TupleRank: Ranking Discovered Content in Virtual Databases

Jacob Berlin and Amihai Motro

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

Abstract. Recently, the problem of data integration has been newly addressed by methods based on machine learning and discovery. Such methods are intended to automate, at least in part, the laborious process of information integration, by which existing data sources are incorporated in a virtual database. Essentially, these methods scan new data sources, attempting to discover possible mappings to the virtual database. Like all discovery processes, this process is intrinsically probabilistic; that is, each discovery is associated with a specific value that denotes assurance of its appropriateness. Consequently, the rows in a discovered virtual table have mixed assurance levels, with some rows being more credible than others. We argue that rows in discovered virtual databases should be *ranked*, and we describe a ranking method, called *TupleRank*, for calculating such a ranking order. Roughly speaking, TupleRank calibrates the probabilities calculated during a discovery process with historical information about the performance of the system. The work is done in the framework of the Autoplex system for discovering content for virtual databases, and initial experimentation is reported and discussed.

1 Introduction

Recently, the problem of data integration has been newly addressed by methods based on *machine learning* and *discovery* [2, 3, 7, 5, 9]. Such methods are intended to automate, at least in part, the laborious process of information integration, by which existing data sources are incorporated in a virtual database. Essentially, these methods scan new data sources, attempting to discover possible mappings to the virtual database.

Like all discovery processes, this process is intrinsically probabilistic; that is, each discovery is associated with a specific value that denotes assurance of its appropriateness. Consequently, the rows in a discovered virtual table have mixed assurance levels, with some rows being more credible than others. This suggests that the content of discovered virtual databases should be *ranked*.

Such ranking is useful in several ways. First, users retrieving data from discovered virtual databases should be made aware of their associated "quality," as such information is essential for any decision process that might be based on the data. Moreover, users could now retrieve from discovered virtual databases on the basis of quality; for example, only information that exceeds a certain level of quality is retrieved.

O. Etzion, T. Kuflik, and A. Motro (Eds.): NGITS 2006, LNCS 4032, pp. 13–25, 2006.

Finally, ranking of discovered content permits fully-automated data integration. Typical data integration systems may be classified at best as semi-automatic [4, 7, 9, 10], requiring domain experts to refine each discovered mapping into a "correct" mapping. By developing discovery methods that annotate their discoveries with accurate predictions of their "correctness," and then using these values for ranking the discoveries, we can offer an automated alternative. In other words, whereas other systems guarantee "perfect" mappings through the use of experts, our approach admits mappings of varying quality, relying instead on a grading method.

In this respect, our approach resembles the approach adopted by most information retrieval systems (and virtually all Internet search engines), where large magnitude searches are now performed automatically, and sophisticated answer-ranking methods substitute for the authoritative answers previously provided by human search experts.

The ranking method used in Autoplex is based on two measurements. The first measurement, called *assurance*, is calculated during the discovery process. Essentially, it is a probabilistic measurement which is calculated for each discovery and indicates how well this discovery satisfies expectations (these expectations are based on various forms of knowledge, both acquired and declared). Assurance is the subject of Section 3. By itself, though, assurance may be misleading. To illustrate, imagine an information retrieval expert who examines the abstract of a document and issues a value between 0 and 1 to indicate the document's relevance to a particular search request. The predictions of the expert may be systemically biased; e.g., too high, too low, or involving a more complex bias. Comparing this expert's predictions with the actual relevance of the documents (established, for example, by examining the actual documents), provides a means for *calibrating* this expert's predictions. The second measurement, called *credibility*, is an indication of the actual performance of the Autoplex system. The calculation of this measurement, and the way in which it is used to calibrate the individual assurance measurements are the subjects of Section 4. The outcome of this process is an individual credibility estimate for each discovery.

These credibility estimates become part of the virtual database system, and are applied to rank every virtual table that is materialized by the system. The ranking method, called TupleRank, is a relatively simple derivative of the estimates and is described in Section 5. The TupleRank methodology has been validated in initial experiments. The experiments and the validation methodology are the subject of Section 6. Section 7 summarizes the findings and outlines ideas for further research. We begin with a brief discussion of virtual databases and the Autoplex discovery system for virtual databases.

2 Background

Our work is conducted within the framework of the Multiplex virtual database system [11] and the Autoplex content discovery system [2], and in this section we provide brief descriptions of both. We note, however, that much of our work is of general applicability to most such database integration systems.

2.1 Multiplex

The multidatabase system Multiplex [11] is an example of a virtual database system. The basic architecture of Multiplex is fairly simple (Figure 1). Whereas a traditional database consists of *schema* and *data* that conforms to the schema, a Multiplex database consists of *schema* and *mapping* that describes data that is stored elsewhere. More specifically, the schema is described in the relational model and the mapping is a list of *contributions*. Each contribution is a pair of expressions: the first expression is a view of the schema (expressed in SQL); the second is a query to one of the member databases (expressed in the language of that system). The answer to 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 fields. The requirement that the second expression is written in the language of the source supports member heterogeneity. Alternatively, heterogeneity can be supported by using *wrappers*, which are software modules that provide a uniform interface to the individual sources [8]; in which case these expressions would be written in a single system-wide language.

The main challenge is to decompose a user query into queries to the individual databases, and then combine the individual answers in an answer to the original query, thus providing complete *transparency*. Additional challenges are to provide *partial* answers in the absence of some of sources, and to resolve *inconsistencies* when sources provide overlapping information.

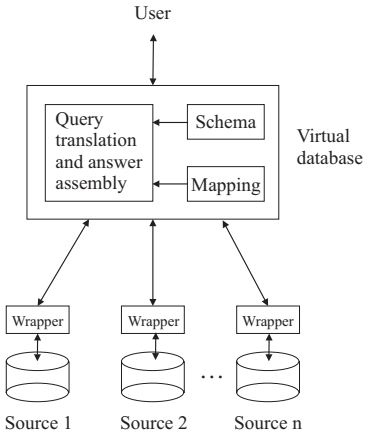

Fig. 1. Multiplex architecture

2.2 Autoplex

A significant limitation of virtually all such multidatabase systems is the difficulty of mapping new sources into the virtual database. Recognizing that

mapping *complete* member schemas is not a viable approach, the Multiplex methodology allows for *ad hoc* mapping of *selected* parts of the sources. Still, in an environment in which the set of member sources is very large and constantly changing, this methodology may not scale up very well.

To address this limitation we have been developing a system called Autoplex [2], for *discovering* member schemas and incorporating them into the virtual schema with only limited human effort. Based primarily on supervised learning, the system acquires knowledge from examples that have already been integrated into the virtual database. With this acquired knowledge, the system can extract "content contributions" from new information sources.

A high level overview of the Autoplex architecture is shown in Figure 2. This architecture includes two main components: a suite of *learners* and a suite *classifiers*. The learners acquire knowledge from examples; the classifiers use this acquired knowledge, as well as various forms of pre-declared knowledge, to infer a contribution from a new source.

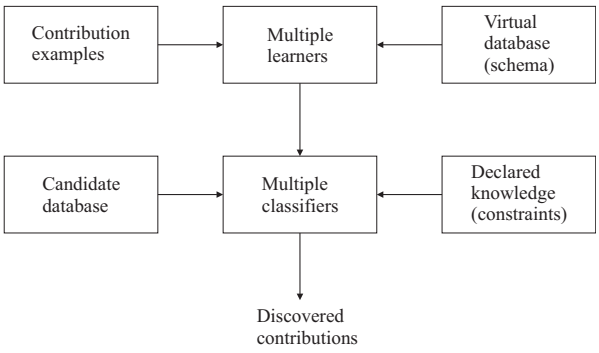

Fig. 2. Autoplex architecture

Each learner is given a virtual database D consisting of tables $R_1, \ldots, R_n$, and a set of contribution examples. Each such example consists of a local scheme S, an instance of this scheme s, and a contribution pair: a selection-projection expression on S and a selection-projection expression on R. The extension of the expression on the instance s generates rows for the expression on the virtual table R. The learner uses this information to acquire knowledge on features of the examples. This knowledge is stored on secondary storage in efficient data structures for future use.

A candidate scheme T and instance t are provided by a new member database as inputs to the corresponding classifier. This classifier uses the acquired knowledge in conjunction with pre-declared knowledge (e.g., constraints) to infer a selection-projection view that defines a contribution of T to a table R in the virtual database. The classifier also outputs assurance metadata that can be used for ranking the discovered content.

The use of multiple learners and classifiers was first applied in the area of database integration in [6, 7], and we use a comparable approach in Autoplex.

Currently, we use learners and classifiers for both domain values and patterns. Furthermore, we distinguish between learners and classifiers for columns and learners and classifiers for rows. This distinction is necessary to support our search for contributions, to be discussed in the next section.

3 Discovery and Assurance

In this section we review the Autoplex content discovery process. In particular, we explain the derivation of the assurance measurement for each discovery.

Autoplex searches for *transformations* of both a candidate table and a virtual table such that the former is a materialization of the latter. For reasons of efficiency, transformations are limited to *projections* and *selections* only. A projection removes columns from a table and possibly reorders the remaining columns; a selection removes rows from the candidate table based on a row predicate.

Furthermore, Autoplex considers projective and selective transformations in two *separate* phases. First, it finds the best projective transformation over a threshold (if any), then, it attempts to improve it by the best possible selective transformation (if any).

3.1 Projection

Finding projective transformations of both the candidate and virtual tables so that the resultant tables are semantically equivalent amounts to finding a *matching* (a one-to-one mapping) between a subset of the columns of the candidate table and a subset of the columns of the virtual table. Let π denote a possible column matching. Autoplex uses a rather complex formula to score the goodness of this matching. A *hill-climbing* search is then used to optimize this formula; i.e., to find the column matching with the highest score. We briefly discuss the Autoplex scoring of column matchings.

The Autoplex score of each column matching is based on the output of multiple classifiers. These classifiers fall into two categories: classifiers that apply *acquired knowledge* and classifiers that apply *declared knowledge*. The former type of classifiers are based on traditional machine learning principles. The primary Autoplex learner is a domain learner. This learner acquires from training examples probabilistic information on the affinities between domain values and virtual columns. Given a new column of values, the corresponding classifier uses this information to score the association between this column and each of the virtual columns. Other learners may be based on column names, string patterns, and so on.

The learners and classifiers consider only individual columns, ignoring any possible interactions among columns. At times, however, the examples may feature such interactions; for instance, the examples may indicate that two virtual columns A and B have a relationship of monotonicity. Learning and classifying multi-column relationships is computationally prohibitive, and for such features Autoplex relies on declared knowledge. Declared knowledge is simply

a pronouncement of relationships among virtual columns (e.g., primary keys, monotonicity, association rules). For every item of declared knowledge, a corresponding classifier scores the level of satisfaction of each column matching. Such classifiers will also be referred to as constraint checkers.

The classifiers in each group are given weights, designating their relative importance or credibility. Because the acquired-knowledge classifiers assign scores to individual column matches, the overall knowledge about each matching may be represented in a bipartite graph. The columns of the candidate and virtual tables are nodes in two separate partitions, and a matching is a set of edges connecting nodes in the two partitions (no two edges share a node). Each edge is labeled with the weighted score of the available classifiers, and the overall score is the product of these labels. For a matching π this score will be denoted $P(\pi)$. For the declared-knowledge classifiers, the overall score is obtained by weighting the scores of the individual constraint checkers. This score will be denoted $S(\pi)$. Finally, these two scores are weighted with a constant α, resulting in this formula

$$ScoreColumn(\pi) = \alpha \cdot S(\pi) + (1 - \alpha) \cdot P(\pi) \tag{1}$$

The optimization process starts with a random, complete matching of the candidate columns to the virtual columns. It then iteratively improves the quality of this matching by swapping and removing edges in the matching.[1] If the best score exceeds a threshold, then Autoplex proceeds to the next phase.

3.2 Selection

The purpose of this phase is to extract an optimal set of rows from the candidate table, and then describe it with an abstract predicate. Let σ denote a subset of rows. Here, too, Autoplex creates a scoring formula for σ that combines verdicts of acquired-knowledge classifiers with declared-knowledge classifiers (constraint checkers).

Recall that the primary learner and classifier in the projection phase were essentially aimed at domains. That is, the learner associated each virtual column with a domain of values, and the classifier matched a candidate column to the most similar domain. Even though the examples show that specific combinations of values (tuples) are more likely to occur than others, this learner has ignored such information. In other words, the corresponding classifier will consider all tuples in the Cartesian product of the domains to be equally likely. The primary learner and classifier in the selection phase address this issue. The learner acquires probabilistic knowledge on the example rows that are selected and those that are discarded, and the classifier assigns each candidate row a probability of being a contribution. In the presence of additional acquired-knowledge classifiers, their verdicts on each row are weighted together to provide a single score.

[1] Note that in the absence of declared-knowledge classifiers, $ScoreColumn(\pi)$ is simply $P(\pi)$ and the problem is reduced to finding a maximal weighted matching in a bipartite graph, for which an efficient $O(n^3)$ algorithm is available.

The score of a subset σ of the candidate rows, denoted $P(\sigma)$, is then a combination of these individual row scores. Finding an appropriate method of combination presents a small challenge. The obvious method of *totaling* the scores of the participating rows is flawed because it encourages the subsequent optimization process to select the entire table. Similarly, *averaging* the scores of the participating rows is flawed as well because it encourages the optimization process to select only the highest scoring row. A good solution appears to be a *product* of scores, in which each row selected contributes its score w, and each row discarded contributes its complement score $1 - w$.

This subset of rows σ is also assigned a score by each available declared-knowledge classifier (constraint checker), reflecting its level of compliance, and these scores are weighted together to provide a single satisfaction score $S(\sigma)$. The scores $P(\sigma)$ and $S(\sigma)$ are then weighted with a constant β. Altogether, the formula

$$ScoreRow(\sigma) = \beta \cdot S(\sigma) + (1 - \beta) \cdot P(\sigma) \tag{2}$$

defines the overall score Autoplex assigns to the prospect that σ is the set of rows that contribute to the virtual table.

As in the projection phase, this scoring function is optimized in a hill-climbing search which begins with a random subset of the rows and improves it iteratively. This optimization process results in set of rows that contribute to the virtual table and a set of discarded rows. The rows are labeled accordingly and a standard classification tree algorithm[2] is applied to find a selection predicate that defines the contributing set of rows. This extension-independent definition will be used to extract the appropriate set of rows should the source be updated.

The constants α and β in the two scoring functions (Equations 1 and 2) denote our willingness to discount the recommendations of the acquired-knowledge classifiers for better constraint satisfaction (and vice versa). This technique of combining general constraints with mapping decisions is due to [7]; however, we extend it to search for both projective and selective transformations.

3.3 Assurance

Finally, the two scores generated in the two phases of the discovery are combined in a single measurement of assurance. Both scores are highly influenced by the cardinalities of the discovery (the number of columns and rows, respectively), with higher cardinalities resulting in lower scores. To compensate, these two scores are normalized by their cardinalities. Let n and m indicate the number of columns and rows, respectively, in the discovery. Assurance is defined as

$$Assurance(\pi, \sigma) = ScoreColumn(\pi)/n + ScoreRow(\sigma)/m \tag{3}$$

Because *ScoreColumn* and *ScoreRow* are generated from products of probabilities and such products tend to become very small, these probabilities are mapped to a logarithmic scale. Consequently, assurance values are always negative.

[2] Autoplex uses the algorithm J48 from the WEKA machine learning package [12].

4 Credibility and Calibration

The assurance measurement is an indication of how well each Autoplex discovery meets expectations. To correct for any systemic biases in this measurement, it is calibrated by the actual performance of the system, termed credibility. We discuss first how credibility is measured, and then how assurance and credibility are combined in a single estimate.

4.1 Measuring Credibility

The success of any discovery process must be judged on the basis of two criteria: (1) its *soundness* — the proportion of the cases discovered that are correct, and (2) its *completeness* — the proportion of the correct cases that are discovered. In Autoplex, a discovery is a mapping of a candidate table to a virtual table, and it is judged with respect to an expert mapping of these tables. Hence, the credibility of Autoplex is measured by *two* dual values.

Soundness and completeness are calculated for every candidate table at the level of the *cell*. Let t be a candidate table, let t_d be the subtable *discovered* by Autoplex and let t_e be the subtable *extracted* by an expert. Together, soundness and completeness measure the overlap of these two subtables. A cell discovered (i.e., in t_d) and also extracted (in t_e) contributes to soundness, whereas a cell extracted (in t_e) and also discovered (in t_d) contributes to completeness.[3] Specifically, the cells of t falls into four disjoint categories:

- A = Cells that are both discovered and extracted (*true positives*).
- B = Cells that are extracted but not discovered (*false negatives*).
- C = Cells that are discovered but not extracted (*false positives*).
- D = Cells that are neither discovered nor extracted (*true negatives*).

The soundness and completeness of a discovery is calculated from the cardinalities of the first three categories:

$$Soundness = \frac{|A|}{|A| + |C|} \tag{4}$$

$$Completeness = \frac{|A|}{|A| + |B|} \tag{5}$$

Soundness is the proportion of valid cases among the discovered cells, and thus measures the accuracy of the Autoplex discovery process. Completeness is the proportion of discovered cases among the valid cells, and thus measures the ability of Autoplex to discover cells. Both measures fall on the $[0, 1]$ interval, and higher values are better.

Soundness and completeness may be viewed also as probabilities. Soundness is the probability that a discovery is a valid contribution, whereas completeness is the probability that a valid contribution is discovered.

[3] We use the primary key of the table to uniquely identify rows and thus cells.

Soundness and completeness correspond to the *precision* and *recall* measures, which are used widely in information retrieval [1]. In some applications, a single performance measure may be preferred, and various ways have been suggested to combine these measures into a single measure (e.g., using their harmonic mean). Here, however, we shall measure the performance of discoveries using both.

4.2 Calibration

The credibility measurements of soundness and completeness are used to *calibrate* the assurance measurement derived in the discovery process. This calibration process uses a set of "training" tables and a numeric discretizing technique from data mining called *equal frequency binning* [12]. The intuition behind this approach is that, while the calculated assurance scores may be inaccurate, higher assurance scores result in higher credibility (i.e., assurance scores are positively correlated to soundness and completeness).

Roughly speaking, the calibration process may be summarized as follows: If the track record of Autoplex shows that when it discovered content with assurance x, its credibility was y, then future discoveries made with assurance x will be estimated to have credibility y as well. A more formal description of the calibration process is given in this 7-step procedure:

1. Autoplex is tasked to discover contributions in the training tables.
2. Experts are used to extract optimal contributions from the same tables.
3. The soundness and completeness of each discovery is calculated.
4. The discoveries are sorted according to their assurance scores.
5. The sorted list of discoveries is divided into b "bins" of equal size.
6. The soundness and completeness of each bin are calculated as the *aggregate* soundness and completeness of all the discoveries in that bin.[4]
7. Finally, boundaries between adjacent bins are calculated by averaging the scores of discoveries that separate the bins.

Since not all training tables are of equal size, equal sized subsets of these tables are used during the discovery phase; this ensures that the bins are all of the same size.

Given a new candidate table, Autoplex discovers the optimal contribution as explained in Section 3. The assurance score and the boundaries (calculated in Step 7) are used to locate the appropriate bin, and the soundness and completeness values of the bin (calculated in Step 6) are associated with the discovery as estimates of its credibility.

Results from actual experimentation in Autoplex are summarized in Table 1. The experiment involved 170 training examples. For each example, assurance and credibility (soundness and completeness) were measured. These measurements showed strong positive correlation between assurance and soundness (0.83) and

[4] *Aggregate* soundness and completeness are obtained from the *unions* of the true positives, false positives and true negatives of the different discoveries, and are usually more informative than *average* soundness and completeness.

between assurance and completeness (0.86), thus validating our working assumption. The 170 examples were assigned to 10 equal-size bins. For each bin, Table 1 shows the boundary assurance value and the aggregate soundness and completeness of the 17 discoveries in the bin. Now consider a future discovery made with assurance -0.8. This discovery will be assigned to bin 4 and will be estimated to have soundness 0.6486 and completeness 0.8205.

Table 1. Example of calibration data

Bin	Assurance (lower boundary)	Soundness (aggregate)	Completeness (aggregate)
1	-0.1232	0.8715	0.9623
2	-0.5977	0.7843	0.9388
3	-0.7704	0.7279	0.9778
4	-0.9638	0.6486	0.8205
5	-1.3222	0.5410	0.8081
6	-1.7155	0.3399	0.7128
7	-1.9522	0.0785	0.5375
8	-2.4041	0.0763	0.3167
9	-2.7884	0.0241	0.2281
10	(none)	0.0109	0.0784

Recall that each contribution is an entry in the mapping of the virtual database (Figure 1). Autoplex discovered contributions are added to the same mapping, but with additional annotations on their estimated credibility.

5 Ranking

The credibility estimates of the discoveries can be used in several ways, and in this section we sketch three possibilities.

A relatively simple deployment of these estimates is to specify in retrieval requests *thresholds of credibility* and require that the evaluation of these requests be based only on discoveries whose credibility exceeds these thresholds. In this way, users can guarantee minimal performance for their queries. If necessary (for example, if the answers that are retrieved are deemed to be too small or too large), these thresholds may be relaxed or strengthened.

A second way in which these estimates can be used is to calculate the *credibility estimates of each answer* that is issued by the system. The main concern here is how to combine the credibility estimates of the different contributions that are used in generating an answer. A straightforward approach is to weigh the estimates of the contributions according to their level of participation in the answer.

A third, and possibly most challenging, way in which the credibility estimates of contributions can be used is to calculate the *credibility of individual rows* of each answer, and then *rank the rows* accordingly. This provides users with information on the quality of each row, and allows them to adopt rows of

higher quality first. We refer to the Autoplex method of ranking answer rows as *TupleRank*.

TupleRank requires an estimate of the credibility of each individual cell of an answer. Of the two measures used to estimate the credibility of contributions, only soundness can be propagated to individual cells. This is because the proportion of discovery cells that are correct can be translated to a probability of each cell being correct; but the proportion of the correct cells in a candidate table that are included in a discovery cannot be translated into a property of the individual cells.

Each cell of an answer is derived from one or more contributions and inherits their credibility. If a cell is obtained from multiple contributions then its soundness is calculated as the average of the soundness of these contributions. The soundness of each row is then calculated as the average soundness of its cells. This final value is used for ranking the answer.

The three methods discussed in this section may be combined. Given a query, only a qualifying set of contributions is used to evaluate its answer; the soundness of each individual row is calculated and the answer is ranked; the sorted answer is accompanied by its overall credibility estimates.

6 Experimentation and Validation

The purpose of the experiment we describe in this section is to validate our calibration methods; that is, to show that calibration improves predictions. Without the use of calibration, the best prediction that can be made regarding the credibility of a discovery is the aggregate performance of Autoplex on a set of test data. Furthermore, this prediction is the same for every discovery. We call this the *uncalibrated prediction*. With calibration, predictions can be made for individual discoveries. We call this the *calibrated prediction*.

To validate the effectiveness of calibration, we ran Autoplex on test data that an expert has mapped into the virtual database. We compared the uncalibrated and calibrated predictions of soundness and completeness relative to the expert mappings, with the expectation that calibrated predictions will be more accurate.

Higher accuracy corresponds to lower error, so we measured the error of uncalibrated and calibrated predictions. Error was measured using Root Mean Squared Error (RMS), which is commonly used to evaluate the effectiveness of numeric prediction. RMS is defined as

$$\sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}} \tag{6}$$

where p_i and a_i are the predicted and actual soundness (or completeness) for the ith discovery of n discoveries. The RMS error for soundness and completeness predictions falls in the $[0, 1]$ interval, and lower values are better.

Two separate experiments were conducted. One experiment used a virtual database on computer retail (originally used in [2]):[5]

[5] Primary keys are indicated in italics.

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

The other experiment used a virtual database on real estate (reported in [7]):

1. Property = (*PropertyId*, Address, Description, Price, Bedrooms, Bathrooms)
2. Agents = (*AgentName*, AgentPhone, AgentEmail)

To experiment with this data, we used a procedure from data mining called *stratified threefold cross-validation* [12], which we briefly describe. Each of the sources was manually mapped into our virtual database by using a mapping table as discussed in Section 2.1. We partitioned the mappings in our mapping table into three folds of approximately equal content. Using two folds for learning and calibration and one fold for testing (i.e., discovery), 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.

For each discovery we measured the error of the calibrated and uncalibrated predictions relative to the correct mappings. Table 2 summarizes the results of the experiments. Both experiments show that the calibrated credibility measures are always more accurate than the uncalibrated ones. Evidently, in three of the four cases the improvement is considerable.

Table 2. RMS error of predicted vs. actual credibility

Credibility	Computer Retail	Real Estate
Calibrated Soundness	0.22	0.31
Uncalibrated Soundness	0.36	0.47
Calibrated Completeness	0.24	0.29
Uncalibrated Completeness	0.34	0.30

7 Conclusion

Recent methods for automatically discovering content for virtual databases result in databases with information of mixed credibility. In this paper we argued that it is important to estimate the credibility of discovered information, so that these estimates could be used to calculate credibility estimates for all information issued from the virtual database. We defined credibility as a pair of measures, soundness and completeness, and we showed how to calculate reliable credibility estimates for each discovery, based on values calculated during the discovery process, and the actual performance of the system on test data. We also sketched various methods in which credibility estimates can be applied.

With respect to experimentation, we have shown so far that the Multiplex virtual database approach is viable [11], that the Autoplex automatic discovery

approach is attractive [2], and (in this paper) that the credibility of discoveries can be estimated reliably. Our focus now is on integrating these results in a single system; that is, incorporate the Autoplex methodology into Multiplex, and modify the Multiplex user interface to provide the type of capabilities described in Section 5.

Undoubtedly, the estimation of the credibility of discoveries and the ranking of tables formed from different discoveries bring to mind information retrieval systems, such as Internet search engines. Extending the Multiplex/Autoplex methodology to discover content in Web pages with semi-structured data (e.g., represented in XML) is a subject currently under investigation.

References

1. R. Baeza-Yates and B.'Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley/ACM Press, 1999.
2. J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. In *Proc. COOPIS 01, Ninth Int. Conf. on Cooperative Information Systems*, pages 108–122, 2001.
3. J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proc. CAiSE 02, Fourteenth Int. Conf. on Advanced Information Systems Engineering*, pages 452–466, 2002.
4. S. Castano and V. De Antonellis. A schema analysis and reconciliation tool environment for heterogeneous databases. In *Proc. IDEAS 99, Int. Database Engineering and Applications Symposium*, pages 53–62, 1999.
5. R. Dhamankar, Y. Lee, A. Doan, A. Y. Halevy, and P. Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proc. SIGMOD 04, Int. Conf. on Management of Data*, pages 383–394, 2004.
6. A. Doan, P. Domingos, and A. Y. Halevy. Learning source description for data integration. In *Proc. WebDB*, pages 81–86, 2000.
7. A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proc. SIGMOD 2001, Int. Conf. on Management of Data*, pages 509–520, 2001.
8. H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *J. Intelligent Information Systems*, 8(2):117–132, 1997.
9. W.-S. Li and C. Clifton. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1):49–84, 2000.
10. J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proc. VLDB 2001, 27th Int. Conf. on Very Large Databases*, pages 49–58, 2001.
11. A. Motro. Multiplex: A formal model for multidatabases and its implementation. In *Proc. NGITS 1999, Fourth Int. Workshop on Next Generation Information Technologies and Systems*, pages 138–158, 1999.
12. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.