Collective Entity Resolution In Relational Data

Indrajit Bhattacharya and Lise Getoor
Department of Computer Science
University of Maryland, College Park, MD 20742, USA

Abstract

An important aspect of maintaining information quality in data repositories is determining which sets of records refer to the same real world entity. This so called entity resolution problem comes up frequently for data cleaning and integration. In many domains, the underlying entities exhibit strong ties between themselves. Friendships in social networks and collaborations between researchers are examples of such ties. In such cases, we stress the need for collective entity resolution where, instead of independently tagging pairs of records as duplicates or non-duplicates, related entities are resolved collectively. We present different algorithms for collective entity resolution that combine relational evidence with traditional attribute-based approaches to improve entity resolution performance in a scalable manner.

1 Introduction

There has been an increase in automated acquisition and integration for data repositories and information sources and, because completely manual curation is impossible in all but the smallest databases, there has been an increasing dependence on automated techniques for maintaining data integrity and quality of information. While we have seen a surge in research interest in this area over the last decade, the problems are quite challenging. Because accuracy is critical in many applications, there is need for further improvement. In addition to the attributes of records that have traditionally been used by data cleaning and integration algorithms, quite often there may be relationships between different database records. In such cases, the models and algorithms for data cleaning can take such relationships into account to improve performance.

Entity resolution is an important problem that comes up frequently for cleaning and integration. In many databases, records refer to real world entities, and as such databases grow, there can many different records that refer to the same entity. For example, a social network database can have different records with names 'J. Doe', 'Jonathan Doe' and 'Jon Doe' that refer to the same person. In the absence of keys such as social security numbers, this duplication issue [13, 15] leads to many different problems, such as redundant records, incorrectness of computed statistics, and several others. This issue also comes up when integrating data from different heterogeneous sources without shared keys and possibly even different schemas [10]. Broadly, we call such database records *references* to real world entities, and the entity resolution problem is to find the underlying *entities* in the domain and tag the references in the database with the entities to which they correspond.

Entity resolution is a difficult problem and cannot be solved using exact matches on tuple attributes. First, there is the *identification* problem, when different representations arising from recording errors or abbreviations

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

refer to the same entity. In our earlier example, figuring out that 'Jon Doe' and 'Jonathan Doe' are the same person is an instance of this problem. Failure in identification affects completeness of information. The second issue is *disambiguation*. It is possible that two records with name 'J. Doe', the same address and the same age refer to two brothers and not to the same person. This affects precision. Early approaches to entity resolution prescribed fuzzy attribute matches and many sophisticated techniques have been developed. However, attribute-based approaches still cannot satisfactorily deal with the problem of false attribute matches and, in general, it may be hard to improve precision and completeness at the same time using just attributes of records.

In many domains, some underlying entities exhibit strong relational ties to certain other entities. For instance, people interact frequently with their close friends in a social network, while in academic circles, researchers collaborate more with their close associates. When such ties exist between entities, co-occurrences between the references to these entities can be observed in the data. In the social network example, we may have the records for the best friends of 'J. Doe' and 'Jon Doe'. Our goal will be to make use of such relationships between references to improve entity resolution performance. However, the problem is that we do not know the entities for these related records either. So how can we use these relations then? One way is to use the attributes of related records as well when computing fuzzy matches. While this is an improvement, it may not always work. For example, we do not want to merge two person records simply because their best friends have similar names. The correct evidence to use is whether their best friends are in fact the same entity. This is the idea behind *collective entity resolution*, where the entity for any reference depends on the entities for its related references. Computationally, it is a more difficult problem to solve than attribute-based resolution. The database cannot be cleaned with a single-pass approach anymore because of the dependent nature of the resolutions. We need to resort to iterative approaches, where each resolution that we make potentially provides evidence to discover other duplicate references. However, there is also the promise that the resolution accuracy can be significantly improved over traditional techniques. In this article, we present a survey of algorithms we have proposed in earlier work [3, 4, 5, 6] that address the computational challenge of collective resolution and combine attributes of records with relational evidence to improve entity resolution performance.

2 Problem Formulation

In domains where the data contains relationships between different entity references, these may be represented using an auxiliary table of relations. We now introduce a generic notion of a *reference database* that records information about references and the relationships between them that are observed in the data. Then we describe the entity resolution problem in such a reference database using examples to illustrate the various issues involved.

2.1 Reference Database

In the simplest formulation, a reference database contains a table of references, $\mathcal{R} = \{r_i\}$, where each reference has an identifier $\mathcal{R}.id$ and a set of attributes $\{\mathcal{R}.A_1, \ldots, \mathcal{R}.A_k\}$. Also, we have the unobserved domain entities $\mathcal{E} = \{e_j\}$. For any particular reference r_i, we denote the entity to which it maps as $E(r_i)$. We will say that two or more references are **co-referent** if they correspond to the same entity. Note however that the database is unresolved — the references do not have any identifiers that disclose the mapping $E(r_i)$. Further, the domain entities $\mathcal{E}$ and even the number of such entities is not known. To model relationships between references in a generic way, we use a hyper-edge table $\mathcal{H}$ with identifier $\mathcal{H}.id$ and attributes $\{\mathcal{H}.A_1 \ldots \mathcal{H}.A_l\}$. Each hyper-edge connects multiple references. We use a mapping table $\mathcal{M} = \{hid, rid\}$ to associate the reference rid to the hyper-edge hid. For convenience, we use the notation $r \in h$ to mean that a reference $r \in \mathcal{R}$ is associated with a hyper-edge $h \in \mathcal{H}$: $r \in h \iff (r.id, h.id) \in \mathcal{M}$. Note that each reference may be associated with zero or more hyper-edges.

Let us now look at a sample domain to see how it can represented in our framework. We consider as our motivating example a database of academic publications similar to DBLP, CiteSeer or PubMed.[1] We consider the problem of resolving the authors of the publications. Each publication in the database has a set of author names. For each author name, we have a reference r_i in $\mathcal{R}$ and $r_i.Name$ records the observed name of the author in the publication. In addition, we can have attributes $\mathcal{R}$.*Affil* and $\mathcal{R}.Email$ to record the affiliation and email of each author reference if they are available in the paper. Additionally, each publication represents a co-author relationship among the references in it. So we have an entry h_i in the hyper-edge table $\mathcal{H}$ for each publication and an tuple $(h_i.id, r_j.id)$ in the mapping table $\mathcal{M}$ for each reference r_j in a publication h_i. If a publication also comes with additional information, such as title, these are represented as attributes ($\mathcal{H}.Title$) of the hyper-edge table $\mathcal{H}$. While in general our representation allows each reference to belong to zero or more hyper-edges, in this domain each author-name in a paper is a distinct reference and therefore occurs in exactly one hyper-edge.

As an example, consider the following four papers.

1. W. Wang, C. Chen, A. Ansari, "A mouse immunity model"

2. W. Wang, A. Ansari, "A better mouse immunity model"

3. L. Li, C. Chen, W. Wang, "Measuring protein-bound fluxetine"

4. W. W. Wang, A. Ansari, "Autoimmunity in biliary cirrhosis"

These may be represented in our notation with 10 references $\{r_1, \ldots, r_{10}\}$ in the reference table $\mathcal{R}$, where r_1 is (id 1; $Name$ 'Wang W'), etc. There are 4 entries $\{h_1, \ldots, h_4\}$ in the hyper-edge table $\mathcal{H}$ for the four papers, where h_1 is (id 1; $Title$ 'The mouse immunity model') and so on. The mapping table $\mathcal{M}$ also has 10 entries, one for each reference, to record which reference appears in which paper. For example, the entry (hid 1; rid 1) records that reference r_1 appears in hyper-edge h_1. This is represented pictorially in Figure 1(a).

2.2 Entity Resolution Problem in a Reference Database

Given the formulation of a reference database, the entity resolution task is to partition or cluster the references according to their underlying entities. To illustrate this for our example, suppose we have six underlying entities, which are shown in Figure 1(a) using six different shades. All references with name 'A. Ansari' are co-referent, as are all the 'L. Li' references. However, the two 'C. Chen's are *not* co-referent and map to two different entities. More interestingly, the four references with name 'Wang' map to two different entities. References r_1, r_4, and r_9 are co-referent, while r_8 maps to a different entity.

A natural task in a reference database is to take all references with a given name and partition them according to the entities to which they correspond. We refer to this as the **disambiguation task**. Consider the name 'W. Wang'. In our example, there are three author references for 'W. Wang': r_1, r_4, and r_8. Our goal is to partition these identically named references according to entities. Then the correct disambiguation for 'W. Wang' is $\{\{r_1, r_4\}, \{r_8\}\}$ indicating that r_1 and r_4 map to the same entity and r_8 maps to a distinct entity. The complete disambiguation for the database would cover the other references as well.

Observe that the disambiguation task handles one part of the resolution process. In our example, while it finds the co-referent pairs with name 'W. Wang', it does not consider references whose names are not exact matches. However, reference r_9 from the fourth paper is co-referent with r_1, even though it has a different recorded name. So, the r_9 reference from the fourth paper should be included in the same entity cluster as the r_1 reference. Therefore, in addition to disambiguation, we need to 'identify' coreferences with different names as well. To handle this, we define the **entity resolution task** as a partitioning *all* references in the database

[1]However, this entity resolution framework is general enough to handle application domains such as customer relationship management, personal information management and others that involve references, entities and complex relationships.

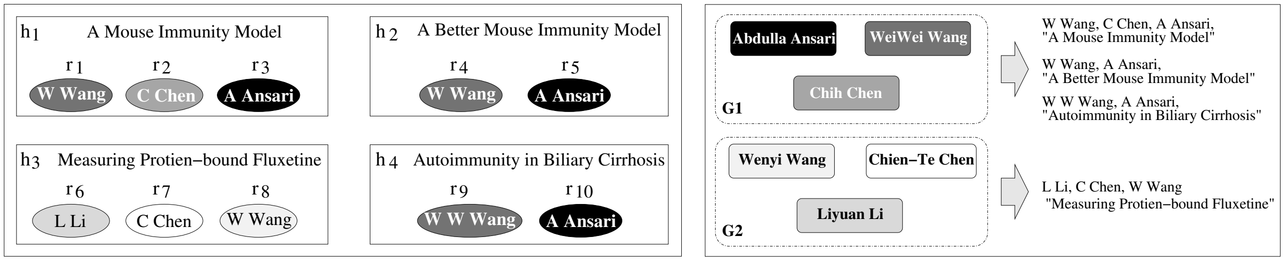

Figure 1: (a) Example papers represented as references connected by hyper-edges, with different entities shaded differently (b) Two underlying groups of collaborating entities, with their generated papers listed alongside.

according to the entities to which they correspond. The entity resolution result for our example should return six clusters: $\{\{r_1, r_4, r_9\}, \{r_8\}, \{r_2\}, \{r_7\}, \{r_3, r_5, r_{10}\}, \{r_6\}\}$. The first two clusters correspond to 'Wang', the next two to 'Chen', the fifth to 'Ansari' and the last to 'Li'.

2.3 Entity Resolution Approaches

Different approaches may be used to resolve the references in a database. Here we briefly look at the intuition behind three of the prevalent ones.

1. Attribute-based Entity Resolution: This is the traditional approach where similarity is computed for each pair of references based on their attributes and only those pairs that have similarity above some threshold are considered to be co-referent. Many sophisticated and efficiently computable similarity measures have been proposed for different types of attributes over the years. However, attributes alone often run into problems, as in the case of the three 'W. Wang' references in our example.

2. Naive Relational Entity Resolution: When relations between references are available, this approach considers the attributes of the related references when computing similarity between pairs of references. In our running example, when computing the similarity between 'W. Wang' and 'W. W. Wang', it would take into account that both have co-authors with name 'A. Ansari'.

3. Collective Entity Resolution: While the naive relational approach improves significantly on the attribute-based approach, it can be misled in domains where many entities have the same name and the relationship graph is dense. In our example, the two 'W. Wang' references r_1 and r_8 are not co-referent, though they both have co-authors with name 'C. Chen'. The correct evidence to use here is that the 'Chen's are not co-referent either. In such a setting, in order to resolve the 'W. Wang' references, it is necessary to *resolve* the 'C. Chen' references as well, and not just consider them as attributes. This is the goal of collective entity resolution, where resolutions are not made independently. Instead one resolution decision affects other resolutions via hyper-edges. This increases the computational expense of the resolution process but improves accuracy significantly in ambiguous domains.

For the first two approaches, all that is needed is a similarity measure between pairs of references. Given such a similarity measure, the algorithm for resolving entities is straight-forward — those reference pairs that have similarity above a given threshold are declared to be co-referent. However, collective entity resolution is more involved. Specifically, the dependencies between the different resolution decisions need to be modeled. Also, as we have already mentioned, the algorithm needs to make multiple passes over the references to capture the dependencies. We next describe two approaches to collective entity resolution that we have developed.

3 Algorithms for Collective Resolution

We first describe a clustering approach to collective resolution and then briefly discuss a probabilistic generative model for the same problem and how we can do inference in it.

3.1 Relational Clustering

Given that the goal of entity resolution is to cluster the database references according to their entities, we have developed a relational clustering algorithm for entity resolution (**RC-ER**) [3]. Given a current set of reference clusters $\mathcal{C} = \{c_i\}$, it iteratively merges the pair of clusters that are the most similar. We associate a cluster label $r.C$ with each reference to denote its current cluster membership. Note that it is the similarity measure that distinguishes the different entity resolution approaches. For the attribute-based approach, the similarity only considers the attributes of references. For the naive relational approach, it additionally considers the attributes of related references. The collective approach, in contrast, considers the cluster labels of the related references.

The similarity of two clusters c_i and c_j is defined as

$$sim(c_i, c_j) = (1-\alpha) \times sim_A(c_i, c_j) + \quad \alpha \times sim_R(c_i, c_j) \quad 0 \leq \alpha \leq 1 \tag{1}$$

where $sim_A()$ is the similarity of the attributes and $sim_R()$ is the relational similarity between the references in the two clusters. The most important and interesting aspect of the collective approach is the dynamic nature of the similarity. In contrast to attribute-based and naive relational resolution, where the similarity between two references is fixed, for collective resolution it depends on the *current* cluster labels of the references and therefore changes with the labels. In our example, the similarity of the two references 'W. Wang' and 'W. W. Wang' increases once the Ansari references are given the same cluster label. Let us now see how the two components of the similarity are computed.

Attribute Similarity: For each reference attribute, 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. In addition, if the hyper-edges have attributes, then the attribute similarity of two references can also take into account the attributes of the hyper-edges with which they are associated. Several sophisticated similarity measures have been developed for names, and popular TF-IDF schemes may be used for other textual attributes such as keywords. The measure that works best for each attribute may be plugged in. Finally, a weighted combination of the similarities over the different attributes yields the combined attribute similarity between two reference clusters.

Relational Similarity: For collective entity resolution, relational similarity considers the cluster labels of the references that each cluster is connected to via the hyper-edges. There are many possible ways to define this similarity; here we discuss one of measures that we have proposed [3, 5].

The hyper-edges relevant for a cluster are the hyper-edges for all references in it. Recall that each reference r is associated with one or more hyper-edges in the hyper-edge table $\mathcal{H}$. Therefore, the hyper-edge set $c.H$ for a cluster c of references is defined as

$$c.H = \bigcup_{r \in \mathcal{R} \wedge r.C = c} \{hid \mid (hid, rid) \in \mathcal{M} \ \wedge r.id = rid\} \tag{2}$$

This set defines the hyper-edges that connect a cluster c to other clusters, and are the ones that relational similarity needs to consider. For instance, when all the references in our running example have been correctly clustered as in Figure 1(b), the edge-set for the larger 'W. Wang' cluster is $\{h_1, h_2, h_4\}$, which are the hyper-edges associated with the references r_1, r_4 and r_9 in that cluster.

The different clusters to which any cluster c of references is connected via its hyper-edge set is called the neighborhood $Nbr(c)$ of cluster c.

$$Nbr(c_i) = \bigcup_{h \in c.H, r \in h} \{c_j \mid c_j = r.C\} \tag{3}$$

Returning to our example, the neighborhood of the 'W. Wang' cluster mentioned above consists of the 'Ansari' and the 'Chen' clusters, which are connected by its edge-set. Now, for the relational similarity measure between two clusters, their neighborhoods are compared using set similarity such as Jaccard similarity:

$$sim_R(c_i, c_j) = Jaccard(Nbr(c_i), Nbr(c_j)) \tag{4}$$

Recall that for two sets A and B, their Jaccard similarity is defined as $Jaccard(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|}$. The similarity can be computed and updated efficiently, in time that is linear in the average number of neighbors per cluster.

Clustering Algorithm: Given the similarity measure for a pair of clusters, a greedy agglomerative clustering algorithm is used for collective entity resolution. The algorithm bootstraps the clusters, identifies the candidate set of potential duplicates and iterates over the following steps. At each step, it identifies the current 'closest pair' of clusters (c_i, c_j) from the candidate set and merges them to create a new cluster c_{ij}. It identifies new candidate pairs and updates the similarity measures for the 'related' cluster pairs. All of these tasks are performed efficiently using an indexed priority queue. The algorithm terminates when the similarity for the closest pair falls below a threshold.

3.2 Probabilistic Group Model

In addition to the relational clustering algorithm, we have also developed a probabilistic generative model for collective entity resolution [6], which we call the Latent Dirichlet Allocation model for Entity Resolution, or **LDA-ER** for short. It describes how the author references in any paper might be generated. Instead of modeling pair-wise collaboration relations between author entities, the novelty of the model is that it uses the notion of collaborating *groups of entities*. For our example, the six relevant entities belong to two different groups, as shown in Figure 1(b). The generative process for each paper first selects one or more groups that collaborate to write the paper. Then each author for the paper is chosen from one of these selected groups. The true name of an author entity determines what the reference name in a paper might be. In the example, papers 1, 2 and 4 are generated by collaborating entities from group G1, while paper 3 is written by entities from group G2. Note that for the author entity with true name "WeiWei Wang", the reference name is "W. Wang" in two of the papers and "W. W. Wang" in another.

We have developed a Gibbs Sampling algorithm for doing inference in this model. Starting from an initial assignment of groups and entities for the references, the algorithm repeatedly samples the group and entity for each reference given those for the others until a stationary distribution is reached. In our example, the algorithm is expected to predict that the 'Wang' references in papers 1, 2 and 4 are likely belong to the same group, and therefore they are more likely to map to the same entity. The other 'Wang' reference in paper 3 maps to a different entity, since most probably it belongs to a different group. Also, one interesting aspect of our inference algorithm is that number of entities does not need to specified as a parameter — it automatically determines the most likely number of entities given the reference database. Another important aspect is that the inference algorithm is completely unsupervised. This is significant given the scarcity of training data for this problem.

4 Experimental Results

We have evaluated our collective entity resolution algorithms [3, 4, 5, 6] for the task of author resolution in synthetic as well real-world citation databases such as CiteSeer (2,892 author references from Machine Learning), arXiv (58,515 author references from High Energy Physics) and BioBase (831,991 author references from Biology). Here we present an overview of our results. The first baseline (**A**) that we compare against uses only attributes of the references for resolution, while the second (**A+N**) additionally uses attributes of neighboring or related references. We also consider the variants **A*** and **A+N*** that take transitive closures over the pair-wise decisions made in **A** and **A*** respectively. For evaluating entity resolution performance, we use the popular F1-measure (the harmonic mean of precision and recall) of the pair-wise decisions over all references.

In Table 1, we show the performance of our relational clustering algorithm algorithm **RC-ER** against the baselines in the three datasets. The best performance for each dataset is shown in bold. We can see that **RC-ER** outperforms the baselines in all cases. Also, the improvement over the baselines increases as we move from CiteSeer to arXiv and then to BioBase. The improvement using collective resolution depends on how densely the references are related to each other and also on what fraction of the references names are ambiguous, or in other words, are shared by more than one entity. The results confirm this since both the density of relations and ambiguity of reference attributes in highest for BioBase and lowest for CiteSeer, which explains the difference in performance. We experimented with different attribute similarity measures and we observed similar improvements with all of them. Performance using our probabilistic model **LDA-ER** is very similar to that of **RC-ER**.

Table 1: Entity resolution performance (F1-measure) of five algorithms on three datasets. Results are for the entire CiteSeer and arXiv datasets and for the 100 most frequent names in BioBase.

	A	**A***	**A+N**	**A+N***	**RC-ER**
CiteSeer	0.980	0.990	0.973	0.984	**0.995**
arXiv	0.974	0.967	0.938	0.934	**0.985**
BioBase	0.701	0.687	0.710	0.753	**0.818**

While Table 1 records improvements over the entire CiteSeer and arXiv datasets, the strength of collective resolution clearly stands out when we look at specific instances of ambiguous names. When a name or its abbreviation is shared between multiple entities, it is hard to resolve different references having that name using attributes alone. In Table 2, we show some examples of ambiguous names from arXiv and the performance of the attribute baselines and our **LDA-ER** model only over references that have this abbreviated name. We can see that for all such cases collective resolution out-performs the baselines by very large margins.

Table 2: Entity resolution performance (F1-measure) for the **LDA-ER** model and the best baseline performance for some example ambiguous names from the arXiv dataset.

	Cho H	Davis A	Sarkar S	Sato H	Shin H	Veselov A	Yamamoto K	Yang Z	Zhang R	Zhu H
Best of A/A*	0.80	0.67	0.67	0.82	0.69	0.78	0.29	0.77	0.83	0.57
LDA-ER	**1.00**	**0.89**	**1.00**	**0.97**	**1.00**	**1.00**	**1.00**	**0.97**	**1.00**	**1.00**

We have done extensive evaluations of the different aspects of our models and algorithms. Figure 2 shows some sample plots. Figure 2(a) shows how performance changes with the combination weight α between attribute and relational similarity for arXiv. We also experimented with synthetic data to see how different structural properties in the data affect the algorithms. Figure 2(b) plots one of the trends, which shows that expected improvements using **LDA-ER** are higher when each relation covers more references on average. Finally, Figure 2(c) shows how **RC-ER** scales with data size once the potential duplicate pairs have been identified. We can see that it takes longer than the attribute baseline, but the growth is still linear.

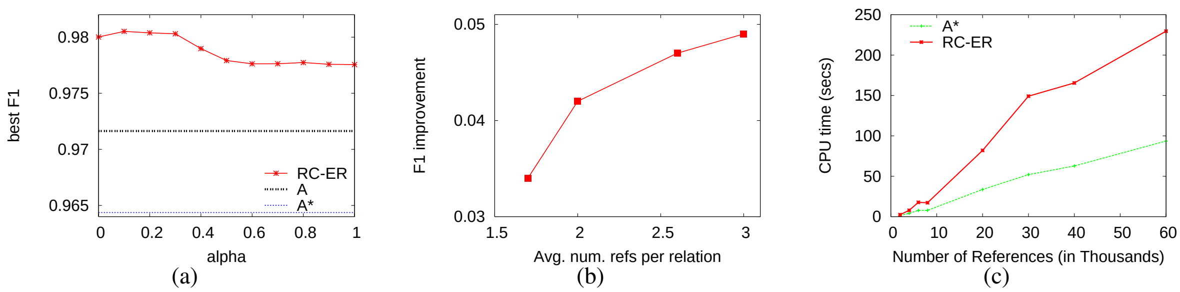

Figure 2: (a) Entity resolution performance using **RC-ER** versus combination weight α for arXiv. (b) Improvement over **A*** using **LDA-ER** against average number of references in each relation. (c) Execution time of **RC-ER** and **A*** for increasing number of references in the data.

5 Related Work

The entity resolution problem has been studied in many different areas under different names — deduplication, record linkage, co-reference resolution, reference reconciliation etc. Most of the work has focused on traditional attribute-based entity resolution. Extensive research has been done on defining approximate string similarity measures [15, 7, 8] that may be used for unsupervised entity resolution. The other approach is to use adaptive supervised algorithms that learn similarity measures from labeled data [18]. The WHIRL system [9] has been proposed for data integration using similarity join queries over textual attributes. Swoosh [2] is generic entity resolution framework that minimizes the number of record-level and feature-level operations when resolving and merging duplicates. Probabilistic techniques have been proposed for quick similarity computation between tuples for fast text-joins [12] and for efficiently looking up candidate matches for incoming tuples [8].

Many recent approaches take relations into account for data integration [1, 3, 5, 14, 11, 16, 17]. Ananthakrishna et al. [1] introduce relational deduplication in data warehouse applications where there is a dimensional hierarchy over the relations. Neville et al. [16] have shown how relations may be combined with attributes for clustering. Kalashnikov et al. [14] enhance attribute similarity between an ambiguous reference and the many entity choices for it with relationship analysis between the entities, such as affiliation and co-authorship. Dong et al. [11] collectively resolve entities of multiple types by propagating relational evidences in a dependency graph, and demonstrate the benefits of collective resolution in real datasets. Singla et al. [17] propose a probabilistic model based on conditional random fields that exploits similar dependencies.

6 Conclusion and Future Directions

Entity resolution is an area that has been attracting growing attention to address the influx of structured and semi-structured data from a multitude of heterogeneous sources. Accurate resolution is important for a variety of reasons ranging from cost-effectiveness and reducing redundancy in data to accurate analysis for critical applications. We have found collective entity resolution to be a powerful and promising technique that combines attribute similarity with relational evidence and significantly improves performance over traditional approaches. The improvements using relations are more dramatic in databases where names are more likely to be ambiguous. While collective resolution is more expensive than attribute-based resolution, the computational cost is not prohibitive. As future directions, we are interested in localized entity resolution, incremental updates and in challenging and important domains such as geo-spatial databases and others with unstructured context as in email archives.

References

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

[2] O. Benjelloun, H. Garcia-Molina, Q. Su, and J. Widom. Swoosh: A generic approach to entity resolution. Technical Report, Stanford University, March 2005.

[3] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD)*, June 2004.

[4] I. Bhattacharya and L. Getoor. Relational clustering for multi-type entity resolution. In *KDD Workshop on Multi-Relational Data Mining (MRDM)*, 2005.

[5] I. Bhattacharya and L. Getoor. *Entity Resolution in Graphs*, Chapter in Mining Graph Data (Lawrence B. Holder and Diane J. Cook, eds.). Wiley, 2006.

[6] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *SIAM Conference on Data Mining (SIAM-SDM)*, 2006.

[7] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.

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

[9] W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *SIGMOD*, 1998.

[10] W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18:288–321, 2000.

[11] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.

[12] L. Gravano, P. Ipeirotis, N. Koudas, and D. Srivastava. Text joins for data cleansing and integration in an rdbms. In *IEEE International Conference on Data Engineering (ICDE)*, 2003.

[13] M. Hernández and S. Stolfo. The merge/purge problem for large databases. In *SIGMOD*, 1995.

[14] D. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM Conference on Data Mining (SIAM SDM)*, 2005.

[15] A. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *SIGKDD*, 1996.

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

[17] P. Singla and P. Domingos. Multi-relational record linkage. In *SIGKDD Workshop on Multi-Relational Data Mining (MRDM)*, 2004.

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