Object Matching for Information Integration: A Profiler-Based Approach

AnHai Doan Ying Lu Yoonkyong Lee Jiawei Han
{anhai,yinglu,ylee11,hanj}@cs.uiuc.edu
Department of Computer Science
University of Illinois, Urbana-Champaign, IL 61801, USA

Abstract

Object matching is a fundamental problem that arises in numerous information integration scenarios. Virtually all existing solutions to this problem have assumed that the objects to be matched share the same set of attributes, and that they can be matched by comparing the similarities of the attributes. We consider the more general problem where the objects can also have disjoint attributes, such as matching tuples that come from relational tables with schemas (age,name) and (name,salary), respectively.

We describe PROM, a solution that also exploits the *disjoint* attributes to improve matching accuracy. In the above example, PROM begins by matching any two given tuples based on the shared attribute name. Then it applies a set of *profilers*, each of which contains some knowledge about what constitutes a typical person. The profilers examine the tuple pair to see if it can plausibly make up a person. For example, a profiler may state that because the age is 9 and the salary is 200K, the tuples do not make up a person and thus do not match. Profilers can be manually specified by domain experts, learned from training data, transferred from other matching tasks, or constructed from external data. Thus, the PROM approach is distinguished in that it not only can exploit disjoint attributes to improve matching accuracy, but can also reuse knowledge from previous object matching tasks.

Introduction

Object matching is the problem of deciding if two given objects (e.g., two relational tuples) refer to the same real-world entity. It is often used to consolidate information about entities and to remove duplicates when merging multiple information sources. As such, it plays an important role in many information management contexts, including information integration, data warehousing, information extraction, and text join in databases (e.g., (Tejada, Knoblock, & Minton 2002; Cohen 1998; McCallum, Nigam, & Ungar 2000; Yih & Roth 2002; Bilenko & Mooney 2002; Lawrence, Bollacker, & Giles 1999; Ananthakrishna, Chaudhuri, & Ganti 2002; Sarawagi & Bhamidipaty 2002; Gravano *et al.* 2003; Hernandez & Stolfo 1995)).

Numerous solutions to object matching have been proposed, in both the AI and database communities (see the related work section). Virtually all of these solutions make the assumption that the objects under consideration *share the same set of attributes*. They then match objects by comparing the similarity of the shared attributes.

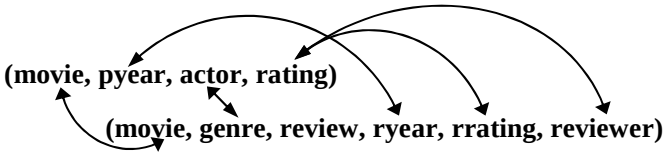

Figure 1: The schemas of two tables in the movie domain. There are many correlations among table attributes (signified with arrows) that can be exploited for object matching.

In this paper we consider the more general matching problem where the objects can also have non-overlapping (i.e., *disjoint*) attributes, such as matching tuples that come from two relational tables with schemas (age,name) and (name,salary). We observe that this problem frequently arises in information integration, when querying a data source, or when merging tuples coming from different sources. In information integration, the sources are typically developed in an independent fashion, and therefore are likely to have overlapping, but different schemas. When dealing with such sources, prior work has not exploited the disjoint schema portions for the purpose of object matching.

In this paper we describe the PROM (Profiler-Based Oject Matching) solution that can exploit the disjoint attributes to maximize matching accuracy. The key observation underlying our approach is that the disjoint attributes are often correlated, and that such correlation can be leveraged to perform a "sanity check" for object matching. For example, consider the two tuples (9,"Mike Smith") and ("Mike Smith",200K). Assuming that they match, we would have a "Mike Smith" who is 9 year old and has a salary of 200K. This appears unlikely, based on our knowledge, specifically, on the *"profile"* that we have about what constitutes a typical person. This profile tells us that such relationship between age and salary is unlikely to exist. Thus, the above two tuples are unlikely to match.

As another example that illustrates the PROM approach, consider two relational tables that contain information about movies and their reviews, respectively. Figure 1 shows the schemas of the two tables. The meaning of most schema attributes are clear from their names, except perhaps pyear

and ryear, which specify the years when the movie was produced and reviewed, respectively, and rrating, which specifies the rating as given by the reviewer.

Given two tuples from the tables, PROM begins by matching the shared attribute movie (i.e., movie name), using any of the existing object matching techniques. If the similarity between the names is low, PROM can discard the tuple pair as no match. Otherwise, PROM applies a set of modules called *profilers* to the tuple pair to perform "sanity check". A profiler contains knowledge about a *specific concept*, such as movie, actor, or review. When given a tuple pair that contain information about the concept, the profiler can examine the pair to decide if it violates any constraint on the concept.

In our movie example, a tuple pair contains information about several concepts in the movie domains, and therefore can be examined by many profilers. For example, a *review profiler* may know that the year in which a review was published must not preceed the production year of the reviewed movie. Thus this profiler can check if the values of the disjoint attributes pyear and ryear satisfy that constraint. This profiler may also know that certain reviewers (e.g., Roger Ebert) have never reviewed any movie with an average rating below 4 (out of 10). Thus, it may also check reviewer and rating for this correlation. An *actor profiler*, on the other hand, may know that a certain actor has never played in action movies, and would check attributes actor and genre. As yet another example, a *movie profiler* may know that the average ratings of a movie tend to be positively correlated. Thus, it may check attributes rating and rrating. Suppose their values are 9 and 2, respectively, then the profiler may conclude that the two tuples probably do not match. PROM knows how to combine the conclusions of the many profilers in order to arrive at a final matching decision for the tuple pair.

A compelling property of profilers is that they contain knowledge about *domain concepts* (e.g., movies, reviews, persons, etc.). Hence, they can be constructed once, then applied to many object matching tasks, as long as the tasks involve the concepts. They can be constructed by domain experts and users, and can also be learned from the data in the domain (e.g., from all movie tuples in the Internet Movie Database at *imdb.com*). Alternatively, they can also be constructed in the context of a specific matching task, from the training data for that task. But afterwards, they can also be transferred to other related matching tasks in the domain.

The PROM approach to object matching therefore possesses several desirable characteristics. First, unlike previous approaches, it can exploit disjoint attributes to maximize matching accuracy. Second, it enables the construction and transfer of matching knowledge (in form of profilers) across matching tasks. Finally, it provides an extensible framework into which to plug newly developed profilers, to further improve matching accuracy. Such frameworks have proven useful for solving other problems, such as schema matching (Doan, Domingos, & Halevy 2001; Do & Rahm 2002; Madhavan *et al.* 2003) and information extraction (Freitag 1998; Craven *et al.* 2000), but to our knowledge have not been considered for object matching.

The key challenges facing PROM are to define, construct, and combine profilers. In this paper we describe the first steps toward solving these challenges. Specifically, the paper makes the following contributions:

- We introduce the general object matching problem where objects can also have disjoint attributes.
- We describe the PROM solution that exploits the disjoint attributes to maximize matching accuracy. The solution can reuse knowledge from previous matching tasks, and provides an extensible framework into which new matching knowledge can be easily incorporated.
- We present preliminary experimental results on two real-world datasets that show the promise of the PROM approach. The results also show that extending existing matching techniques in a straightforward manner to exploit disjoint attributes may actually decrease rather than increase matching accuracy.

Problem Definition

We now describe the specific object matching problem that we consider in this paper. Let T_1 and T_2 be two relational tables. We say two tuples from the tables *match* if they refer to the same real-world entity. A table attribute is called a *shared attribute* iff it appears in both tables and any two tuples that match must agree on the value of that attribute. Other attributes are called *disjoint attributes*. We assume that tables T_1 and T_2 have a non-empty set of shared attributes.

For example, the two tables in Figure 1 share the attribute movie. A matching pair of tuples from the two tables must share the same movie name. In constrast, attributes rating and rrating are not shared, because the same movie may have different ratings.

Given the two tables T_1 and T_2, the matching problem that we consider is to find all matching tuples between the two tables. This is a very general problem setting which arises in many contexts, including data integration (Tejada, Knoblock, & Minton 2002), data warehousing (Ananthakrishna, Chaudhuri, & Ganti 2002), and text join in databases (Gravano *et al.* 2003). In the rest of the paper, we shall use the terms "object" and "tuple" interchangeably when there is no ambiguity.

The performance of matching algorithms have typically been evaluated with *matching accuracy* and *runtime efficiency* (Hernandez & Stolfo 1995; Ananthakrishna, Chaudhuri, & Ganti 2002). As the first step, in this paper we shall focus on improving matching accuracy. Improving runtime is the subject of future research. In the experiment section we describe our accuracy measures in detail. In the next section we describe the PROM approach to solving the above object matching problem.

The MOBS Approach

Figure 2 illustrates the working of PROM. Given two tuples t_1 and t_2 from the input tables, the Similarity Estimator computes a similarity value for the tuple pair and decides if they can potentially match. Note that this similarity value is

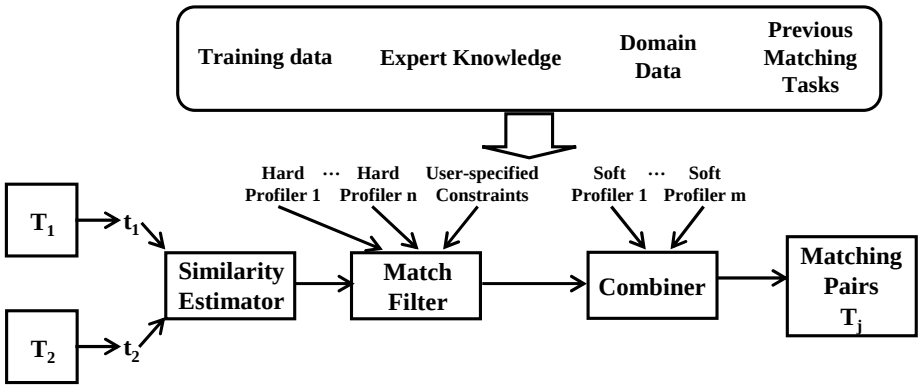

Figure 2: The working of the PROM system

computed *based solely on the shared attributes*. If this module decides that the similarity value is low, it discards the tuple pair, otherwise it passes the pair to the Match Filter.

The Match Filter uses a set of hard profilers to check if the tuple pair could possibly match. A hard profiler contains hard constraints on the concept that it profiles. If *any* hard profiler says no, then the pair is discarded from further consideration. Notice that the Match Filter can also take user specified hard constraints (treating them as belonging to yet another hard profiler).

Any tuple pair surviving the Match Filter is passed to the Meta Profiler. This module employs a set of soft profilers. Each soft profiler issues a confidence score that indicates how well the tuple pair fits the profile maintained by the profiler. The Combiner merges the confidence scores to obtain a single overall score, then decides based on this score if the tuple pair fit the profile and thus likely to match. If the decision is "yes", the pair is stored in the result table T_J, otherwise it is discarded.

We now describe the PROM module in more detail. We note that the Similarity Estimator can employ any of existing object matching techniques (see the related work section), and hence is not discussed further. In the experiment section we describe specific instantiations of the PROM architecture for the real-world datasets.

Profilers: As mentioned earlier, a profiler contains a "profile" of a concept, that is, the knowledge about what constitute a typical instance of the concept. Most importantly, given a tuple pair that include information about the concept, the profiler can issue a confidence score on how well the pair fits the concept "profile" (in a sense, on how well the data of the two tuples fit together).

A hard profiler specifies "hard" constraints about a concept, that is, constraints that *any* instance of that concept *must* satisfy. An example of such constraints is that the review year must not preceed the year when the movie is produced. As another example, an actor hard profiler covers actors and may specify that a specific actor has never played in a movie with the average rating less than 4.

Hard profilers can be constructed manually, or automatically by examining the data in the domain, given that the data is *complete*. For example, "hard" constraints about an actor and his/her movie rating can be automatically derived by examining *all* movies that involve the actor.

Note that the user can also specify "hard" constraints about the matching process, and these constraints can be thought of as making up a *temporary* hard profiler (see Figure 2). While other hard profilers cover general concepts and thus can be transferred across matching tasks, some user-supplied "hard" constraints may be task-specific and thus not transferrable.

A soft profiler also covers a certain concept, but specifies "soft" constraints that any instance of that concept is *likely* to satisfy. A movie soft profiler may specify that the IMDB rating and the Ebert rating of a movie are strongly correlated, in that they would differ by less than 3. Most movies, but not all, would satisfy this constraint.

Like hard profilers, soft profilers can also be constructed in several ways. They can be elicited manually from domain experts and users (then evaluated on some training data to obtain confidence scores for the elicited rules). They can also be constructed from domain data. For example, we can learn a Bayesian network from movie instances in the IMDB database. This Bayesian network would form a soft profiler for movies. Soft profilers can also be constructed directly from training data for a particular matching task. Given a set of matching and non-matching pairs, virtually any learning technique can be applied to construct a classifier that in essence represents a soft profiler.

Combining Profilers: Since the hard profilers issue "yes/no" predictions whereas the soft profilers issue confidence scores, we separate the combination of the two types of profilers, as represented by the Match Filter and the Combiner. We also believe that separating the combination of profilers this way improves matching accuracy over methods that combine all profilers in a single stage; we are verifying this with current research.

The Match Filter uses an AND combination to merge hard profilers' predictions. That is, if any hard profiler says no, then the overall prediction is no and the tuple pair is discarded. The Combiner merges soft profilers' predictions by computing the weighted sum of the confidence scores. The

weights are currently set manually, based on some experiments on holdout data. In the future, we shall explore methods to set weights automatically, in a fashion similar to that of (Doan, Domingos, & Halevy 2001).

Empirical Evaluation

We now present preliminary results that demonstrate the utility of exploiting disjoint attributes and the potential of the PROM approach.

Data: We evaluated PROM on two datasets, Citeseer and Movies. The dataset Citeseer was obtained from *http://citeseer.nj.nec.com/mostcited.html*, which lists highly cited authors together with their homepages. An actual line from this page is "J. Gray homepage-url1 ... homepage-url5", where the five homepage urls were suggested by a search engine. The homepages belong to James Gray at Walker Informatics, Jeffrey Gray at University of Alabama, and so on. Only one homepage actually belongs to the correct Jim Gray (at Microsoft Research). Thus, the object matching problem here is to match author names such as "J. Gray" with their correct homepage urls.

We downloaded the top 200 authors, together with the suggested homepages. Since in this first step we only consider matching relational tuples, we manually converted each homepage into a tuple, by extracting from the homepage information such as name, name and rank of current university, position, and graduation year. We removed authors who have no associated homepage and performed some simple text processing. (The exact tuple format and data transformation procedures will be given in the full paper.) The final dataset consists of 150 author names and 254 homepage tuples, for an average of 1.7 homepage tuples per author.

The dataset Movies consists of two tables, with formats (movie-name1,production-year,avg-rating) and (movie-name2,review-year,ebert-rating,review). They are obtained from the Internet Movie Database (*imdb.com*) and Roger Ebert's Review Page (*suntimes.com/ebert/ebertser.html*), respectively. The tables consist of about 10000 tuples and 3000 tuples, respectively.

Algorithms & Methodologies: We begin by describing algorithms applied to the Citeseer dataset. First, we applied Baseline, an algorithm that matches tuples based only on the shared attributes: author name with homepage owner's name in our case. Baseline converts the values of the shared attributes into a set of tokens, then compares the obtained sets of tokens.

Next, we applied three *extended traditional* algorithms, which extend existing object matching techniques that exploit only shared attributes to exploit also disjoint attributes. Extended-Manual manually specifies the matching rules (e.g., "if similarity(name1,name2) ≥ 0.8 but position=student then the two tuples do not match"). Thus, in a sense this method extends the manual method described in (Hernandez & Stolfo 1995), which would exploit only shared attributes such as "name1" and "name2". Extended-AR is similar to Extended-Manual, but uses the association rule classification method of (Li, Han, & Pei 2001) to guide the process of generating rules. The rules of Extended-AR are then manually picked among the generated rules. In constrast to the above two (semi)-manual methods, Extended-DT is completely automatic. It extends the decision tree method in (Tejada, Knoblock, & Minton 2002), by adding to the training dataset all disjoint attributes, and a new attribute that specifies for each tuple pair its similarity value, as computed based on the shared attributes.

Finally, we applied PROM. For the Similarity Estimator, we used the Baseline algorithm described above. We currently used no hard profilers. We use three soft profilers: one that consists of several "soft" manually specified rules, one that uses decision tree techniques, and one that uses association rule techniques.

We applied similar algorithms to the Movies dataset. We then evaluated matching accuracy using three measures: *recall* (number of correct matching pairs in the join table divided by total number of correct matching pairs), *precision* (number of correct matching pairs in the join table divided by total number of pairs in the join table), and *F-value* (defined as $2*recall*precision/(recall+precision)$). These measures have been used widely in the object matching literature. They also suit our objectives of developing matching methods that maximize precision and recall.

On each dataset, we performed 4-fold cross validation, and report the *average* recall, precision, and F-value. We took care to create folds that are representative of the overall dataset (see (Bilenko & Mooney 2002) for an example of such fold creation).

Results: Since the results on both datasets are similar, we report only those of Citeseer. Table 1 shows the evaluation results for this dataset. Each column in the table lists recall, precision, and F-value (in that order) for a specific object matching algorithm.

The results for Baseline (first column) show that it achieves high recall (99%) but low precision (67%), thereby demonstrating that matching based on the shared attributes only (names in this case) can be quite inaccurate. Extended-Manual (second column) decreases recall slightly (by 2%) but increases precision substantially (by 16%), thus demonstrating that exploiting disjoint attributes (any attribute other than names in this case) can significantly boost matching accuracy. Extended-AR (third column) shows similar, albeit slightly worse, performance to Extended-Manual.

The automatic method Extended-DT (fourth column) shows some surprising results: its precision is substantially lower than that of Baseline (58% vs. 67%). This is unusual because one would expect that Extended-DT improve matching precision, by virtue of exploiting disjoint attributes. A close inspection reveals that many rules that Extended-DT constructed do not refer to the similarity values of the input tuples at all. In other words, these rules match tuples based *solely on exploiting the correlation among the disjoint attributes*, ignoring the shared attributes. (The previous two methods do not have any such rules because those rules are manually constructed or verified.) It's thus clear that such rules will not be very accurate on the

		Baseline	Extended Traditional			PROM			
			Man	AR	extDT	DT	Man+DT	Man+AR	Man+DT+AR
CiteSeer	R	0.99	0.97	0.96	0.91	0.95	0.67	0.96	0.97
	P	0.67	0.83	0.71	0.58	0.78	0.87	0.82	0.86
	F	0.80	0.89	0.81	0.71	0.85	0.76	0.88	0.91

Table 1: Experimental results on the Citeseer dataset

testing data. This surprising result suggests that extending prior matching techniques in a straightforward manner to handle disjoint attributes may actually decrease rather than increase matching accuracy.

For the PROM algorithm, besides examining its performance with respect to the baseline and extended algorithms, we also want to know if adding more profilers would be better accuracy-wise than fewer profiler. Thus, we ran four variations of PROM (see the last four columns of Table 1. The DT variation uses only one soft profiler, which is the decision tree method. Man+DT uses the soft manual profiler and the soft decision tree profiler. Man+AR is similar to the above variation, but replacing the decision tree with the association rule classifier. Finally, Man+DT+AR is the complete PROM algorithm.

The results of PROM show that the DT variation beats the Extended-DT. This suggests that extending prior matching techniques to exploit disjoint attributes in the PROM manner is promising and potentially better than a straightforward extension of traditional techniques. The results also show that the complete PROM system (last column) achieves the highest F-value (0.91) over any previous method, due to high precision and recall. (In particular, this algorithm found the correct Jim Gray homepage that the Baseline algorithm could not.) The results suggest that PROM obtains the best performance and that adding more profilers may improve matching accuracy, because more matching knowledge can be utilized.

Summary: The preliminary results on the two datasets suggest that:

- exploiting disjoint attributes can substantially improve matching accuracy, but
- exploiting them by straightforwardly extending existing techniques may actually decrease rather than increase matching accuracy, and
- the PROM approach can exploit disjoint attributes and domain knowledge to improve accuracy over baseline and extended traditional methods.

Discussion: We are currently experimenting with several new methods to learn profilers in these domains (e.g., Naive Bayes as well as methods that do not require training data). We also plan to transfer the profilers constructed in these matching tasks (e.g., the decision tree soft profiler) to other related matching tasks to examine the effect of transferring such knowledge. We are also particularly interested in learning profilers from domain data, independently of any matching task (e.g., learning movie and actor profilers from *imdb.com*), then applying these profilers to matching tasks in the domain.

Related Work

Our work builds upon numerous matching solutions that have been developed in the AI, database, and data mining communities (e.g. (Tejada, Knoblock, & Minton 2002; Cohen 1998; McCallum, Nigam, & Ungar 2000; Yih & Roth 2002; Bilenko & Mooney 2002; Lawrence, Bollacker, & Giles 1999; Ananthakrishna, Chaudhuri, & Ganti 2002; Sarawagi & Bhamidipaty 2002; Gravano *et al.* 2003; Hernandez & Stolfo 1995; Galhardas *et al.* 2000; Raman & Hellerstein 2001)). Earlier solutions employ manually specified rules to match objects (Hernandez & Stolfo 1995). Many subsequent solutions learn matching rules from a set of training data created from the input tables (Tejada, Knoblock, & Minton 2002; Bilenko & Mooney 2002; Sarawagi & Bhamidipaty 2002). Several solutions focus on efficient techniques to match strings (Monge & Elkan 1996; Gravano *et al.* 2003). Others also address techniques to scale up to very large number of objects (McCallum, Nigam, & Ungar 2000; Cohen & Richman 2002). The commonality underlying these solutions is that they match objects by comparing the shared attributes. Our solution extends these previous solutions by adding another layer that utilizes the correlations among the disjoint attributes, to maximize matching accuracy. Our use of attribute correlation bears some resemblance to the work (Kang & Naughton 2003), in which the authors exploit statistical correlation among schema attributes to find semantic mappings between the attributes of two relational tables.

The topics of knowledge reuse and incorporating prior knowledge have been studied actively in the AI community. More closely related to our approach, several AI works have considered the issue of reusing classifiers that are learned in other domains (e.g., (Cohen & Kudenko 1997)). Our work differs from these in several aspects. First, we also reuse knowledge types other than classifiers (e.g., the manual profilers). Second, when reusing classifiers we do not attempt to reuse *arbitrary* classifiers from other domains. Instead, we advocate building task-independent classifiers and reusing only those. This is possible in our context due to the frequent recurrence of common concepts in matching tasks within a domain. For example, any matching task in the movie domain is likely to involve the concepts of movie, review, actor, and so on.

Recently, knowledge reuse has received increasing attention in the database community, and several works on schema matching (Berlin & Motro 2002; Do & Rahm 2002; Madhavan *et al.* 2003; Doan, Domingos, & Halevy 2001) and data integration (e.g., (Rosenthal *et al.* 2001)) have investigated the issue. Our work can be seen as a step in this direction. To our knowledge, this is the first work that attempts to reuse knowledge in the context of object matching.

Conclusion & Future Work

Object matching plays an important role in a wide variety of information management applications. Previous solutions to this problem have typically assumed a uniform setting where objects share the same attributes. In this paper we have considered a more general setting where objects can have different but overlapping sets of attributes. Such a setting commonly arise in practice, where data sources are independently developed and thus are unlikely to share the same schemas.

We have proposed the PROM solution that builds upon previous work, but exploits the disjoint attributes to substantially improve matching accuracy. To do this, PROM employs multiple profilers, each of which contains information about a certain concept in the matching task. The profilers can be specified by domain experts, learned from training data that is obtained from the input objects, transferred from related matching tasks, or constructed from domain data. Most importantly, the profilers contain task-independent informations and thus can be reused once constructed. This makes the PROM approach labor-saving and maximizing accuracy on any particular matching task. Preliminary experiments on two real-world datasets show the promise of the PROM approach.

Our approach also suggests a broader knowledge-reuse methodology: within any particular task, isolate knowledge that is task-dependent (e.g., similarity knowledge) from that which is task-independent (e.g., profile knowledge). The latter, once learned, can be reused across tasks. This methodology is clearly not always applicable, but can be effective in appropriate settings, as we have demonstrated. Our future research, besides developing the PROM solution – as discussed in the experiment section, will aim to further explore this idea.

References

Ananthakrishna, R.; Chaudhuri, S.; and Ganti, V. 2002. Eliminating fuzzy duplicates in data warehouses. In *Proc. of 28th Int. Conf. on Very Large Databases*.

Berlin, J., and Motro, A. 2002. Database schema matching using machine learning with feature selection. In *Proceedings of the Conf. on Advanced Information Systems Engineering (CAiSE)*.

Bilenko, M., and Mooney, R. 2002. Learning to combine trained distance metrics for duplicate detection in databases. Technical Report Technical Report AI 02-296, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, TX.

Cohen, W., and Kudenko, D. 1997. Transferring and retraining learned information filters. In *Proc. of the AAAI Conf. (AAAI-97)*.

Cohen, W., and Richman, J. 2002. Learning to match and cluster entity names. In *Proc. of 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*.

Cohen, W. 1998. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Procceedings of SIGMOD-98*.

Craven, M.; DiPasquo, D.; Freitag, D.; McCallum, A.; Mitchell, T.; Nigam, K.; and Slattery, S. 2000. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence* 118(1-2):69–113.

Do, H., and Rahm, E. 2002. Coma: A system for flexible combination of schema matching approaches. In *Proceedings of the 28th Conf. on Very Large Databases (VLDB)*.

Doan, A.; Domingos, P.; and Halevy, A. 2001. Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach. In *Proceedings of the ACM SIGMOD Conference*.

Freitag, D. 1998. Multistrategy learning for information extraction. In *Proc. 15th Int. Conf. on Machine Learning (ICML-98)*.

Galhardas, H.; Florescu, D.; Shasha, D.; and Simon, E. 2000. An extensible framework for data cleaning. In *Proc. of 16th Int. Conf. on Data Engineering*.

Gravano, L.; Ipeirotis, P.; Koudas, N.; and Srivastava, D. 2003. Text join for data cleansing and integration in an rdbms. In *Proc. of 19th Int. Conf. on Data Engineering*.

Hernandez, M., and Stolfo, S. 1995. The merge/purge problem for large databases. In *SIGMOD Conference*, 127–138.

Kang, J., and Naughton, J. 2003. On schema matching with opaque column names and data values. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD-03)*.

Lawrence, S.; Bollacker, K.; and Giles, C. L. 1999. Autonomous citation matching. In *Proc. of the 3rd Int. Conf. on Autonomous Agents*.

Li, W.; Han, J.; and Pei, J. 2001. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proc. of the Int. Conf. on Data Mining (ICDM-01)*.

Madhavan, J.; Bernstein, P.; Chen, K.; Halevy, A.; and Shenoy, P. 2003. Matching schemas by learning from a schema corpus. In *Proc. of the IJCAI-03 Workshop on Information Integration on the Web*.

McCallum, A.; Nigam, K.; and Ungar, L. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*.

Monge, A., and Elkan, C. 1996. The field matching problem: Algorithms and applications. In *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*.

Raman, V., and Hellerstein, J. 2001. Potter's wheel: An interactive data cleaning system. In *The VLDB Journal*, 381–390.

Rosenthal, A.; Renner, S.; Seligman, L.; and Manola, F. 2001. Data integration needs an industrial revolution. In *Proceedings of the Workshop on Foundations of Data Integration*.

Sarawagi, S., and Bhamidipaty, A. 2002. Interactive deduplication using active learning. In *Proc. of 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*.

Tejada, S.; Knoblock, C.; and Minton, S. 2002. Learning domain-independent string transformation weights for high accuracy object identification. In *Proc. of the 8th SIGKDD Int. Conf. (KDD-2002)*.

Yih, W., and Roth, D. 2002. Probabilistic reasoning for entity and relation recognition. In *Proc. of COLING'02*.