \times \xi_2$. Since there can be several rules for each premise, The overall clustering correlation between two attribute values, γ, is the sum of the individual correlations. That is, if a_1 and a_2 are values of attribute A, and there are m attributes B_1, B_2, ..., B_m in the relation. then

$$\gamma(a_1, a_2) = \sum_{i=1}^{m} \xi(A = a_1 \rightarrow B_i = b) \times \xi(A = a_2 \rightarrow B_i = b)$$

If B = A, then the two terms in the summation are

$$\xi\,(A = a_1 \rightarrow A = b) \text{ and } \xi\,(A = a_2 \rightarrow A = b)$$

For the first term to be non-zero, a_1 must equal b_1 and for the second term to be non-zero, a_2 must equal b. Therefore, for the product of the terms to be non-zero, $a_1 = a_2 = b$. But a_1 and a_2 are distinct, leading to a contradiction. Therefore, the clustering correlation is zero if B = A. From this, it follows that γ ranges between zero and m - 1. Thus, we divide by m-1 to normalize.

$$\overline{\gamma}(a_1, a_2) = \frac{1}{m-1}\gamma(a_1, a_2)$$

It follows that $\overline{\gamma}(a_1, a_2) = \overline{\gamma}(a_2, a_1)$.

$\overline{\gamma}$ measures the "closeness" of two attribute values, and can thus be used for binary clustering. More general versions of $\overline{\gamma}$ are possible, but prove unnecessary, as will be shown in section 4.3. Table 2 shows values of $\overline{\gamma}$ for the size attribute in the *VEHICLES* relation.

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

A_1	A_2	$\bar{\gamma}(a_1, a_2)$
Huge	Large	.82
Huge	Medium	.63
Huge	Small	.62
Small	Large	.57
Large	Medium	.57
Small	Medium	.54
Small	Enormous	.49
Large	Enormous	.41
Small	Tiny	.41
Enormous	Medium	.40
Tiny	Medium	.39
Huge	Enormous	.33
Tiny	Enormous	.26
Large	Tiny	.24
Huge	Tiny	.24

Table 2: Values of $\bar{\gamma}$ for SIZE in *VEHICLES*.

4.2. The Binary Clustering Algorithm

We first present a binary clustering algorithm that uses $\bar{\gamma}$ values to identify clusters. The binary clustering algorithm is a "greedy" algorithm, putting the two attribute values with the highest overall $\bar{\gamma}$ together. Then, the two values with the next highest $\bar{\gamma}$ are clustered, but only if those values have not already been clustered, and so on, until $\bar{\gamma}$ falls below a specified threshold value or all values are clustered. This algorithm creates a forest of binary clusters.

Using a threshold of 0.5 yields the initial clustering of "Huge" and "Large," since they have the highest $\bar{\gamma}$ in Table 2. The next four highest $\bar{\gamma}$ values involve either "Huge" or "Large" and must be ignored. Thus, the next allowable clustering is between "Small" and "Medium." The last remaining values, "Tiny" and "Enormous" do not get clustered, because their $\bar{\gamma}$ is well below the threshold. Figure 2 shows the initial clusters.

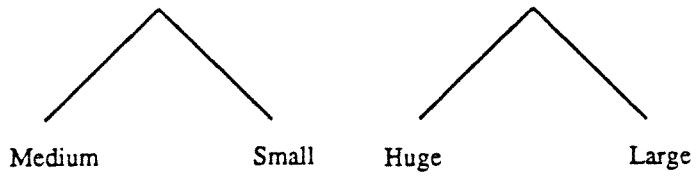

Figure 3: Second Iteration Binary Clustering with $\bar{\gamma} > 0.5$.

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

To construct the next level of the hierarchy, each of these clusters is now given a name. The Huge-Large cluster might be called "Big," while the Small-Medium cluster might be called "Little." Now the algorithm is rerun with "Small" and "Medium" replaced by "Little", and "Huge" and "Large" replaced by "Big." The new $\bar{\gamma}$ values are used to generate the next layer of the hierarchy, which turns out to only cluster "Big" and "Little" as shown in Figure 3.

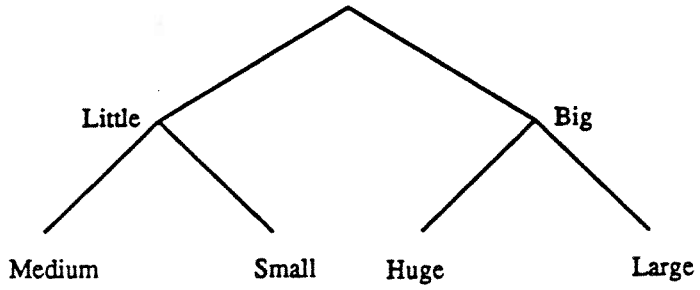

Figure 3: Second Iteration Binary Clustering with $\bar{\gamma} > 0.5$.

The process repeats, but none of the $\bar{\gamma}$ values are above the threshold. To complete the hierarchy, all the individual pieces are linked at the top of the hierarchy, as shown in Figure 4. At this point, "Tiny" and "Enormous" are linked in to the rest of the hierarchy.

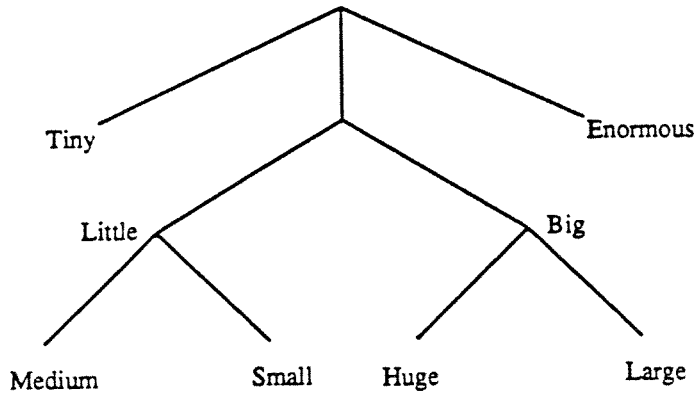

Figure 4: Final Binary Clustering Result with $\bar{\gamma} > 0.5$.

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

4.3. The n-ary Clustering Algorithm

The binary clustering algorithm yields a binary tree hierarchy, except at the top level where all the sub-trees are combined. If a more general n-ary hierarchy is desired, then the algorithm must be extended. The obvious method to achieve n-ary clustering is to use a more sophisticated $\overline{\gamma}$ that measures the correlation between an arbitrary n values instead of just two. However, evaluating such a formula would require considering all possible combinations of values instead of all possible pairs, and would thus be too inefficient.

Instead of modifying $\overline{\gamma}$ to allow n-ary clustering, we approximate the n-ary $\overline{\gamma}$ using a combination of the binary values. Three values should be clustered together if the $\overline{\gamma}$ between each of the three is above the threshold. That is, a_1, a_2 and a_3 are clustered together if $\overline{\gamma}(a_1, a_2)$, $\overline{\gamma}(a_1, a_3)$, and $\overline{\gamma}(a_2, a_3)$ are all above the threshold. In general, if all combinations of a set of attributes have $\overline{\gamma}$ above the threshold, then the entire set are clustered. In the example, "Huge," "Large," "Small," and "Medium" can be clustered together on the first pass, since any pair of these four have a $\overline{\gamma}$ above the threshold. The cluster of four is given a single label, and the algorithm is rerun yielding the cluster shown in Figure 5.

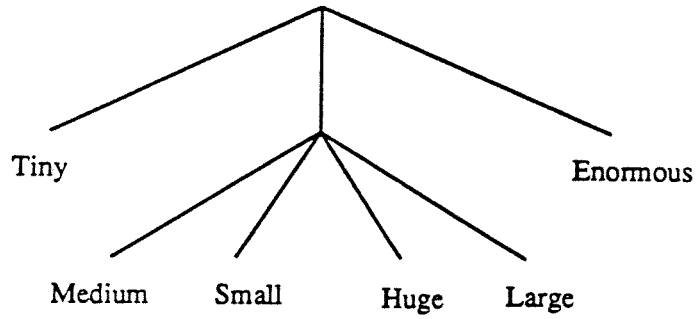

Figure 5: Final n-ary Clustering with $\overline{\gamma} > 0.5$.

The algorithm generates hierarchies with no overlapping clusterings. In some cases, values are allowed to belong to more than one cluster. In such cases, the selection criterion must again be modified. All pairs with $\overline{\gamma}$ above the threshold are clustered, not just the ones belonging to a set as above. Sets of mutually clustered values can be combined as above, but a value may belong to more than one such set.

4.4 Directed Clustering

The final modification to the clustering algorithm allows expert directed rule selection and use. In the example, each attribute was weighted equally for classification. The binary attribute, PACKABLE, was as important as SIZE and WEIGHT. In general, different attributes should have

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

different impact on the final clustering. Suppose we were clustering a new attribute, MPG, the gas mileage of each vehicle. For that attribute, CARGO STATUS is not important, but SIZE and WEIGHT are.

To allow expert control over the selection of attributes for clustering, we propose to assign weights to each attribute. Let $\overline{\gamma_w}$ be the normalized weighted sum of the rules. Let $w_{Bi}(A)$ be the weight assigned to attribute B_i when clustering attribute A, then,

$$\overline{\gamma_w}(a_1, a_2) = \frac{1}{W(A)} \sum_{i=1}^{m} w_{B_i}(A) \times \xi(A = a_1 \rightarrow B_i = b) \times \xi(A = a_2 \rightarrow B_i = b)$$

where

$$W(A) = \sum_{i=1}^{m} w_{B_i}(A)$$

A database expert can now control the behavior of the algorithm by assigning different weights for each attribute.

5. DISCUSSION

Currently, the clustering algorithm works well for discrete values. It would not be suitable, however, for continuous domains, such as numeric values. In the example, the numbers of the WEIGHT attribute have already been pre-clustered into ranges. In general, clustering of continuous numeric domains is pointless, since the clusterings fall naturally based on the data. However, if clustering of numeric domains is necessary, the preprocessing of the domains into small ranges based on application context will vastly improve the performance of the algorithm.

The PKI algorithm for rule detection is exponential on the number of attribute values. Thus, the algorithm does not scale well to large domains. Several techniques can be used to reduce the complexity of the problem. First, the pre-clustering of attributes, as described above, will decrease the number of values in numeric domains and thus reduce the number of atomic patterns and potential rules. Pre-clustering can also help in non-numeric domains, if the expert makes an initial pass through the database and selects values which are closely related. The domain expert can also be consulted to dictate which pairs of attributes are most likely to generate "useful" rules and further direct the PKI algorithm and reduce the computational complexity. For example, the domain expert, knowing that color and m.p.g. are independent, can direct the rule finder to ignore that possibility for rule generation. Further, for very large database, statistical sampling methods can be used for rule gathering. Instead of considering all tuples when evaluating a rule, a randomly selected sample of the tuples in the relation can be used to estimate the rule's confidence and popularity.

ISSN: 2324-9773

Merzbacher, M., & Chu, W. (1992). Instance-Based Clustering for Databases. 3rd ASIS SIG/CR Classification Research Workshop, 101-114. doi:10.7152/acro.v3i1.12599

6. CONCLUSIONS

In this paper, we have presented a Pattern-Based Knowledge Induction mechanism that generates rules relating attribute values in a relational database. The rules can then be used to cluster attribute values and construct abstraction hierarchies suitable for use with cooperative query answering systems such as CoBase [2]. The algorithm works well for discrete non-numeric domains and can use semantics and domain expert direction for pre-clustering to reduce computational complexity. The expert can also add weights that indicate the relevance of certain attributes, thus tailoring the hierarchy based on the semantics of a specific application.

REFERENCES

[1] Q. Chen, W. Chu, and R. Lee. Pattern-based knowledge induction from databases. In *Database Systems for Next Generation Applications*. World Science Publishing Co., 1992.

[2] W. W. Chu, Q. Chen, and R. Lee. Cooperative query answering via type abstraction hierarchy. In S.M. Deen, editor, *Cooperating Knowledge Based Systems*. North-Holland, Elsevier Science Publishing Co., Inc., 1991.

[3] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IT-13(1):21--27, January 1967.

[4] Frederic Cuppers and Robert Demoloube. Cooperative answering: a methodology to provide intelligent access to databases. In *Proc. 2nd International Conference on Expert Database Systems*, Virginia, USA, 1988.

[5] S. A. Dudani. The distance-weighted k-nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325--327, April 1976.

[6] Evelyn Fix and J. L. Hodges, Jr. *Discriminatory analysis: Nonparametric discrimination: Consistency properties*. Technical Report 21-49-004 no. 11, USAF School of Aviation Medicine, 1951.

[7] L. Goodman and W. Kruskal. Measures of association for cross-classifications. *Journal of the American Statistical Association*, 49:732--764, 1954.

[8] L. Goodman and W. Kruskal. Measures of association for cross-classification ii: Further discussions and references. *Journal of the American Statistical Association*, 54:123--163, 1959.

[9] J. L. Kuhns. The continuum of coefficients of association. In Stevens et al., editors, *Statistical Association Methods for Mechanised Documentation*, pages 33--39. National Bureau of Standards, Washington, 1965.

[10] J. Minker, G.A. Wilson, and B.H. Zimmerman. Query expansion by the addition of clustered terms for a document retrieval system. *Information Storage and Retrieval*, 8:329--348, 1972.

ISSN: 2324-9773