# Autoplex: Automated Discovery of Content for Virtual Databases

Jacob Berlin and Amihai Motro

Information and Software Engineering Department
George Mason University, Fairfax, VA 22030
{jberlin, ami}@gmu.edu

**Abstract.** Most virtual database systems are suitable for environments in which the set of member information sources is small and stable. Consequently, present virtual database systems do not scale up very well. The main reason is the complexity and cost of incorporating new information sources into the virtual database. In this paper we describe a system, called Autoplex, which uses machine learning techniques for automating the discovery of new content for virtual database systems. Autoplex assumes that several information sources have already been incorporated ("mapped") into the virtual database system by human experts (as done in standard virtual database systems). Autoplex learns the *features* of these examples. It then applies this knowledge to new candidate sources, trying to infer views that "resemble" the examples. In this paper we report initial results from the Autoplex project.

## 1 Introduction

The integration of information from multiple databases has been an enduring subject of research for over twenty years (for example, [16, 7, 10, 5, 21, 27, 26, 2, 11]). Indeed, while the solutions that have been advanced tended to reflect the research approaches prevailing at their time, the overall goal has remained mostly unchanged: to provide flexible and efficient access to information residing in a collection of distributed, heterogeneous and overlapping databases (more generally, other kinds of information sources may be considered as well).

A common approach to this problem has been to integrate the independent databases by means of a comprehensive *global scheme* that models the information contained in the entire collection of databases. This global scheme is fitted with a *mapping* that defines the elements of the global scheme in terms of elements of the schemes of the member databases. Algorithms are designed to interpret queries on the global scheme. Such *global queries* are translated (using the information captured in the mapping) to queries on the member databases; the individual answers are then combined to an answer to the global query. The global scheme and the scheme mapping constitute a *virtual database*; the main difference between a virtual database and a conventional database is that whereas a conventional database contains data, a virtual database points to other databases that contain the data (Figure 1). An important concern is that this

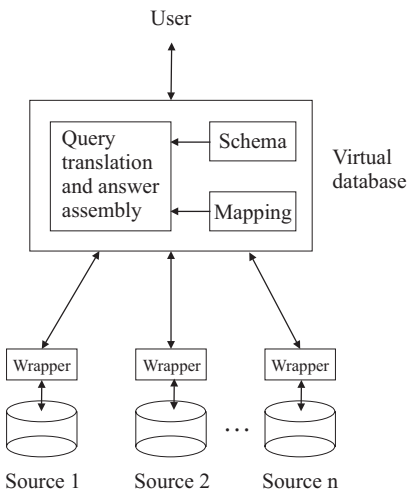


Fig. 1: Typical architecture for integrating heterogeneous information sources.

query processing method be *transparent*; i.e., users need not be aware that the database they are accessing is virtual.

Although there are a number of recent systems that follow this general scheme, for example [4, 13, 22], none of these systems scale up to an environment in which the number of potential sources is very large, and which is constantly changing (such as the World Wide Web). The primary limitation is that the process of incorporating new member schemes into the global scheme is complex and costly. Consequently, such systems tend to be useful only when the community of member databases is small and stable. Indeed, it is commonly agreed upon in this field and the related field of data warehousing that future research should find ways to automate the integration and maintenance process [14, 25].

Given the vast amount of information available and the cost of locating and incorporating such information into a virtual database, we have been developing a system, called Autoplex, for *discovering* member schemes and incorporating them into the global scheme with only limited human effort. Based primarily on Bayesian learning, the system acquires probabilistic knowledge from examples that have already been integrated into the virtual database. Our approach thus follows a supervised learning paradigm. From the acquired probabilistic knowledge, the system can discover "content contributions" in new, previously unseen information sources. Although not treated in detail in this paper, we believe this approach can also be used to maintain the contribution definitions in a dynamic environment where the underlying schemes may change over time.

The authors are unaware of any prior work that attempts to automate the discovery and mapping of new data sources. Two recent works, however, share with Autoplex the general goal of accelerating the mapping process. In [15] a neural network-based method is described, that classifies attributes of data sources. This important integration step is also part of the Autoplex discovery process.

The recent Clio system [18] introduces an interactive process that facilitates the mapping of a given source (such as a legacy database) to a target schema. However, mappings are derived from prespecified source-target relationships (called value correspondences). The translation of heterogeneous data is also the subject of [1, 19]. These approaches, too, are based on prespecified correspondence rules. It must be mentioned that file translation, schema mapping, and database restructuring are ancient database problems, with seminal work, often-ignored, done over 25 years ago (for example, [17, 24, 23]).

The remainder of this paper is organized as follows. Section 2 states the problem formally, and provides an overview of the architecture. Section 3 discusses how probabilistic knowledge is acquired from the examples. Section 4 discusses how that knowledge is used to discover a contribution from a new source. Section 5 describes the experimental environment and initial results. Finally, Section 6 concludes with a brief discussion of proposed future work. A fuller discussion may be found in [6].

## 2 Basic Issues and Assumptions

In this section we describe the framework of this project, and we outline the overall architecture of Autoplex. Our work is conducted within the framework of the Multiplex virtual database system, but is of general applicability to most such systems. We begin with a brief description of Multiplex.

### 2.1 Multiplex

The multidatabase system Multiplex [22] is an example of a virtual database system. The basic architecture of Multiplex is fairly simple. The virtual database consists of a *global scheme* (described in the relational model) and a *mapping table*. Each entry in this table is called a *contribution* and consists of two expressions. The first expression is a view (expressed in SQL) of the global scheme; the second expression is a query to one of the member databases (expressed in the language of that system). The first expression is called a *global view*; the second expression is called a *local view*. The result obtained from the second expression is assumed to be a materialization of the view described in the first expression. The complexity of these expressions can vary greatly: they could range from a complex calculation, to a statement that simply denotes the equivalence of two attribute names.

### 2.2 Statement of the Problem

In its most general form, a contribution is a pair of arbitrary view expressions: one on a member scheme, the other on the global scheme. For reasons of complexity, Autoplex places limitations on these expressions. Specifically, it assumes that the global view is a single global relation, and that the local view is a *selection-projection* expression on a single relation. The challenge of Autoplex can be stated in terms of the Multiplex system as follows. *Given:*

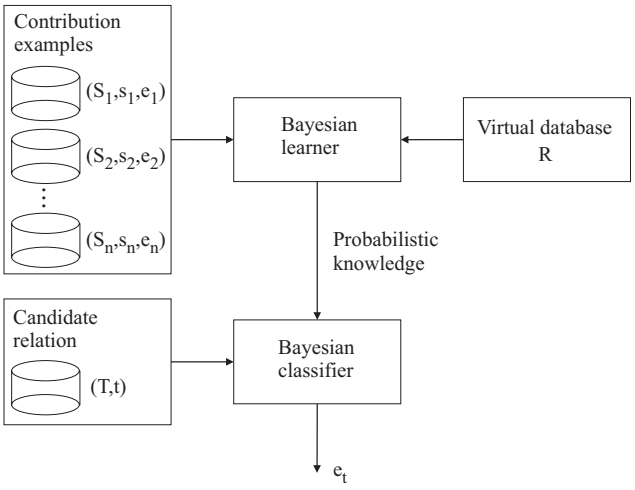


Fig. 2: Autoplex architecture.

1. A relation scheme $R = (X_1, \ldots, X_n)$. This is the virtual database. Each column $X_i$ of $R$ is labeled as either *required* or *optional*.
2. A set of contribution examples, each consisting of a relation scheme $S = (Y_1, \ldots, Y_k)$, a relation instance $s$ of scheme $S$, and a selection-projection expression $e$ on relation $S$ that defines a contribution to $R$.
3. A new, previously unseen relation scheme $T = (Z_1, \ldots, Z_m)$ and a relation instance $t$ of $T$. We shall often refer to $T$ as a *candidate* relation.

*Determine:* Whether $T$ contains an acceptable contribution to $R$, and if so, find the expression $e_t$ that defines it. An acceptable contribution is one that satisfies all required columns in $R$ and exceeds a predetermined threshold.

### 2.3 Autoplex Architecture

A high level overview of the Autoplex architecture is shown in Figure 2. This architecture includes two main components: a learner and a classifier.

**The learner.** The Bayesian learner is given the virtual relation scheme $R$ and a set of contribution examples. Each such example consists of a local relation scheme $S$, an instance of this scheme $s$, and a selection-projection expression $e$ on scheme $S$. The extension of this expression in the instance $s$ generates tuples for the virtual relation. The Bayesian learner uses this information to acquire probabilistic knowledge on features of the examples. This knowledge is stored on secondary storage in efficient data structures for future use.

**The classifier.** A candidate relation scheme $T$ and instance $t$ are provided by a new local database as inputs to the Bayesian classifier. This classifier uses the acquired probabilistic knowledge to infer a selection-projection view that defines a contribution of $T$ to $R$.

### 2.4 Classification Methodology

In Figure 2, classification is a single process. More precisely, classification comprises four different phases. Given a candidate relation $T$ and a global relation $R$, in the first phase the classifier considers each column of $T$ and each column of $R$ and determines the probability that the former is an "instance" of the latter. In the second phase, the classifier finds an assignment of the local relation to the global relation that maximizes total column probabilities. At this point, if successful, the classifier has found a projection of the candidate relation that offers the best "match."

The rows of this projection now must be pruned to retain only those rows that "resemble" rows in the examples. In the third phase, the classifier partitions the instance $t$ into two sets of rows: those that should be included in the contribution and those that should be excluded from it. Because the new contribution should be usable even after the extension of the candidate relation is updated, an *intensional* description of the included rows is desirable. In phase four, a classification tree algorithm is used to derive a selection predicate that conforms to the set of included rows.

The final output from this entire process is a selection-projection expression, or *False* in the event an acceptable contribution could not be found. Our approach is biased to search within the space of projections and selections on the candidate relation. Furthermore, our approach is greedy in that we search for a projection followed by a selection. Clearly, other approaches could produce better results. For example, a better projection may be found if some rows are first removed from the candidate. Furthermore, a better mapping may be found if we allow *transformations* of the values in the candidate (such as conversions of measurement units). Our approach represents a reasonable tradeoff between the advantage of more a powerful search (which will discover additional contributions) and the need to keep the problem tractable. More general mappings are currently under investigation (see Section 6).

## 3 Learning from Example Contributions

Learning from example relations is accomplished in two stages. First, we acquire probabilistic knowledge on the "behavior" of columns (to be used in determining the projective transformation of a candidate relation). Next, we acquire probabilistic knowledge on the "behavior" of rows (to be used in determining the optimal selective transformation of a candidate relation). In this section, we address both of these procedures in order. We begin with a brief survey of the Bayesian concepts that will be used throughout this work.

### 3.1 Bayesian Framework

Our discussion is mostly conventional in the application of Bayes Theorem to machine learning. In machine learning terminology, the problem of associating

a new instance with one of previously learned classes is called *classification*. Let $P(c)$ be the *prior* probability that classification $c$ holds before observing any data, let $P(D)$ represent the unconditional probability of observing data $D$, and let $P(D|c)$ represent the conditional probability of observing the data $D$ given that the classification $c$ holds. Bayes Theorem states:

$$P(c|D) = \frac{P(D|c)P(c)}{P(D)} \ . \tag{1}$$

$P(c|D)$ is referred to as the *posterior* probability of $c$, because it reflects the probability of classification $c$ after the data $D$ has been observed. In machine learning problems, we wish to find the best classification. Translated into Bayesian terms, we wish to find the classification with the greatest posterior probability given the data. Since the probability of the data $P(D)$ is common for all classifications, the most probable classification is the one that maximizes the numerator in Equation 1. Letting $D$ be a sequence of attribute values $(d_1, d_2, \ldots, d_n)$ and $C$ a set of possible classifications $(c_1, c_2, \ldots, c_m)$, we wish to determine the classification $c_i \in C$ such that $\forall c_j \in C$ and $c_j \neq c_i$,

$$P(d_1, d_2, \ldots, d_n|c_i)P(c_i) \geq P(d_1, d_2, \ldots, d_n|c_j)P(c_j) \ . \tag{2}$$

The terms in Equation 2 do not produce probabilities since we have removed the denominator; however, the terms can be easily converted to probabilities by normalization. For our purposes, we shall make the *naive* assumption that data values are conditionally independent given the classification. This assumption is the basis for the well-known Naive Bayes classifier. Both theoretical and experimental results show that even when the independence assumption does not hold, the Naive Bayes classifier performs comparable to or better than other more sophisticated approaches in many problem domains [20, 12, 8]. The key to this paradox is that good classifications can be made from this approach even when the estimated probabilities are wrong. From the independence assumption, we can rewrite Equation 2 as

$$P(c_i) \prod_{k=1}^{n} P(d_k|c_i) \geq P(c_j) \prod_{k=1}^{n} P(d_k|c_j) \ . \tag{3}$$

Equation 3 is useful in our problem domain for several reasons. First, since we assumed conditional independence, the number of probability terms that we need to estimate has been reduced to a manageable level. Second, the equation is robust to noise since observed data only causes incremental changes to our estimates. Finally, the equation is robust to missing features since we simply ignore the conditional probability terms for the missing features. All of these aspects of Equation 3 make a Bayesian approach towards automated discovery of contributions attractive.

### 3.2 Learning Column Behavior

Our strategy is to learn the "behavior" of each column $X$ of $R$ and, upon encountering a new column $Z$, determine the posterior probability that the new

column is in the class of $X$, given the data values of $Z$. For this, we use a collection of Naive Bayes (NB) classifiers: one classifier for each column $X$. Building a collection of classifiers instead of just one gives us the ability to determine that a candidate column should *not* be mapped to any of the columns of $R$ or that it may be mapped to *more* than one column of $R$.

To construct a classifier for column $X$ we need to learn the prior probability $P(X)$ that an arbitrary column maps to $X$, and, for each column, the conditional probability of that column among the columns that were mapped to $X$. Following our assumption of conditional independence among the values of a column, the latter is substituted by the conditional probability $P(v|X)$ that an *individual* column value $v$ occurs in the set comprising the values in columns that were mapped to $X$. Together, these will allow us to calculate the right hand side of Equation 3 and subsequently the best classification for the column $Z$. The learning algorithms are discussed next.

**Learning Prior Probabilities** $P(X)$, the probability that an arbitrary column maps to $X$, is estimated by the proportion of columns in the entire set of examples that have been mapped to $X$.

**Learning Conditional Probabilities** In estimating conditional probabilities we distinguish between values that are *words* and values that are *numbers*. The algorithm for learning conditional probabilities for words is shown in Table 1. This algorithm is similar in spirit to the typical Naive Bayes approach for document classification discussed in [20]. For a text column $X$, the positive examples come from all the text columns of examples that are mapped to $X$. The negative examples for the same $X$ come from all the text columns of examples that are not mapped to $X$.

$P(v|X)$, the probability that a vocabulary word $v$ occurs in a column mapped to $X$, is estimated by the proportion of the occurrences of $v$ among words from columns that are mapped to $X$. This ratio is adjusted with the $m$-estimate to prevent zero probability terms which would dominate all calculations. We assume uniform priors for the words and weight them by the size of the vocabulary.

To estimate the probabilities of numeric values, we assume that these values conform to some distribution function; currently, a normal distribution is assumed. The learning phase consists simply of calculating the mean and standard deviation for the positive and negative examples. For a numeric column $X$, the positive examples come from all the numeric columns of examples that are mapped to $X$. The negative examples for the same $X$ come from all the numeric columns of examples that are not mapped to $X$. Given a numeric value from a candidate column, we can now calculate the conditional probabilities by using the probability density function for the normal distribution.

### 3.3 Learning Row Behavior

Our learning process is similar to that for column behavior, except that now our evidence is rows of data instead of columns of data. In the learning phase we shall

Table 1: Algorithm for learning column conditional probabilities.

$V \leftarrow$ All distinct words in all the example relations $S$
For each text column $X$ in $R$:
    Initialize bag variables $P$ and $N$ to $\emptyset$.
    For each text column $Y$ in all example relations $S$:
        If $Y$ is mapped to $X$
            $P \leftarrow P \cup \{$All words in $Y\}$
        else
            $N \leftarrow N \cup \{$All words in $Y\}$
    For each word value $v$ in $V$:
        $C_P \leftarrow$ Number of times $v$ appears in $P$
        $C_N \leftarrow$ Number of times $v$ appears in $N$
        $P(v|X) \leftarrow \frac{C_P+1}{|P|+|V|}$
        $P(v|\neg X) \leftarrow \frac{C_N+1}{|N|+|V|}$

examine the rows of each example relation $S$ (noting which rows were selected and which were discarded). Upon encountering a new table (whose columns have already been matched successfully to the columns of $R$) we shall use our acquired knowledge to determine, for each row, whether it should be selected or not.

To determine whether a row of a future table should contribute to $R$, we need to learn the prior probability $P(R)$ that an arbitrary row is a contributor to $R$, and, for each row (a *sequence* of values), the conditional probability of that row within the set of selected rows. Again, following our assumption of conditional independence among the components of a row, we learn instead the conditional probabilities $P(v|R)$ that an *individual* row component $v$ occurs in the set comprising the values taken from the same column position in all the selected rows. Together, these will allow us to calculate the right hand side of Equation 3 and subsequently the appropriate classification for each row.

**Learning Prior Probabilities** $P(R)$, the probability that an arbitrary row would contribute to $R$, is estimated by the proportion of rows in the entire set of examples that were selected.

**Learning Conditional Probabilities** For textual columns, we use the algorithm in Table 2 to learn the conditional probabilities of words. This algorithm is similar to the one in Table 1, except that now we process rows of data. For numeric columns, we again assume a normal distribution on the values and calculate the mean and standard deviation for the examples. For both textual and numerical values, positive examples are drawn from the set of rows that satisfy the selection predicate of the contribution. Negative examples are drawn from the set of rows that do not satisfy the selection predicate.

Table 2: Algorithm for learning row conditional probabilities.

For each text column $X$ of $R$:
  $V \leftarrow$ All distinct words from all example columns mapped to $X$
  Initialize bag variables $P$ and $N$ to $\emptyset$.
  For each example $S$ and its selection predicate $\sigma$:
    For each row $t$ from $s$:
      For each column $Y$ of $S$ that was mapped to $X$:
        If $t$ satisfies $\sigma$
          $P \leftarrow P \cup t[Y]$
        else
          $N \leftarrow N \cup t[Y]$
  For each word value $v$ in $V$:
    $C_P \leftarrow$ Number of times $v$ appears in $P$.
    $C_N \leftarrow$ Number of times $v$ appears in $N$.
    $P(v|R) \leftarrow \frac{C_P+1}{|P|+|V|}$
    $P(v|\neg R) \leftarrow \frac{C_N+1}{|N|+|V|}$

## 4 Discovering New Contributions

Given a candidate relation, discovery is accomplished in two phases. First we search for a projective transformation of the candidate relation to match the global relation $R$. If successful, we then search for a selective transformation to be applied subsequently. We shall discuss each of these procedures in order.

### 4.1 Projective Transformations

Discovering a projective transformation of a candidate relation is a two-step process. For each column of the candidate relation and for each column of the virtual relation we use a Naive Bayes (NB) classifier to estimate the probability that the columns match. This information is represented in a bipartite weighted graph, and a graph algorithm is applied to find the optimal overall matching of the candidate relation to the virtual relation. This optimal mapping is our projective transformation.

The process for estimating probabilities is shown in Table 3. Because we assume conditional independence among the values (even when this assumption is inaccurate), the outcome of this algorithm may not be strictly the probability of a match; hence, we use the notation $\hat{P}$.

The output of the NB mapping algorithm is a bipartite graph in which the columns of $T$ and $R$ are represented by nodes in two partitions, each column mapping is represented by an edge and the mapping probabilities are edge weights. The simplest way to approach optimality is to look for a maximum weighted matching in this bipartite graph [9]. After removing "weak" edges that fall below a user specified threshold, we search the graph for a subset of the edges that

Table 3: Estimating the probabilities for columns of $T$ matching columns of $R$.

For each column $X$ in $R$:
  For each column $Z$ in a candidate relation $T$:
    Let $v_1, \ldots, v_n$ be the values of $Z$.
    $\hat{P}(match) = P(X) \prod_i P(v_i|X)$
    $\hat{P}(\neg match) = P(\neg X) \prod_i P(v_i|\neg X)$
    Normalize $\hat{P}(match)$ and $\hat{P}(\neg match)$ to sum to 1.
    Output $\hat{P}(match)$ as probability that $Z$ maps to $X$.

Table 4: Estimating the probabilities for rows of $T$ contributing to $R$.

For each row $t$ in a candidate relation $T$:
  $\hat{P}(R|t) = P(R) \prod_i P(v_i|R)$
  $\hat{P}(\neg R|t) = P(\neg R) \prod_i P(v_i|\neg R)$
  Normalize $\hat{P}(R|t)$ and $\hat{P}(\neg R|t)$ to sum to 1.
  Output $\hat{P}(R|t)$ as probability that row $t$ contributes to $R$.

connects nodes in one partition to the nodes in the other partition such that no two edges share a node and the sum of the weights is maximal. The classifier uses a polynomial-time algorithm [3] to find the maximum weighted matching. This matching is then translated into a relational algebra expression that defines the projective contribution of $T$ to $R$.

### 4.2 Selective Transformations

Discovering a selective transformation of a candidate relation is also a two-step process. First, we partition the rows of the candidate into a set of contributing rows and a set of non-contributing rows, and we label the rows accordingly. Next, we apply a classification tree algorithm to the labeled rows, to learn a set of rules that *defines* the partition; these rules are then converted to a selection predicate. A more detailed discussion follows.

**Labeling Rows of a Candidate Table** Given a candidate relation, we use the acquired probabilistic knowledge to partition its set of rows to those that should be selected and those that should be discarded. We use the algorithm in Table 4 to estimate the probability that a candidate row should be included as a contribution to $R$. Rows that have a probability greater than a user-specified threshold are labeled as members of the contributing set; the remaining rows are labeled as members of the non-contributing set.

**Inferring Selection Predicates** We use a standard classification tree algorithm (J48 from the WEKA machine learning package [28]) to find a selection

predicate that defines the contributing set of rows. Before applying the candidate rows to the classification tree learner, some preprocessing is necessary to initialize the learning algorithm with the column names, column types, and the set of values for each column.

The learned classification tree represents a disjunction of conjunctive rules on the column values. These rules partition the data according to their labels. Since the classification tree considers all of the columns of the candidate, it is likely that the tree will use the unmapped columns of the candidate in the set of rules. This is interesting because it allows us to define selection predicates on columns that are neither in the virtual database nor in the example contributions.

For our problem, we are interested in the rules that define the contributing rows. Through simple string parsing, we build a selection predicate as a disjunction of the conjunctive rules that identify contributing rows.

## 5 Experimentation

We built a prototype in the Java programming language to test the ideas in this paper. This prototype is not yet integrated with a complete virtual database system, such as Multiplex. Specifically, the experimental data was collected offline from the World Wide Web and stored locally in relational database tables.

### 5.1 Measures of Performance

To measure performance, the outputs of Autoplex are regarded as four types of Boolean decisions:

1. *Column Mapping:* For each combination of a candidate column and a virtual column, decide whether or not the columns match.
2. *Table Mapping:* For each combination of a candidate table and a virtual table, decide whether or not the tables match.
3. *Tuple Partitioning:* For each tuple in a candidate table, decide whether to assign it to the contributing set or to the non-contributing set.
4. *Tuple Selection:* After we infer a selection predicate from the partitioned tuples, decide for each tuple whether or not it satisfies the selection predicate.

Obviously, the last two decisions are made only if we have made a positive table mapping decision. For each type of decision, the output falls into four disjoint categories:

*A.* Decision is *True* and the correct answer is *True* (*true positives*).
*B.* Decision is *False* and the correct answer is *True* (*false negatives*).
*C.* Decision is *True* and the correct answer is *False* (*false positives*).
*D.* Decision is *False* and the correct answer is *False* (*true negatives*).

The ratio $|A|/(|A| + |C|)$ is the proportion of true positives among the cases thought to be positive; i.e. it measures the accuracy of Autoplex when it decides *True*. The ratio $|A|/(|A| + |B|)$ is the proportion of positives detected by Autoplex among the complete set of positives; i.e. it measures the ability to detect positives. Specifically to our application, the former ratio measures the *soundness* of the content that has been discovered, and the latter ratio measures the *completeness* of the discovery process. These two ratios are known from the field of information retrieval as *precision* and *recall*, but we shall refer to them here as the soundness and completeness of the discovery process. Thus, we can measure the soundness and completeness of column mapping, table mapping, tuple partitioning, and tuple selection.

### 5.2 Setting Up the Experiment

To experiment with the prototype, we defined a virtual database for computer retail information with the following relations:

1. Desktops = (*Retailer, Manufacturer, Model*, Cost, Availability)
2. Monitors = (*Retailer, Manufacturer, Model*, Cost, Availability)
3. Printers = (*Retailer, Manufacturer, Model*, Cost, Availability)

Italicized attributes denote primary keys. The *Retailer* attribute is derived from the information source (for example, we use the web address). For the purposes of our experiment, the primary keys are *required* for a candidate to be accepted as a contribution to the virtual database. All other fields are *optional*.

Data for this experiment was taken from the web sites of 15 different computer retailers (e.g. Gateway, Egghead, etc). The data was collected off-line from HTML web pages and imported into relational database tables accessible through the ODBC protocol. The data from each retailer was imported into a single local table and then mapped to one or more virtual tables through selective and projective transformations.

To experiment with this data, we used a procedure from data mining called *stratified threefold cross-validation* [28], which we briefly describe. Each of the 15 web sources was manually mapped into our virtual database by using a mapping table as discussed in Section 2.1. We partitioned the 21 mappings in our mapping table into three folds of approximately equal content. Using two folds for learning and one fold for testing, we repeated the experiment for the three possible combinations of folds. To measure the soundness and completeness of the discoveries, the information in the mapping table was assumed to be the correct mapping of these sources.

### 5.3 Results

Table 5 shows the soundness and completeness for the four types of decisions made by Autoplex. A perfect sequence of decisions would result in $Soundness = Completeness = 1$. It is interesting to note that soundness and completeness

Table 5: Soundness and completeness for column mapping decisions.

| Category | A | B | C | D | Soundness | Completeness |
|---|---|---|---|---|---|---|
| Column Mapping | 74 | 17 | 17 | 660 | 0.81 | 0.81 |
| Table Mapping | 18 | 3 | 0 | 24 | 1.00 | 0.86 |
| Tuple Partitioning | 969 | 60 | 117 | 612 | 0.89 | 0.94 |
| Tuple Selection | 967 | 62 | 104 | 625 | 0.90 | 0.94 |

for column mapping are approximately equal. This behavior is caused by the matching constraint of the weighted bipartite graph algorithm (no two edges may share a node). Due to this constraint, an incorrect decision to map two columns typically causes both a false positive and a false negative.

Soundness and completeness for table mapping are higher than column mapping due to our constraint that all required columns of the virtual table must be mapped to make a positive table mapping decision. Given a candidate table and a virtual table that should not be mapped, we must determine that only one of the required virtual columns should not be mapped to any of the columns in the candidate to prevent a false positive. Thus, false positives in column mapping do not entail false positives in table mapping. Furthermore, optional columns that are not mapped do not cause a false negative for table mapping.

Table 5 also shows the performance for tuple selection is slightly better than the performance for tuple partitioning. This improvement is due to the pruning strategy employed by the J48 classification tree algorithm that learns selection predicates. Pruning prevents the overfitting of noise in partitioned sets of tuples. This strategy works well in our experiment because we have errors in the partitioned sets of tuples that are used for classification tree learning.

## 6 Conclusion

In this paper we described a novel approach to virtual databases, aimed at solving a serious limitation of all such systems: the cost and complexity of incorporating new data sources into the global system. This limitation hampers scalability, in effect restricting the virtual database paradigm to applications in which the community of sources is relatively small and stable.

The results of our initial experimentation are encouraging enough to support our main thesis of automatic discovery of content for virtual databases. Among the many research issues that are on our agenda, we discuss briefly four issues.

**Support more general views.** The ideal virtual database system can map a local source to a global database by matching arbitrary views of the local scheme with arbitrary views of the global scheme. In the version of Autoplex we described, the local view is a selection-projection of a single relation, and the global view is simply a relation. Our first priority is to support more general views, and we mention here two examples: (1) Allow local and global views that involve *joins*; i.e., discover content in a join of two *local* relations, and discover content for a join of two *virtual* relations. (2) Discover content that

becomes suitable for a virtual database after an appropriate *transformation*; e.g., a local column would be mapped if it matches a virtual column after a linear transformation (such as the conversion of Fahrenheit to Celsius).

**Use intensional information.** The features considered in this paper were purely extensional. Yet, intensional information, such as *integrity constraints* on the virtual database, could be used to improve the discovery process. Roughly speaking, with extensional features, discoveries are based on "similarity" to example data. With intensional features, discoveries would also be based on the satisfaction of constraints. We are confident that the future incorporation of intensional features will improve the performance of Autoplex.

**Assurance** Contributions in Autoplex are adopted with different levels of assurance in their suitability. These levels of assurance should be remembered so that answers to database queries could be annotated accordingly. The initial work in this research focused on the statistical performance of the discovery process using standard cross-validation techniques. This work will be extended to combine the statistical performance of the discovery process with confidence measures of individual discoveries to produce a meaningful assurance measure.

**Application to data warehouses.** The premise of virtual databases is closely related to that of *data warehouses*, in that both create global repositories that integrate data from multiple, heterogeneous data sources. The techniques described in this paper have the potential for developing a type of data warehouse, which (after a preliminary phase of design and learning) is populated and maintained automatically by "crawlers" that periodically visit the data sources.

## References


1. S. Abiteboul, S. Cluet, and T. Milo. Correspondence and translation for heterogeneous data. *Proc ICDT*, pages 351–363, 1997.
2. R. Ahmed, P. De Smedt, W. Du, W. Kent, M. A. Ketabchi, W. A. Litwin, A. Rafii, and M. C. Shan The Pegasus heterogeneous multidatabase system. *IEEE Computer*, 24(12):19–27, 1991.
3. Algorithmic Solutions. *The LEDA Users Manual (Version 4.2.1)*, 2001.
4. Y. Arens. Query Reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6:99–130, 1996.
5. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *Computing Surveys*, 18(4):323–364, Dec 1986.
6. J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. Technical Report ISE-TR-00-04, George Mason University, August 2000.
7. U. Dayal and H. Hwang. View definition and generalization for database integration in a mutlidatabase system. *IEEE ToSE*, SE-10(6):628–644, November 1984.
8. P. Domingos and M. Pazzani. Conditions for the optimality of the simple Bayesian classifier. *Proc ICML*, pages 105–112, 1996.
9. Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys*, 18(1):23–38, March 1986.
10. D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM ToIS*, 3(3):253–278, July 1985.
11. Y. Kambayashi, M. Rusinkiewicz, and A. Sheth, editors. *Proc RIDE-IMS*, 1991.

12. P. Langley, W. Iba, and K. Thompson. An Analysis of Bayesian Classifiers. *Proc of the Tenth National Conference on AI*, pages 223–228, 1992.
13. A. Levy. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5(2):121–143, September 1995.
14. A. Levy, C. Knoblock, S. Minton, and W. Cohen. Information integration. *IEEE Intelligent Systems*, 13(5):12–24, 1998.
15. W-S. Li and C. Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proc VLDB*, pages 1–12, 1994.
16. W. Litwin. MALPHA: A relational multidatabase manipulation language. In *Proc ICDE*, pages 86–93, 1984,
17. A. G. Merten and J. P. Fry. A data description language approach to file translation. In *Proc of ACM-SIGFIDET*, 1974.
18. R. Miller, L. Haas, and M. Hernández. Schema mapping as query discovery. *Proc VLDB*, pages 77–88, 2000.
19. T. Milo and S. Zohar. Using schema matching to simplify heterogeneous data translation. *Proc VLDB*, pages 122–133, 1998.
20. T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
21. A. Motro. Superviews: Virtual integration of multiple databases. *IEEE Transactions on Software Engineering*, SE–13(7):785–798, July 1987.
22. A. Motro. Multiplex: a formal model for multidatabases and its implementation. *Proc NGITS*, pages 138–158, 1999.
23. S. B. Navathe and J. P. Fry. Restructuring for large databases: three levels of abstraction. *ACM ToDS*, 1(2), June 1976.
24. J. A. Ramirez, N. A. Rin, and N.S. Prywes. Automatic generation of data conversion programs using a data description language. In *Proc ACM-SIGFIDET*, 1974.
25. E. A. Rundensteiner, A. Koeller, and X. Zhang. Maintaining data warehouses over changing information sources. *Communications of the ACM*, 43(6):57–62, 2000.
26. P. Scheuermann, C. Yu, A. Elmagarmid, H. Garcia-Molina, F. Manola, D. McLeod, A. Rosenthal, and M. Templeton. Report on the workshop on heterogeneous database systems. *SIGMOD Record*, 19(4):23–31, December 1990.
27. A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous and autonomous databases. *Computing Surveys*, 22(3):183–236, Sep 1990.
28. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.