Relational Clustering for Multi-type Entity Resolution

Indrajit Bhattacharya
Department of Computer Science
University of Maryland
College Park, MD 20742, USA
indrajit@cs.umd.edu

Lise Getoor
Department of Computer Science
University of Maryland
College Park, MD 20742, USA
getoor@cs.umd.edu

ABSTRACT

In many applications, there are a variety of ways of referring to the same underlying entity. Given a collection of references to entities, we would like to determine the set of true underlying entities and map the references to these entities. The references may be to entities of different types and more than one type of entity may need to be resolved at the same time. We propose similarity measures for clustering references taking into account the different relations that are observed among the typed references. We pose typed entity resolution in relational data as a clustering problem and present experimental results on real data showing improvements over attribute-based models when relations are leveraged.

1. INTRODUCTION

In many applications, there are a variety of ways of referring to the same underlying entity. Given a collection of references to entities, we would like to a) determine the collection of 'true' underlying entities and b) correctly map the entity references in the collection to these entities. This problem comes up in many guises throughout computer science. Examples include computer vision, where we need to figure out when regions in two different images refer to the same underlying object (the correspondence problem); natural language processing when we would like to determine which noun phrases refer to the same underlying entity (co-reference resolution); and databases, where, when merging two databases or cleaning a database, we would like to determine when two tuple records are referring to the same real world object (deduplication and/or record linkage).

Deduplication is important in data cleaning for both accurate analysis and for cost-effectiveness. In information integration, determining approximate joins is necessary for consolidating information from multiple sources; most often there will not be a unique key that can be used to join tables in distributed databases, and we must infer when two records from different databases, possibly with different schemas, refer to the same entity.

Often times data may have errors, for example typographical errors, or multiple representations, such as abbreviations, so an exact comparison does not suffice for resolving duplicates in such cases. Traditional approaches to entity resolution are based on approximate string matching criteria. More recent approaches also take relational similarity into account. For example, if we are comparing two census records for 'Jon Doe' and 'Jonathan Doe', they are more likely to be the same individual if they are both married to 'Jeannette Doe'. In other words, the string similarity of the attributes is taken into account, but so too is the similarity of the people to whom the person is related.

The problem becomes more interesting when we do not assume that the related entities have already been resolved. In fact, determining that two records refer to the same individual may in turn allow us to make additional inferences. In other words, the resolution process becomes *iterative*. Many domains involve multiple types of entities, many of which may be potentially ambiguous. The problem then involves resolving multiple types of entities at the same time using the different relations that are observed among them.

The entity resolution problem has generated considerable interest in the machine learning literature recently [24, 16, 23, 15] and some of these approaches resolve multiple types of entities[24, 23, 15]. In this paper, we pose typed entity resolution in relational data as a clustering problem. We propose similarity measures that consider relations between entities of multiple types in addition to their attributes. We also present preliminary experimental results on two real citation datasets that demonstrate the advantages of relational entity resolution over attribute-only baselines.

2. RELATED WORK

There has been a large body of work on deduplication, record linkage, and co-reference resolution. Here we review some of the main work, but the review is not exhaustive. For a nice summary report, see Winkler [29].

The traditional approach to entity resolution looks at textual similarity in the descriptions of the entities. There has been extensive work on defining approximate string similarity measures [19, 21, 7, 6] that may be used for unsupervised entity resolution. The other approach is to use adaptive supervised algorithms that learn string similarity measures from labeled data [26, 5, 8, 28].

The problem is known to be hard computationally. So an important focus is on efficiency issues in data cleaning, where the goal is to come up with inexpensive algorithms

This paper appears in the Proceedings of the Fourth International Workshop on Multi-Relational Data Mining (MRDM-2005), August 21, 2005, Chicago. The proceedings were edited by Sašo Džeroski and Hendrik Blockeel.

for finding approximate solutions to the problem. The key mechanisms for doing this involve computing the matches efficiently and employing techniques commonly called 'blocking' to quickly find potential duplicates [13, 20, 17].

The groundwork for posing record linkage as probabilistic classification problem was done by Fellegi and Sunter [11], who labeling pairs of records from two different files to be merged as "match" or "non-match" on the basis of agreement among their different fields. Winkler [30] and more recently Ravikumar et al [25] have built upon this work.

Approaches that take relational features into account for data integration have been proposed [1, 3, 14, 9]. Ananthakrishna et al [1] introduce relational deduplication in data warehouse applications where there is a dimensional hierarchy over the relations. Kalashnikov et al [14] enhance feature-based similarity between an ambiguous reference and the many entity choices for it with relationship analysis between the entities, like affiliation and coauthorship. Bhattacharya and Getoor [3] propose different measures for relational similarity and show how this can be combined with attribute similarity for improved author resolution in synthetic collaboration graphs. Dong et al [9] resolve enitites of multiple types by propagating relational evidences in a dependency graph. They adopt a pair-wise reconciliation approach so that the graph has nodes for all potential duplicate pairs and all pairs of similar attributes. Further, they have a unique node in the graph for each attribute value pair, thereby resorting to 'global' resolution which we discuss in Section 3.3. Emde and Wettschereck [10] and Neville et al [22] have proposed similarity measures for relational data but not in the context of entity resolution.

Probabilistic models that take into account interaction between different entity resolution decisions have been proposed for named entity recognition in natural language processing and for citation matching. McCallum et al [16] use conditional random fields for noun coreference and use clique templates with tied parameters to capture repeated relational structure. Li et al [15] address the problem of disambiguating "entity mentions", potentially of multiple types, in the context of unstructured textual documents. They propose a probabilistic generative model that captures a joint distribution over pairs of entities in terms of co-mentions in documents. Parag et al [23] use the idea of merging evidence to allow the flow of reasoning between different pair-wise decisions over multiple entity types. Pasula et al[24] propose a generic probabilistic relational model framework for posing the citation matching problem. They resort to sampling algorithms for reasoning over the unknown set of entities. Milch et al [18] present a formal generative language for defining probability distribution over worlds with unknown objects and identity uncertainty. Bhattacharya and Getoor [4] propose a generative group model for joint entity resolution. Instead of performing a pair-wise comparison task, we use a latent group variable for each reference to predict the entity label.

3. A MOTIVATING EXAMPLE

We will now motivate the problem of relational entity resolution and highlight the different issues that come up using an illustrative example from the bibliographic domain. Consider the problem of trying to construct a database of papers, authors and citations, from a collection of paper references, perhaps collected by crawling the web, like CiteSeer [12]. Such systems often have multiple references to the same paper, citations are not always resolved and authors are not always correctly identified [27, 24].

The most commonly studied bibliographic entity resolution task is resolving paper citations. Consider the following example from [27]:

- R. Agrawal, R. Srikant. *Fast algorithms for mining association rules in large databases.* In VLDB-94, 1994.
- Rakesh Agrawal and Ramakrishnan Srikant. *Fast Algorithms for Mining Association Rules.* In Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994.

These very different strings are citations of the same paper and clearly string edit distance will not work. Sometimes, paper resolution can be done based simply on the title. We can use one of the many existing methods for string matching, perhaps even tuned to the task of title matching. However, there is additional relational information, in terms of the venue, the authors of the paper, and the citations made by the paper; this additional information may provide further evidence to the fact that two references are the same. This type of entity resolution has been the focus of much of the work in citation matching [17, 24, 27].

3.1 Multiple Entity Resolution

While the above two citation strings were used to motivate the paper citation problem, it is more interesting to note that they present an illustrative example of multi-entity resolution. To elaborate, the strings refer to papers, but in addition to the paper title, contain references to other kinds of entities, which themselves may be ambiguous. This brings up a scenario where multiple types of entities need to be resolved simultaneously. In particular, the strings refer to *author* and *venue* entities in addition to *paper* entities. Assuming that the strings have been correctly parsed to separate out the different references — which is a difficult problem by itself — the first citation string mentions 'R. Agrawal' and 'R. Srikant' as the authors and 'VLDB-94, 1994' as the venue, while the second has 'Rakesh Agrawal' and 'Ramakrishnan Srikant' as the authors and 'Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994' as the venue reference. Not all of these pairs are easy to disambiguate individually. While it may not be too difficult to resolve 'R. Srikant' and 'Ramakrishnan Srikant', 'Agrawal' is an extremely common Indian last name and it is certainly not obvious that 'R. Agrawal' refers to the same author entity as 'Rakesh Agrawal'. As for the venue references, it is very difficult, if not impossible, to resolve the two venue references without specialized domain knowledge.

3.2 Joint Resolution Using Entity Relations

For our example, it is not hard to observe the dependence among the different resolution decisions across multiple types. We can make the resolution decisions *depend* on each other in cases were they are *related*. Significantly, when the resolution decisions are made *collectively*, they can be much easier to make. Let us illustrate the relations for the citation domain, where the relevant entities are papers, authors and venues. Authors *write* papers, which get *published in* venues. An author is a *collaborator* of another author if

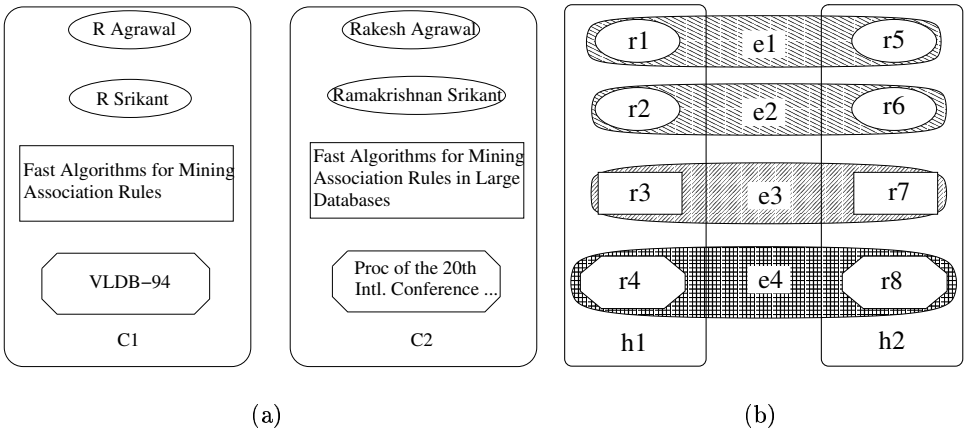


Figure 1: (a) Example citations parsed into observed author, paper title and venue strings. (b) The corresponding author, paper and venue references with the resolved entities.

they write the same paper. We may use these relations to make one resolution decision lead to another. We may begin by resolving 'R. Srikant' and 'Ramakrishnan Srikant', of which we are most confident. This may lead us to believe 'R. Agrawal' and 'Rakesh Agrawal' are the same author, since they *collaborate* with the resolved author entity 'Srikant'. Now we can go back to the paper citations which, in addition to having very similar titles, have now been determined to be *written by* the same author entities. This makes us more confident that the two paper references map to the same paper entity. Following the same thread, two identical papers citations must have identical venue citations. So we may resolve the apparently disparate venue references.

Similar dependencies between pairwise resolution decisions in the presence of relations among references can be captured by the relational markov network model for entity resolution proposed by Parag et al [23]. Note that joint resolution can be performed even when the observed data is not relational. Different pairwise decisions can still be made to depend on each other using two main ideas. The first uses tied parameters for repeating features over pairs of references and their resolution decisions. The second source of dependence is via overlaps between pairs of references. Two different decisions involving the same reference are dependent and this dependence can flow over 'chains' of overlapping pairs. This flavor of joint resolution was introduced by McCallum et al [16]. As we have illustrated using our example, other relations between references, when available in the data, can be taken into account to introduce and leverage more resolution dependencies.

3.3 Local And Global Resolution

It has been noted [23, 9] that some resolution decisions may be 'exported' to apply in other cases as well. For example, the *local resolution* made in the case of the two venue strings 'Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994' and 'VLDB-94, 1994' in the context of our example papers may be extended for another set of papers that have the same venues to resolve them. This we will call *global resolution* for some attribute values for an entity type. However, it may not always be appropriate to export local resolutions. In case of names for example, while 'R. Agrawal' refers to the 'Rakesh Agrawal' entity in this case, some other instance of it might refer to a different 'Rajeev Agrawal'.

3.4 Positive And Negative Relational Evidence

Apart from the ones illustrated in our example, other bibliographic relations can also potentially be used as additional evidence when available. Papers are *cited by* other papers and two papers are *co-cited* if they are cited by the same paper entity. If two similar paper references are co-cited by the same paper entity, that may serve as additional evidence that they are the same. However, relational evidence may also be *negative* in some cases. For example, two paper references that are *cited by* the same paper entity are unlikely to be duplicates, as are two author references that are *co-authors* of the same paper entity.

4. PROBLEM FORMULATION

In the relational entity resolution problem, we have some collection of references to entities and from this set of references we would like to identify the (unique, minimal) collection of individuals or entities to which they should be mapped. In what follows, we will use lower case characters e and r to denote entities and references. We will use qualified upper case letters like $e.A$ and $r.E$ to denote variables, and we will use $e.a$ and $r.e$ to denote values of variables (short-hand for $e.A = a$).

We are given a set of references $\mathcal{R} = \{r_i\}$, where each reference r has its attributes $r.A$. The references are typed. We are given the possible types $\mathcal{T} = \{t_i\}$ in the domain and the type $r.T$ for each reference is also observed. The references correspond to entities $\mathcal{E} = \{e_i\}$ so that each reference r has its hidden entity label $r.E$. Note that a reference and its corresponding entity are necessarily of the same type. Each entity e also has its own attributes $e.A$, but the entities are not directly observed. What we observe are the attributes $r.A$ of individual references. We can imagine $r.A$ to be generated by some distortion process from the attributes of the corresponding entity $r.E$. Obviously, the entity labels of the references are not observed. Our problem is to recover the

hidden set of entities $\mathcal{E} = \{e_i\}$ and the entity labels $r.E$ of individual references given the observed attributes and types of the references.

We will consider relational information between references to help us in collective entity resolution. We will assume that the references are observed, not individually, but as members of hyper-edges. We are given a set of hyper-edges $\mathcal{H} = \{h_i\}$ and the membership of a reference in a hyper-edge is captured by its hyper-edge label $r.H$. If reference r occurs in hyper-edge h, then $r.H = h$. Note that unlike the entity labels, we *know* the association of hyper-edges and references. Any hyper-edge may relate references of arbitrary types. The hyper-edges can help us make better predictions if we assume that they are indicative of associative patterns among the entities. In other words, the entity labels of references that occur in the same hyper-edge are related to each other. Now the resolution decisions are not independent. Instead of finding the entity labels for each reference individually, our task is to predict the entity labels of the references collectively, where the entity label $r.E$ of any reference r is directly influenced by the choice of entity label $r'.E$ for another reference r' if they are associated with the same hyper-edge, ie. $r.H = r'.H$.

To make this more concrete, consider the earlier example shown in Figure 1(a) and the corresponding references resolved into entities in Figure 1(b). We have a total of eight references with four in each citation. In citation C1, r_1 and r_2 are author references with attributes $r_1.A =$'R. Agrawal' and $r_2.A =$'R. Srikant' and types $r_1.T = r_2.T =$'author'. Reference r_3 is of type paper, so that $r_3.T =$'paper' and $r_3.A =$'Fast Algorithms for Mining Association Rules'. The fourth reference r_4 is a venue with $r_4.T =$'venue' and $r_4.A =$ 'VLDB-94'. Similarly, citation C4 has four references; r_5 and r_6 of type 'author', r_7 of type 'paper' and r_8 of type 'venue'.

The set of underlying entities $\mathcal{E}$ is $\{e_1, e_2, e_3, e_4\}$, where e_1 and e_2 are the author entities with names $e_1.A =$'Rakesh Agrawal' and $e_2.A =$'Ramakrishnan Srikant', e_3 is a paper with title $e_3.A =$'Fast Algorithms for Mining Association Rules in Large Databases' and e_4 is a conference venue with name $e_4.A =$'20th International Conference on Very Large Data Bases, 1994'. Clearly, author references r_1 and r_5 correspond to author entity e_1 so that $r_1.E = r_5.E = e_1$, while r_2 and r_6 correspond to author entity e_2 so that $r_2.E = r_6.E = e_2$. Also, paper references r_3 and r_7 correspond to entity e_3 and venue references r_4 and r_8 to entity e_4.

We also have the hyper-edges, which correspond to citations. Here we have two citations, so that $\mathcal{H} = \{h_1, h_2\}$. The references r_1 through r_4 are associated with hyper-edge h_1, since they are all obtained from citation C1. This relation is represented as $r_1.H = r_2.H = r_3.H = r_4.H = h_1$. We similarly associate the other four references with hyper-edge h_2. In rest of this paper, we will use the term edge as a substitute for hyper-edge. It will be understood that an edge may involve more than two references.

5. RESOLUTION BY CLUSTERING

The task of collective entity resolution may be rephrased as a relational clustering problem where the goal is to cluster the references so that those that correspond to the same entity end up in the same cluster. We may imagine a greedy agglomerative clustering algorithm, where at any stage the current set $\mathcal{C} = \{c_i\}$ of *entity clusters* reflects our current belief about the underlying entities. In other words, each entity cluster corresponds to one reconstructed entity and all references in a cluster correspond to the same entity. For our running example, we would like the algorithm to find four clusters, one for each underlying entity. Membership of references in the constructed clusters are represented using a cluster label $r.C$ for each reference. All references that are members of a cluster need to be of the same type, which is also the type $c.T$ for the cluster. To start off, each reference belongs to a separate cluster and at each step the pair of clusters (or entities) that have the highest evidence for being the same entity are merged. The key to the success of a clustering algorithm is the distance measure (or alternatively, the similarity measure) that is employed. The similarity measure takes into account the similarity in the attributes of the references in the two clusters. In addition, we want the similarity measure to incorporate the idea of joint or collective resolution. The measure should take into account the *related* resolution decisions that have been made previously. Accordingly, the similarity measure needs to be extended to consider both reference attributes and relational patterns.

The *attribute similarity component* of the similarity measure looks at the similarity of the attributes $r.A$ of the references in the two clusters. In our case, it measures the similarity between two observed reference names. In addition, we have the *relational similarity component* that looks at the similarity of the relations that the two entities or clusters participate in. Each cluster is associated with a set of edges to other references that we want to take into account. For the other references, we want look not at their attribute values but at the resolution decisions that have been taken on them. Specifically, we want to look at the cluster labels $r.C$ of these references. To illustrate this using our example from Figure 1, suppose we have already resolved the 'Agrawal' and 'Srikant' references in clusters c_1 and c_2 respectively. Consider the current similarity of the two paper references which are yet to be resolved and still belong to separate entity clusters, say c_{3a} having r_3 and c_{3b} having r_7. The two clusters are associated with one edge each, h_1 and h_2 respectively. We want to factor into the similarity measure the fact that the two edges are associated with the same resolved clusters c_1 and c_2, which is what makes the clusters c_{3a} and c_{3b} similar.

An issue that is brought out by the above discussion is the dynamic nature of the relational similarity component. Initially, when all references belong to distinct clusters, the two paper clusters will not be considered similar enough. But their relational similarity goes up in stages as first the 'Srikant' references and then the 'Agrawal' references are resolved. In the following sections, we describe an iterative clustering algorithm that leverages this dynamic nature of relational similarity.

6. SIMILARITY MEASURES

In this section, we formally define the similarity measure between two entity clusters as a weighted combination of the attribute similarity and relational similarity between them and highlight the computational and other algorithmic issues that are involved.

For two entity clusters c_i and c_j, their similarity may be

defined as

$$sim(c_i, c_j) = (1-\alpha) \times sim_{attr}(c_i, c_j) + \quad \alpha \times sim_{rel}(c_i, c_j)$$
$$0 \leq \alpha \leq 1$$

where $sim_{attr}()$ is the similarity of the attributes and $sim_{rel}()$ is the relational similarity between the two entity clusters and they are linearly combined with weights α and $1-\alpha$. In the following two subsections, we discuss the two similarity components in detail.

6.1 Attribute Similarity

We assume the existence some basic similarity measure that takes two reference attributes and returns a value between 0 and 1 that indicates the degree of similarity between them. A higher value indicates greater similarity between the attributes. We are not making any other assumptions about the attribute similarity measure. Any measure that satisfies these assumptions can be used in our scheme. We may potentially use different attribute similarity measures for different clusters pairs depending on their types.

However, $sim_{attr}()$ defines the similarity of attributes between two entity clusters. Each entity cluster is a collection of references with their own attributes. So we need to use the similarity measure that takes two attributes to define the similarity between two clusters of attributes. This is similar to finding the aggregate distance between two clusters given a pairwise distance measure. Many approaches have been proposed for aggregating over pair-wise distances between the two clusters [2]. The duplicate relation is typically transitive: if references r_i are r_j are duplicates, then all other duplicates of r_i will also be duplicates of r_j. So the single link measure that takes the minimum pair-wise distance (or the maximum pairwise similarity) between two clusters is the most relevant for our purposes. Computing single link similarity is quadratic in the average number of distinct attribute values in a cluster. A more efficient alternative to pairwise comparison might be to maintain a representative attribute for each entity cluster, which is the most likely attribute value for the underlying entity given the observed attributes of all references in that cluster. Then the attribute similarity for a cluster pair is the similarity of their representative attributes.

6.2 Relational Similarity

Next, we address the relational similarity measure between two entity clusters considering the clusters that they are related to via the observed edges. There are many possible ways to define this similarity. We explore some possibilities here and focus on some issues that are of relevance.

6.2.1 Edge Detail Similarity

Each entity cluster is associated with a set of edges that the references contained in it belong to. Recall that each reference r is associated with an observed hyper-edge $r.H$. Then the edge set $c.H$ for an entity cluster c may be defined as $c.H = \{h \mid r.H = h \wedge r.C = c\}$. To ground this in terms of our example in Figure 1, after we have resolved the 'Agrawal' references so that cluster c_1 contains references r_1 and r_5, then the edge set for the cluster is $c_1.H = \{h_1, h_2\}$ having the edges corresponding to the two citations involving 'Agrawal'. We will now define a similarity measure for a pair of edges, so that given this pair-wise similarity measure, we can again use a linkage metric like single link to measure the relational similarity between two clusters.

Let us first define a pairwise similarity measure between two hyper-edges. We have already noted that what we want to consider for relational similarity are the the cluster labels of the references in each edge, and not their attributes. So for an edge, we consider the multi-set of cluster labels, one for each reference associated with it. However, the references and accordingly the cluster labels in a hyper-edge can be of multiple types. To look at the clusters of each type separately, we define a type projection for an edge as the set of cluster labels from it that are of a particular type. For example, the 'author projection' for edge h_1 in our example has the two author clusters c_1 and c_2. Formally, for a hyper-edge h and a type t, the projection $\Pi_t(h)$ is defined as

$$\Pi_t(h) = \{c \mid r.C = c \wedge r.H = h \wedge c.T = t\}$$

To check two edges for similarity, we will consider their projections for each type separately. A common measure for set similarity is the Jaccard similarity. For two sets A and B, their Jaccard similarity is defined as $Jaccard(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|}$. Then the similarity of two edges h_i and h_j for a particular type t is

$$sim_t(h_i, h_j) = Jaccard(\Pi_t(h_i), \Pi_t(h_j))$$

The overall similarity for the edges may then be obtained by using some appropriate aggregation operation Σ over all types:

$$sim(h_i, h_j) = \Sigma_t[sim_t(h_i, h_j)]$$

The aggregation operation could be a max or a weighted combination over the types.

Given this pairwise similarity measure for edges, we can use an aggregation operation like max to calculate the relational similarity between two entity clusters as follows:

$$sim_{rel}(c_i, c_j) = \quad max_{(h_i, h_j)}\{sim(h_i, h_j)\}$$
$$h_i \in c_i.H, h_j \in c_j.H$$

This we will call the *edge detail similarity* between two clusters since it explicitly considers every edge associated with each cluster. Observe that computing the similarity between two edges is linear in the average number of references in an edge, while computing the edge detail similarity of two clusters is quadratic in the average number of edges associated with each cluster.

6.2.2 Neighborhood Similarity

Clearly, an issue with edge detail similarity is the computational complexity. Also, it is not trivial to incrementally update edge detail similarity when clusters merge. A second and probably more pertinent issue is whether the detailed pair-wise similarity computation for edges is really necessary for the task at hand. While it may make sense for some other applications, it may not be necessary to look at the structure of each edge separately for the task of relational entity resolution. Using a more concrete example, for two author entities e_1 and e_1' to be considered similar, it is not necessary for both of them to co-author a paper with entities e_2, e_3 and e_4 together. Instead, if e_1 participates in an edge $\{e_1, e_2, e_3, e_4\}$ and e_1' participates in three separate

edges $\{e'_1, e_2\}$, $\{e'_1, e_3\}$ and $\{e'_1, e_4\}$, then that also should count as significant relational evidence for their being the same author entity (if they have similar attributes as well). In other words, whether or not two entity clusters with similar attributes have the same edge structures, if they have the *same neighbor clusters* to which they have edges, that is sufficient relational evidence for bibliographic entity resolution.

We will now formalize the notion of neighborhood for a cluster. Recall that we have defined $\Pi_t(h))$ as the typed projection of cluster labels for an edge h. Also, for an entity cluster c, $c.H$ is the set of edges associated with it. Then we can formally define the typed neighborhood $N_t(c)$ for c as follows:

$$N_t(c) = \bigcup_m \Pi_t(h), \quad h \in c.H$$

Note that $\bigcup_m$ is the multi-set union operator and the neighborhood is also a multi-set in general. Intuitively, we collapse the edge structure and for each type just look at how many times c has participated in the same edge with a cluster of that particular type. Going back to our example from Figure 1, suppose the 'Agrawal' and 'Srikant' references have been resolved into author clusters c_1 and c_2 but the other references are still dispersed over paper clusters c_{3a} and c_{3b} and venue clusters c_{4a} and c_{4b}. Then the author neighborhoods for the paper clusters c_{3a} and c_{3b} are identical: $N_{author}(c_{3a}) = N_{author}(c_{3b}) = \{c_1, c_2\}$. But their venue neighborhoods are different: $N_{venue}(c_{3a}) = \{c_{4a}\}$ and $N_{venue}(c_{3b}) = \{c_{4b}\}$.

Now, for the relational similarity measure between two entity clusters for a particular type, we take the Jaccard similarity between their neighborhoods of that type. Note that Jaccard similarity can be naturally extended for multi-sets. Finally, for the overall similarity between clusters we can aggregate over all types as before:

$$sim_{rel}(c_i, c_j) = \Sigma_t[Jaccard(N_t(c_i), N_t(c_j))]$$

We will call this the *neighborhood similarity* between two clusters. Computing and updating neighborhood similarity is significantly cheaper computationally compared to the *edge detail similarity*. It is linear in the average number of neighbors per entity.

6.3 Handling Negative Relational Evidence

So far, we have considered relational structure as additional evidence for two author references actually referring to the same underlying author entity. However, relational evidence can be negative as well. A 'soft' aspect of negative evidence is directly captured by the combined similarity measure. Imagine two references with identical names. If we only consider attributes, their similarity would be very high. However, if they do not have any similarity in their edge structures, then we are less inclined to believe that they correspond to the same entity. This is reflected by the drop in the their overall similarity when the relational similarity measure is factored in as well.

We may also imagine stronger relational constraints for clustering. In many relational domains, there is the constraint that no two references appearing in the same edge can be duplicates of each other. In our example, no matter how similar the names 'R. Srikant' and 'R. Agrawal' are deemed to be by the attribute similarity measure that is employed, they *cannot* be the same entity since they are co-authors. We have such constraints for every edge that has more than one reference. This can be taken into account by the relational similarity measure. The similarity between two cluster pairs is zero if merging them violates any relational constraint.

7. IMPLEMENTATION

Given the similarity measure for a pair of clusters, we can use a greedy agglomerative clustering algorithm that finds the closest cluster pair at each step and merges them. Here we discuss several implementation and performance issues regarding our relational clustering algorithms for entity resolution (**RC-ER**).

The algorithm inserts all candidate duplicate pairs — identified using a 'blocking' approach — into a priority queue considering their similarities. Then it iteratively picks the pair with the highest similarity and merges them. The algorithm terminates when the similarity for the closest pair falls below a threshold.

The relational clustering algorithm begins with each reference assigned to a separate entity cluster. So to start with, the clusters are disconnected and there is no relational evidence at all between clusters. As a result, all initial merges would occur based solely on attribute similarity, which is what we want to avoid. To deal with this, we would like to bootstrap the clusters so that some relational evidence may be leveraged. It would also speed up the algorithm if initial clusters could be merged without similarity computations and updations. One option is to assign the same initial cluster to any two references that have attributes v_1 and v_2, where either v_1 is identical to v_2, or v_1 is an initialed form of v_2. For example, we may merge 'Alfred Aho' references with other 'Alfred Aho' references or with 'A. Aho' references. However, for domains where last names repeat very frequently, like Chinese, Japanese or Indian names, this can affect precision quite adversely. For the case of such common last names[1], the same author label can be assigned to pairs only when they have document co-authors with identical names as well. For example, two 'X. Wang' references will be merged when they are co-authors with 'Y. Li'. (We will also merge the 'Y. Li' references.) This should improve bootstrap precision significantly under the assumption that while it may be common for different authors to have the same (initialed) name, it is extremely unlikely that they will collaborate with the same author, or with two other authors with identical names.

In addition to using a secondary source for determining common names, a data-driven approach may also be employed. A [last name, first initial] combination in the data is ambiguous, if there exist multiple first names with that initial for the last name. For example, though 'Zabrinsky' is not a common last name, 'K. Zabrinsky' will be considered ambiguous if 'Ken Zabrinsky' and 'Karen Zabrinsky' occur as author references in the data. Ambiguous references or references with common last names are not bootstrapped in the absence of relational evidence in the form of co-authorships, as described above.

Once potential duplicate entity clusters have been identified and clusters have been bootstrapped, the algorithm

[1]A list of common last names is available at http://en.wikipedia.org/wiki/List_of_most_popular_family_names

iterates over the following steps. At each step, it identifies the currently closest pair of clusters (c_i, c_j) from the candidate set and merges them to create a new cluster c_{ij}. It removes from the candidate set all pairs that involve either c_i or c_j and inserts relevant pairs for c_{ij}. It also updates the similarity measures for the 'related' cluster pairs. All of these tasks are performed efficiently using an indexed priority queue to make the algorithm scalable.

8. EXPERIMENTAL RESULTS

We perform evaluations on two citation datasets from different research areas and compare the relational entity resolution algorithm (**RC-ER**) with others based solely on attributes. Here we report results for the single entity case where we only resolve references of the author type. We do not resolve venues or papers.

The first of the citation datasets is the CiteSeer dataset containing citations to papers from four different areas in machine learning, originally created by Giles et al.[12]. This has 2,892 references to 1,165 authors, contained in 1,504 documents. The second dataset is significantly larger; arXiv (HEP) contains papers from high energy physics used in KDD Cup 2003[2]. This has 58,515 references to 9,200 authors, contained in 29,555 papers. The authors for both datasets have been hand-labeled.[3]

To evaluate the algorithms, we measure the performance of the algorithms for detecting duplicates in terms of the traditional precision, recall and F1 on pairwise duplicate decisions. It is practically infeasible to consider all pairs, particularly for HEP, so a 'blocking' approach is employed to extract the potential duplicates. This approach retains $\sim$ 99% of the true duplicates for both datasets. The number of potential duplicate pairs of author references after blocking is 13,667 for CiteSeer and 1,534,661 for HEP.

As our baseline (**ATTR**), we compare with the hybrid *SoftTF-IDF* measure [7] that has been shown to outperform other unsupervised approaches for text-based entity resolution. Essentially, it augments the TF-IDF similarity for matching token sets with approximate token matching using a secondary string similarity measure. Jaro-Winkler is reported to be the best secondary similarity measure for *SoftTF-IDF*. We also experiment with the Jaro and the Scaled Levenstein measures. **ATTR** only reports pairwise match decisions. Since the duplicate relation is transitive, we also evaluate **ATTR*** which removes inconsistencies in the pairwise match decisions in **ATTR** by taking a transitive closure. Note that this issue does not arise with **RC-ER**; it does not make pairwise decisions. All of these unsupervised approaches **ATTR**, **ATTR*** and **RC-ER** need a similarity threshold for deciding duplicates. We consider the best $F1$ that can be achieved over all thresholds.

Table 1 records F1 achieved by the four algorithms with various string similarity measures coupled with SoftTF-IDF while Table 2 shows the best F1 and the corresponding precision and recall for the four algorithms for each dataset over all secondary similarity measures. The recall includes blocking, so that the highest recall achievable is 0.993 for CiteSeer and 0.991 for HEP.

[2]http://www.cs.cornell.edu/projects/kddcup/index.html

[3]We would like to thank Aron Culotta and Andrew McCallum for providing the author labels for the CiteSeer dataset and David Jensen for providing the author labels for the HEP dataset. We performed additional cleaning for both.

Table 1: Performance of ATTR, ATTR* and RC using neighborhood and edge detail similarity in terms of F1 using various secondary similarity measures with SoftTF-IDF. The measures compared are Scaled Levenstein (SL), Jaro (JA), and Jaro Winkler (JW).

	CiteSeer			HEP		
	SL	JA	JW	SL	JA	JW
ATTR	0.980	**0.981**	0.980	**0.976**	0.976	0.972
ATTR*	0.989	**0.991**	0.990	**0.971**	0.968	0.965
RC(Nbr)	**0.994**	**0.994**	**0.994**	0.979	**0.981**	0.981
RC(Edge)	**0.995**	**0.995**	**0.995**	0.982	**0.983**	0.982

Table 2: Best F1 and corresponding precision and recall for ATTR, ATTR* and RC-ER with neighborhood and edge detail similarity for CiteSeer and HEP datasets.

	CiteSeer			HEP		
	P	R	F1	P	R	F1
ATTR	0.990	0.971	0.981	0.987	0.965	0.976
ATTR*	0.992	0.988	0.991	0.976	0.965	0.971
RC(Nbr)	0.998	0.991	**0.994**	0.990	0.972	**0.981**
RC(Edge)	0.997	0.993	**0.995**	0.992	0.974	**0.983**

The best baseline performance is with Jaro as secondary string similarity for CiteSeer and Scaled Levenstein for HEP. Transitive closure affects the baseline differently in the two datasets. While it adversely affects precision for HEP reducing the F1 measure as a result, it improves recall for CiteSeer and thereby improves F1 as well.

RC-ER outperforms both forms of the baseline for both datasets. Also, for each secondary similarity measure **RC-ER** with neighborhood similarity outperforms the baselines with that measure and is in turn outperformed by **RC-ER** using edge detail similarity. For CiteSeer, **RC-ER** gets close to the highest possible recall with very high accuracy. Improvement over the baseline is greater for HEP. While the improvement may not appear large in terms of F1, note that **RC-ER** reduces error rate over the baseline by 44% for CiteSeer (from 0.009 to 0.005) and by 29% for HEP (from 0.024 to 0.017). Also, HEP has more than 64, 6000 true duplicate pairs, so that a 1% improvement in F1 translates to more than 6, 400 correct pairs.

Looking more closely at the resolution decisions from CiteSeer, we were able to identify some interesting combination of decisions by **RC-ER** that would be difficult or impossible for an attribute-only model. There are instances in the dataset where reference pairs are very similar but correspond to different author entities. Examples include *(liu j, lu j)* and *(chang c, chiang c)*. **RC-ER** correctly predicts that these are not duplicates. At the same time, there are other pairs that are not any more similar in terms of attributes than the examples above and yet are duplicates. These are also correctly predicted by **RC-ER** using the same similarity threshold by leveraging common collaboration patterns. The following are examples: *(john m f, john m st)*, *(reisbech c, reisbeck c k)*, *(shortliffe e h, shortcliffe e h)*, *(tawaratumida s, tawaratsumida sukoya)*, *(elliott g, elliot g l)*, *(mahedevan s, mahadevan sridhar)*, *(livezey b, livezy b)*,

(brajinik g, brajnik g), *(kaelbing l p, kaelbling leslie pack)*, *(littmann michael l, littman m)*, *(sondergaard h, sndergaard h)* and *(dubnick cezary, dubnicki c)*. An example of a particularly pathological case is *(minton s, minton andrew b)*, which is the result of a parse error. The attribute-only baselines cannot make the right prediction for both these sets of examples simultaneously, whatever the decision threshold, since they consider names alone.

Figure 2 shows how performance varies for **RC-ER** for the two datasets with varying combination weight α for attribute and relational similarity. Recall that when α is 0, the similarity is based only on attributes and when α is 1 it is wholly relational. The plots show that **RC-ER** with both neighborhood and edge detail similarity outperform the baselines over *all* values of α. Note that **RC-ER** takes advantage of relational bootstrapping in these experiments which explains why it is better than the baseline even when alpha is 0. The best performance for CiteSeer is around 0.5 while for HEP performance peaks around 0.1 and then trails off. It can also be observed that edge detail similarity is more stable in performance over varying α than neighborhood similarity. Significantly, once clusters have been bootstrapped using attribute and relational evidence, **RC-ER** outperforms the baselines even when alpha is 1, which means that attributes are being overlooked altogether and clusters are merged using relational evidence alone.

Figure 3 shows performance of **RC-ER** without using relational bootstrapping. When α is 0, **RC-ER** is identical to **ATTR*** which is verified by the results. As α increases from 0, performance improves over the baseline and then drops again. For HEP, performance falls sharply with higher α with neighborhood similarity. Edge detail similarity however still performs surprisingly well. Even when α is 1, it does better than the baseline for CiteSeer and is able to achieve close to 0.9 F1 for HEP. This suggests that edge detail is a reliable indicator of identity even without considering attributes. It should however be noted these results include blocking, which uses attributes to find potential duplicates. This suggests that given people with similar names, it is possible to identify duplicates with a high degree of reliability using edge detail similarity alone.

Table 3: Execution time of RC-ER, ATTR and ATTR* in CPU seconds for CiteSeer and HEP datasets.

	CiteSeer	HEP
ATTR	1.70	164.34
ATTR*	1.97	191.40
RC(Nbr w/ Bootstrap)	3.28	1123.41
RC(Edge w/ Bootstrap)	3.56	1462.77
RC(Nbr w/o Bootstrap)	14.50	4099.88
RC(Edge w/o Bootstrap)	14.34	4143.17

Finally, we look at the execution times of the algorithms. All experiments were run on a 1.1GHz Dell PowerEdge 2500 Pentium III server. First, in Table 3 we record the execution times in CPU seconds of the baselines and different versions of **RC-ER** on the CiteSeer and HEP datasets. **RC-ER** expectedly takes more time than the baselines. But it is quite fast for CiteSeer taking only twice as much time as **ATTR***. It takes longer for HEP; about 7 times as long compared to the baseline. While using edge detail is more expensive than neighborhood similarity, it does not take significantly longer for either dataset. The complexity of edge detail depends on a number of factors. It grows quadratically with the average number of edges per entity and linearly with average number of references in each edge. While the average edge size is same for both datasets, average number of edges per entity is 2.5 for CiteSeer and 6.36 for HEP which explains the difference in execution times. In contrast, complexity of neighborhood similarity is linear in the average number of neighbors per entity, which is 2.15 for CiteSeer and 4.5 for HEP.

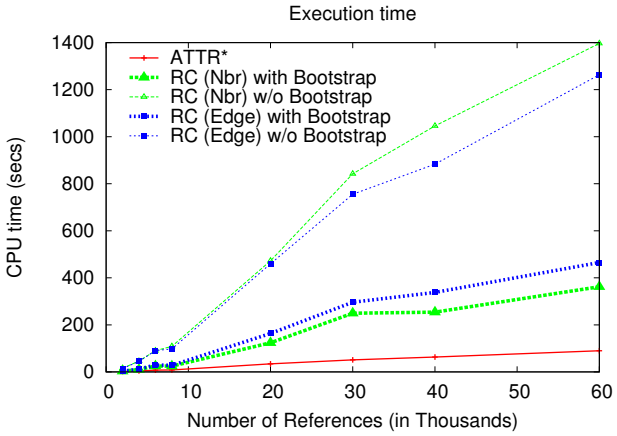


Figure 4: Execution time for ATTR* and RC-ER with neighborhood and edge detail similarity over varying number of references in synthetic datasets.

To see how the algorithms scale with increasing number of references in the dataset, we used a synthetic generator for ambiguous data. We preserved features of the real datasets wherever possible, like the average number of references and edges per entity, the degree of neighborhood for each entity and the average number of references per edge. The execution times of **ATTR*** and different versions of **RC-ER** are plotted in Figure 4 against varying number of references in the dataset. We would like to stress that the execution time depends on other factors as well like the number of potential duplicate pairs in the data, which we were not able to control directly. So these numbers should not be compared with the execution times on the real datasets but instead should serve only for a comparative study of the different algorithms. The curves confirm that **RC-ER** takes longer than the baseline but they also show that the trend is roughly linear in the number of references for all versions of it. The plots also show the significant speedup that is achieved with relational bootstrapping in addition to the performance benefits that it provides.

9. CONCLUSIONS

In this paper, we address the problem of resolving references to multiple types of entities in relational data. We propose two different similarity measures for references that consider relational similarity among them in addition to the attribute similarities. We show how these similarity mea-

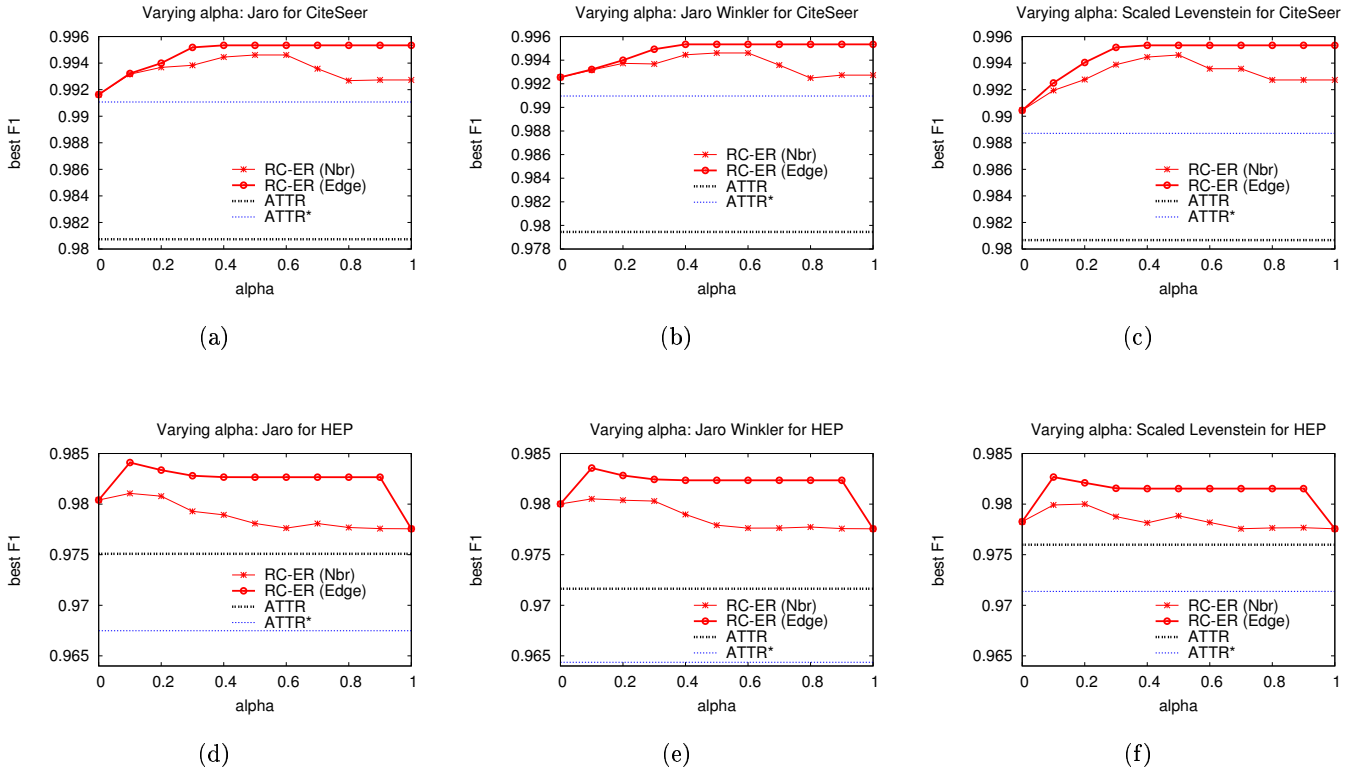


Figure 2: The best F1 measures achieved by RC-ER with neighborhood and edge detail similarities over varying combination weight α for (a-c) CiteSeer and (d-f) HEP using single link for attribute similarity with Jaro, Jaro-Winkler and Scaled Levenstein respectively.

sures may be used to resolve references into entities by clustering them. Experiments on the CiteSeer and HEP bibliographic datasets demonstrate that relational similarity when used in addition to attribute similarity results in improved entity resolution performance. In fact, one of our relational similarity measures does surprisingly well in resolving ambiguous references even without the assistance of attributes. While relational clustering expectedly takes longer than attribute clustering, we show that the algorithms scale gracefully with the size of the data. We present a relational bootstrapping scheme for clustering that significantly reduces execution time in addition to improving performance. We also illustrate the diverse issues that need to be addressed when resolving typed references in relational data.

10. REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, 2002.

[2] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, 2002.

[3] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *DMKD*, 2004.

[4] I. Bhattacharya and L. Getoor. A latent dirichlet model for entity resolution. Technical report, University of Maryland, College Park, 2005.

[5] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, 2003.

[6] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD*, 2003.

[7] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IJCAI-2003 Workshop on Information Integration on the Web*, 2003.

[8] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *KDD*, 2002.

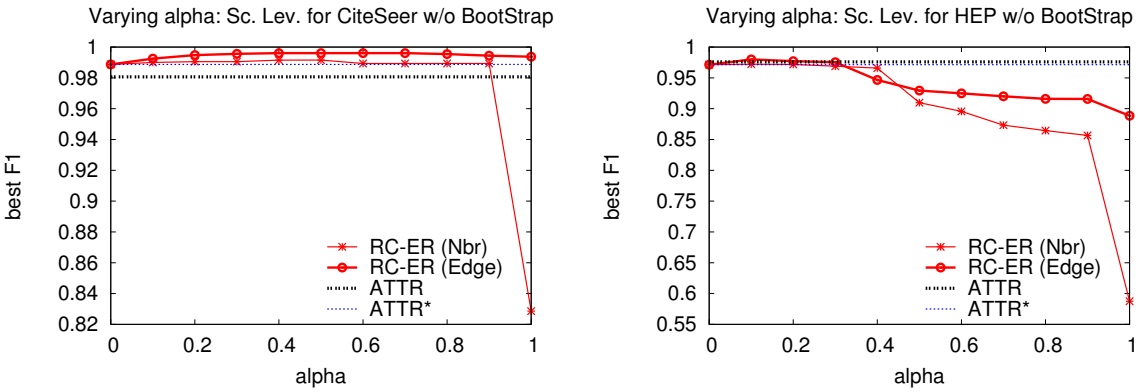
[9] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *Proceedings of the ACM SIGMOD Conference*, Baltimore, MD, 2005.

[10] W. Emde and D. Wettschereck. Relational instance based learning. In L. Saitta, editor, *Proceedings of The 13th International Conference on Machine Learning*, pages 122 – 130. Morgan Kaufmann Publishers, 1996.

[11] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[12] C. L. Giles, K. Bollacker, and S. Lawrence. CiteSeer: An automatic citation indexing system. In *ACM Conference on Digital Libraries*, 1998.

[13] M. A. Hernández and S. J. Stolfo. The merge/purge



(a) (b)

Figure 3: The effect of not using bootstrapping for initializing the entity clusters for (a) CiteSeer and (b) HEP using Scaled Levenstein as secondary similarity.

problem for large databases. In *SIGMOD*, 1995.

[14] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM SDM*, Newport Beach, CA, USA, April 21–23 2005.

[15] X. Li, P. Morie, and D. Roth. Semantic integration in text: From ambiguous names to identifiable entities. *AI Magazine. Special Issue on Semantic Integration*, 2005. to appear.

[16] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS*, 2004.

[17] A. K. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, 2000.

[18] B. Milch, B. Marthi, D. Sontag, S. Russell, D. L. Ong, and A. Kolobov. Blog: Probabilistic models with unknown objects. In *IJCAI*, 2005.

[19] A. E. Monge and C. P. Elkan. The field matching problem: Algorithms and applications. In *KDD*, 1996.

[20] A. E. Monge and C. P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *DMKD*, 1997.

[21] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[22] J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. In *Text Mining and Link Analysis Workshop, IJCAI*, 2003.

[23] Parag and P. Domingos. Multi-relational record linkage. In *KDD Workshop on Multi-Relational Data Mining*, 2004.

[24] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS*, 2003.

[25] P. Ravikumar and W. W. Cohen. A hierarchical graphical model for record linkage. In *UAI*, 2004.

[26] E. Ristad and P. Yianilos. Learning string edit distance. *IEEE Transactions on PAMI*, 20(5):522–532, 1998.

[27] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *KDD*, 2002.

[28] S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Information Systems Journal*, 26(8):635–656, 2001.

[29] W. E. Winkler. The state of record linkage and current research problems. Technical report, U.S. Census Bureau, 1999.

[30] W. E. Winkler. Methods for record linkage and Bayesian networks. Technical report, U.S. Census Bureau, 2002.