Adaptive Graphical Approach to Entity Resolution*

Zhaoqi Chen Dmitri V. Kalashnikov Sharad Mehrotra

Computer Science Department
University of California, Irvine

ABSTRACT

Entity resolution is a very common Information Quality (IQ) problem with many different applications. In digital libraries, it is related to problems of citation matching and author name disambiguation; in Natural Language Processing, it is related to coreference matching and object identity; in Web application, it is related to Web page disambiguation. The problem of Entity Resolution arises because objects/entities in real world datasets are often referred to by descriptions, which might not be unique identifiers of these entities, leading to ambiguity. The goal is to group all the entity descriptions that refer to the same real world entities. In this paper we present a *graphical approach* for entity resolution. It complements the traditional methodology with the analysis of the entity-relationship graph constructed for the dataset being analyzed. The paper demonstrates that a technique that measures the degree of interconnectedness between various pairs of nodes in the graph can significantly improve the quality of entity resolution. Furthermore, the paper presents an algorithm for making that technique self-adaptive to the underlying data, thus minimizing the required participation from the domain-analyst and potentially further improving the disambiguation quality.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; H.2.m [**Database Management**]: Miscellaneous—*data cleaning*; H.3.7 [**Information Storage and Retrieval**]: Digital Libraries

General Terms

Algorithms, Management

Keywords

Entity Resolution,Graph Analysis,Entity Relationship Graph, SNA, Self-tuning

*This work was supported by NSF grants 0331707, 0331690

JCDL'07, June 17–22, 2007, Vancouver, British Columbia, Canada.

1. INTRODUCTION

In digital libraries, a big challenge is to automatically group bibliographic references that refer to the same publication. This problem is known as *citation matching* [24, 33]. Not only publications, but also authors, venues, etc. are often referred to by different representations and need to be disambiguated. For example, the dataset might describe publications written by two people, John Smith and Jane Smith, but refer to both of them as 'J. Smith', leading to ambiguity. This is a more general disambiguation challenge known as *entity resolution*. It is also known as *fuzzy grouping* [10] and *object consolidation* [2]. In the generic settings of this disambiguation problem, a dataset $\mathcal{D}$ describes a set of entities $E = \{e_1, e_2, \ldots, e_m\}$ and the relationships in which they participate. Entities can be of different types, such as publications, authors, and venues. Entities in $\mathcal{D}$ are represented as a set of instantiated attributes $R = \{r_1, r_2, \ldots, r_n\}$ referred to as entity *representations* or *references*. The goal is to correctly group the representations in R that co-refer, that is, refer to the same entity.

Most of the traditional domain-independent entity resolution techniques are *feature-based similarity (FBS)* methods, as they employ similarity functions that compare values of entity features (or attributes) in order to determine if two object descriptions co-refer. The similarity metrics of features can be learned from the data by adaptive approaches [9, 12, 30, 34, 36]. More recently, techniques have been developed that in addition to analyzing object features can also analyze the information derived from the object *context*, which is usually the directly associated entities/records, when computing reference similarities [3, 5, 15, 31]. For example, in a publication database, the *authors* are usually represented by names (either full names or first initials and last names) in the *context* of *papers*. It is not hard to derive and use the *coauthorship* information from the *papers* as additional information for the authors being considered when detecting the author duplication.

In our past work, we have developed a novel graphical methodology for entity resolution [11, 21–23]. The proposed domain-independent approach views the underlying dataset $\mathcal{D}$ as an entity-relationship graph, wherein the nodes represent the entities in $\mathcal{D}$ and the edges represent the relationships among the entities. For any two entity representations, the co-reference decision is made not only based on the entity features, or information derived from the context, but also based on the inter-entity relationships, including indirect ones, that exist among the two representations.

While this methodology has been demonstrated to work

well in practice, it has several limitations. In particular, the solution employs an intuition based mathematical model for computing how strongly two entities are interconnected via relationships in the entity relationship graph. This model is the key component of the overall domain-independent approach. However, because it is fixed and purely intuition-based, it might not be adequate for a particular application domain or might lead to only minor quality improvements. To overcome this problem, in this paper we present an algorithm that makes the overall approach self-adaptive to the data being processed, leading to the improvement of the quality and efficiency of the proposed methodology.

The rest of this paper is organized as follows. Section 2 presents an overview of related work. Section 3 takes up a graphical view of the problem. The basic entity resolution algorithm is covered in Section 4. Section 5 presents an algorithm for making the approach self tuning to dataset being processed. The overall algorithm is then extensively empirically evaluated in Section 6 and compared to some of the state of the art solutions. Finally, we conclude in Section 7.

2. RELATED WORK

Real-world datasets, especially when created by combining many diverse heterogeneous data sources into a unified dataset, are known to have various disambiguation issues: missing, erroneous, duplicate data and so on. It has been realized early that the quality of data analysis and decision making is only as good as the quality of the data being analyzed, leading to creation of various techniques and methodologies to deal with various types of errors in data. In this section, we will mostly concentrate on methodologies that deal with the *Disambiguation/Entity Resolution* problem. The related problems include *Deduplication* [3, 34], *Name Disambiguation* [17, 25, 29], *Citation Matching* [13, 24, 33], *Reference Reconciliation* [15], etc. The cause of various types of the disambiguation problem is the fact that in real-world datasets, entities are referred to via descriptions (a set of instantiated attributes) instead of unique identifiers, leading to ambiguity.

There are different kinds of information present in a dataset that an approach can utilize for disambiguation: attributes, context, relationships, etc. The traditional approach to solving the disambiguation challenge is to compare the values in attributes of two entity descriptions (references) to decide whether they co-refer (i.e., refer to the same real-world entity), for all pairs of entity references. These are called *feature-based similarity* (FBS) methods [16, 18]. Recently, there has been a number of interesting techniques developed that enhance the traditional techniques by being able to utilize certain types of *context* to improve the quality [3, 5, 15]. All these methods utilize directly linked entities as additional information when computing record similarity. We refer to these methods as context-based methods.

There are also techniques that can analyze another type of information: inter-entity relationships [22, 26]. In addition to analyzing the information present in the dataset, some of the disambiguation techniques can also analyze meta information about the dataset, specified by the analyst [13]. Some disambiguation approaches take into account the dependence among the co-reference decisions and no longer make pair-wise matching decisions independently. These are called *relational* methods [6, 15, 27, 33, 35].

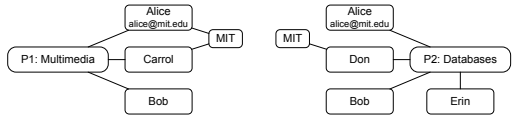
The technique proposed in the paper is domain-independent and aims at minimizing analyst participation. It is a graphical approach, as it visualizes the dataset as the standard entity-relationship graph. There are other graphical disambiguation approaches, which visualize different graphs:

- **Web graph.** To disambiguate appearances of people on the web, it has been proposed to analyze the Web graph [4], wherein the webpages are represented as nodes, and hyperlinks as edges.
- **Co-reference dependence graph.** Some of the relational approaches, e.g. [15], visualize the dependence (edges) among co-reference decisions (nodes) as a graph.
- **Entity-relationship graph (ER graph).** In the standard entity-relationship graph, the nodes represent the entities in the dataset and the edges represent the relationships among the entities [22,26,29,31]. The method proposed in this paper also analyzes the ER graph for the dataset.

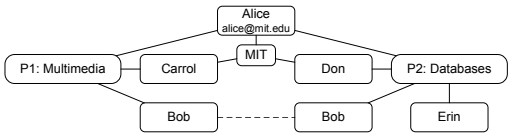
The work on entity resolution is quite extensive. The most related work include recent approaches developed by Andrew McCallum, William Cohen, Bradley Malin, Lise Getoor, Lee Giles, etc. [8, 13, 14, 17, 26, 29]. The differences of these techniques from our methodology are:

- **ER graph.** Our methodology analyzes the whole ER graph "as is". Existing most related methods analyze either only portions of the ER graph or they analyze a different graph – derived by the analyst from the ER graph.
- **Algorithms for analysis.** Existing techniques are frequently based on probabilistic methodologies, whereas we rely primarily on the mathematical apparatus from the area of Operation Research.
- **Domain-independence.** Unlike our approach, some of those existing techniques are intended for specific instances of Entity Resolution (e.g., email alias disambiguation, citation matching), or applicable to domains with certain properties.

It should be noted that virtually all of the existing techniques analyze only portions of the ER graph, if at all. For instance, the approach in [7] by Bhattacharya and Getoor analyzes only co-occurrences of names of authors via publications for a publication dataset. The approach [7], for a fixed domain, understands only one type of relationship ("writes" in this case) and only two types of entities: person ("author") and container ("publication"). Our approach can analyze all of the types of relationships and entities present. The approach in [7] is applicable to only those domains where certain (co-occurrence) properties hold. Another example is [20], which analyzes only the shortest path between two nodes in the graph. The adaptive approach in [29] is most related, but it analyzes paths, in the ER graph, of length no more than 2. Another interesting solution [26] looks at people and connect them via one single type of 'are-related' relationships. However, that approach does not take into account that people can be related for different reasons and there can be different types of relationships (and entities) in a given domain. We will compare the proposed method with some of the state of the art solutions in Section 6.



(a) Initial graphical representation.



(b) Graph representation after merging high-confidence cases and adding similarity edges.

Figure 1: Graphical representations of a sample dataset.

3. GRAPHICAL VIEW OF THE PROBLEM

The proposed entity resolution algorithm employs an entity-relationship graph as a representation for the dataset. To introduce the necessary concepts, let us consider the following simple scenario. Assume some extraction software is applied to a dataset consisting of research publications. For each paper, the software is capable of extracting its title, authors, author affiliations and author email addresses, if present. Figure 1(a) and 1(b) shows an example of graph representation for a sample dataset with two papers. For paper P_1 with title "Multimedia", the software extracts that its authors are Alice, Bob, and Carroll; Alice and Carroll are affiliated with MIT; and Alice's email address is `alice@mit.edu`. For paper P_2 with title "Databases", the software extracts its authors Alice, Bob, Don, and Erin; Don is affiliated with MIT; and Alice's email address is `alice@mit.edu`. The information extracted from the two papers are initially represented as shown in Figure 1(a). For each entity representation, there is a distinct node in the graph. For instance, the 'Alice' from P_1 and 'Alice' from P_2 get two separate nodes. The relationships among entities are represented by edges between corresponding nodes. For example, the edge between 'Carroll' and P_1 represents an 'author-writes-paper' relationship. The essence of the problem is to merge those references that in reality refer to the same entity.

The traditional approaches (and/or domain knowledge) often can be effectively employed to determine for each pair of entity representations whether they: (1) are highly likely to co-refer (very similar) and thus should be merged, (2) are highly unlikely to co-refer (too dissimilar) and thus should not be merged, (3) might co-refer (uncertain). For instance, if organization names are known to be unique identifiers of organizations in a given application domain, then handling organizations might not be a challenge for that domain. For the scenario illustrated in Figure 1(a), it means that both 'MIT' references refer to the same organization and the two nodes of 'MIT' should be merged. Also email addresses often serve as unique identifiers of people, so the traditional approach can conclude with high confidence that the two representations for 'Alice', with the same email address, are likely to co-refer and should be merged.

The most interesting and challenging cases are the uncertain cases that fall under the third category. To capture those cases graphically we introduce a new edge type: a *similarity* edge. A similarity edge is created for a pair of representations which are similar to some degree that they potentially refer to the same entity, but need further analysis. The degree of such a similarity can be based on domain knowledge, ad-hoc techniques, or on feature-based similarity, such as the Edit Distance measure. In the latter case, entities are considered to be similar when their similarity exceeds a certain threshold value. For instance, for the graph illustrated in Figure 1(a), a similarity edge will be created for the two representations of 'Bob', since they might potentially refer to the same person. The rest of the author names are less similar, and thus no similarity edges will be created for them. The new graph is demonstrated in Figure 1(b) with the similarity edge represented as a dashed line.

```
CREATE-GRAPH(R, τ, ε)
1  V ← ∅ //V is the node set
2  E_R ← ∅ //E_R is the regular edge set
3  E_S ← ∅ //E_S is the similarity edge set
4  for each r_i ∈ R
5     Create a node v_i corresponding to r_i
6     V ← V ∪ {v_i}
7  for each pair of nodes u, v ∈ V
8     if there is a relationship between u and v
9        E_R ← E_R ∪ {(u, v)} //create an edge (u,v)
10 P ← BLOCKING(V)
11    //P contains pairs of nodes {{u_1, v_1}, ..., {u_n, v_n}}
12    //where u_i, v_i are close to each other
13    //according to BLOCKING(V)
14 for each pair of nodes {u_i, v_i} ∈ P
15    if f(u_i, v_i) > τ then
16       MERGE-NODES(u_i, v_i)
17    else if f(u_i, v_i) > ε then
18       E_S ← E_S ∪ {(u_i, v_i)} //create a similarity edge
19 E ← E_R ∪ E_S
20 return V, E
```

Figure 2: Graph Creation Algorithm

3.1 Creating the Entity-Relationship Graph

The generic algorithm for creating the graphical entity-relationship representation of a dataset is presented in Figure 2. The algorithm first creates an initial representation of the dataset. It does so by creating a distinct node per each entity representation found in the dataset, and then connecting nodes via an edge when there is a corresponding relationship among the nodes. It then applies domain knowledge and other standard techniques to merge high-confidence cases. Usually the merging of two nodes u and v is decided by comparing the similarity score $f(u, v)$ between them with a high-confidence threshold τ. Notice that the current standard techniques frequently employ a blocking process [32] to bring similar representations together so that there is no need to check every single pair of nodes in the dataset, leading to a speedup. Finally, the algorithm creates similarity edges between each pair of nodes that has similarity scores lower than τ, but higher than a low-confidence threshold ϵ. This means this pair of nodes can potentially represent the same entity but needs further analysis. Using higher threshold τ to merge high-confidence cases and lower threshold ϵ to bring together nodes for further analysis is essentially an adaptation of the canopy method [28]. With regard to the similarity edges, the goal of entity resolution can be rephrased as determining for each two nodes, connected via a similarity edge, whether they co-refer.

3.2 Virtual Connected Subgraph

An interesting property of the resulting entity-relationship

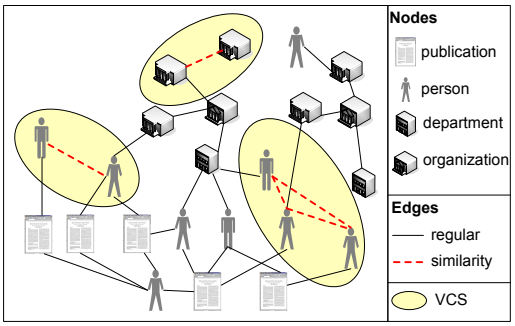


Figure 3: Virtual Connected Subgraphs (VCS).

graph with respect to similarity edges is that it contains several Virtual Connected Subgraphs (VCS). Each similarity edge is a part of a VCS as illustrated in the example in Figure 3. A VCS is a subgraph of the overall graph G, s.t.:

- it is virtual – it consists of only similarity edges,
- it is connected – there is a path between any pair of nodes that belong to the VCS,
- it is complete – no more nodes or edges from G can be added to the VCS such that the above two properties still hold.

The concept of a VCS is important because the overall task can be conceptually viewed as partitioning each VCS. This is because according to the similarity edges, entity representations in different VCSs cannot refer to the same entity. Figure 3 demonstrates an example of three VCSs: two for entities of type *people* and one for entities of type *department*.

There are two situations that are possible with respect to splitting VCSs into clusters of entity representations: (1) the number of entities in each VCS is known to the algorithm beforehand; and (2) the general case where that number is unknown. In the first case, since the number of entities in each VCS is known, the algorithm knows the exact number of clusters to split each VCS into. That restricted scenario has been studied in [11]. In a more common case studied in this paper, that number is unknown and the overall problem is thus more complex.

3.3 The idea behind the graphical approach

We will build on a well-established technique of analyzing relationships for disambiguation [11, 20–22, 26], to apply to the problem of Entity Resolution . While that work has been shown to significantly improve the quality for certain types of the disambiguation problem, applying it to the general case of entity resolution remains a challenge. The overall idea behind using relationships is to look at the direct and indirect (long) relationships that exist between specific pairs of entity representations in order to make a disambiguation decision. In terms of the entity-relationship graph that means analyzing paths that exist between various pairs of nodes.

For instance, to decide whether the two representations of 'Bob' co-refer in Figure 1(b), among other factors the proposed technique will analyze various paths that exist between the two representations for Bob in the graph and measure the connection strength that those paths carry. If the algorithm decides that the connection strength between the two 'Bob' nodes is sufficiently strong, it would then use it as an extra evidence that the two representations might co-refer. The hypothesis and the premise of this approach is that each path carries in itself a certain degree of affinity, and if the overall affinity is sufficiently large, then the two representation are likely to co-refer.

More specifically, the approach relies on the notion of the *connection strength* $c(u, v)$ between two nodes u and v in the graph G. The connection strength between u and v reflects how strongly u and v are connected to each other via paths that exist between them. The key idea of the methodology that we propose is to complement existing techniques with analysis of $c(u, v)$ to better the quality of entity resolution and, most importantly, make $c(u, v)$ adapt to the underlying dataset.

4. CONSOLIDATION ALGORITHM

We now have developed all the concepts and notation needed to explain the proposed entity resolution approach. The solution exploits both *features* and *relationships* for the purpose of consolidation, and outputs the resulting clustering as the outcome. The algorithm starts by constructing the Entity-Relationship (ER) graph for the dataset and then identifies all the VCSs in the ER graph. It then employs a merging algorithm to consolidate entities in each VCS.

Merging is a bottom-up clustering approach employed by many disambiguation algorithms. The basic idea is to identify the two nodes with the highest similarity, and to merge them in case their similarity is above a predefined threshold. The merging process continues until there are no nodes left whose similarity is above the threshold. If the merging of the nodes affects the similarities of other nodes, then the similarity scores need to be updated and compared with the threshold again. The algorithm formalization is presented in Figure 4.

Various disambiguation algorithms that employ merging differ significantly in how exactly the merging is carried out. One of the key differences is in the way similarity $s(u, v)$ between two representations u and v is computed. For example, the traditional techniques compute $s(u, v)$ using only entity features, i.e., $s(u, v) = f(u, v)$, where $f(u, v)$ is a feature-based similarity function. Context-based methods, utilize a merging approach that combines feature-based similarity with the similarity of attributes derived from the context. This combined strategy leads to significant improvement in quality of the result, compared to that of pure feature-based methods [5, 15].

As elaborated in the previous section, a feature-based similarity approach is first employed to construct the ER graph for the dataset being processed. The merging technique discussed next essentially enhances that feature-based approach, by applying analysis of relationships to the remaining cases that cannot be resolved with FBS techniques with high confidence.

The proposed merging approach, like the traditional methods, also looks at all pairs of nodes that can be similar: in this case, these are the nodes connected via similarity edges, see Figure 4. However, the proposed approach computes $s(u, v)$ as a combination of the feature-based similarity $f(u, v)$ *and* the connection strength $c(u, v)$:

$$s(u, v) = c(u, v) + \lambda f(u, v), \quad (1)$$

where λ is a parameter that controls the amount of relative contribution of the connection strength and feature-based

```
CONSOLIDATE-GRAPH(V, E, λ, δ)
 1  Q ← ∅ //Q is a priority queue
 2  for each similarity edge(u, v) ∈ E_S
 3      s(u, v) ← c(u, v) + λf(u, v)
 4      Insert {u, v, s(u, v)} into Q
 5          based on descending order of s(u, v)
 6  while Q is not empty
 7      Take the first element {u, v, s(u, v)} in Q
 8      if s(u, v) > δ then
 9          MERGE-NODES(u, v)
10          Q ← UPDATE-SCORE(Q)
11      else
12          break
13  return V, E
```

Figure 4: Consolidation Algorithm

Figure 5: Utilizing external data/knowledge.

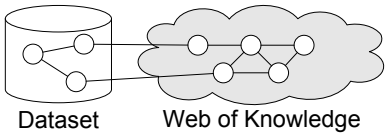


Figure 6: Web of knowledge.

similarity. Section 5.3 will explain how the value of λ is obtained. In Section 6 we shall see that considering connection strength $c(u, v)$ in the merging process leads to a powerful $s(u, v)$ function which is very effective in improving the quality of the final result.

When $c(u, v)$ is computed by a disambiguation algorithm, only L-short simple paths between u and v are considered, where a path is L-short if its length does not exceed L and a path is simple if it does not contain duplicate nodes. The set of L-short simple paths is denoted by $\mathcal{P}_L(u, v)$. The intuition is that longer paths are less important and thus can be omitted. The total connection strength between nodes u and v is computed as the sum of connection strengths of paths in $\mathcal{P}_L(u, v)$:

$$c(u, v) = \sum_{p \in \mathcal{P}_L(u,v)} c(p). \quad (2)$$

When computing connection strengths, the proposed algorithm considers only paths that go through regular edges. For nodes x and y, the $x \rightsquigarrow y$ paths connecting x and y with similarity edges are discovered, but not taken into account when computing $c(x, y)$. Instead they are hashed on those similarity edges in a hash table. The merging proceeds iteratively. Merging of any two nodes u and v, connected via a similarity edge (u, v), can lead to appearance of new paths that do not contain similarity edges. This new paths are efficiently discovered via a lookup in the hash table with the key $\{u, v\}$. After that the corresponding connection strength and the priority queue are updated by the UPDATE-SCORE function on Line 10 in Figure 4.

5. ADAPTIVENESS TO DATA

Since there can be many path types, it can be quite difficult to pick good weights for those path types based on the intuition. The number of path types, and hence the number of their weights, depends on the particular graph being analyzed and on the value of the parameter L (maximum length of paths). In general, assigning good weights manually is proved to be a non-trivial challenge. Therefore, we propose a self-tuning algorithm that can learn the path type weights automatically so that it is adaptive to the data being analyzed.

5.1 Motivation for Adaptiveness

Before we describe the self-tuning capabilities of our approach, let us analyze which benefits you can expect if you succeed in creating a self-tunable connection strength (CS) model in the context of the disambiguation task. First, such a CS model would minimize the analyst participation. This is important since nowadays various data-integration solutions are incorporated in real Database Management Systems (DBMS). For example, entity resolution is known as *fuzzy grouping* operation in the data-integration module of Microsoft SQL Server DBMS [10]. Having a less analyst-dependent technique makes that operation of wide applicability, so that non-expert users can apply it to their datasets.

The second advantage of such a CS model is that it expects to increase the quality of the disambiguation technique. There are also less obvious advantages. For example, the technique is capable of detecting which path types are marginal in their importance. Thus, the algorithm that discovers paths when computing $c(u, v)$ can be sped up, since the path search space can be reduced by searching only for important paths while skipping marginally important ones. Speeding up the algorithm that discovers paths is important since it is the bottle-neck of the overall approach.

Finally, it is often desirable to be able to (aggregate and then) use external data/knowledge sources to disambiguate a dataset, as illustrated in Figure 5. This is indeed possible with the approach proposed in [21, 22] and with an adaptive CS model, as illustrated in Figure 6. For that, the dataset and external data need to be first represented as the standard entity-relationship (ER) graph. Then the nodes that are known to correspond to the same entity in the dataset and the external 'web of knowledge' will need to be merged. Now the disambiguation technique, when applied to the dataset, will find paths that go through not only this dataset, but also the web of knowledge. The importance of various paths will be automatically determined by the adaptive CS model. So the web of knowledge can be potentially automatically employed to improve the disambiguation quality, or, it is also possible that the model will detect that all paths going through the web of knowledge are unimportant and thus it should not be used.

In the rest of this section, we first discuss how to generalize many of the existing Connection Strength (CS) models to create an adaptive CS model in Section 5.2. We then present an algorithm for calibrating the adaptive model from data in Section 5.3. We illustrate the above algorithms via an example in Section 5.5.

5.2 Adaptive CS Model

The goal of any connection strength model is for any two nodes u and v in the graph G to be able to compute how strongly they are connected to each other via paths in G. To create an adaptive CS model, we observe that many of the non-adaptive CS models can be generalized.

Namely, assume that all paths that can be potentially discovered by the algorithm can be classified into a finite set of path types $S_T = \{T_1, T_2, \ldots, T_n\}$. That is, there is a function $T(p, G) \rightarrow S_T$, that for any given path p maps it to one of those path types. If any two paths p_1 and p_2 are of the same type T_j, then they will be treated as identical by the algorithm. Then, for any two nodes u and v we can characterize the connections among them with a path-type count vector $Tuv = (c_1, c_2, \ldots, c_n)$, where each c_i is the number of paths of type T_i that exist between u and v. If we assume that there is a way to associate weight w_i with each path type T_i, then we can see that many existing connection strength models compute $c(u, v)$ as:

$$c(u, v) = \sum_{i=1}^{n} c_i w_i. \tag{3}$$

The difference among the existing CS models is (a) in the way they classify paths into types, and (b) in the way they assign weights to path types, where both (a) and (b) are either already given and fixed or, alternatively, decided by the algorithm designer based on what is intuitively reasonable for a particular domain.

Assigning those weights manually based on intuition might be a suboptimal solution. It requires a smart analyst who should know the dataset and the algorithm in detail. That often is undesirable, and the goal is to make the analyst work simpler by minimizing her participation. So instead the right weights can be learned from data automatically. Below we present such a learning algorithm that reduces this learning task to solving a linear programming problem.

5.3 Learning Algorithm

As any supervised learning algorithm, the proposed algorithm uses training data for learning. In the training data for each pair of object description x and y we know whether or not they co-refer. The learning is based on the generic disambiguation principle, that the affinity/similarity between two entity representations that co-refer is *likely* to be noticeably greater than that of two descriptions that do not co-refer. Using this principle, we can define the learning formulae for the problem of entity resolution. In defining constraints for the corresponding linear programming problem, we look at each VCS at a time. For each similarity edge (x_i, x_j), where x_i and x_j belong to the same cluster in the VCS, we formally define a set of constraints according to the generic disambiguation principle:

$$\begin{array}{l} s(x_i, x_j) \geq s(x_i, x_k) \quad (*) \\ s(x_i, x_j) \geq s(x_j, x_\ell) \quad (**) \end{array} \tag{4}$$

Here, $s(u, v)$ is computed according to (1), in which λ is a non-negative variable, and $(*)$ denotes the set of additional conditions on selecting x_i, x_j, x_k. Specifically, x_i, x_j, x_k should be chosen such that:

- there should be similarity edges (x_i, x_j), (x_i, x_k) in G,
- those edges should be in the same VCS,
- x_i and x_j co-refer, x_i and x_k do not co-refer.

The conditions specified as $(**)$ are similar to $(*)$, but for x_ℓ instead of x_k. System (4) might be overconstrained and as such might not have a solution. This system does not yet address the 'likely' part of the principle: the inequalities should hold most of the time, but not always. To address that, a slack is added to the system:

$$\left\{ \begin{array}{l} \text{Constraints:} \\ s(x_i, x_j) + \xi_{ijk} \geq s(x_i, x_k) \quad (*) \\ s(x_i, x_j) + \xi_{ij\ell} \geq s(x_j, x_\ell) \quad (**) \\ \xi_{ijk}, \xi_{ij\ell} \geq 0 \text{ for all } i, j, k, \ell \\ \\ \text{Objective:} \\ \text{Minimize } \left[\sum_{ijk} \xi_{ijk} + \sum_{ij\ell} \xi_{ij\ell}\right]. \end{array} \right. \tag{5}$$

System (5) always has a solution. Here, ξ_{ijk} are slack variables that take on any real nonnegative values. The objective is to minimize their sum. In addition, to make clear decisions, the connections strength between two descriptions that co-refer should clearly outweigh that of two descriptions that do not co-refer. This translates into requiring that the difference between $s(x_i, x_j)$ and $s(x_i, x_k)$ be maximized, and the same for x_ℓ.

$$\left\{ \begin{array}{l} \text{Constraints:} \\ \delta_{ijk} = s(x_i, x_j) - s(x_i, x_k) \quad (*) \\ \delta_{ij\ell} = s(x_i, x_j) - s(x_j, x_\ell) \quad (**) \\ \delta_{ijk} + \xi_{ijk} \geq 0 \quad (*) \\ \delta_{ij\ell} + \xi_{ij\ell} \geq 0 \quad (**) \\ \\ \text{Objective:} \\ \text{Minimize } \alpha \left[\sum_{ijk} \xi_{ijk} + \sum_{ij\ell} \xi_{ij\ell}\right] + \\ \qquad + (1 - \alpha) \left[-\sum_{ijk} \delta_{ijk} - \sum_{ij\ell} \delta_{ij\ell}\right]. \end{array} \right. \tag{6}$$

In (6), $c(u, v)$ is computed according to (3). The task is to learn the optimal path type weights w_i's along with the parameter λ by minimizing the objective. System (6) is a linear programming (LP) problem which can be solved by any known LP method or using an off-the-shelf LP solver. Linear programming in general is known to have efficient solutions [19].

5.4 Classifying Paths into Types

There are many possible ways to classify paths. The Path Type Model (PTM) classifies a path by looking at the types of edges the path is composed of. That is, each path can be viewed as a sequence of edges $\langle e_1, e_2, \ldots, e_k \rangle$. Each edge has a type associated with it. For example, in a publication database authors write papers and are affiliated with organizations. Hence there are two types of edges that correspond to the two types of relationships: E_1 for 'writes' and E_2 for 'is affiliated with'. A sequence of edge types can be viewed as a string, and PTM assigns different weight to each distinct string.

It is possible to think the number of path types will be huge: t^k, where t is the number of edge types and k is the length of paths. However, because each node type can only connect to certain types of edges, the number of path types is limited. For example, an *author* node can be associated with edges of types 'writes' (a paper) and 'is affiliated with' (an organization), but a *paper* node can only be incident to a 'writes' edge type. Therefore the number of path types will be far less than t^k and in practice, the size of training data is generally sufficient for the purpose of learning.

5.5 Example

To illustrate the concepts from the previous sections, consider the small scenario in Figure 7. It shows a small ER

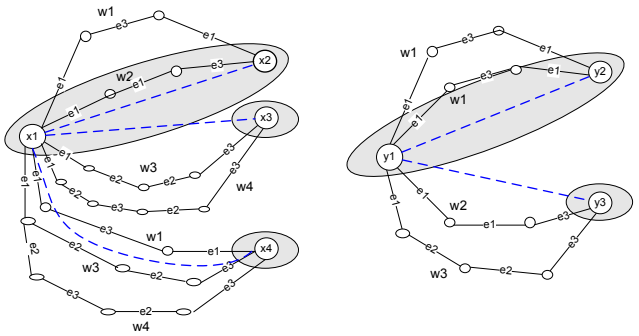


Figure 7: Example Graph.

graph with two VCSs. The first VCS has four references x_1, x_2, x_3, x_4 connected by three similarity edges about three real-world entities $\{\{x_1, x_2\}, \{x_3\}, \{x_4\}\}$. The second VCS has three references y_1, y_2, y_3 with two similarity edges about two real-world entities $\{\{y_1, y_2\}, \{y_3\}\}$. For simplicity, let us assume $\lambda = 1$ in (1) and thus $s(u, v) = c(u, v)$. Then $s(x_1, x_2) = w_1 + w_2$, $s(y_1, y_2) = 2w_1$, and so on, where w_i's are as illustrated in Figure 7. Thus we have:

$$\begin{cases} \text{Constraints:} \\ \delta_1 = [w_1 + w_2] - [w_3 + w_4] \\ \delta_2 = [w_1 + w_2] - [w_1 + w_3 + w_4] \\ \delta_3 = [2w_1] - [w_2 + w_3] \\ \delta_1 + \xi_1 \geq 0 \\ \delta_2 + \xi_2 \geq 0 \\ \delta_3 + \xi_3 \geq 0 \\ 0 \leq \xi_1, \xi_2, \xi_3 \\ 0 \leq w_1, w_2, w_3, w_4 \leq 1 \\ \\ \text{Objective: Minimize} \\ \alpha\,[\xi_1 + \xi_2 + \xi_3] + (1 - \alpha)\,[-3w_1 - w_2 + 3w_3 + 2w_4] \end{cases} \tag{7}$$

After solving the linear program (7) using a LP solver, we will get the answer for the PTM: $w_1 = w_2 = 1$ and $w_3 = w_4 = 0$.

6. EXPERIMENTAL EVALUATION

In this section, we experimentally study the proposed techniques on three datasets taken from the Movie and Publication domains. We first present the experimental setup in Section 6.1, followed by the experiments that test the proposed entity resolution approach in Section 6.2.

6.1 Experimental Setup

6.1.1 Datasets

We will use three datasets: `RealPub`, `RealMov` and `Cora` to test our algorithms. RealPub and RealMov are derived from real manually cleaned data, so they initially do not contain uncertain references. To test our approach, we will use a standard technique employed by data cleaning practitioners: we introduce uncertainty/errors in our datasets manually in a controlled fashion, and then examine the effect of uncertainty on the quality of various entity resolution algorithms.

RealPub. RealPub is a real dataset derived from two public-domain sources: CiteSeer and HPSearch. It is a publication dataset that describes authors, their publications and their affiliations. It contains entities of four types: *papers* (11,682 entities), *authors* (14,590 entities), *departments* (3084 entities) and *organizations* (1494 entities). This dataset contains four types of (regular) relationships: *paper_author*, *author_dept*, *author_org*, and *org_dept*, which map papers to their authors, authors to their affiliations, and the departments to the organizations respectively.

Introducing Uncertainty. In RealPub, all author references are in the format of '*full first name + last name*'. We introduce uncertainty in this dataset by pretending we only know '*first initial + last name*' for the author descriptions. For example, for two representations r_1 ='John Smith' and r_2 ='Jane Smith' we pretend we only know r_1 ='J. Smith' and r_2 ='J. Smith', so that r_1 and r_2 will be put in the same VCS. We refer to the authors who have the same "*first initial + last name*" as *uncertain authors* and their names as *uncertain names*. The goal is to correctly resolve the uncertain *author* entities, i.e., separate the uncertain authors into groups such that each group consists of author references with the same full names.

RealMov. RealMov is a real dataset constructed from the Stanford Movies Dataset [37]. It contains entities of three types: *movies* (11,453 entities), *studios* (992 entities), and *people* (22,121 entities). It also contains five types of (regular) relationships: *movie_actor*, *movie_director*, *movie_producer*, *producingStudio*, and *distributingStudio*, which map movies to their actors, directors, producers, producing studios and distributing studios respectively.

Introducing Uncertainty. In our experiments, the type of entity we are dealing with is *people*, and specifically, *director*. Assume that RealMov stores information about $d_1, d_2, \ldots, d_n$ director entities. The uncertainty will be introduced in a fashion similar to that of the recent KDD Cup. We choose randomly a fraction ρ of those directors and make all their references uncertain. To achieve that, we group every two directors and simulate the desired uncertainty by changing all the director references in the same group such that each reference can refer to either of the directors in this group. For example, if the directors d_1, d_2, ..., d_{10} were chosen, we group the "uncertain" directors d_1, d_2, ..., d_{10} in groups of two, e.g. $\{d_1, d_2\}$, $\{d_3, d_4\}$, ..., $\{d_9, d_{10}\}$. We assume all references of d_1 can refer to either d_1 or d_2, and so on, whereas for the rest of the directors $d_{11}, d_{12}, \ldots, d_n$, all their references will still uniquely identify the right director. Based on the typical degree of uncertainty in real-world dataset [22], we set ρ to 10% in our experiments.

Cora Cora dataset has been used for disambiguation by many other researchers. It was prepared by the RIDDLE project [1] and then cleaned further. It contains 1295 citations for 132 paper entities. Each record of citations consists of 12 attributes: *authors*,*title*,*institution*,*venue*,*address*,*publisher*, *year*,*month*, *volume*,*pages*,*editors*,and *notes*. The goal is to correctly identify the duplicate citations.

6.1.2 Measuring Quality

Traditionally, the quality of record linkage is measured using *precision* and *recall*, and their harmonic mean known as *F1* measure. Although precision and recall capture the quality of results well most of the time, it is now often argued that such measures are not fully adequate for the problem of entity resolution, because they are representation-centric measures, whereas entity-centric measures are often preferred for entity resolution.

Let us illustrate this by an example. Assume a database of ten objects $\{o_1, \ldots, o_{10}\}$. Assume o_1 has many representations $\{r_{11}, \ldots, r_{1n_1}\}$, where $n_1 = 100$ for example. Suppose the other objects have only a few representations, say two: $\{r_{i1}, r_{i2}\}$ for each o_i. Now let us assume that two different algorithms have been applied to this database and we will compare their outcome:

- **Algorithm 1** has correctly consolidate all the representations of o_1 by placing them in one cluster $\{r_{11}, \ldots, r_{1n_1}\}$. But it failed on each of the other objects $o_2, \ldots, o_{10}$, by placing their representations into two clusters: $\{r_{i1}\}$ and $\{r_{i2}\}$ for each o_i, instead of one $\{r_{i1}, r_{i2}\}$ cluster.

- **Algorithm 2** has correctly consolidated the representations of objects $o_2, \ldots, o_{10}$. But it slightly failed on the representations of object o_1, by creating two clusters $\{r_{11}\}$ and $\{r_{12}, \ldots, r_{1n_1}\}$ instead of one $\{r_{11}, r_{12}, \ldots, r_{1n_1}\}$ cluster.

If we use precision/recall as measures, Algorithm 1 will be significantly better than Algorithm 2 because it has consolidated more pairs correctly. For this example, Algorithm 1 missed 9 pairs of representations that co-refer (i.e.,$\{r_{i1}, r_{i2}\}$, $i = 2, \ldots, 10$), while Algorithm 2 missed 99 pairs (i.e., $\{r_{11}, r_{12}\}$, $\ldots, \{r_{11}, r_{1,100}\}$). However, if we consider the effectiveness of the algorithms as the number of objects that the algorithms can correctly resolve, then Algorithm 2 is significantly better since it failed slightly only on one object, while Algorithm 1 failed on 9 out of 10 objects.

Thus, we will also use other well-known measures: *purity, inverse purity* and their harmonic mean F_P*-measure*. Purity measures the homogeneity of the clusters when evaluated against pre-classification, while inverse purity measures the stability of the classes when split into clusters. F_P-measure is the harmonic mean of the purity and inverse purity. [1]

6.1.3 Resolution Algorithms

We test and compare four algorithms that employ features, contexts and relationships.

FBS technique. FBS technique is the traditional approach of using features only for entity resolution.

Context-based techniques. To compare our approach with context-based approaches that use directly associated neighborhoods, we have implemented one of the best known context-based methods that could be applied to our datasets, which we will refer to as Context [5]. We always set all the parameters of Context to their optimal values, thus giving it an advantage over all other tested techniques.

[1]The purity measure is based on the precision. Each resulting cluster P from a partitioning $\mathcal{P}$ of the overall representation set $\mathcal{D}$ is treated as if it were the result of a query. Each set L of representations of a partitioning $\mathcal{L}$, which is obtained by manually labeling the classes, is treated as if it were the desired set/class of representations for a query. The two partitioning $\mathcal{P}$ and $\mathcal{L}$ are then compared as follows. The precision of a cluster $P \in \mathcal{P}$ for a given category $L \in \mathcal{L}$ is given by $\text{Precision}(P; L) := \frac{|P \cap L|}{|P|}$. The overall value for purity is computed by taking the weighted average of maximal precision values: $\text{Purity}(\mathcal{P}, \mathcal{L}) = \sum_{P \in \mathcal{P}} \frac{|P|}{|D|} \max_{L \in \mathcal{L}} \text{Precision}(P, L)$. The inverse purity is computed in a similar fashion: $\text{InversePurity}(\mathcal{P}, \mathcal{L}) = \sum_{L \in \mathcal{L}} \frac{|L|}{|D|} \max_{P \in \mathcal{P}} \text{Precision}(L, P)$.

Algorithms	RealPub		RealMov	
	F1	F_P	F1	F_P
FBS	0.538	0.740	0.597	0.785
Context	0.725	0.853	0.760	0.849
RelER	0.766	0.889	0.760	0.849
RelAA	0.774	0.892	0.848	0.869

Table 1: Comparing Different Approaches

Relationship-based techniques. We will test the basic and adaptive relationship-based techniques presented in Sections 4 and 5. We refer to the former as RelER (Relationship-based Entity Resolution) and the latter as RelAA (Relationship-based Adaptive Algorithm). For the basic algorithm RelER, the connection strengths are computed using WM model [22] based on random walk since it is the best known non-adaptive model. For the adaptive algorithm RelAA, we will randomly divide the dataset to two parts: 50% of the dataset is used for training the model to get proper path weights and the rest 50% of the dataset is used for testing. We choose the parameter $L = 6$ for all experiments. The results reported in this section will be averaged from multiple runs. To fairly compare the adaptive algorithm with all others, we report the results of all algorithms on the same part of (test) data.

We will report the results of different approaches on RealPub and RealMov datasets using both $F1$ and F_P measures. For Cora dataset, since the number of citations is not enough for training purpose, we will just use the basic algorithm RelER and compare the precision/recall/F1 with those reported in other papers.

6.2 Experiments on RealPub and RealMov

In this section, we empirically evaluate the quality and efficiency of the overall approach, which are traditionally of interest in the context of entity resolution.

Experiment 1 (Comparing approaches). In Table 1, we compare the quality of the various entity resolution approaches on RealPub and RealMov datasets using $F1$ and F_P measures. In the table, we can see that for both RealPub and RealMov datasets, RelAA performs the best. Context-based and relationship-based algorithms always get better results than FBS. This is expected since each of the three methodology complement the previous one with the analysis of more information that is available in a dataset: features, then context, and then relationships. Notice that for RealPub, the results of RelER and RelAA are very close to each other and much better than Context and FBS. For RealMov, the results of basic RelER algorithm is the same as Context, while adaptive algorithm RelAA is much better than all the other algorithms. This is because in RealMov, there are more types of relationships, and when the paths' length is long (i.e., parameter L is large), using a fixed model (e.g., WM model in our experiments) to compute connection strengths sometimes may confuse the basic resolution algorithm. On the other hand, the adaptive algorithm instead learns the weights from data, leading to better quality of entity resolution. □

Experiment 2 (Learning threshold). Varying the merging threshold employed by merging algorithms (see Section 4) causes those algorithms to produce various results that form a trade-off between the resulting precision/recall and purity/inverse purity. Figure 8 plots average purity against

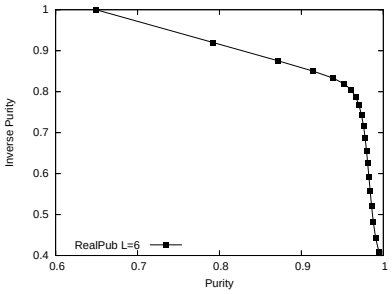


Figure 8: Trade-off between Purity and Inverse Purity when varying threshold.

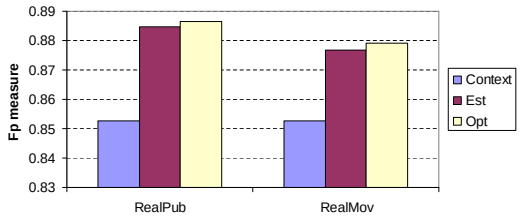


Figure 9: Learning threshold.

average inverse purity with the merging threshold varying between 0 and 1 on RealPub dataset. From this figure, we can see that there is a trade-off between the resulting quality of clusters and entities, and there exists an optimum result for a certain threshold value in terms of F_P measure.

In our experiments we set the the merging threshold by learning it from data. First, we load the ER graph for the RealPub or RealMov dataset and partition all the VCS's of this ER graph into two sets: the training and test sets. We determine the optimal threshold value on the training set. This value serves as an *estimation* of the optimal threshold value for the test set. We then apply that threshold on the test set. Figure 9 illustrates the learning results of F_P for the two datasets, RealPub and RealMov, where 10% of all VCS's are utilized for training purposes and the rest 90%—for testing. We report the results on the learned threshold as 'Est' and the true optimal value on testing data as 'Opt'. For Context method, we always use the optimal threshold value. Figure 9 demonstrates that the learned threshold serves as a good estimation of the optimal threshold and that the proposed algorithm significantly outperforms the Context method. □

Experiment 3 (Efficiency). This experiment tests the efficiency and shows the scalability of the proposed approach. The bottleneck of our approach is the algorithm for discovering all L-short simple paths and the figure essentially plots the cost of that phase of the algorithm. In [21] we study several optimizations of that algorithm, which improve the performance by 1–2 orders of magnitude. We employ the same optimizations in this paper as well. To simulate the increasing size of the data, we tested on different fraction of dataset we have. Figure 10 shows the execution time as a function of the fraction of publications loaded from RealPub dataset, e.g., 0.1 corresponds to 10% of the dataset, 1.0 corresponds to the whole dataset. For a graph with 40K nodes, it takes less than 0.03 seconds. In our ongoing work, we are trying to improve the scalability of the algorithm. □

Experiment 4 (Choosing alpha). Figure 11 studies the effect of parameter α, which controls the contribution of the

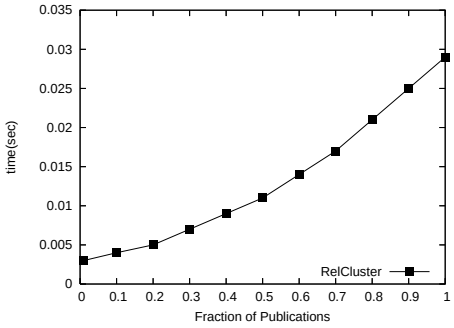


Figure 10: Execution time vs. database size.

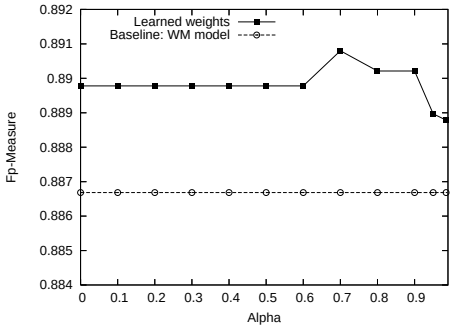


Figure 11: Comparing the learning algorithm with baseline and the impact of Alpha on RealPub.

two objectives in system (6) from Section 5, on the resolution quality of the algorithm. It shows that there is a trade-off between the two objectives, and that the algorithm that learns weights outperforms the baseline, which uses WM model, for any $\alpha \in (0, 1)$. In our experiment, the optimal result for RealPub was achieved when α is equal to 0.7. □

Experiment 5 (Handling random noise). To further demonstrate the advantage of the adaptive model, we gradually add uniform random noise to the datasets by introducing random relationships/edges that randomly connect two nodes in the graph. The random relationships are added to both training and test data. These random relationships are meaningless for the purpose of disambiguation, and therefore the paths with random edge(s) should ideally have no effect on the connection strengths. Figure 12 shows the impact of random relationships added to RealPub data on the result of the resolution algorithm. The more random relationships are added, the worse the baseline algorithm performs since it is not adaptive. The learning algorithm, on the other hand, can correctly capture all the meaningless path types. Thus its curve stays flat as the amount of noise increases. □

6.3 Experiments on Cora

We apply our relationship-based algorithm on Cora dataset and compare the results of precision/recall/F1 with those reported by other researchers. One of the best results published in the past is by [15], which used similar methodology as ours. The approach they proposed is referred by `DepGraph` and the baseline approach is referred by `InDepDec` in that paper. We compare our approach with these two and the results are presented in Table 2. Although the improvement might look marginal, notice that our approach almost doubled the improvement of DepGraph to baseline. Another

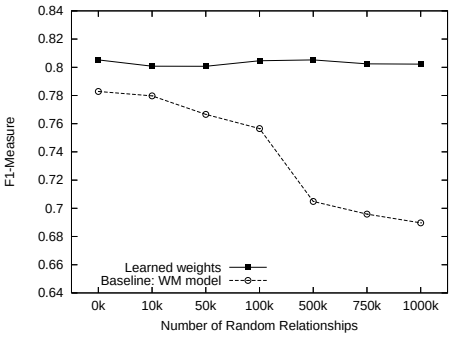


Figure 12: Impact of Random Relationships.

Algorithms	Precision	Recall	F1	Improve
InDepDec	0.985	0.913	0.948	-
DepGraph	0.985	0.924	0.954	0.6%
RelER	0.935	0.984	**0.959**	1.1%

Table 2: Comparing different approaches on Cora

observation is that the precision of our approach is worse although the recall is better. This is because we employed the adaptation of a canopy method that brings all the references that can potentially co-refer together to the same VCS and therefore guaranteed the matching references will almost always be found. In the meantime, this strategy reduces the precision since it will group references that do not co-refer.

7. CONCLUSION

In this paper, we have developed a novel domain-independent entity resolution approach and have presented its empirical evaluation on datasets taken from two different domains. The approach leverages an analysis of the entity-relationship graph of the dataset being processed for improving the quality of the entity resolution. In addition, we have developed a method that minimizes the required domain-analyst participation and achieves even higher disambiguation quality by being able to self-tune the overall approach to the dataset being processed.

8. REFERENCES

[1] Riddle:http://www.cs.utexas.edu/users/ml/riddle/.

[2] Kdd workshop on data cleaning, record linkage, and object consolidation, 2003.

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

[4] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *WWW*, 2005.

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

[6] I. Bhattacharya and L. Getoor. Relational clustering for multi-type entity resolution. In *MRDM Workshop*, 2005.

[7] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. 29(2):4–12, June 2006.

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

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

[10] S. Chaudhuri, K. Ganjam, V. Ganti, R. Kapoor, V. Narasayya, and T. Vassilakis. Data cleaning in Microsoft SQL Server 2005. In *SIGMOD*, 2005.

[11] Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Exploiting relationships for object consolidation. In *IQIS Workshop at ACM SIGMOD Conference*, 2005.

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

[13] I. G. Councill, H. Li, Z. Zhuang, S. Debnath, L. Bolelli, W. C. Lee, A. Sivasubramaniam, and C. L. Giles. Learning metadata from the evidence in an on-line citation matching scheme. In *JCDL*, 2006.

[14] A. Culotta and A. McCallum. Joint deduplication of multiple record types in relational data. In *CIKM*, 2005.

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

[16] I. Fellegi and A. Sunter. A theory for record linkage. *Journal of Amer. Statistical Association*, 64(328):1183–1210, 1969.

[17] H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *JCDL*, 2004.

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

[19] F. Hillier and G. Lieberman. *Introduction to operations research*. McGraw-Hill, 2001.

[20] R. Holzer, B. Malin, and L. Sweeney. Email alias detection using social network analysis. In *LinkKDD Workshop*, 2005.

[21] D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM TODS*, 31(2), June 2006.

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

[23] D. V. Kalashnikov, S. Mehrotra, Z. Chen, R. Nuray-Turan, and N. Ashish. Disambiguation algorithm for people search on the web. In *ICDE poster*, 2007.

[24] S. Lawrence, K. Bollacker, and C. L. Giles. Autonomous citation matching. In *Proceedings of the Third International Conference on Autonomous Agents*, 1999.

[25] X. Li, P. Morie, and D. Roth. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *AAAI*, 2004.

[26] B. Malin. Unsupervised name disambiguation via social network similarity. In *Workshop on Link Analysis, Counterterrorism, and Security*, 2005.

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

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

[29] E. Minkov, W. W. Cohen, and A. Y. Ng. Contextual search and name disambiguation in email using graphs. In *ACM SIGIR*, 2006.

[30] R. Nuray-Turan, D. V. Kalashnikov, and S. Mehrotra. Self-tuning in graph-based reference disambiguation. In *DASFAA*, 2007.

[31] B.-W. On, E. Elmacioglu, D. Lee, J. Kang, and J. Pei. Improving grouped-entity resolution using quasi-cliques. In *ICDM*, 2006.

[32] B.-W. On, D. Lee, J. Kang, and P. Mitra. Comparative study of name disambiguation problem using a scalable blocking-based framework. In *JCDL*, 2005.

[33] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS*, 2002.

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

[35] P. Singla and P. Domingos. Multi-relational record linkage. In *MRDM Workshop*, 2004.

[36] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *SIGKDD*, 2002.

[37] G. Wiederhold. The movies dataset. `http://www-db.stanford.edu/pub/movies/doc.html`.