RJ 10264 (A0211-017) November 5, 2002
Computer Science

IBM Research Report

Attribute Classification Using Feature Analysis

Felix Naumann, Ching-Tien Ho, Xuqing Tian, Laura Haas, Nimrod Megiddo
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Attribute Classification Using Feature Analysis

Felix Naumann, Ching-Tien Ho, Xuqing Tian, Laura Haas, and Nimrod Megiddo
IBM Almaden Research Center, San Jose, CA 95120
{felix,ho,laura,megiddo}@almaden.ibm.com, tianxq@acm.org

Abstract

Database integration and migration are important, but labor-intensive tasks. To transform data from one representation to another, an expert user must identify and express correspondences between different attributes of different schemata. There are potentially many attributes in a source schema that might correspond to a particular target attribute. Our aim is to ease the burden of the user by classifying source attributes so that they can be automatically and intelligently matched to target attributes.

For categorical data, we present a novel variation of existing Naïve Bayes classification techniques based on domain-independent feature selection. For numerical data, we use a quantile-based classification method, discovering characteristic distributions of the data. We show through extensive experiments that automatic classification of attributes is both feasible and useful for identifying potential matches. The techniques are exploited for several different tasks in Clio, a tool for semi-automatic schema mapping.

1 Finding Attribute Correspondences

The basis of many systems that integrate data from multiple sources is a set of correspondences between source schemata and a target schema. In general, correspondences express a relationship between sets of source attributes, possibly from multiple sources, and a set of target attributes. These correspondences determine how data from the sources is transformed and combined to appear at the target. Although different approaches define the correspondences differently, they all have in common that a user or domain expert must identify and express the correspondences.

Clio is an integration tool, that assists users in defining value correspondences between attributes [HMH+01]. It presents an easy-to-use drag-and-drop interface displaying the source schemata and a target schema. To express a correspondence between two attributes, the user draws an arrow from an attribute of a source schema to an attribute of the target schema, as shown in Fig. 1. Each arrow in the figure represents a correspondence. Based on the source and target schema information, Clio interprets these lines to deduce the final mapping [MHH00]. This schema-based mapping can be improved by an instance-based matching component, which uses existing data in source and target databases to suggest mappings. We address two opportunities to help the users by predicting and suggesting such correspondences in Clio. First, in real life scenarios there may be many sources and the source relations may have many attributes. The users can get lost and might miss or be unable to find some correspondences. Second, in many real life schemata the attribute names reveal little or nothing about the semantics of the data values. Only the data values in the attribute columns can convey the semantic meaning of the attribute.

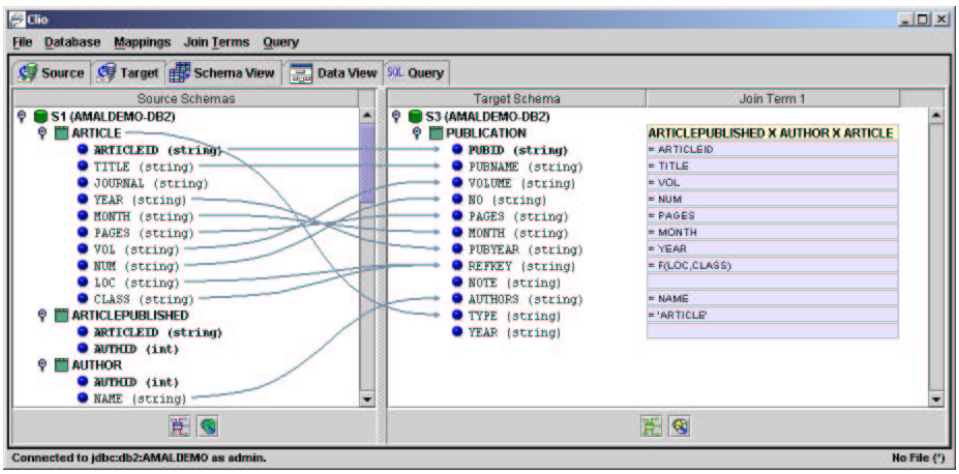


Figure 1: A screen shot of Clio

Example 1. Consider again Fig. 1, where we examine two different schemata for bibliographic data. On the left-hand side is a source schema with relations about scientific articles, authors, etc. On the right hand side is a target schema with the same type of information, but modelled differently.

In order to migrate data from the source schema to the target schema, correspondences between source and target attributes must be identified. For instance, it is unclear whether the source attribute `LOC` refers to a geographic address or a bibliographic reference. The methods presented in this paper can automatically detect that `LOC` is a reference and suggest an appropriate correspondence (e.g., to `REFKEY`). □

The work presented in this paper relieves users of the problems of too many attributes and meaningless attribute names, by automatically suggesting correspondences between source and target attributes. For each attribute, we analyze the data values and derive a set of features. The overall feature set forms the characteristic signature of an attribute. There are more likely to be correspondences between attributes with similar signatures than between others. The automation of the process solves the first problem, because our methods systematically scan all available attributes. Our analysis of the data values solves the second problem, because our methods do not depend on the attribute names but rather on the actual data values stored in the tables. Other applications benefiting from methods for automatically detecting attributes that store similar data are query formation tools, data exploration tools, tools to check consistency and cleanliness, etc.

Our hypothesis is that a properly chosen small set of domain-independent features can mostly capture structural information of categorical (i.e., non-numerical) attributes and distribution of numerical attributes. The focus of the paper is to verify our hypothesis through experiments on several real world databases and one synthetic database.

Contributions. We propose a feature set composed of Boolean features based on the existence of a single character or a set of related characters (Sec. 2). For classifying categorical attributes, we propose the Naïve Bayes classifier, for which we also present an intuitive confidence measure (Sec. 3). We show

through experiments that the feature set captures structural information as an overall good signature, and we show good performance of the Naïve Bayes classifier for our problem domains (Sec. 4).

For classifying numerical attributes, we show through experiments that quantile-based features capture distribution information of attribute values as an overall good signature. We propose to combine two sets of quantiles, one for all values and another for non-zero values only (Sec. 5). The resulting accuracy improves significantly over the use of either set of quantiles alone. Finally, we propose three new similarity metrics, given the feature set, and show their effects and trade-offs on the accuracy of the classifier (Sec. 6). Based on these successful studies, the methods were included in Clio in several deployment modes, from automatically suggesting attribute correspondences to dependency discovery (Sec. 7).

2 Attribute Signatures Using Features

To determine a signature for an attribute, we make use of the properties (features) of the values stored for that attribute. For each data value, we examine certain features and note in a Boolean feature vector whether the value has the feature or not. Features include the presence of certain characters, such as the @-symbol or a space character, in the data field. Also, we examine aggregate features, such as the presence of any upper case character. An attribute signature vector stores the average number of occurrences (as a fraction) of the Boolean features for all of its values.

By determining the similarity of two vectors, we are able to classify attributes using signatures, and hence, make suggestions about which attributes might correspond to each other. An `email`-attribute at the source has a signature with a value close to 1 for the presence of @-symbols and periods, a lower value for the presence of hyphens, and a value of zero for the presence of other symbols such as parentheses. An attribute at the target having the same or a similar signature is very likely to be an attribute also storing email addresses. A strong similarity between the two signatures suggests a correspondence.

Three factors influence the success of our approach. The first two factors are under our control; the third lies beyond our control for a given environment. Hence, we present results that suggest appropriate choices for the first two, and examine the effects of the third.

Choice of features. The chosen feature set must be general enough to highlight the signatures of all possible types of attributes, including numerical attributes, text attributes, dates etc. Knowledge about the data type of the attribute is only a little help: For instance, a `price` can be stored as a string, as an integer, or as a float and is thus indistinguishable by type from, say, an `age` attribute.

Choice of classifier. Different classifiers have different discriminatory power and deliver different results. We examined the accuracy of several popular classifiers. For categorical data, we used the Naïve Bayes classifier, for numerical data a novel distribution-distance method.

Column size. All else being equal, the more data values are available to the classifier, the better the signature represents typical data of that attribute. In order to find corresponding attributes in

two schemata, data should already be present in both.

In our scenario of mapping source schemata to some target schema there is usually a sufficiently large amount of data at the sources. However, since the data is to be migrated to the target schema, data for this schema may not yet exist. If there is no data, we ask the user to enter data examples. Our experiments show good results, even when target attributes have only a small number of data values, e.g., fewer than five entries.

We divide attributes into two categories based on their data types: Categorical attributes include all non-numerical types like `Char` and `Varchar`; numerical attributes include `Smallint`, `Integer`, `Decimal`, and `Double` type attributes. We assume attributes to be database-like attributes with short data, as opposed to text-like attributes with long paragraphs. First, we treat *all* attributes as categorical and present classification techniques and an evaluation. That is, we convert numerical data to `Varchar` type and apply classification techniques. Then, we present specialized techniques for classifying numerical attributes only, again with an evaluation.

3 Classifying Categorical Attributes

Classification consists of two steps: Describing the classes and objects to be classified, and applying a classifier to assign the objects to classes. Here, both the classes and the objects to be classified are columns of data values from an attribute. We describe an attribute with a signature, reflecting innate features of the values. After defining the signatures, we formulate the attribute classification problem. We give a brief review of the Naïve Bayes classifier and propose a confidence measure.

3.1 The Feature Set

A *feature* is a Boolean value describing a certain property of a data value. Features are combined to form a Boolean *data signature* vector describing multiple properties of a single data value. The signatures of all data values in a column are finally aggregated to an *attribute signature* vector with values in $[0, 1]$.

Definition 1 (Feature) *A feature f is a Boolean function that takes a data value t as input and generates 1 (for true) or 0 (for false) as output.*

A feature might check for the presence of an @-symbol, whether the data value contains the upper-case character `A`, or whether it contains any digit. For the purpose of classification, a set of carefully chosen features, called a *feature set*, is used as the basis for generating a data *signature* for each data value. Instead of using Boolean features, we have considered using numeric features that count occurrences of characters. McCallum and Nigam compare the Naïve Bayes classifier as we use it (see Section 3.2) with such a multinomial variation [MN98]. They conclude that the former generally performs better at small feature set sizes, which is the case for our setup. Additionally, Boolean features minimize computational complexity and enable efficient ad hoc classification requests.

Definition 2 (Data Signature) *Given a data value t and a feature set $F = \{f_1, f_2, \cdots, f_k\}$, the data signature of t with respect to F is $F(t) \coloneqq (f_1(t), f_2(t), \cdots, f_k(t))$.*

Example 2. Let data values t_1 = `ho@almaden.ibm.com`, t_2 = `naumann@hu-berlin.de`, and t_3 = `123@yahoo.com`. Let feature f_1 check whether the input contains an @-symbol, f_2 check whether the input contains a hyphen, and f_3 check whether the input contains any digit. Then with respect to $F = \{f_1, f_2, f_3\}$, the signature of t_1 is $F(t_1) = (1, 0, 0)$, and $F(t_2) = (1, 1, 0)$, and $F(t_3) = (1, 0, 1)$. □

Definition 3 (Attribute Signature) *Given an attribute a with n data values, $t_1, t_2, \cdots, t_n$, and a feature set F, the attribute signature of a is* $\mathfrak{S}(a) := \frac{1}{n} \sum_{i=1}^{n} F(t_i)$.

Thus, the j-th element in the attribute signature $\mathfrak{S}(a)$ is the average of the j-th value of the data signatures of all values in a. This value is between 0 and 1 and represents the probability that a data value in the attribute has this feature. We use this information to decide whether a new data value should be classified as belonging to this attribute.

Example 3. Figure 2 shows three attributes with sample data. The attribute signatures of the `emails` attribute (a_1), the `eAddresses` attribute (a_2), and the `Full_name` attribute (a_3) with respect to the three features described in Example 2 are, respectively:

$$\mathfrak{S}(a_1) = \frac{1}{10}\left(\begin{pmatrix}1\\0\\0\end{pmatrix} + \begin{pmatrix}1\\1\\0\end{pmatrix} + \cdots + \begin{pmatrix}1\\0\\0\end{pmatrix}\right) = \begin{pmatrix}1\\\frac{1}{10}\\\frac{3}{10}\end{pmatrix}, \quad \mathfrak{S}(a_2) = \ldots = \begin{pmatrix}1\\\frac{1}{7}\\\frac{3}{7}\end{pmatrix}, \quad \mathfrak{S}(a_3) = \ldots = \begin{pmatrix}0\\\frac{1}{8}\\0\end{pmatrix}$$

Already from this small example, the signatures of the two first attributes suggest a similarity in the data, and a dissimilarity of the third. □

a_1: `emails`

ho@almaden.ibm.com
naumann@cs.hu-b.de
123@yahoo.com
superman@web.ac.uk
dan_99@acm.org
bill@hotmail.com
lou@us.ibm.com
tianxq@berkeley.edu
mike@123go.com
BillClinton@mme.net

a_2: `eAddresses`

help@altavista.com
HIGH5@internet.org
tim@w3c.org
laura@almaden.ibm.com
12monkeys@tu-d.de
sombody@somewhere
felix@almaden.ibm.com

a_3: `Full_name`

Xuqing Tian
Howard Ho
Laura Haas
Felix Naumann
Dan Rather
Bill Clinton
Peter John-Carter
Barbara Bush

Figure 2: Three attributes with data

We consider two types of features: *singleton* features and *aggregate* features. A singleton feature checks if a data value contains a certain character. An aggregate feature checks if a data value contains any character in a predefined aggregate set of characters. The aggregate set intends to capture characters that are closely related at various levels. In this paper, we consider all singleton features that check for a delimiter, a letter, or a digit. For the aggregate features, we consider the following: delimiter, lower-case vowel, upper case vowel, lower-case letter, upper case letter, letter, digit, and alphanumeric. Table 1 summarizes these 103 features.

Using these features, attributes with different semantics have different signatures. E.g., a `phone number` attribute has high scores for the singleton digit features, for the singleton hyphen feature, and a very high score for the aggregate digit feature. A `last name` attribute has low scores in all of those, but high scores for the aggregate character and vowel features, and characteristic scores for the space

Singleton Features	Aggregate Features
{a}, ..., {z},	{@,-, ..., /, \}, {0,...,9}
{A}, ..., {Z},	{a,e,i,o,u}, {A,E,I,O,U}
{0}, ..., {9},	{a,...,z}, {A,...,Z}
{@}, ..., {\}	{a,..., z,A,...,Z}
	{a,..., z,A,...,Z,0,...,9}

Table 1: The feature set

and hyphen delimiters, reflecting the typical frequency of double last names. Recall that we assume attributes to store database-like, short data values. For text-like attributes such as a book review, all features will score 1 or close to 1. If only one such attribute is present, our classifier will correctly classify it, if more are present, other features are necessary (see below).

Other features. In this paper we highlight only character-based features, to assess their potential and show their suitability for our application scenario of small test sets and data-type attributes. Other types of features are potential candidates for successful classification: schema meta-data, n-grams, entire words, and domain-dependent features. All mentioned features are useful to some extent and in some scenarios. A combination through some weighted aggregation of different classification results can be quite powerful, and is to some extent used in Clio, but not reported on here.

Schema meta-data like data type, attribute length (number of Bytes allocated to the attribute), attribute name, and functional dependencies are usually readily available, but often are of little use. Typically, data types and length alone do not convey the semantics of an attribute, e.g., a varchar-type attribute could store a date, an ID, a name etc.

An attribute name can contain some semantic meaning, and Clio indeed uses an edit-distance measure between attribute names to complement and enhance automatic classification. We do not report on these enhancements in this paper.

n-grams are character sequences of length n, i.e., the singleton features are 1-grams. N-grams were introduced as features for text classification and have proven to be very useful [Dam95]. A disadvantage of using n-gram features is the increased time- and space-complexity of the classifier. In the dramatically increased signature space small data sets will not have a signature that is detailed enough to be comparable. Consider a training set of say 10 last names. Only a few of the numerous n-grams appear in those names, rendering it likely that a test set with say 5 other names will have no matching n-gram and is thus not recognized as a match.

Entire words as features, an approach developed for text classification, are used for instance in [AJS00, DDH01]. Word-based features, possibly enhanced by the use of an ontology, are not appropriate for the problem at hand: We assume data-type attributes, not text-like attributes. Using word-based features would drastically reduce the ability of the feature vector to express the signature of an attribute category, for the same reasons as for n-grams.

Domain-dependent features and domain-dependent matching techniques are suggested by Doan et al. [DDH01]. For instance, the occurrence of predefined words such as `beautiful` and `great` point to the description attribute of a real-estate database. The goal of our paper is to present a general

classifier, that is not restricted to a specific domain and that must not be adapted with each new usage. Domain-dependent features can and should complement our approach at deployment.

3.2 The Attribute Classification Problem

We describe the problem of finding corresponding attributes as a classification problem. Given m attributes and n data values per attribute, we generate $m \cdot n$ training data records. Each contains k input values (corresponding to the k feature values) and a class label: the attribute ID. For the testing, we are given t data values from some (unknown) attribute $a_x, 1 \leq x \leq m$, with equal probability and undisclosed to the classifier, and ask the classifier to predict the attribute ID. In general, any type of classifier can be used. In the following paragraphs we present the Naïve Bayes classifier and discuss several alternatives.

Naïve Bayes classifier. The Naïve Bayes model as a classifier was introduced by Good [Goo65]. We use the multi-variate Bernoulli model for the Naïve Bayes classification, as laid out by McCallum and Nigam [MN98]. Given a test data value t, the Naïve Bayes classifier estimates the posterior probability of t belonging to source attribute a_i (training data) via Bayes' rule:

$$P(a_i|t) = \frac{P(t|a_i) \cdot P(a_i)}{P(t)} \tag{1}$$

This term is evaluated for each attribute a_i, and we assign the test column to the attribute that maximizes the product of the probabilities of all t in the test column. We ignore $P(t)$ in the denominator and $P(a_i)$ in the numerator, because in our model they are the same for all a_i. To estimate $P(t|a_i)$, we "naïvely" assume that all the features of t, $f_j(t)$, $1 \leq j \leq k$, are independent to get

$$P(t|a_i) = \prod_{j=1}^{k} P(f_j(t)|a_i). \tag{2}$$

$P(f_j(t)|a_i)$ is the fraction of occurrences of the particular feature in all data values of a_i. To avoid situations where $P(f_j(t)|a_i) = 0$ and thus $P(t|a_i) = 0$, we apply Laplace's law of succession [Goo65] to arrive at

$$P(f_j(t)|a_i) = \begin{cases} \frac{|a_i| \cdot \mathfrak{S}_j(a_i)+1}{|a_i|+2} & \text{if } f_j(t) = 1 \\ \frac{|a_i| \cdot (1-\mathfrak{S}_j(a_i))+1}{|a_i|+2} & \text{if } f_j(t) = 0 \end{cases} \tag{3}$$

where $\mathfrak{S}_j(a_i)$ is the average value of feature j for training column a_i, and $|a_i|$ is the size of the training column.

Example 4. Given the `emails` attribute a_1 and the `Full_name` attribute a_3 of Fig. 2, we want to

classify the data values of the `eAddresses` column. The first value, `help@altavista.com`, is classified:

$$P(\texttt{help@alta}\ldots|a_1) = \prod_{j=1}^{3} P(f_j(\texttt{help@alta}\ldots)|a_1) = \frac{11}{12} \cdot \frac{10}{12} \cdot \frac{8}{12} = \frac{880}{1728} \approx 0.51$$

$$P(\texttt{help@alta}\ldots|a_3) = \prod_{j=1}^{3} P(f_j(\texttt{help@alta}\ldots)|a_3) = \frac{1}{10} \cdot \frac{8}{10} \cdot \frac{9}{10} = \frac{72}{1000} = 0.072$$

In the same way, we classify the six other data values and compare the probability that all 7 data values belong to a_1 against all of them belonging to a_3. We correctly conclude that a_2 and a_1 match better than a_2 and a_3. □

Other classifiers. Distance-based classifiers calculate the distance of signature vectors in the k-dimensional cube, where k is the number of features. The vector closest to the test signature vector is then suggested as a corresponding attribute. We performed tests using different distance measures (Manhattan distance L_1, Euclidean distance L_2, and maximum distance L_∞) and found Naïve Bayes to consistently outperform all distance-based classifiers. This observation matches the results of other work like [LIK92, DP97]. Lewis and Ringuette compared the Naïve Bayes classifier with a decision-tree learning algorithm [LR94]. The authors conclude that both perform similarly well for the categorization of text documents.

Finally, Agrawal and Srikant propose a revised Naïve Bayes classification for catalog integration [AS01]. The authors make use of the intuition that if two documents are in one category of a source catalog, it is likely that they belong to one category of the target catalog. The analogy to our problem is that if two attributes are in one source table, it is likely that they correspond to attributes of the same target table. Testing the usefulness of this intuition in the schema mapping domain remains future work. In Sec. 8 we discuss related work on attribute classification using other classifiers, such as neural networks.

3.3 A Confidence Measure for the Naïve Bayes Classifier

In order to suppress suggestion of which Clio is not sure and to rank suggestions, we propose a confidence measure for the results of the Naïve Bayes classifier[1]. Recall that a target attribute a_x is classified as corresponding to source attribute a_i, if a_i maximizes the probability that *all* data values from a_x "belong" to a_i. That is, a_i maximizes $\prod_{t \in a_x} P(a_i|t)$. The value $P(a_i|t)$ in turn is calculated as the probability that *each* feature of t corresponds to the average feature value of a_i. That is $P(a_i|t) = \prod_{f_i} P(f_j(t)|a_i)$. Together, the final probability that a target attribute corresponds to a source attribute is $\prod_{t \in a_x} \prod_{f_i} P(f_j(t)|a_i)$. This probability is in $[0, 1]$, where 1 is reached only if for *all* values of a_i and a_x all features are 1.

For data-type attributes we expect this never to occur, and never observed it for the databases of our tests. For most features, probabilities are 0 or close to 0. In consequence, the probability that all 103

[1] We do not use the word confidence in the mathematical sense, but as a measure to give users a feeling for the surety of the result.

features match is small, and thus, the final probability is also very small, decreasing with the number of features and test tuples—a typical probability result in our experiments ranges between 10^{-3000} for the top ranked attribute and 10^{-5000} for the lowest ranked attribute. Clearly, these numbers cannot be used as confidence values for our choice—a user will not accept suggestions that have a confidence of virtually 0, and whose confidence is almost indistinguishable from that of other choices.

A simple and effective method to solve this problem is to report only a ranking of suggestions without confidence scores, or to scale the scores so that the top choice has confidence/probability 1 and the others are below. However, apart from the probability as calculated above, a confidence score for a correspondence suggestion in Clio should also reflect the number of test and training tuples involved. Intuitively, the more training and test tuples there are, the higher the confidence is. This intuition is strongly supported by the results of our experiments. There is no single, correct way to scale the scores to meet these requirements; we present a method meeting these criteria and successfully used in Clio.

To deal with the extremely small probability values, we used their logarithmic values for all calculations. With 100 test and training data, a typical result lies within -3000 for the top ranked and -5000 for the 10th ranked attribute (in Clio, we only consider the top 10 attributes). For scaling purposes, we enter the test attribute itself as an additional training attribute. Because this new training attribute will have the same signature as the test attribute, it is always ranked first, and it is ranked with the best possible score. We scale all other values against this score: Let $P(a_t)$ be the final (logarithmic) score for the newly added attribute, and let $P(a_i)_{i=1,\ldots,m}$ be the scores for all other attributes. We define classification confidence $c(P(a_i)) := P(a_t)/P(a_i)$. This scaling has the property that the best possible matching of a_t—the one with itself—has confidence 1, and all others have a confidence that takes into account the number of participating tuples. The higher the number of tuples, the lower the probability of a match as calculated by the Naïve Bayes classifier will be, but the higher the confidence will be, because the probability appears in the denominator. In the experiments we observed practical confidence scores: Good matches have confidence scores of over 0.9.

4 Evaluation of Classification Methods – Categorical Attributes

We used several different domains to test the features and Naïve Bayes classifier. The first is a set of three bibliography databases from academia. The second is a set of columns taken from three different real estate Web sites. We also conducted experiments with a commercial database collecting information about semiconductor manufacturers and an insurance database storing data about life insurance contracts. Due to space limitations, we do not present the results for those databases here, but these results are consistent with the ones we do present.

In each experimental run for a database, for each attribute of the training database we choose $n \in \{50, 100, 250, 500, 1000\}$ random non-null data values as a *training column* to construct the signature for that attribute. Then, we choose for each attribute of the test database a set of size m of random non-null data values as a *test column*. We vary m as appropriate for each experiment. Given this input, we classify test data values as a column, i.e., we let the classifier decide, from which of the attributes this column was chosen. Knowing the correct answer, we note whether the classification of the corresponding attribute was correct or not. Such a run is repeated 1,000 times and an average misclassification rate

is taken over all runs and over all attributes.

First, we use this experimental setup to decide, which feature set produces the best result (4.1). To quantify the difficulty of classifying attributes of the three databases, we perform confusion distance measurements (4.2). Using the Naïve Bayes classifier and a combination of singleton and aggregate features as the best performers, we present the classification results for the different databases (4.3–4.4).

4.1 Choosing a Feature Set

Table 1 in Sec. 3.1 lists two classes of features: singleton features and aggregate features. To find out which feature set is the most useful, we performed five tests with different combinations of features using the Naïve Bayes classifier. The results for the bibliography database are shown in Fig. 3. Each line represents a misclassification rate obtained by averaging the individual rates for different sized training columns.

We observe that using aggregate features alone creates a mismatch for about every third attribute. Delimiter features alone already yield good results, but using all singleton features (including delimiters) performs best. Complementing the latter two feature sets with aggregate features yields a slight gain for each. Using all singleton and aggregate features together yields the best result. Using all singleton and aggregate features covers properties of both a wide range of string attributes, such as names and addresses, and numerical attributes, treated as strings, such as date or phone. For the following experiments we use the singleton and aggregate features together.

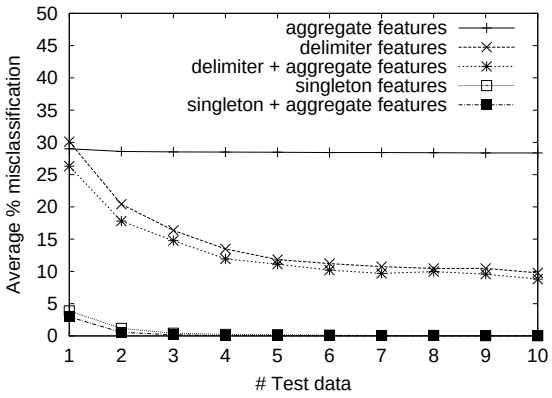


Figure 3: Choosing a feature set

4.2 Confusion Distance Measurement

Intuitively, the misclassification rate depends on the classifier chosen, the features chosen, and the contents of the data. In this section, we give a metric for measuring the degree of difficulty in classifying attributes of a database.

Definition 4 (Attribute Distance Matrix) *Given m attributes $a_1, \ldots, a_m$, each with a signature, $\mathfrak{S}(a_i)$, we define the attribute distance matrix M as an $m \times m$ symmetrical matrix, where a matrix element $x_{ij} = d_2(a_i, a_j)$ is the Euclidean distance between signatures $\mathfrak{S}(a_i)$ and $\mathfrak{S}(a_j)$.*

Definition 5 (Confusion Distance) *The confusion distance $c_M(a_i)$ of an attribute a_i is $c_M(a_i) := \min_{i \neq j}\{x_{ij}\}$.*

Thus, the confusion distance of an attribute is the Euclidean distance to its closest neigbour in the vector space of signatures.

Figure 4 shows the confusion distances for the attributes of the bibliography and real estate databases. Each curve shows the confusion distances of the attributes in a database, sorted in ascending order. Lower values represent a higher difficulty in accurately matching the attribute. For the real estate database R_2, we have already clustered attributes with a distance of less than 0.01.

We observe the real estate database R_2 to be the most difficult, and database R_3 the easiest. R_2 stores many attributes of similar semantics and contents, such as boolean values about whether there is a fireplace/patio/golf course... or not. The confusion distance gives us a better feel for how well our classifier is really working with respect to the similarity of attributes of the databases to be classified.

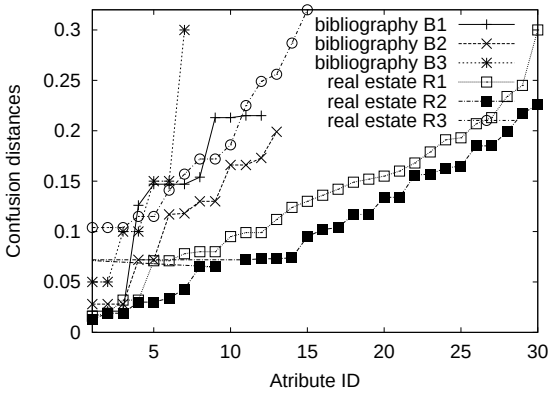


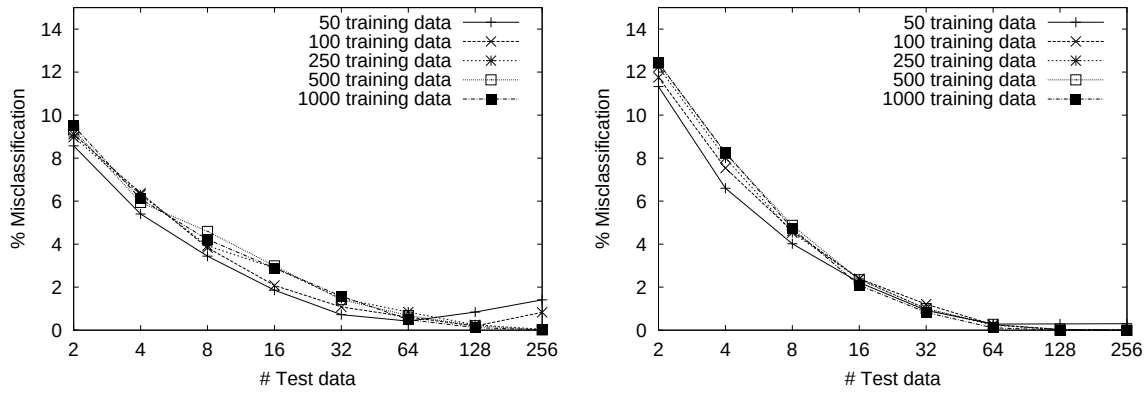
Figure 4: Confusion distances

4.3 The Bibliography Domain

For testing out techniques, we used three different bibliography databases (B_1, B_2, B_3) containing categorical attributes, such as article ID, title, authors, pages, month, etc., and numerical attributes, such as volume, number, and year. Each contained over $1,000$ records from different sources on the Web. For the purpose of this testing, we treat all numerical attributes as categorical, increasing the difficulty. Bibliography data typically has many null-values due to the many different types of publications. After removing from consideration those attributes that do not have enough non-null values to fill the training column, 11 attributes remain in B_1, 12 in B_2, and 8 attributes in B_3.

Tests for accuracy. In this test, we partition all columns of a database into two parts—a training part and a test part. The classifier returns a correct result if it classifies a test column as belonging to the training column taken from the same database column. Figure 5 shows the misclassification rate as a function of test column sizes for various training column sizes. For example for database B_2, with a training column size of 500 and a test column size of 8, the misclassification rate is 5%, meaning 95% of all attempts, the classifier predicts the correct corresponding attribute over 7 attributes. Clearly, the larger the test column is, the more accurate the classifier is. We emphasize that in a typical application scenario for Clio, for which we developed these methods, there is little to no test data. In cases where no test data is available, the user is asked to enter sample tuples. We observe that misclassification is already very low for only four test values. Hence, this is not an undue burden for users.

Throughout the experiments we observed misclassification rates to remain fairly constant after the test set size surpasses the training set size. As the test set size increases, its signature variance approaches zero, i.e., the test signature reflects more and more the true signature. Further test values



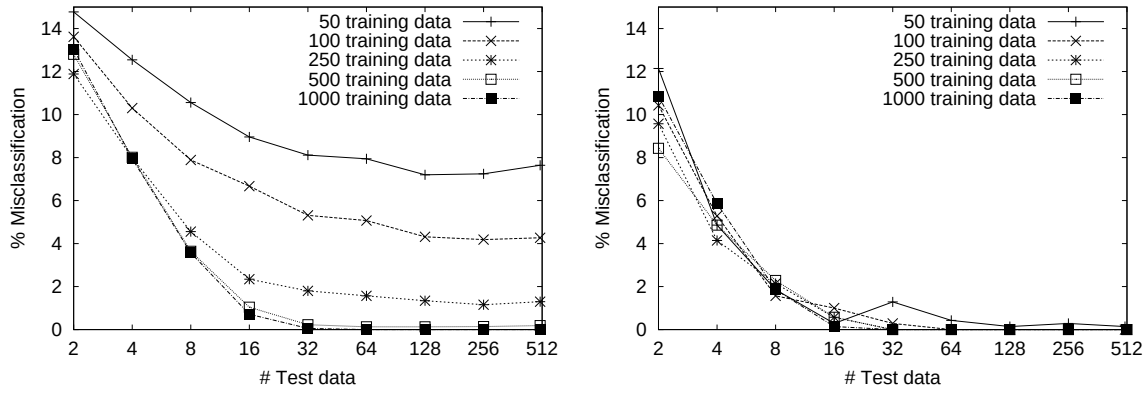
(a) Database B_1

(b) Database B_2

Figure 5: Misclassification for bibliography databases

do not decrease variance, and thus do not improve classification, which is determined by the constant variance of the training set signature. For a more detailed analysis of this behavior, see Appendix A.

Tests for sensitivity. The previous experiment draws the training column data values and disjoint test column data values from the same database. However, the aim of Clio is to create mappings from multiple sources to a different target source. Therefore, an attribute-matching classifier should recognize matching attributes using training data from one source and test data from another. Figure 6 shows the results of two experiments, both with B_3 as test database and with B_1 and B_2 as training databases, respectively. The attributes of B_3 are a subset of each of the training databases, i.e., we ask the classifier to find a match *for each* test attribute.



(a) 11 training attr. (B_1) and 8 test attr. (B_3)

(b) 12 training attr. (B_2) and 8 test attr. (B_3)

Figure 6: Misclassification among different bibliography databases

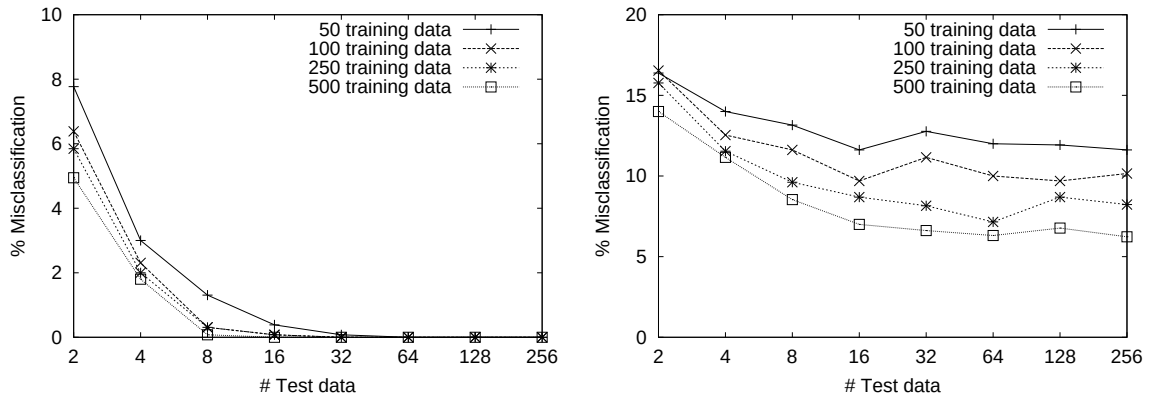
The Naïve Bayes classifier shows an overall slightly lower performance, because of different format-

ting of the data. Despite the difficulties, the classification results are satisfying. Intuitively, this is because our choice of feature set is based on the character set distribution (histogram), not the commonly used word frequencies of a typical text classifier. So even when the data comes from different sources with different content, the classifier detects semantically matching attributes.

4.4 The Real Estate Domain

We use data collected from three real estate Web sites (R_1, R_2, R_3) providing information about houses for sale[2]. The real estate domain is considerably more difficult than the bibliography domain, because there are more attributes that are indistinguishable looking only at their data values. For instance, no system will be able to distinguish a phone number from a fax number, using only data values[3]. There are also attributes that are unique to a certain source, i.e., there is no correct match for that attribute. In general, the domain lends itself more to word-based classification and for us presents an excursion to non-traditional information sources.

To account for these difficulties we change the paradigm of attribute matching slightly: We no longer ask for the best match for each attribute; instead we ask for the best match *or no match* for each attribute. This shift is in the spirit of Clio, which does not overwhelm users with an enormous amount of possibly incorrect matches, but only suggests matches of which it is confident.



(a) 30 training attr. (R_1) and 20 test attr. (R_3)

(b) 50 training attr. (R_2) and 20 test attr. (R_3)

Figure 7: Misclassification among different real estate databases

The results presented in Figure 7 are achieved by attempting a match only if its confidence is greater than 90%, according to the confidence measure introduced in Section 3.3. In all experiments, approximately one third of all suggestions have confidence greater than 90%; for the remaining attributes, no match is suggested. If no match with that confidence is found, Clio can suggest the top k matches, even at lower confidence. We observe that the problem becomes more difficult with an increasing number of training attributes to choose from. However, the results remain at an acceptable level, especially

[2] It is the same data as reported on in [DDH01], which the authors kindly supplied.

[3] Clio additionally considers attribute names when suggesting matches, but we do not report on this technique here.

considering that they are to be confirmed by a user in Clio.

5 Classifying Numerical Attributes

Recall that the objective of the paper is to help users identify semantically equivalent attributes, given a target attribute, and so specify correspondences between different schemata. We support this process by finding such attributes using feature analysis. Consider an insurance database containing many numerical attributes of similar range and distribution, such as the number of eligible drivers in the family and the number of insured cars, the start and end date in 8 digit form etc. Clearly, many simple techniques can be used to distinguish numerical attributes with mostly disjoint ranges. Also, the Naïve Bayes classification of the previous sections is suitable to classify numerical attributes in many cases. However, to distinguish numeric attributes with similar ranges and similar distribution, a more advanced algorithm is required. As before for categorical attributes, we characterize numerical attributes using features.

5.1 The Feature Set

Definition 6 (Numerical Feature) *A numerical feature g is a function that takes a column of numerical data and returns some statistical value of the data distribution as output.*

Note that generally a numerical feature is not Boolean. Simple examples of numerical features are min, max, mean, and median. To best model the range and distribution of the values of an attribute, we choose the following 18 features for our implementation.

- The 10%, 20%, to 90% quantiles $q_1, \ldots, q_9$ of each data column. That is, $q_i(a)$ is the data value at position $\lfloor \frac{i \cdot |a|}{10} \rfloor$, after sorting the values of attribute a.
- The 10%, 20%, to 90% quantiles $q'_1, \ldots, q'_9$ of each data column after removing all data of value zero.

We show from experiments on two real-world databases later that combining the two types of quantiles classifies better than either one of them. We do not choose min and max features to avoid being misled by possible outliers. We have seen many cases in which the "unknown" value for numerical data is stored as a zero value rather than using null. In real-world databases, there are many numerical attributes with mostly null values, which are now stored as zero. Hence the need for the second set of quantiles.

We choose quantiles in 10% intervals to avoid over-sampling and to reduce computational complexity. To obtain more precise signatures, the intervals could be reduced when more training and test data is available.

Definition 7 (Numerical Signature) *The signature of a numerical attribute a is the vector* $\mathfrak{N}(a) := (q_1(a), \ldots, q_9(a), q'_1(a), \ldots, q'_9(a))$.

5.2 Quantile-based Classification

We introduce three quantile-based classification methods—VOTE, SUM, and RANK and give an example below. As before, we generate one signature for each numerical attribute from the training column. We then generate the signature for the numerical test attribute to be classified. Let a_i $(1 \leq i \leq m)$ be the training attribute and let a_x be the test attribute. With this input the classifiers are:

VOTE: For each feature value $q_j(a_x)$, $1 \leq j \leq 9$, a vote is given to that attribute a_i where $|q_j(a_x) - q_j(a_i)| < |q_j(a_x) - q_j(a_k)|$ for all $1 \leq k \leq m$, $(k \neq i)$. When a tie occurs, the vote is evenly distributed among the winners. An analogous rule applies to $q'_j(a_x)$. The voting method classifies a_x as belonging to the attribute with the most votes. In case of a tie, in our experiment, the attribute is randomly classified to one of the winners. In Clio, we let the user choose from all the winners.

SUM: For each attribute a_i this method determines the sum of feature value differences $\sum_{j=1}^{9} |q_j(a_x) - q_j(a_i)| + \sum_{j=1}^{9} |q'_j(a_x) - q'_j(a_i)|$. It classifies a_x as belonging to the a_i that minimizes this sum.

RANK: RANK is similar to SUM, but it sums differences in feature value ranks of interpolated values. Specifically, let $rd(i, x, j)$ be the rank deviation between the ideal rank j and the projected rank of $q_j(a_x)$ in the list of $q_j(a_i)$, $1 \leq j \leq 9$, using linear interpolation.[4] We define $rd'(i, x, j)$ similarly for the second set of quantiles q'_j, $1 \leq j \leq 9$. RANK classifies a_x as belonging to the a_i that minimizes $\sum_{j=1}^{9}(rd(i, x, j) + rd'(i, x, j))$. This method is stable, even when data values of an attribute do not follow a single distribution. We apply the SUM method as a tie breaker.

Example 5. Figure 8 shows an example of the three quantile-based classifiers for the first set of quantiles $q_1, \ldots, q_9$ with two training attributes (a_1, a_2) and one test attribute (a_x). For the RANK classifier, consider the derivation of $rd(1, x, 4)$ as an example. We are interested in projecting the rank of $q_4(a_x) = 43$ in between $q_4(a_1) = 47$ (of rank 4) and $q_3(a_1) = 35$ (of rank 3). With linear interpolation, the value 43 is projected into rank $4 - (47 - 43)/(47 - 35) = 4 - 1/3$. Thus, $rd(1, x, 4) = 1/3$, i.e., a $1/3$ deviation from ideal rank 4. □

	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	total	winner
Train a_1	10	22	35	47	56	64	73	81	90		
Train a_2	10	21	32	44	54	63	72	80	90		
Test a_x	10	22	32	43	54	64	72	80	90		
VOTE a_1	0.5	1	0	0	0	1	0	0	0.5	3	
VOTE a_2	0.5	0	1	1	1	0	1	1	0.5	6	$\surd$
SUM a_1	0	0	3	4	2	0	1	1	0	11	
SUM a_2	0	1	0	1	0	1	0	0	0	3	$\surd$
RANK a_1	0	0	3/13	4/12	2/9	0	1/9	1/8	0	1.02	
RANK a_2	0	1/11	0	1/12	0	1/9	0	0	0	0.29	$\surd$

Figure 8: Example of three quantile-based classifiers

[4] or extrapolation for $q_j(a_x)$ smaller than $q_1(a_i)$ or larger than $q_9(a_i)$.

6 Evaluation of Classification Methods – Numerical Attributes

We evaluate the quantile-based classification methods by applying them to two real world databases and a synthetic data set. The first database is a collection of numerical attributes taken from a large biochemical database (Sec. 6.4); the second is an insurance database (6.5). Often, real world attributes have a distinct and easy-to-recognize distribution. Therefore, to test classification in a more difficult situation, we generated several data columns with synthetic data and only slightly differing distributions (6.6-6.7).

6.1 Confusion Distance Measurement

As before for categorical attributes, we want to examine the difficulty of classifying numerical attributes. That is, we need a measure to compare the distance/similarity of two distributions.

Definition 8 (Distribution Distance) *The L_1 distance between any two distributions of attributes a_i and a_x of domain D is defined as $d_1(a_i, a_x) = \sum_{t \in D} |P_i(t) - P_x(t)|$, where $P_i(t)$ is the probability of data value t occurring in a_i.*

The L_1 distance is between 0, reflecting identical distributions, and 2, reflecting disjoint distributions. Figure 9 quantifies the difficulty of classifying the attributes from the real-world databases. It shows the L_1 distance for each attribute to the closest other attribute. The higher the distances, the easier the attribute is to classify. For convenience, we scale the L_1 distance to a percentage.

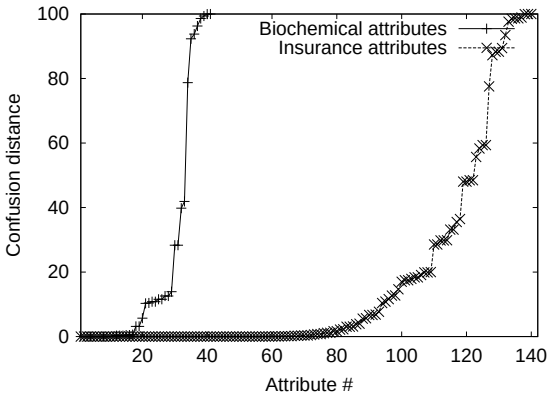


Figure 9: Confusion distances

We deem any numerical attributes with a distance of less than 5% as virtually indistinguishable. Therefore, we cluster such attributes, leaving 61 attributes for the insurance database and 26 attributes for the biochemical database. Eventually, Clio presents all elements of such a cluster to users, leaving them to decide which correspondence is semantically correct, possibly using the attribute name as an additional indicator.

To evaluate the difficulty of classifying data from the biochemical and the insurance database, the two graphs of Figure 10 show respectively the distribution of the columns cumulatively—one graph for each attribute. Even though the curves are scattered over the entire plot signalling differing signatures and hence easy classification, in both databases there are several columns of similar distributions accounting for higher misclassification rates.

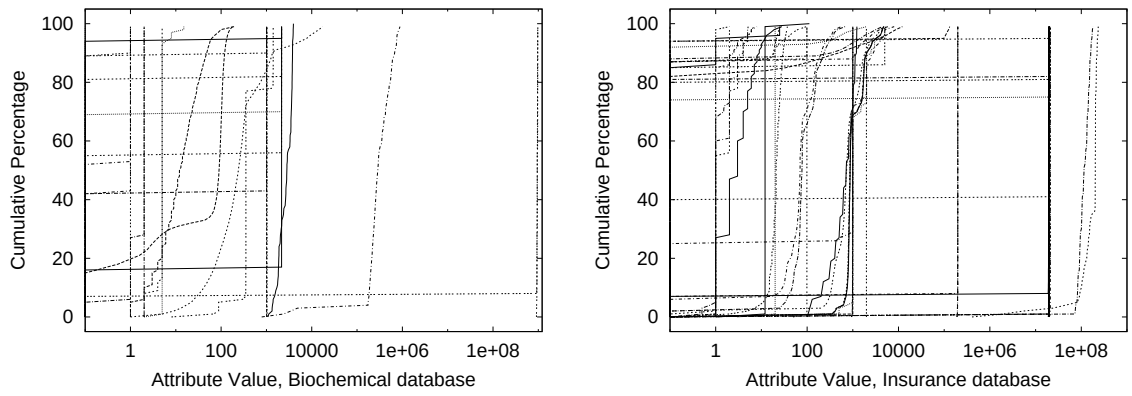


Figure 10: Cumulative distribution of numerical attribute values (one graph per attribute)

6.2 Choosing a Classifier

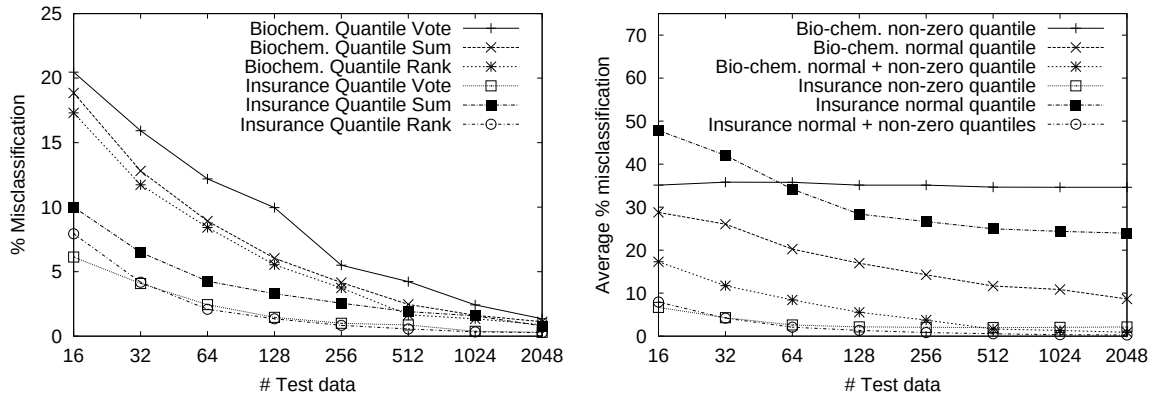
To compare the classifiers presented in Sec. 5.2, we performed measurements for both real life database and several synthetic data columns. For each experiment we chose equal training and test column sizes, but observed similar results for other settings. From the results shown in Fig. 11(a) and the measurements on different synthetic data columns not shown here for space limitations, we can conclude that the RANK method outperforms the SUM and VOTE methods. Hence, for the remainder of this paper, we use the RANK method. Additionally, we observe that VOTE performs better where data values do not follow one distribution (insurance database), SUM performs better in classifying a set of similar distributions (biochemical database).

6.3 Choosing a Feature Set

As for categorical data, we must choose the features (quantiles) to use for classification. In Sec. 5 we proposed three variations of using quantiles—the first (normal) using all available data, the second using only non-zero data, the third as a combination of the first two. To decide which variation classifies most accurately, we performed tests on both real world databases. Figure 11(b) shows misclassification rates for different features, for equal training and test column sizes.

We observe that a combination of both normal and non-zero variations performs best. Interestingly, for the biochemical database the first two variations produce mediocre results but they complement each other, so that the combination gets good misclassification rates. This is because the database stores two types of attributes. The first type reflects some measured values (observed as curved lines in Fig. 10, left), the second type reflects category-type values with only 2 or 3 distinct values (observed as steps in Fig. 10, left). Each type profits from a different feature set for classification.

For the insurance database, the normal quantile alone produces poor results because of the large number of zero-values in the data (seen in Fig. 10, right, as many lines beginning with high cumulated value percentages). Only by disregarding the zero-values do we produce low misclassification rates.



(a) Choosing a classifier

(b) Choosing a feature set

Figure 11: Averaged misclassification results

6.4 Biochemical Database

The biochemical database is a large customer database with the results of experimental tests of various potential drug candidates against possible targets. We extracted 26 numerical columns from the main results table, consisting of the results of various experiments, status fields showing what stage of analysis or certification the results have reached, fields that cross-index other tables in the database for further descriptions of the experiments, time-stamps for various steps of the experiment, and so on. In each experimental run and each attribute, we chose different columns of training and test data and counted the number of misclassifications. We performed several hundred runs, depending on the stability of the results (standard deviation), and display the average misclassification results in Fig. 12.

A first observation is the high rate of misclassification when little training or test data is available. Only when both columns of data are larger than 1,000 does the classification achieve results comparable to that of the Naïve Bayes classification for categorical attributes, i.e., under 5%. The reason is that many of the numerical attributes are really used as categorical attributes. For instance, one attribute encodes a date as an integer; another acts as a boolean value, containing only 0 and 1 values. A classification of this categorical data using the Naïve Bayes classifier produced equally good results. The quantile method proves its effectiveness for "truly" numerical data.

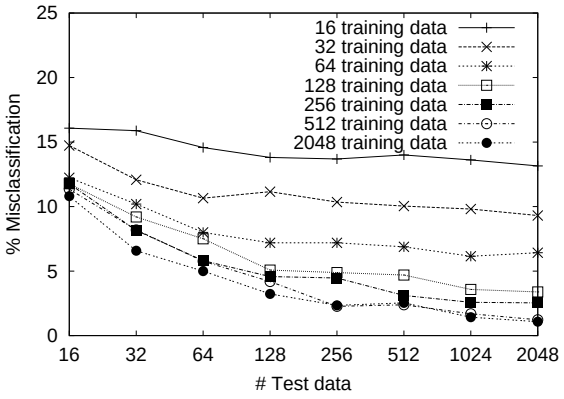


Figure 12: Biochemical database

As for categorical data, we also observe that the more training data we use, the lower the misclassification rate. Further, although additional test data also improves accuracy, it does not compensate for a smaller training column. Briefly, the misclassification rate is related to the variance of the statistics

of the training columns. Even when the variance of the same statistic in the test column is small, the misclassification rate is still bounded from below by a function of the sum of variances of the training columns (see Appendix A for details).

6.5 Insurance Database

The insurance database has a size of 2 Gigabytes with 19 tables and 141 numerical attributes total. After clustering (see Section 6.1) 61 attributes remain to be classified. Figure 13 shows the classification results for those attributes, obtained in the same manner as for the biochemical database. For test and training set sizes over 32, we observe very good misclassification rates below 2%.

In summary, the misclassification rates for real world databases are low enough to successfully employ the RANK method for attribute classification in Clio. To prove the virtue of these methods in more difficult settings, we test the methods on synthetically generated data values with only very subtle differences between attributes.

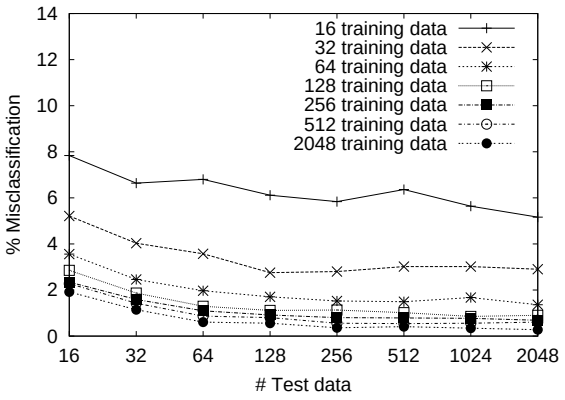


Figure 13: Misclassification for insurance database

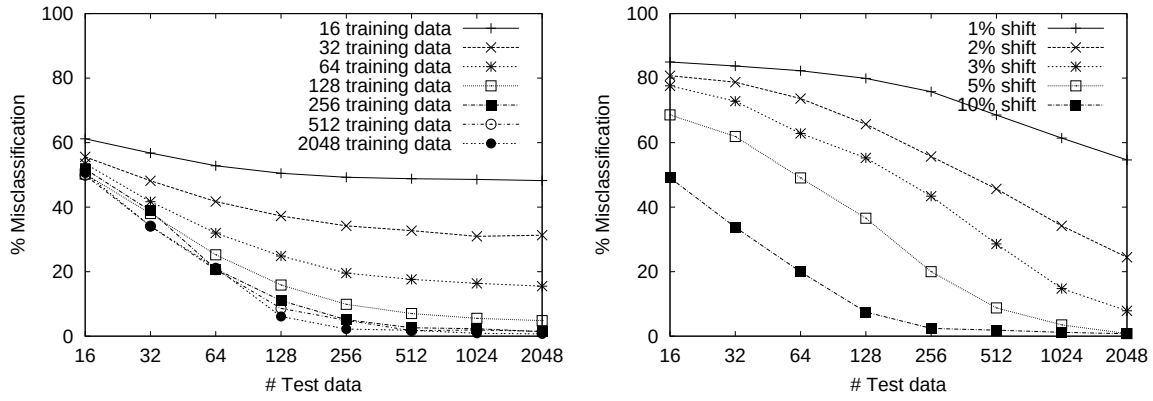
6.6 Uniformly Distributed Data

In this experiment we randomly generate data for 10 numerical attributes with increasingly shifted uniform distribution. That is, the ranges are $[0, n], [ns, n + ns], \cdots, [9ns, n + 9ns]$, respectively, where s is the *shift factor* and n was chosen as 100. In Fig. 14(a) we fix the shift factor as 10% and vary the training column size for different curves. As expected, the misclassification rates are higher than for the real world data, but are at an acceptable level for training and test column sizes above 128. The Naïve Bayes classification for this data failed, with all misclassification rates usually at a rate equivalent to random picks, and never better than 50%. This insight justifies the use of different classifiers for different types of attributes.

In Fig. 14(b) we fix the training column size for each of the ten attributes at 2048, and vary the shift factor s from 1% to 10%. Data with a lower shift are more similar, and hence, more difficult to classify. In this experiment the limitations of automatic classification become apparent: With a small amount of test data and with only slight variations in the values, automatic classification of attributes is sometimes reduced to mere guesswork.

6.7 Gaussian Distributed Data

The same experiments for Gaussian distributed data yielded very similar results. We generated data for ten Gaussian distributions, (μ, σ), where the mean $\mu = 50 + 100 \cdot i \cdot s$ for $i \in \{0, 1, \cdots, 9\}$ and shift



(a) Misclassification with shift factor 10%

(b) Misclassification with 2048 training data

Figure 14: Uniform distribution

factor s. We chose standard deviation $\sigma = 25$. For the first experiment we chose a fixed shift of 10% and varied the number of training data for each curve, for the second we fixed the training column size at 2048 and varied the shift factor.

In both experiments we observed the same behavior as for the uniform data distribution, i.e., good results for test and training column sizes above 128 with a 10% shift factor, and poor results for low shift factors.

7 Deployment in Clio

Currently, Clio has four deployment modes of the attribute matching techniques: correspondence discovery, dependency discovery, rapid table matching, and relation hiding.

Correspondence discovery. In a typical deployment of Clio, the user knows the target schema well but knows little about the source schemata. In particular, source attribute names may mean nothing to the user or may convey a different semantics than the one intended. To help, the user can ask Clio to find source attributes that are similar to a certain target attribute. Clio uses the attribute matching techniques to find the top matches and present them to the user, annotated with the confidence level (Fig. 15). The user can then browse the suggestions and accept the correct ones.

Dependency discovery. The overall goal of Clio is to create a mapping between a source schema and a target schema. This problem is equivalent to discovering a transformation query on the source schema that produces data for the target schema [MHH00]. Foreign key dependencies are used to suggest the join paths of the transformation queries. When these dependencies are not given as schema meta-data, Clio discovers them. Foreign key discovery is tedious, because one must expensively test every combination of attributes with same data types between two relations. Attribute pairs of the

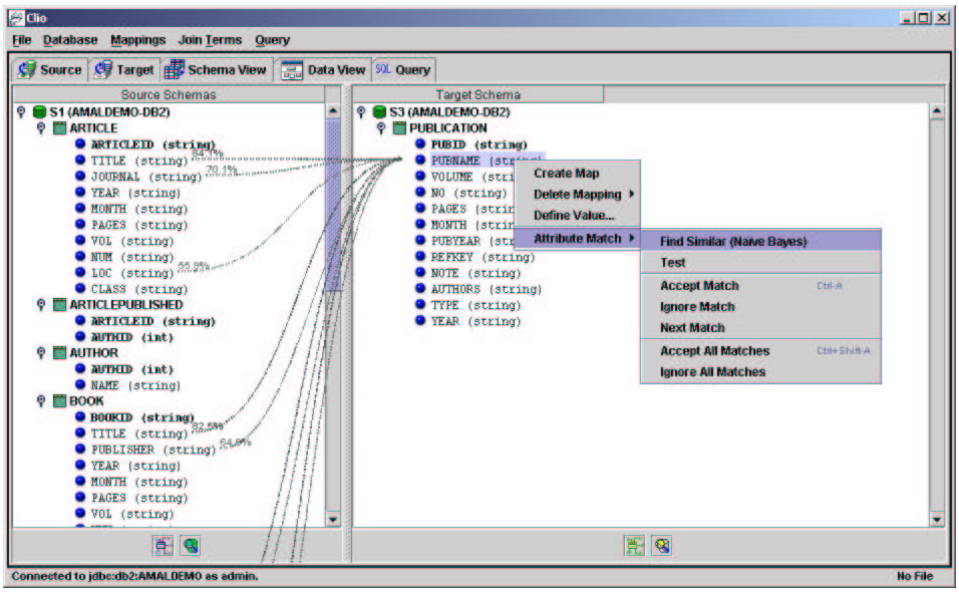


Figure 15: Schema Discovery with Clio

same source schema, discovered with our matching techniques, are good candidates for testing, and we thus save a large amount of computation and database access[5].

Rapid table matching. Because source databases and target database store similar information, tables in source and target often have similar attributes. Recognizing this, Clio relieves the user of having to tediously specify obvious matchings: After the user specifies one matching between two tables, Clio suggests the top matching attribute in the same source table for each remaining target attribute—up to a certain confidence threshold. If they are correct, the user can accept all suggestions at once (Fig. 16), rapidly matching an entire set of attributes.

Relation hiding. Clio aims at scenarios with possibly very large source schemata that the user is not familiar with. Users can be overwhelmed by the large number of relations and may not know where to start and on which part to focus. On the other hand, we assume some user familiarity with the target schema. Users can focus on a certain part of the target schema (e.g., a target relation), and Clio will hide source relations whose relevance to that part is below a certain threshold. Currently, relevance is calculated as the average matching confidence of all attributes in the target relation to the best match in the source relation. We are investigating more refined confidence aggregation.

8 Related Work

With the growing number of sources available for integration, the bottleneck of creating mappings from the source schemata to the target schema has been recognized by several projects. For instance, there

[5]Note that for this deployment we do not find matchings between source and target attributes. Rather, we compare only source attributes with one another.

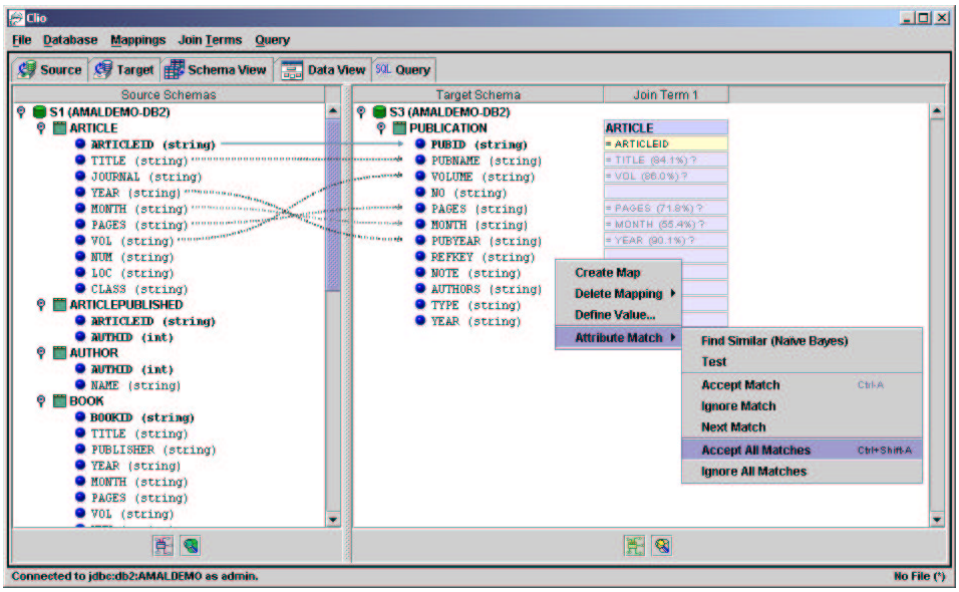


Figure 16: Rapid Table Matching with Clio

are tools to simplify the creation of such mappings in the form of wrappers [LPH00, SA99]. Several attempts have been made to automate this task (see [RB01]), including the work of Gibson et al. [GKR98], the rule-based schema matching approach of Milo and Zohar [MZ98], and the approach of Sciore et al. introducing a semantics-aware variation of SQL [SSR94]. The Cupid approach employs a tree matching algorithm to find similarities between two schemas [MBR01]. While projects like Cupid have the same goal as we do, i.e., mapping one schema to the other, we are aware of only two projects using not only meta-data (schema-based matching) but also the data to improve the mapping (instance-based matching):

Doan et al. describe the LSD system, which provides methods to match nodes of a schema tree derived from XML-documents [DDH01]. The authors use four classifiers—among them a Naïve Bayes classifier—whose results are combined to produce a final classification. The usage of the Naïve Bayes classifier differs in that the authors do not use alphanumeric features, as in our approach, but word frequencies, as often used to classify documents. The word-based approach does not adapt well to the attribute matching problem because data values rarely contain more than one or two words. However, their ability to combine different classification methods should allow them to include our new classification methods, promising even better results.

Li and Clifton present a neural network classifier for the same problem in the Semint prototype [LC95]. The authors encode two types of meta-data into a feature vector: (i) specification of attributes, such as data type, constraints, data formats, etc., and (ii) aggregate features determined from the data values, such as the average number of digits in a value. The authors do not use singleton features for classification, as we do—the features they extract from the data are complementary to ours. A neural network must be constructed of the training attributes under user supervision; test attributes are then classified using the network. In all experiments, similar attributes are clustered and treated as one, and the authors only compare columns taken from the same database. As expected in such a setup, the reported misclassification rates are low, but not comparable to ours.

9 Conclusion

The contributions of this paper are the application and adaptation of well known classification techniques, new types of features, and novel similarity measures to the problem of finding correspondences between attributes of different schemata. In particular, this work examined which classification techniques are most suitable for categorical and numerical values, which features are most discriminatory, and which training and test column sizes are necessary to reach satisfactory classification results.

To show the effectiveness of our approach, we applied them to well studied real world data sets and to synthetically created data. The misclassification results were very low, justifying deployment of the methods in the Clio system. There, users are successfully supported in finding correspondences between attributes.

For future work, we plan to extend these approaches to find not only corresponding attributes, but also corresponding tables. Other work will refine the feature selection and will include the ability to apply different weights to the features, reflecting different significance. With this work we have instantiated a further component of Clio, which further enhances the user's ability to perform schema transformations.

Acknowledgment. We wish to thank Mauricio Hernández, Renée Miller, Ramakrishnan Srikant, AnHai Doan, and Ling-Ling Yang for their helpful comments.

References

[AJS00] Rakesh Agrawal, Roberto J. Bayardo Jr. and Ramakrishnan Srikant. Athena: Mining-based interactive management of text database. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, volume 1777 of *Lecture Notes in Computer Science*, pages 365–379, Konstanz, Germany, 2000. Springer.

[AS01] Rakesh Agrawal and Ramakrishnan Srikant. On integrating catalogs. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 603–612, Hong Kong, 2001.

[Dam95] Marc Damashek. Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267:843 – 848, February 1995.

[DDH01] AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 509–520, Santa Barbara, CA, 2001.

[DP97] Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[GKR98] David Gibson, Jon M. Kleinberg, and Prabhakar Raghavan. Clustering categorical data: An approach based on dynamical systems. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 311–322, New York City, NY, 1998. Morgan Kaufmann.

[Goo65] Irving John Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods.* Research monograph no. 30. M.I.T. Press, Cambridge, MA, 1965.

[HMH+01] Mauricio A. Hernández, Renée J. Miller, Laura M. Haas, Lingling Yan, C.T. Ho, and Xuqing Tian. Clio: A semi-automatic tool for schema mapping. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2001. Demo.

[LC95] Wen-Syan Li and Chris Clifton. Semint: A system prototype for semantic integration in heterogeneous databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, page 484, San Jose, CA, 1995.

[LIK92] Pat Langley, Wayne Iba, and KevinThompson. An analysis of Bayesian classifiers. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI 92)*, San Jose, CA, 1992.

[LPH00] Ling Liu, Calton Pu, and Wei Han. XWRAP: An XML-enabled wrapper construction system for Web information sources. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 611–621, 2000.

[LR94] David D. Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Symposium on Document Analysis and IR*, Las Vegas, NV, 1994.

[MBR01] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with Cupid. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, Rome, Italy, 2001.

[MHH00] Renée J. Miller, Laura M. Haas, and Mauricio A. Hernández. Schema mapping as query discovery. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 77–88, Cairo, Egypt, 2000. Morgan Kaufmann.

[MN98] Andrew McCallum and Kamal Nigam. A comparison of event models for naïve Bayes text classification. In *Proceedings of the AAAI Workshop on Learning for Text Categorization*, Madison, WN, 1998.

[MZ98] Tova Milo and Sagit Zohar. Using schema matching to simplify heterogeneous data translation. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 122–133, New York City, NY, 1998.

[RB01] Erhard Rahm and Philip A. Bernstein. On matching schemas automatically. Technical Report MSR-TR-2001-17, Microsoft Research, Redmond, WA, Feb. 2001.

[SA99] Arnaud Sahuguet and Fabien Azavant. Building light-weight wrappers for legacy Web datasources using W4F. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 738–741, 1999.

[SSR94] Edward Sciore, Michael Siegel, and Arnon Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems (TODS)*, 19(2):254–290, 1994.

A Probabilistic Analysis of the Classifiers

Let X and Y be independent normal random variables with expectations μ_1 and μ_2, respectively, each with variance $1/N$, where N is the size of a sample. Intuitively, X and Y are sample means from two different distributions, each with variance 1. Furthermore, they could be the approximate values of sample means from binomial distributions. The connection to our problem is that X and Y represent training sample means from two distinct columns. Thus N is proportional to the size of the training sample. Let Z be a normal random variable independent of X and Y, with expectation μ_1 and variance $1/n$. The connection to our problem is that is a test sample mean from the same column as X, but in reality we do not know that and the goal is to decide whether Z is from the same column as X, or the same column as Y. Thus, n is proportional to the size of the test sample. We are interested in the probability of the event $A = \{|Z - X| < |Z - Y|\}$, i.e., the event in which Z turns out to be closer to X than to Y, in which case we would conclude correctly that Z is from the same column as X. The remainder of the analysis is aimed at understanding the dependence of the probability of A on the sample sizes.

Denote by E the event $\{X < Y\}$ and denote by F the event $\{Z < \frac{X+Y}{2}\}$. Thus, $\Pr(A) = \Pr(E)\Pr(F|E) + (1 - \Pr(E))\Pr(\bar{F}|\bar{E})$. It is well-known that $X + Y$ and $X - Y$ are independent, hence

$$\Pr(A) = \Pr(E)\Pr(F) + (1 - \Pr(E))(1 - \Pr(F)) \ .$$

Since $X - Y$ is normal with expectation $\mu_1 - \mu_2$ and variance $\frac{2}{N}$,

$$\Pr(E) = \Phi\left(\sqrt{N} \cdot \frac{\mu_1 - \mu_2}{\sqrt{2}}\right) \ .$$

Since $Z - \frac{X+Y}{2}$ is normal with expectation $\frac{\mu_1 - \mu_2}{2}$ and variance $\frac{1}{2N} + \frac{1}{n}$,

$$\Pr(F) = \Phi\left(\frac{1}{\sqrt{\frac{1}{2N} + \frac{1}{n}}} \cdot \frac{\mu_2 - \mu_1}{2}\right) \ .$$

Denote $\delta = \frac{\mu_2 - \mu_1}{\sqrt{2}}$. Thus,

$$\Pr(A) = \Phi(\sqrt{N} \cdot \delta)\Phi(\frac{1}{\sqrt{\frac{1}{N} + \frac{\sqrt{2}}{n}}} \cdot \delta) + \Phi(-\sqrt{N} \cdot \delta)\Phi(-\frac{1}{\sqrt{\frac{1}{N} + \frac{\sqrt{2}}{n}}} \cdot \delta)$$

It is easy to see that for any fixed n, $\Pr(A)$ tends to 1 as N tends to infinity, whereas if N is fixed, then as n tends to infinity, $\Pr(A)$ tends to

$$[\Phi(\sqrt{N} \cdot \delta)]^2 + [\Phi(-\sqrt{N} \cdot \delta)]^2 \ .$$