Profile-Based Object Matching for Information Integration

AnHai Doan, Ying Lu, Yoonkyong Lee, and Jiawei Han, *University of Illinois, Urbana-Champaign*

Traditional object-matching methods rely on similarities among shared attributes. Profile-Based Object Matching builds on this approach but also correlates disjoint attributes to improve matching accuracy.

Object-matching systems attempt to determine whether two objects (such as two relational tuples) refer to the same real-world entity. When an organization merges information sources, for example, it might use an object-matching application to help consolidate information about entities and remove duplicates. Object matching thus plays an important role in many information management contexts, including information integration, data warehousing, information extraction, and text joins in databases.

As the "Related Work" sidebar describes, researchers have proposed numerous object-matching solutions in both the AI and database communities. Virtually all these solutions assume that the target objects, or *tuples*, share the same set of attributes, and they match tuples by comparing attribute similarity. However, tuples often have nonoverlapping, or *disjoint*, attributes. This is frequently true in information integration, especially when merging tuples from different data sources. Because these data sources are typically developed independently, they often have overlapping, but different schemas.

As an example, suppose a government agency is trying to merge two branch offices and their databases (S1 and S2), which both list information about people living in Champaign, Illinois. As Figure 1 shows, each tuple contains information about a single person. Because the databases cover the same geographical area, they contain many duplicate tuples. For example, S1's tuple **S. Riley, 105 Spring St, 61801, $95,000** refers to the same person as S2's **Sarah Riley, E. Spring St, 61801, 38**; the shared attributes are name, street, and zip, while the disjoint attributes are income (S1) and age (S2).

Existing systems can effectively match attributes, despite variations and errors: the name "Sarah Riley," for example, might be abbreviated as "S. Riley" or even mistyped as "Sarah Rilye." However, these methods don't exploit disjoint attributes in the matching process. Our Profile-Based Object Matching solution—as embodied in the PROM system we're developing—exploits disjoint attributes to maximize matching accuracy. The key to PROM is that disjoint attributes are often correlated, and we can leverage this correlation to perform a "sanity check" on object matching. So if we had the S1 tuple **Mike Smith, E. White Street, 61820, $100,000** and the S2 tuple **Mike Smith, E. White Street, 61820, 6**, prior solutions would declare a match because the shared attributes match perfectly. However, the disjoint attributes (income in S1 and age in S2) indicate that, combined, the two tuples give us "Mike Smith," a six-year-old with a $100,000 yearly income. This is possible but very unlikely. Thus, PROM would reject the match.

PROM overview

To illustrate the PROM approach, we'll use two relational tables: one contains information about movies, the other about movie reviews (see Figure 2). The meaning of most schema attributes are clear from their names; exceptions are **pyear** and **ryear**, which indicate the year the movie was produced and reviewed, respectively; and **rrating**, which specifies the reviewer's film rating.

Given two tuples from the tables, PROM begins by matching the shared attribute **movie** (that is, the movie's name) using existing object-matching techniques. If the name similarity is low, PROM discards the pair as not matching. Otherwise, PROM performs a sanity check using modules that apply different *profilers* to the tuple pair. A profiler contains knowledge about a specific concept, such as movie, actor,

1094-7167/03/$17.00 © 2003 IEEE
Published by the IEEE Computer Society

or review. Given a tuple pair that contains concept information, the profiler can examine the pair to decide if it violates any concept constraints.

Because our movie tuple pair has information about several concepts in the movie domain, PROM examines it using several different profilers. A review profiler, for example, might know that the year the review was published must not precede the year that the movie was produced. The profiler thus checks to see if the disjoint attribute values **ryear** and **pyear** satisfy that constraint. The review profiler might also know that certain reviewers, such as Roger Ebert, have never reviewed a movie with an average rating below 4 (out of 10) and might correlate the reviewer and rating accordingly. PROM then applies other profilers, such as the actor or movie profiler, to check for other correlations. It then combines the profilers' output to arrive at a final matching decision for the tuple pair.

A compelling property of profilers is that they contain knowledge about domain concepts (such as movies, reviews, and people). Profilers can thus be built once, then applied to many object matching tasks as long as the tasks involve the concepts. The profilers can be built by domain experts and users, as well as be trained on domain data (such as all movie tuples in the Internet Movie Database at www.imdb.com). Alternatively, users can build profilers in the context of a specific matching task, using that task's training data. Thereafter, the profilers can be transferred to other related matching tasks in the domain.

The PROM approach to object matching therefore

- lets users construct and transfer matching knowledge (in the form of profilers) across matching tasks
- provides an extensible framework into which users can plug newly developed profilers, further improving matching accuracy

Although researchers have used similar frameworks for solving other problems, such as schema matching[1–3] and information extraction,[4,5] they have not to our knowledge considered such a framework for object matching.

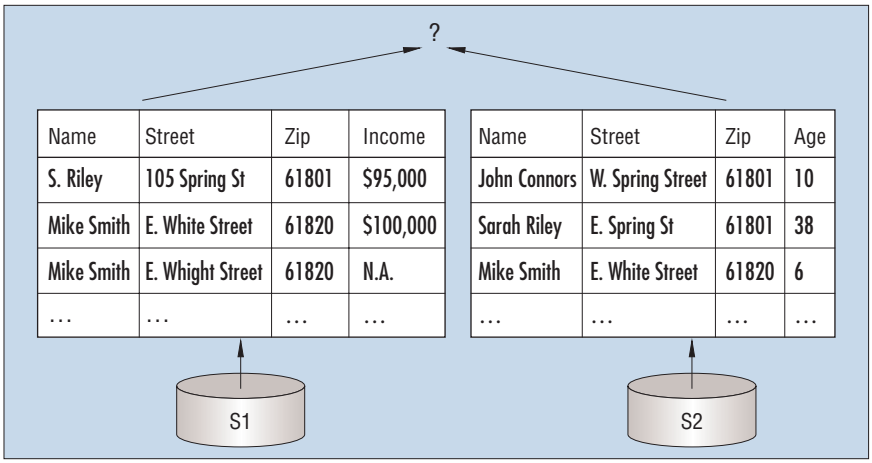

Figure 1. Integrating the S1 and S2 databases. Many tuples refer to the same person. Object-matching systems must detect and merge such tuples to ensure data-processing accuracy.

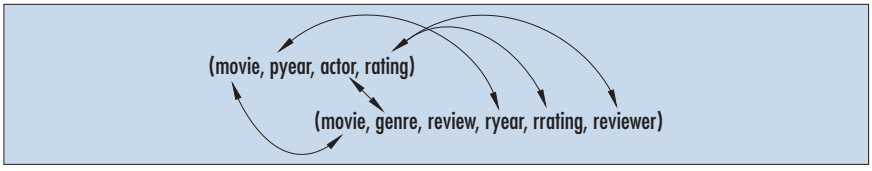

Figure 2. Two table schemas in the movie domain. PROM exploits several attribute correlations (signified by the arrows) for object matching. For each movie, pyear is the production year, ryear is the review year, and rrating is the reviewer's rating.

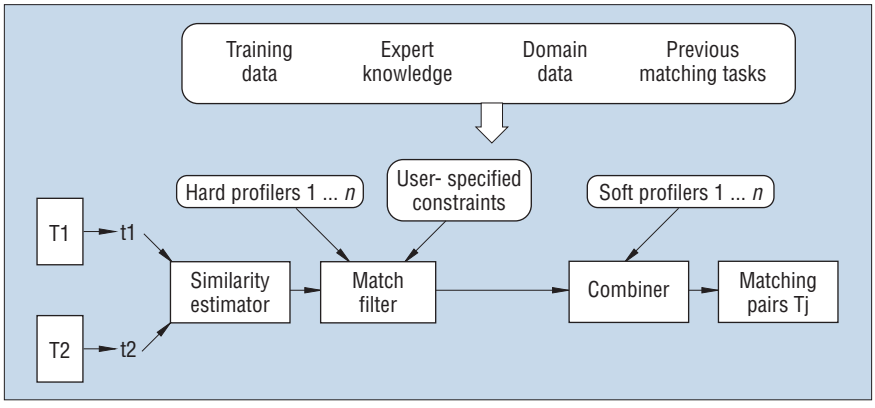

Figure 3. The PROM system. If the similarity estimator computes a low value for the tuple pair, it discards them. Otherwise, it passes them on, and profilers in the match filter and combiner further evaluate the pair.

PROM components

To discuss specific PROM components in detail, we'll use the relational tables T1 and T2 (see Figure 3). We say that two tuples from the tables *match* if they refer to the same real-world entity. An attribute appearing in both tables is a shared attribute if any two tuples that match must agree on that attribute's value. In Figure 2, for example, the two tables share the **movie** attribute, so a matching pair of tuples from those tables must share the same movie name. In contrast, the **rating** and **rrating** attributes are disjoint because a single movie can have different ratings. Here, we assume that tables T1 and T2 have a nonempty set of shared attributes.

Our matching problem for T1 and T2 is to find all matching tuples between them. This general problem setting arises in many contexts, including data integration and warehousing. Researchers have typically evaluated a matching algorithm's performance by measuring accuracy and runtime efficiency.[6,7] We focus here on improving accuracy; we'll focus on improving runtime efficiency in future research.

Similarity estimator

Figure 3 shows two tuples—t1 and t2. PROM's similarity estimator computes a similarity value for the input tuples and decides if they're a potential match. It computes this

Related Work

Our work builds upon numerous matching solutions developed by researchers in the AI, database, and data mining communities.[1–12] Early solutions use manually specified rules to match objects,[6] while many subsequent solutions learn matching rules from training data created from the input tables.[2,10,11] Several solutions focus on efficient techniques to match strings;[5,13] others address techniques to scale up to numerous objects.[8,14] All the solutions match objects by comparing their shared attributes.

Our solution extends existing solutions by adding a layer that correlates disjoint attributes to maximize matching accuracy. This use of attribute correlation bears some resemblance to work in which researchers exploit statistical correlation among schema attributes to find semantic mappings between two relational tables' attributes.[15]

The AI research community has actively studied knowledge reuse and prior-knowledge incorporation. And, in work that relates more closely to ours, several AI researchers have considered reusing classifiers that are trained in other domains.[16] Our work differs from this AI work in two primary ways. First, we reuse knowledge types other than classifiers (such as manual profilers). Second, when we reuse classifiers, we don't try to reuse arbitrary classifiers from other domains. Instead, we advocate building task-independent classifiers and reusing only those. We can do this in our context because common concepts frequently recur in a domain's matching tasks. Any matching task in the movie domain, for example, will likely involve the concepts of movie, review, actor, and so on.

Recently, database researchers have paid increasing attention to knowledge reuse, and several have investigated schema matching[17–20] and data integration.[21] Our work is a step in this direction; to our knowledge, it's the first work that attempts to reuse knowledge in the object-matching context.

References

1. R. Ananthakrishna, S. Chaudhuri, and V. Ganti, "Eliminating Fuzzy Duplicates in Data Warehouses," *Proc. 28th Int'l Conf. Very Large Databases* (VLDB 2002). Morgan Kaufmann, 2002, pp. 586–597.
2. M. Bilenko and R. Mooney, *Learning to Combine Trained Distance Metrics for Duplicate Detection in Databases*, tech. report AI 02-296, Artificial Intelligence Laboratory, Univ. Texas at Austin, 2002.
3. W. Cohen, "Integration of Heterogeneous Databases without Common Domains Using Queries Based on Textual Similarity," *Proc. ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 98), ACM Press, 1998, pp. 201–212.
4. H. Galhardas et al., "An Extensible Framework for Data Cleaning," *Proc. 16th Int'l Conf. Data Eng.* (ICDE 00), IEEE CS Press, 2000, p. 312.
5. L. Gravano et al., "Text Joins for Data Cleansing and Integration in an RDBMS," *Proc. 19th Int'l Conf. Data Eng.* (ICDE 03), IEEE CS Press, 2003.
6. M. Hernández and S. Stolfo, "The Merge/Purge Problem for Large Databases," *Proc. 1995 ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 95), ACM Press, 1995, pp. 127–138.

value solely on the basis of shared attributes. The similarity estimator can employ any existing object-matching technique (see the sidebar for a discussion of these techniques). If the estimator decides that the similarity value is low, it discards the tuple pair; otherwise it passes them to the match filter.

Applying hard profilers in the match filer

To determine whether the tuple pair match, the match filter uses a set of hard profilers, which specify a concept's hard constraints. For example, a hard constraint on a movie review is that a review year must not precede the year that the movie was produced. As another example, a hard constraint on actors might specify that a specific actor has never played in a movie with an average rating of less than 4.

Several ways to build hard profilers exist. We can construct them manually, or build them automatically by examining domain data, assuming the data is complete. We can automatically generate hard constraints about an actress and her movie rating, for example, by examining all the movies she's appeared in. When users specify hard constraints, PROM treats them as a temporary hard profiler (see Figure 3). While other hard profilers cover general concepts and can be transferred across matching tasks, user-supplied hard constraints are typically task-specific and thus not transferable.

If any hard profiler says no, the match filter discards the tuple pair from further consideration. Otherwise, it passes them to the combiner for further evaluation.

Applying soft profilers in the combiner

The combiner uses a set of soft profilers, each of which issues a confidence score indicating how well the tuple pair fits its profile (and thus how well the two tuples' data might fit together). Like hard profilers, soft profilers cover a particular concept, but they specify soft constraints that concept instances will likely satisfy. A movie soft profiler, for example, might specify that the movie's IMDb (Internet Movie Database) rating and Ebert rating are strongly correlated when they differ by less than 3. Most movies would satisfy this constraint.

Soft profilers can be constructed in several ways. We can elicit them manually from domain experts and other users (then evaluate them on training data to obtain confidence scores). Or, we can build them on the basis of domain data, training a Bayesian network on IMDb's movie instances, for example. We can also build a soft profiler directly from a particular matching task's training data. Soft profilers are essentially classifiers; given a set of matching and nonmatching pairs, we can build one use virtually any learning technique.

Once the profilers issue their confidence scores, the combiner merges the scores to obtain a single overall score. On the basis of this score, it decides whether the tuple pair will likely match. If the decision is yes, it stores the pair in the result table Tj; otherwise it discards them.

Combining profilers

Because hard profilers issue yes-or-no predictions and soft profilers issue confidence scores, we separate the combination of the two profiler types. The match filter combines hard profilers, and the combiner handles soft profilers. We believe that separating the profilers improves matching accuracy over

7. S. Lawrence, K. Bollacker, and C.L. Giles, "Autonomous Citation Matching," *Proc. 3rd Int'l Conf. Autonomous Agents* (Agents 99), ACM Press, 1999, pp. 392–393.

8. A. McCallum, K. Nigam, and L. Ungar, "Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching," *Proc. 6th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (KDD 2000), 2000, ACM Press, pp. 169–178.

9. V. Raman and J. Hellerstein, "Potter's Wheel: An Interactive Data Cleaning System," *Proc. 27th Conf. Very Large Data Bases* (VLDB 2001), Morgan Kaufmann, pp. 381–390.

10. S. Sarawagi and A. Bhamidipaty, "Interactive Deduplication Using Active Learning," *Proc. 8th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (SIGKDD 02), ACM Press, 2002, pp. 269–278.

11. S. Tejada, C. Knoblock, and S. Minton, "Learning Domain-Independent String Transformation Weights for High Accuracy Object Identification," *Proc. 8th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (SIGKDD 02), ACM Press, 2002, pp. 350–359.

12. W. Yih and D. Roth, "Probabilistic Reasoning for Entity and Relation Recognition," *Proc. 19th Int'l Conf. Computational Linguistics* (COLING 02), Morgan Kaufmann, 2002.

13. A. Monge and C. Elkan, "The Field Matching Problem: Algorithms and Applications," *Proc. 2nd ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, AAAI Press, 1996, pp. 267–270.

14. W. Cohen and J. Richman, "Learning to Match and Cluster *Large High-Dimensional Data Sets for Data Integration*," *Proc. 8th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (KDD 02), ACM Press, 2002, pp. 475–480.

15. J. Kang and J. Naughton, "On Schema Matching with Opaque Column Names and Data Values," *Proc. 2003 ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 03), ACM Press, 2003, pp. 205–216.

16. W. Cohen and D. Kudenko, "Transferring and Retraining Learned Information Filters," *Proc. 14th Nat'l Conf. Artificial Intelligence* (AAAI 97), AAAI Press, 1997, pp. 583–590.

17. J. Berlin and A. Motro, "Database Schema Matching Using Machine Learning with Feature Selection," *Proc. 14th Int'l Conf. Advanced Information Systems Engineering* (CAiSE 02), LNCS 2348, Springer-Verlag, 2002, pp. 452–466.

18. H. Do and E. Rahm, "Coma: A System for Flexible Combination of Schema Matching Approaches," *Proc. 28th Conf. Very Large Databases* (VLDB 2002), Morgan Kaufmann, 2002, pp. 610–621.

19. A. Doan, P. Domingos, and A. Halevy, "Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach," *Proc. ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 01), ACM Press, 2001, pp. 509–520.

20. J. Madhavan et al., "Matching Schemas by Learning from a Schema Corpus," *Proc. IJCAI-03 Workshop Information Integration on the Web*, AAAI Press, 2003, pp. 59–65.

21. A. Rosenthal et al., "Data Integration Needs an Industrial Revolution," *Proc. Workshop Foundations of Models for Data Integration* (FMII 2001), 2001; www.fmldo.org/FMII-2001/proceedings.html.

methods that combine all profilers in a single stage; we are currently verifying this.

The match filter uses an AND combination to merge hard profiler predictions. That is, if any hard profiler says no, the overall prediction is no and PROM discards the tuple pair. The combiner merges soft profilers' predictions by computing the weighted sum of the confidence scores. Currently, we set the weights manually, on the basis of our experiments on holdout data. In the future, we'll explore methods to set weights automatically.[2]

Empirical evaluation

We evaluated PROM on two data sets, Citeseer and Movies. Because the two evaluations had similar results, we present only the Citeseer results here. We obtained the Citeseer data set from a Web page that lists highly cited authors and their homepages (http://citeseer.nj.nec.com/mostcited.html). For example, the page says, "J. Gray [1] [2] [3] [4] [5]" offering five homepage URLs suggested by a search engine. The homepages belong to various J. Grays: 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). So, the object-matching problem here is to match author names with their correct homepage URLs.

We downloaded the top 200 authors and their suggested homepages. Because we consider only matching relational tuples at this stage, we manually converted each homepage into a tuple by extracting homepage information such as name, name and rank of current university, position, and year that the person obtained his or her PhD. We also removed authors without homepages and performed some simple text processing. Our final data set consisted of 150 author names and 254 homepage tuples, for an average of 1.7 homepage tuples per author.

Algorithms and methodologies

We applied several algorithms to the Citeseer data set. First, we applied *Baseline*, an algorithm that matches tuples only on the basis of shared attributes—in this case, it matched the author names with homepage owner names. Baseline converts the values of the shared attributes into a set of tokens, then compares the token sets.

Next, we applied three algorithms that extend existing object-matching techniques to also exploit disjoint attributes.

- *Extended Manual* manually specifies the matching rules, such as `if similarity(name1, name2) >= 0.8 but position=student, then the two tuples do not match`. So, in a sense, this method extends the manual method that Mauricio Hernandez and Salvatore Stolfo described, which exploits only shared attributes such as `name1` and `name2`.[7]
- *Extended Association Rule* (Extended AR) is similar to Extended Manual but uses CMAR (*C*lassification Based on *M*ultiple *A*ssociation *R*ules)[8] to generate a set of rules. We then manually examine these rules to select a small set of rules.
- Unlike the two previous methods, *Extended Decision Tree* (Extended DT) is completely automatic. It extends the decision tree method of Sheila Tejada, Craig Knoblock, and Steven Minton[9] by adding to the training data all disjoint attributes and a new attribute that specifies a similarity value for each tuple pair (based on their shared attributes).

Table 1. Experimental results on the Citeseer data set.

	Extended algorithms				PROM			
	Baseline	Manual	AR	DT	DT	Man+DT	Man+AR	Man+DT+AR
Recall	0.99	0.97	0.96	0.91	0.95	0.67	0.96	0.97
Precision	0.67	0.83	0.71	0.58	0.78	0.87	0.82	0.86
F-value	0.80	0.89	0.81	0.71	0.85	0.76	0.88	0.91

Next, we applied PROM, using the *Baseline* algorithm described above for the similarity estimator. We used no hard profilers. We used three soft profilers: one based on soft, manually specified rules; another on decision tree techniques; and the third on association rule techniques. We then evaluated matching accuracy using three common object-matching measures:

- *Recall*: The number of correct matching pairs in the join table divided by the total number of correct matching pairs
- *Precision*: The number of correct matching pairs in the join table divided by the total number of pairs in the join table
- *F-value*: 2 ∗ recall ∗ precision/(recall + precision)

Together these measures suit our goal of developing matching methods that maximize precision and recall.

Results

We performed fourfold cross validation on the Citeseer data set and recorded the average recall, precision, and F-value. We took care to create folds that represent the overall data set (Mikhail Bilenko and Raymond Mooney describe similar fold creation[10]). Table 1 shows the results.

As the first column shows, Baseline achieves high recall (99 percent) but low precision (67 percent), demonstrating the inaccuracy of matching based on shared attributes only (in this case, names). Extended Manual's results (column two) show a slight recall decrease (2 percent) but a substantial precision increase (16 percent), demonstrating that exploiting disjoint attributes (in this case, any attribute other than names) can significantly boost matching accuracy. Extended AR produced similar (although slightly worse) results.

Extended DT produced surprising results (column four): its precision is substantially lower than that of Baseline (58 percent compared with 67 percent). This is unusual; because Extended DT exploits disjoint attributes, we might expect it to improve matching precision. However, many rules that Extended DT constructed didn't refer to the input tuples' similarity values at all. In other words, the rules matched tuples solely on the basis of the correlation among disjoint attributes, ignoring the shared attributes. Such rules would clearly not be very accurate on the testing data. So, extending prior matching techniques in a straightforward manner to handle disjoint attributes might actually decrease rather than increase matching accuracy.

With PROM, we wanted to examine both its performance with respect to Baseline and the extended algorithms and to discern whether adding more profilers would improve accuracy. So, we ran four variations of PROM (columns five through eight):

- DT used only one soft profiler (the decision tree method).
- Man+DT used the soft manual profiler and the soft decision tree profiler.
- Man+AR is similar to Man+DT but replaced the decision tree with the association rule classifier.
- Man+DT+AR is the complete PROM algorithm.

PROM's results show that the DT variation beats Extended DT's results, which suggests that extending prior matching techniques to exploit disjoint attributes using PROM is promising and potentially better than a straightforward extension of traditional techniques. The results also show that the complete PROM system (column eight) achieves the highest F-value (0.91) of any method, owing to high precision and recall. (In particular, this algorithm found the correct Jim Gray homepage, which *Baseline* could not.) Our results suggest both that PROM performs best and that adding more profilers might improve matching accuracy, because it would give PROM access to more matching knowledge.

We're experimenting with several new profiler training methods—including naive Bayes—and methods that require no training data. We also plan to use some of the profilers we've constructed for these matching tasks (such as the decision tree soft profiler) in related matching tasks to examine the effect of transferring such knowledge. We're particularly interested in training profilers on domain data, independently of matching tasks, then applying these profilers to matching tasks in the domain.

The PROM approach also suggests a broader knowledge-reuse methodology: within any particular task, we can isolate task-dependent knowledge (such as similarity knowledge) from task-independent knowledge (such as profile knowledge). Once PROM learns the latter, it can reuse it across tasks. Clearly, this reuse methodology is not always applicable, but it can be effective in appropriate settings, as we've demonstrated here. We aim to further explore its application. ⬒

References

1. H. Do and E. Rahm, "Coma: A System for Flexible Combination of Schema Matching Approaches," *Proc. 28th Conf. Very Large Databases* (VLDB 2002), Morgan Kaufmann, 2002, pp. 610–621.
2. A. Doan, P. Domingos, and A. Halevy, "Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach," *Proc. ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD 01), ACM Press, 2001, pp. 509–520.
3. J. Madhavan et al., "Matching Schemas by Learning from a Schema Corpus," *Proc. IJCAI-03 Workshop Information Integration on the Web*, AAAI Press, 2003, pp. 59–65.
4. M. Craven et al., "Learning to Construct Knowledge Bases from the World Wide Web," *Artificial Intelligence,* vol. 118, nos. 1-2, 2000, pp. 69–113.
5. D. Freitag, "Multistrategy Learning for Information Extraction," *Proc. 15th Int'l Conf. Machine Learning* (ICML 98), Morgan Kaufmann, 1998, pp. 161–169.
6. R. Ananthakrishna, S. Chaudhuri, and V. Ganti, "Eliminating Fuzzy Duplicates in Data Warehouses," *Proc. 28th Int'l Conf. Very Large Databases* (VLDB 2002). Morgan Kaufmann, 2002, pp. 586–597.
7. M. Hernández and S. Stolfo, "The Merge/Purge Problem for Large Databases,"

Proc. 1995 ACM SIGMOD Int'l Conf. Management of Data (SIGMOD 95), ACM Press, 1995, pp. 127–138.

8. W. Li, J. Han, and J. Pei, "CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules," *Proc. Int'l Conf. Data Mining* (ICDM 01), IEEE CS Press, 2001, pp. 369–376.

9. S. Tejada, C. Knoblock, and S. Minton, "Learning Domain-Independent String Transformation Weights for High Accuracy Object Identification," *Proc. 8th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (SIGKDD 02), ACM Press, 2002, pp. 350–359.

10. M. Bilenko and R. Mooney, *Learning to Combine Trained Distance Metrics for Duplicate Detection in Databases*, tech. report AI 02-296, Artificial Intelligence Laboratory, Univ. Texas at Austin, 2002.

The Authors

AnHai Doan is an assistant professor of computer science at the University of Illinois at Urbana-Champaign. His research interests are in databases and artificial intelligence, including data integration and sharing, schema matching, data mining, information discovery on the Web, metadata management, the Semantic Web, and machine learning. He received his PhD in computer science from the University of Washington. Contact him at the Dept. of Computer Science, Univ. of Illinois, Urbana, IL 61801; anhai@cs.uiuc.edu.

Ying Lu is a PhD student in computer science at the University of Illinois at Urbana-Champaign. Her research interests include data mining, data integration, and bioinformatics. She received her MS in computer science from the University of Wisconsin-Madison. Contact her at the Dept. of Computer Science, Univ. of Illinois, Urbana, IL 61801; yinglu@cs.uiuc.edu.

Yoonkyong Lee is a PhD student in computer science at the University of Illinois at Urbana-Champaign. Her research interest is in data integration. She holds a BS in computer science from Korea Advanced Institute of Science and Technology. Contact her at the Dept. of Computer Science, Univ. of Illinois, Urbana, IL 61801; ylee11@cs.uiuc.edu.

Jiawei Han is a professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. His research interests are in data mining, data warehousing, spatial and multimedia databases, deductive and object-oriented databases, and biomedical databases. He is the lead author of *Data Mining: Concepts and Techniques* (Morgan Kaufmann, 2001). He serves or has served on the editorial boards of *Data Mining and Knowledge Discovery: An International Journal*, *IEEE Transactions on Knowledge and Data Engineering*, and the *Journal of Intelligent Information Systems*. He has received the IBM Faculty Award and the ACM Service Award. Contact him at the Dept. of Computer Science, Univ. of Illinois, Urbana-Champaign, IL 61801; hanj@cs.uiuc.edu.