Appeared in *SIAM International Conference on Data Mining (SDM)*, April 21–23, 2005

Exploiting relationships for domain-independent data cleaning*†

Dmitri V. Kalashnikov Sharad Mehrotra Zhaoqi Chen

Computer Science Department
University of California, Irvine

Abstract

In this paper, we address the problem of *reference disambiguation*. Specifically, we consider a situation where entities in the database are referred to using descriptions (e.g., a set of instantiated attributes). The objective of reference disambiguation is to identify the unique entity to which each description corresponds. The key difference between the approach we propose (called RelDC) and the traditional techniques is that RelDC analyzes not only object features but also inter-object relationships to improve the disambiguation quality. Our extensive experiments over two real datasets and over synthetic datasets show that analysis of relationships significantly improves quality of the result.

1 Introduction

Recent surveys [3] show that more than 80% of researchers working on data mining projects spend more than 40% of their project time on cleaning and preparation of data. The data cleaning problem often arises when information from heterogeneous sources is merged to create a single database. Many distinct data cleaning challenges have been identified in the literature: dealing with missing data [20], handling erroneous data [21], record linkage [6, 7], and so on. In this paper, we address one such challenge, which we refer to as *reference disambiguation*.

The reference disambiguation problem arises when entities in a database contain references to other entities. If entities were referred to using unique identifiers, then disambiguating those references would be straightforward. Instead, frequently, entities are represented using properties/descriptions that may not uniquely identify them leading to ambiguity. For instance, a database may store information about two distinct individuals 'Donald L. White' and 'Donald E. White', both of whom are referred to as 'D. White' in another database. References may also be ambiguous due to differences in the representations of the same entity and errors in data entries (e.g., 'Don White' misspelled as 'Don Whitex'). The **goal** of reference disambiguation is for each reference to correctly identify the unique entity it refers to.

The reference disambiguation problem is related to the problem of *record deduplication* or *record linkage* [7, 6] that often arises when multiple tables (from different data sources) are merged to create a single table. The causes of record linkage and reference disambiguation problems are similar; viz., differences in representations of objects across different datasets, data entry errors, etc. The differences between the two can be intuitively viewed using the relational terminology as follows: while the record linkage problem consists of determining when two records are the same, reference disambiguation corresponds to ensuring that references (i.e., "foreign keys"[1]) in a database point to the correct entities.

Given the tight relationship between the two data cleaning tasks and the similarity of their causes, existing approaches to record linkage can be adapted for reference disambiguation. In particular, *feature-based similarity (FBS)* methods that analyze similarity of record attribute values (to determine whether two records are the same) can be used to determine if a particular reference corresponds to a given entity or not. This paper argues that the quality of disambiguation can be significantly improved by exploring additional semantic information. In particular, we observe that references occur within a context and define relationships/connections between entities. For instance, 'D. White' might be used to refer to an author in the context of a particular publication. This publication might also refer to different authors, which can be linked to their affiliated organizations etc, forming chains of relationships among entities. Such knowledge can be exploited alongside attribute-based similarity resulting in improved accuracy of disambiguation.

In this paper, we propose a domain-independent

*RelDC project (`http://www.ics.uci.edu/~dvk/RelDC`)

†This work was supported in part by NSF grants 0331707, 0331690, and IRI-9703120.

[1]We are using the term foreign key loosely. Usually, foreign key refers to a unique identifier of an entity in another table. Instead, foreign key above means the set of properties that serve as a reference to an entity.

data cleaning approach for reference disambiguation, referred to as Relationship-based Data Cleaning (RelDC), which systematically exploits not only features but also relationships among entities for the purpose of disambiguation. RelDC views the database as a graph of entities that are linked to each other via relationships. It first utilizes a feature-based method to identify a set of candidate entities (choices) for a reference to be disambiguated. Graph theoretic techniques are then used to discover and analyze relationships that exist between the entity containing the reference and the set of candidates.

The primary contributions of this paper are: (1) developing a systematic approach to exploiting both attributes as well as relationships among entities for reference disambiguation (2) establishing that exploiting relationships can significantly improve the quality of reference disambiguation by testing the developed approach over 2 real-world datasets as well as synthetic datasets.

This paper presents the core of the RelDC approach, details of RelDC can be found in [16] where we discuss various implementations, optimizations, computational complexity, sample content and sample graphs for real datasets, and other issues not covered in this paper. The rest of this paper is organized as follows. Section 2 presents a motivational example. In Section 3, we precisely formulate the problem of reference disambiguation and introduce notation that will help explain the RelDC approach. Section 4 describes the RelDC approach. The empirical results of RelDC are presented in Section 5. Section 6 contains the related work, and Section 7 concludes the paper.

2 Motivation for analyzing relationships

In this section we will use an instance of the "author matching" problem to illustrate that exploiting relationships among entities can improve the quality of reference disambiguation. We will also schematically describe one approach that analyzes relationships in a systematic domain-independent fashion. Consider a

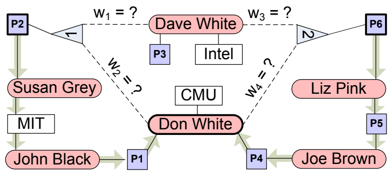


Figure 1: Graph for the publications example

database about *authors* and *publications*. Authors are represented in the database using the attributes ⟨`id`, `authorName`, `affiliation`⟩ and information about papers is stored in the form ⟨`id`, `title`, `authorRef1`, ..., `authorRefN`⟩. Consider a toy database consisting of the following *authors* and *publications* records.

1. ⟨A_1, 'Dave White', 'Intel'⟩,
2. ⟨A_2, 'Don White', 'CMU'⟩,
3. ⟨A_3, 'Susan Grey', 'MIT'⟩,
4. ⟨A_4, 'John Black', 'MIT'⟩,
5. ⟨A_5, 'Joe Brown', unknown⟩,
6. ⟨A_6, 'Liz Pink', unknown⟩.

1. ⟨P_1, 'Databases ...', 'John Black', 'Don White'⟩,
2. ⟨P_2, 'Multimedia ...', 'Sue Grey', **'D. White'**⟩,
3. ⟨P_3, 'Title3 ...', 'Dave White'⟩,
4. ⟨P_4, 'Title5 ...', 'Don White', 'Joe Brown'⟩,
5. ⟨P_5, 'Title6 ...', 'Joe Brown', 'Liz Pink'⟩,
6. ⟨P_6, 'Title7 ...', 'Liz Pink', **'D. White'**⟩.

The goal of the author matching problem is to identify for each `authorRef` in each paper the correct author it refers to.

We can use existing feature-based similarity (FBS) techniques to compare the description contained in each `authorRef` in papers with values in `authorName` attribute in authors. This would allow us to resolve almost every `authorRef` references in the above example. For instance, such methods would identify that 'Sue Grey' reference in P_2 refers to A_3 ('Susan Grey'). The only exception will be 'D. White' references in P_2 and P_6: 'D. White' could match either A_1 ('Dave White') or A_2 ('Don White').

Perhaps, we could disambiguate the reference 'D. White' in P_2 and P_6 by exploiting additional attributes. For instance, the titles of papers P_1 and P_2 might be similar while titles of P_2 and P_3 might not, suggesting that 'D. White' of P_2 is indeed 'Don White' of paper P_1. We next show that it may still be possible to disambiguate the references 'D. White' in P_2 and P_6 by analyzing relationships among entities even if we are unable to disambiguate the references using title (or other attributes).

First, we observe that author 'Don White' has co-authored a paper (P_1) with 'John Black' who is at MIT, while the author 'Dave White' does not have any co-authored papers with authors at MIT. We can use this observation to disambiguate between the two authors. In particular, since the co-author of 'D. White' in P_2 is 'Susan Grey' of MIT, there is a higher likelihood that the author 'D. White' in P_2 is 'Don White'. The reason is that the data suggests a connection between author 'Don White' with MIT and an absence of it between 'Dave White' and MIT.

Second, we observe that author 'Don White' has co-authored a paper ($P4$) with 'Joe Brown' who in turn has co-authored a paper with 'Liz Pink'. In contrast, author 'Dave White' has not co-authored any papers with either 'Liz Pink' or 'Joe Brown'. Since 'Liz Pink' is a co-author of P_6, there is a higher likelihood that 'D. White' in P_6 refers to author 'Don White' compared to author 'Dave White'. The reason is that often co-author networks form groups/clusters of authors that do related research and may publish with each other. The data suggests that 'Don White', 'Joe Brown' and 'Liz Pink' are part of the cluster, while 'Dave White' is not.

At first glance, the analysis above (used to disambiguate references that could not be resolved using conventional feature-based techniques) may seem ad-hoc and domain dependent. A general principle emerges if we view the database as a graph of inter-connected entities (modeled as nodes) linked to each other via relationships (modeled as edges). Figure 1 illustrates the entity-relationship graph corresponding to the toy database consisting of *authors* and *papers* records. In the graph, entities containing references are linked to the entities they refer to. For instance, since the reference 'Sue Grey' in P_2 is unambiguously resolved to author 'Susan Grey', paper P_2 is connected by an edge to author A_3. Similarly, paper P_5 is connected to authors A_5 ('Joe Brown') and A_6 ('Liz Pink'). The ambiguity of the references 'D. White' in P_2 and P_6 is captured by linking papers P_2 and P_6 to both 'Dave White' and 'Don White' via two "choice nodes" (labeled '1' and '2' in the figure). These "choice nodes" serve as `OR`-nodes in the graph and represent the fact that the reference 'D. White' refers to either one of the entities linked to the choice nodes.

Given the graph view of the toy database, the analysis we used to disambiguate 'D. White' in P_2 and P_6 can be viewed as an application of the following general principle:

Context Attraction Principle (CAP): *If reference r made in the context of entity x refer to an entity y_j, whereas the description provided by r matches multiple entities $y_1, \ldots, y_j, \ldots, y_N$, then x and y_j are likely to be more strongly connected to each other via chains of relationships than x and y_l ($l = 1, 2, \ldots, N$; $l \neq j$).* $\square$

The first observation we made, regarding disambiguation of 'D. White' in P_2, corresponds to the presence of the following path (i.e., *relationship chain* or *connection*) between the nodes 'Don White' and P_2 in the graph: $P_2 \rightarrow$ 'Susan Grey' $\rightarrow$ 'MIT' $\rightarrow$ 'John Black' $\rightarrow P_1 \rightarrow$ 'Don White'. Similarly, the second observation, regarding disambiguation of 'D. White' in P_6 as 'Don White', was based on the presence of the following path: $P_6 \rightarrow$ 'Liz Pink' $\rightarrow P_5 \rightarrow$ 'Joe Brown' $\rightarrow P_4 \rightarrow$ 'Don White'. There were no paths between P_2 and 'Dave White' or between P_6 and 'Dave White' (if we ignore '1' and '2' nodes). Thus, after applying the CAP principle, we concluded that the 'D. White' references in both cases probably corresponded to the author 'Don White'. In general, there could have been paths not only between P_2 (P_6) and 'Don White', but also between P_2 (P_6) and 'Dave White'. In that case, to determine if 'D. White' is 'Don White' or 'Dave White' we should have been able to measure whether 'Don White' or 'Dave White' is more strongly connected to P_2 (P_6).

The generic approach therefore first *discovers connections* between the entity, in the context of which the reference appears and the matching candidates for that reference. It then *measures the connection strength* of the discovered connections in order to give preference to one of the matching candidates. The above discussion naturally leads to two questions:

1. Does the context attraction principle hold over real datasets. That is, if we disambiguate references based on the principle, will the references be correctly disambiguated?
2. Can we design a generic solution to exploiting relationships for disambiguation?

Of course, the second question is only important if the answer to the first is positive. However, we cannot really answer the first unless we develop a general strategy to exploiting relationships for disambiguation and testing it over real data. We will develop one such general, domain-independent strategy for exploiting relationships for disambiguation, which we refer to as RelDC in Section 4. We perform extensive testing of RelDC over both real data from two different domains as well as synthetic data to establish that exploiting relationships (as is done by RelDC) significantly improves the data quality. Before we develop RelDC, we first develop notation and concepts needed to explain our approach in Section 3.

3 Problem formalization

3.1 Notation Let $\mathcal{D}$ be the database which contains references that are to be resolved. Let $X = \{x_1, x_2, \ldots, x_{|X|}\}$ be the set of all entities in $\mathcal{D}$. Entities here have the same meaning as in the E/R model. Each entity x_i consists of a set of properties and contains a set of n_{x_i} references $x_i.r_1, x_i.r_2, \ldots, x_i.r_{n_{x_i}}$. Each reference $x_i.r_k$ is essentially a description and may itself consist of one or more attributes $x_i.r_k.b_1, x_i.r_k.b_2, \ldots$. For instance, in the example from Section 2, *paper* entities contain one-attribute `authorRef` references in the form $\langle$*author name*$\rangle$. If, besides author names, author affiliation were also stored in the *paper records*, then `authorRef` references would have consisted of two attributes – $\langle$*author name*, *author affiliation*$\rangle$.

Choice set. Each reference $x_i.r_k$ semantically refers to a single specific entity in X which we denote by $d[x_i.r_k]$. The description provided by $x_i.r_k$ may, however, match a set of one or more entities in X. We refer to this set as the *choice set* of reference $x_i.r_k$ and denote it by $CS[x_i.r_k]$. The choice set consists of all the entities that $x_i.r_k$ could potentially refer to. We assume

$CS[x_i.r_k]$ is given for each $x_i.r_k$. If it is not given, we assume a feature-based similarity approach is used to construct $CS[x_i.r_k]$ by choosing all of the candidates such that FBS similarity between them and $x_i.r_k$ exceed a given threshold. To simplify notation, we will always assume $CS[x_i.r_k]$ has N (i.e., $N = |CS[x_i.r_k]|$) elements y_1,y_2,...,y_N.

3.2 The Entity-Relationship Graph RelDC views the resulting database $\mathcal{D}$ as an undirected entity-relationship graph (also known as *Attributed Relational Graph (ARG)*) $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node corresponds to an entity and each edge to a relationship. Notation $v[x_i]$ denotes the vertex in G that corresponds to entity $x_i \in X$. Note that if entity u contains a reference to entity v, then the nodes in the graph corresponding to u and v are linked since a reference establishes a relationship between the two entities. For instance, `authorRef` reference from paper P to author A corresponds to "A writes P" relationship.

In the graph G, edges have weights, nodes do not have weights. Each edge weight is a real number in $[0, 1]$, which reflects the degree of confidence the relationship, corresponding to the edge, exists. For instance, in the context of our author matching example, if we are 100% confident 'John Black' is ffiliated with MIT, then we assign weight of 1 to the corresponding edge. However, if we are only 80% confident, we assign the weight of 0.80 to that edge. By default, all weights are equal to 1. Notation "edge label" means the same as "edge weight".

References and linking. If $CS[x_i.r_k]$ has only one element, then $x_i.r_k$ is resolved to y_1, and graph G contains an edge between $v[x_i]$ and $v[y_1]$. This edge is assigned a weight of 1 to denote that the algorithm is 100% confident that $d[x_i.r_k]$ is y_1. If $CS[x_i.r_k]$ has

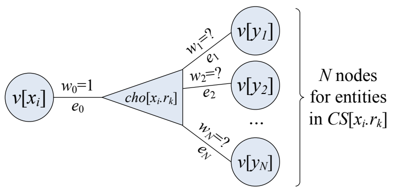


Figure 2: A choice node

more than 1 elements, then graph G contains a **choice node** $cho[x_i.r_k]$, as shown in Figure 2, to reflect the fact that $d[x_i.r_k]$ can be one of $y_1, y_2, \ldots, y_N$. Node $cho[x_i.r_k]$ is linked with node $v[x_i]$ via edge $e_0 = (v[x_i], cho[x_i.r_k])$. Node $cho[x_i.r_k]$ is also linked with N nodes $v[y_1], v[y_2], \ldots, v[y_N]$, for each y_j in $CS[x_i.r_k]$, via edges $e_j = (cho[x_i.r_k], v[y_j])$ $(j = 1, 2, \ldots, N)$. Nodes $v[y_1], v[y_2], \ldots, v[y_N]$ are called the *options* of choice $cho[x_i.r_k]$. Edges $e_1, e_2, \ldots, e_N$ are called the *option-edges* of choice $cho[x_i.r_k]$. The weights of option-edges are called *option-edge weights* or simply *option weights.* The weight of edge e_0 is 1. Each weight w_j of edges e_j $(j = 1, 2, \ldots, N)$ is undefined initially. Since these option-edges $e_1, e_2, \ldots, e_N$ represent mutually exclusive alternatives, the sum of their weights should be 1: $w_1 + w_2 + \cdots + w_N = 1$.

3.3 The objective of reference disambiguation To *resolve* reference $x_i.r_k$ means to choose one entity y_j from $CS[x_i.r_k]$ in order to determine $d[x_i.r_k]$. If entity y_j is chosen as the outcome of such a disambiguation, then $x_i.r_k$ is said to be *resolved to* y_j or simply *resolved.* Reference $x_i.r_k$ is said to be *resolved correctly* if this y_j is $d[x_i.r_k]$. Notice, if $CS[x_i.r_k]$ has just one element y_1 (i.e., $N = 1$), then reference $x_i.r_k$ is automatically resolved to y_1. Thus reference $x_i.r_k$ is said to be *unresolved* or *uncertain* if it is not resolved yet to any y_j and also $N > 1$.

From the graph theoretic perspective, to resolve $x_i.r_k$ means to assign weights of 1 to one edge e_j, $1 \leq j \leq N$ and assign weights of 0 to the other $N - 1$ edges $e_1, e_2, \ldots, e_{j-1}, e_{j+1}, \ldots, e_N$. This will indicate that the algorithm chooses y_j as $d[x_i.r_k]$.

The **goal** of reference disambiguation is to resolve all references as correctly as possible, that is for each reference $x_i.r_k$ to correctly identify $d[x_i.r_k]$. We will use notation $Resolve(x_i.r_k)$ to refer to the procedure which resolves $x_i.r_k$. The *goal* is thus to construct such $Resolve(\cdot)$ which should be as accurate as possible. The *accuracy* of reference disambiguation is the fraction of references being resolved that are resolved correctly.

The **alternative goal** is for each $y_j \in CS[x_i.r_k]$ to associate weight w_j that reflects the degree of confidence that y_j is $d[x_i.r_k]$. For that alternative goal, $Resolve(x_i.r_k)$ should label each edge e_j with such a weight. Those weights can be **interpreted** later to achieve the main goal: for each $x_i.r_k$ try to identify only one y_j as $d[x_i.r_k]$ correctly. We emphasize this alternative goal since most of the discussion of RelDC approach is devoted to one approach for computing those weights. An interpretation of those weights (in order to try to identify $d[x_i.r_k]$) is a small final step of RelDC. Namely, we achieve this by picking y_j such that w_j is the largest among $w_1, w_2, \ldots, w_N$. That is, the outcome of $Resolve(x_i.r_k)$ is $y_j : w_j = \max_{l=1}^{N} w_l$.

3.4 Connection Strength and Context Attraction Principle As mentioned before, RelDC resolves references based on context attraction principle that was discussed in Section 2. We now state the principle more formally. Crucial to the principle is the notion of *connection strength* between two entities x_i and y_j (denoted $c(x_i, y_j)$ which captures how strongly x_i and y_j are connected to each other through relationships. Many different approaches can be used to measure $c(x_i, y_j)$ which will be discussed in Section 4. Given the concept of $c(x_i, y_j)$, we can restate the context attraction principle as follows:

Context Attraction Principle: Let $x_i.r_k$ be a reference and $y_1, y_2, \ldots, y_N$ be elements of its choice set $CS[x_i.r_k]$ with corresponding option weights $w_1, w_2, \ldots, w_N$ (recall that $w_1+w_2+\cdots+w_N = 1$). The CAP principle states that for all $l, j \in [1, N]$, if $c_l \geq c_j$ then $w_l \geq w_j$, where $c_l = c(x_i, y_l)$ and $c_j = c(x_i, y_j)$.

4 The RelDC approach

We now have developed all the concepts and notation needed to explain RelDC approach for reference disambiguation. Input to RelDC is the entity-relationship graph G discussed in Section 3 in which nodes correspond to entities and edges to relationships. We assume that feature-based similarity approaches have been used in constructing the graph G. The choice nodes are created only for those references that could not be disambiguated using only attribute similarity. RelDC will exploit relationships for further disambiguation and will output a resolved graph G in which each entity is fully resolved.

RelDC disambiguates references using the following four steps:

1. **Compute connection strengths**. For each reference $x_i.r_k$ compute the connection strength $c(x_i, y_j)$ for each $y_j \in CS[x_i.r_k]$. The result is a set of equations that relate $c(x_i, y_j)$ with the option weights: $c(x_i, y_j) = g_{ij}(\overline{w})$. Here, $\overline{w}$ denote the set of all option weights in the graph G.
2. **Determine equations for option weights**. Using the equations from Step 1 and the CAP, determine a set of equations that relate option weights to each other.
3. **Compute weights.** Solve the set of equations from Step 2.
4. **Resolve References.** Utilize/interpret the weights computed in Step 3 as well as attribute-based similarity to resolve references.

We now discuss the above steps in more detail in the following subsections.

4.1 Computing Connection Strength

Computation of $c(x_i, y_j)$ consists of two phases. The first phase discovers connections between x_i and y_j. The second phase computes/measures the strength in connections discovered by the first phase.

4.1.1 The connection discovery phase.

In general there can be many connections between $v[x_i]$ and $v[y_j]$ in G. Intuitively, many of those (e.g., very long ones) are not very important. To capture most important connections while still being efficient, the algorithm computes the set of all L-short simple paths $\mathcal{P}_L(x_i, y_j)$ between nodes $v[x_i]$ and $v[y_j]$ in graph G. A path is *L-short* if its length is no greater than parameter L. A path is *simple* if it does not contain duplicate nodes.

Illegal paths. Not all of the discovered paths are considered when computing $c(x_i, y_j)$ (to resolve reference $x_i.r_k$). Let $e_1, e_2, \ldots, e_N$ be the option-edges associated with the reference $x_i.r_k$. When resolving $x_i.r_k$, RelDC tries do determine the weights of these edges via connections that exist in the remainder of the graph not including those edges. To achieve this,

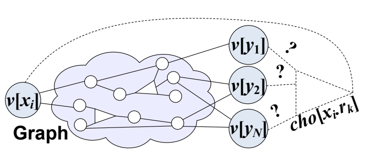


Figure 3: Graph

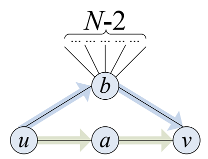


Figure 4: $c(p)$?

RelDC actually discovers paths not in graph G, but in $\widetilde{G} = G - cho[x_i.r_k]$, see Figure 3. That is, $\widetilde{G}$ is graph G with node $cho[x_i.r_k]$ removed. Also, in general, paths considered when computing $c(x_i, y_j)$ may contain option-edges of some choice nodes. If a path contains an option-edge of a choice node, it should not contain another option-edge of the same choice node. For instance, if a path used to compute connection strength between two nodes in the graph contained an option edge e_j of the choice node shown in Figure 2, it must not contain any of the rest of the option-edges $e_1, e_2, \ldots, e_{j-1}, e_{j+1}, \ldots, e_N$.

4.1.2 Computing connection strength

A natural way to compute the connection strength $c(u, v)$ between nodes u and v is to compute it as the probability to reach node v from node u via random walks in graph G where each step is done with certain probability. Such problems have been studied for graphs in the previous work under Markovian assumptions. The graph in our case is not Markovian due to presence of illegal paths (introduced by choice nodes). So those approaches cannot be applied directly. In [16] we have developed the *probabilistic model (PM)* which treats edge weights as probabilities that those edges exist and which can handle illegal paths. In this section we present the *weight-based model (WM)* which is a simplification of PM. Other models can be derived from [11, 24].

WM is a very intuitive model, which is suited well for illustrating issues related to computing $c(u, v)$. WM computes $c(u, v)$ as the sum $\sum_{p \in \mathcal{P}_L(u,v)} c(p)$ of the connection strength $c(p)$ of each path p in $\mathcal{P}_L(u, v)$. The connection strength $c(p)$ of path p from u to v is the probability to follow path p in graph G. Next we explain how WM computes $c(p)$.

Motivating $c(p)$ formula. Which factors should be taken into account when computing the connection strength $c(p)$ of each individual path p?

Figure 4 illustrates two different paths (or connections) between nodes u and v: $p_a{=}u{\rightarrow}a{\rightarrow}v$ and $p_b{=}u{\rightarrow}b{\rightarrow}v$. Assume that all edges in this figure have weight of 1. Let us understand which connection is better.

Both connections have an equal length of two. One connection is going via node a and the other one via b. The intent of Figure 4 is to show that b "connects" many things, not just u and v, whereas a "connects" only u and v. We argue the connection between u and v via b is much weaker than the connection between u and v via a: since b connects many things it is not surprising we can connect u and v via b. For example, for the author matching problem, u and v can be two authors, a can be a publication and b a university.

To capture the fact that $c(p_a) > c(p_b)$, we measure $c(p_a)$ and $c(p_b)$ as the probabilities to follow paths p_a and p_b respectively. Notice, measures such as path length, network flow do not capture this fact. We compute those probabilities as follows. For path p_b we start from u. Next we have a choice to go to a or b with probabilities of $\frac{1}{2}$, and we choose to follow (u, b) edge. From node b we can go to any of the $N-1$ nodes (cannot go back to u) but we go specifically to v. So the probability to reach v via path p_b is $\frac{1}{2(N-1)}$. For path p_a we start from u, we go to a with probability $\frac{1}{2}$ at which point we have no choice but to go to v, so the probability to follow p_a is $\frac{1}{2}$.

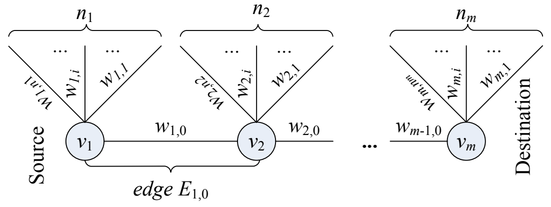


Figure 5: Computing $c(p)$ of path $p = v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_m$. Only "possible-to-follow" edges are shown.

General WM formula. In general, each L-short simple path p can be viewed as a sequence of m nodes $v_1, v_2, \ldots, v_m$, where $m \leq L + 1$, as shown in Figure 5. Figure 5 shows that from node v_i it is *possible to follow*[2] $n_i{+}1$ edges labeled $w_{i,0}, w_{i,1}, \ldots, w_{i,n_i}$. The probability to follow the edge labeled $w_{i,0}$ is proportional to weight $w_{i,0}$ and computed as $w_{i,0}/(\sum_{j=0}^{n_i} w_{i,j})$. The probability to follow path p is computed as the probability to follow each of its edges:

$$c(p) = \prod_{i=1}^{m-1} \frac{w_{i,0}}{\sum_{j=0}^{n_i} w_{i,j}}. \tag{4.1}$$

[2]It is not possible to follow zero-weight edges, and edges following which would make the path not simple.

The total connection strength between nodes u and v is computed as the sum of connection strengths of paths in $\mathcal{P}_L(u, v)$:

$$c(u, v) = \sum_{p \in \mathcal{P}_L(u,v)} c(p). \tag{4.2}$$

Measure $c(u, v)$ is the probability to reach v from u by following only L-short simple paths, such that the probability to follow an edge is proportional to the weight of the edge.

For instance, for the toy database we have:

1. $c_1 = c(P_2, \text{'Dave White'}) = c(P_2 \rightarrow \text{Susan} \rightarrow \text{MIT} \rightarrow \text{John} \rightarrow P_1 \rightarrow \text{Don} \rightarrow P_4 \rightarrow \text{Joe} \rightarrow P_5 \rightarrow \text{Liz} \rightarrow P_6 \rightarrow \text{'2'} \rightarrow \text{Dave White}) = \frac{w_3}{2}$.
2. $c_2 = c(P_2, \text{'Don White'}) = c(P_2 \rightarrow \text{Susan} \rightarrow \text{MIT} \rightarrow \text{John} \rightarrow P_1 \rightarrow \text{'Don White'}) = 1$.
3. $c_3 = c(P_6, \text{'Dave White'}) = \frac{w_1}{2}$
4. $c_4 = c(P_6, \text{'Don White'}) = 1$

4.2 Determining equations for option-edge weights

Given the connection strength measures $c(x_i, y_j)$ for each unresolved reference $x_i.r_k$ and its options y_j, we can use the context attraction principle to determine the relationships between the weights associated with the option-edges in the graph G. Note that the context attraction principle does not contain any specific strategy on how to relate weights to connection strengths. Any strategy that assigns weight such that if $c_l \geq c_j$ then $w_l \geq w_j$ is appropriate, where $c_l = c(x_i, y_l)$ and $c_j = c(x_i, y_j)$. In particular, we use the strategy where weights $w_1, w_2, \ldots, w_N$ are proportional to the corresponding connection strengths: $w_j \cdot c_l = w_l \cdot c_j$. Using this strategy weight w_j $(j = 1, 2, \ldots, N)$ is computed as:

$$w_j = \begin{cases} c_j/(\sum_{l=1}^{N} c_l) & \text{if } \sum_{l=1}^{N} c_l > 0; \\ \frac{1}{N} & \text{if } \sum_{l=1}^{N} c_l = 0. \end{cases} \tag{4.3}$$

For instance, for the toy database we have:

1. $w_1 = c_1/(c_1 + c_2) = \frac{w_3}{2}/(1 + \frac{w_3}{2})$
2. $w_2 = c_2/(c_1 + c_2) = 1/(1 + \frac{w_3}{2})$
3. $w_3 = c_3/(c_3 + c_4) = \frac{w_1}{2}/(1 + \frac{w_1}{2})$
4. $w_4 = c_4/(c_3 + c_4) = 1/(1 + \frac{w_1}{2})$

4.3 Determining all weights by solving equations.

Given a system of equations, relating option-edge weights as derived in Section 4.2, our goal next is to determine values for the option-edge weights that satisfy the equations. Before we discuss how such equations can be solved in general, let us first solve the option-edge weight equations in the toy example. These equations, given an additional constraint that all weights should be in $[0, 1]$, have a unique solution $w_1 = 0$, $w_2 = 1$, $w_3 = 0$, and $w_4 = 1$. Once we have computed the weights, RelDC will interpret these weights to resolve the references. In the toy example, weights $w_1 = 0$,

$w_2 = 1$, $w_3 = 0$, and $w_4 = 1$ will lead RelDC to resolve 'D. White' in both P_2 and P_6 to 'Don White'.

In general case, Equations (4.3), (4.1), and (4.2) define each option weight as a function of other option weights: $w_i = f_i(\overline{w})$. The exact function for w_j is determined by Equations (4.3), (4.1), and (4.2) and by the paths that exist between $v[x_i]$ and $v[y_j]$ in G. Often, in practice, $f_i(\overline{w})$ is constant leading to the equation of the form $w_i = const$.

The goal is to find such a combination of weights that "satisfies" the system of $w_i = f_i(\overline{w})$ equations along with the constraints on the weights. Since a system of equations, each of the type $w_i = f_i(\overline{w})$, might not have an exact solution, we transform the equations into the form $f_i(\overline{w}) - \delta_i \leq w_i \leq f_i(\overline{w}) + \delta_i$. Here variable δ_i, called *tolerance*, can take on any real nonnegative value. The problem transforms into solving the NLP problem where the constraints are specified by the inequalities above and the objective is to minimize the sum of all δ_i's. Additional constraints are: $0 \leq w_i \leq 1$, $\delta_i \geq 0$, for all w_i, δ_i. In [16] we argue that such a system of equations always has a solution.

The straightforward approach to solving the resulting NLP problem is to use one of the off-the-shelf math solvers, such as SNOPT. Such solvers, however, do not scale to large problem sizes that we encounter in data cleaning as will be discussed in Section 5. We therefore exploit a simple iterative approach, which is outlined below.[3] The iterative method first iterates over each reference $x_i.r_k$ and assigns weight of $\frac{1}{|CS[x_i.r_k]|}$ to each w_j. It then starts its major iterations in which it first computes $c(x_i, y_j)$ for all i and j, using Equation (4.2). Then it uses those $c(x_i, y_j)$'s to compute all w_j's using Equation (4.3). Note that the values of w_j's will change from $\frac{1}{|CS[x_i.r_k]|}$ to new values. The algorithm performs several major iterations until the weights converge (the resulting changes across iterations are negligible) or the algorithm is explicitly stopped.

Let us perform an iteration of the iterative method for the example above. First $w_1 = w_2 = \frac{1}{2}$ and $w_3 = w_4 = \frac{1}{2}$. Next $c_1 = \frac{1}{4}$, $c_2 = 1$, $c_3 = \frac{1}{4}$, and $c_4 = 1$. Finally, $w_1 = \frac{1}{5}$, $w_2 = \frac{4}{5}$, $w_3 = \frac{1}{5}$, and $w_4 = \frac{4}{5}$. If we stop the algorithm at this point and interpret w_j's, then the RelDC's answer is identical to that of the exact solution: 'D. White' is 'Don White'.

Note that the above-described iterative procedure computes only an *approximate* solution for the system whereas the solver finds the exact solution. Let us refer to iterative implementation of RelDC as *Iter-RelDC* and denote the implementation that uses a solver as *Solv-RelDC*. For both Iter-RelDC and Solv-RelDC, after the weights are computed, those weights are **interpreted** to produce the final result, as discussed in Section 4. It turned out that the accuracy of Iter-RelDC (with a small number of iterations, such as 10–20) and of Solv-RelDC is practically identical. This is because even though the iterative method does not find the exact weights, those weights are close enough to those computed using a solver. Thus, when the weights are *interpreted*, both methods obtain similar results.

[3]Methods different from Iter-RelDC can be used to compute an approximate solution as well: e.g. [16] sketches another solution which is based on computing the bounding intervals for the option weights and then applying the techniques from [9, 8, 10].

4.4 Resolving references by interpreting weights. When resolving references $x_i.r_k$ and deciding which entity among $y_1, y_2, \ldots, y_N$ from $CS[x_i.r_k]$ is $d[x_i.r_k]$, RelDC chooses such y_j that w_j is the largest among $w_1, w_2, \ldots, w_N$. Notice, to resolve $x_i.r_k$ we could have also combined w_j weights with feature-based similarities $FBS(x_i, y_j)$ (e.g., as a weighted sum), but we do not study that approach in this paper.

5 Experimental Results

In this section we experimentally study RelDC using two real (*publications* and *movies*) and synthetic datasets. RelDC was implemented using C++ and SNOPT solver [4]. The system runs on a 1.7GHz Pentium machine. We test and compare the following implementations of RelDC:

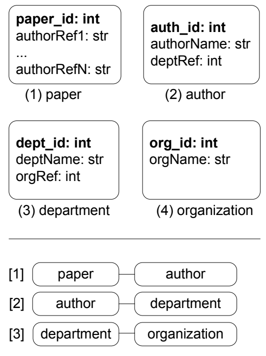
1. **Iter-RelDC** vs. **Solv-RelDC**. If neither 'Iter-' nor 'Solv-' is specified, Iter-RelDC is assumed.
2. **WM-RelDC** vs. **PM-RelDC**. The prefixes indicate whether the weight-based model (WM) from Section 4.1.2 or probabilistic model (PM) from [16], is used for computing connection strengths. By default WM-RelDC is assumed.

In each of the RelDC implementations, the value of L used in computing the L-short simple paths is set to 7 by default. In [16] we show that WM-Iter-RelDC is one of the best implementations of RelDC in terms of both accuracy and efficiency. That is why the bulk of our experiments use that implementation.

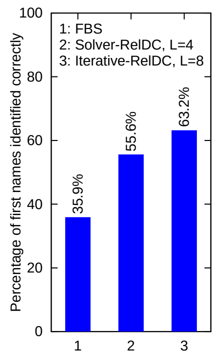
5.1 Case Study 1: the publications dataset

5.1.1 Datasets In this section, we will introduce RealPub and SynPub datasets. Our experiments will solve *author matching (AM)* problem, defined in Section 2, on these datasets.

RealPub dataset. RealPub is a real dataset constructed from *two* public-domain sources: CiteSeer[1] and HPSearch[2]. CiteSeer can be viewed as a collection of research publications, HPSearch as a collection of information about authors. HPSearch can be viewed as a set of ⟨`id`, `authorName`, `department`, `organization`⟩ tuples. That is the affiliation consists of not just organization like in Section 2, but also of department. Information stored in CiteSeer is in the same form as specified in Section 2, that is ⟨`id`, `title`, `authorRef1`, . . . , `authorRefN`⟩ per each paper. [16] contains sample content of CiteSeer and HPSearch as well as the corresponding entity-relationship graph.



(a) Types of Ent./Rel.



(b) Ident. first names

Figure 6: Experiments

The various types of entities and relationships present in RealPub are shown in Figure 6(a). RealPub consists of 4 *types* of entities: papers (255K), authors (176K), organizations (13K), and departments (25K). To avoid confusion we use "`authorRef`" for author names in *paper* entities and "`authorName`" for author names in *author* entities. There are 573K `authorRef`'s in total. Our experiments on RealPub will explore the efficacy of RelDC in resolving these references.

To test RelDC, we first constructed an entity-relationship graph G for the RealPub database. Each node in the graph corresponds to an entity of one of these types. If author A is affiliated with department D, then there is $(v[A], v[D])$ edge in the graph. If department D is a part of organization U, then there is $(v[D], v[U])$ edge. If paper P is written by author A, then there is $(v[A], v[P])$ edge. For each of the 573K `authorRef` references, feature-based similarity (FBS) was used to construct its choice set.

In the RealPub dataset, the paper entities refer to authors using **only their names (and not affiliations)**. This is because the paper entities are derived from the data available from CiteSeer, which did not directly contain information about the author's affiliation. As a result, only similarity of author names was used to initially construct the graph G.

This similarity has been used to construct choice sets for all `authorRef` references. As the result, 86.9% (498K) of all `authorRef` references had choice set of size one and the corresponding papers and authors were linked directly. For the remaining 13.1% (75K) references, 75K choice nodes were created in the graph G. RelDC was used to resolve these remaining references. The specific experiments conducted and results will be discussed later in this section. Notice that the RealPub dataset allowed us to test RelDC only under the condition that a majority of the references are already correctly resolved. To test robustness of the technique we tested RelDC over synthetic datasets where we could vary the uncertainty in the references from 0 to 100%.

SynPub dataset. We have created two synthetic datasets SynPub1 and SynPub2, which emulate RealPub. The synthetic datasets were created since, for the RealPub dataset, we do not have the true mapping between papers and the authors of those papers. Without such a mapping, as will become clear when we describe experiments, testing for accuracy of reference disambiguation algorithm requires a manual effort (and hence experiments can only validate the accuracy over small samples). In contrast, since in the synthetic datasets, the *paper-author* mapping is known in advance, accuracy of the approach can be tested over the entire dataset. Another advantage of the SynPub dataset is that by varying certain parameters we can manually control the nature of this dataset allowing for the evaluation of all aspects of RelDC under various conditions (e.g., varying level of ambiguity/uncertainty in the dataset).

Both the SynPub1 and SynPub2 datasets contain 5000 papers, 1000 authors, 25 organizations and 125 departments. The average number of choice nodes that will be created to disambiguate the `authorRef`'s is 15K (notice, the whole RealPub dataset has 75K choice nodes). The difference between SynPub1 and SynPub2 is that author names are constructed differently: SynPub1 uses unc_1 and SynPub2 uses unc_2 as will be explained shortly.

5.1.2 Accuracy experiments

In our context, the *accuracy* is the fraction of all `authorRef` references that are resolved correctly. This definition includes the references that have choice sets of cardinality 1.

Experiment 1 (RealPub: manually checking samples for accuracy). Since the correct *paper-author* mapping is not available for RealPub, it is infeasible to test the accuracy on this dataset. However, it is possible to find a portion of this *paper-author* mapping *manually* for a sample of RealPub: by going to authors web pages and examining their publications.

We have applied RelDC to RealPub in order to test the effectiveness of analyzing relationships. To analyze the accuracy of the result, we concentrated only on the 13.1% of uncertain `authorRef` references. Recall, the cardinality of the choice set of each such reference is at least two. For 8% of those references there were no $x_i \rightsquigarrow y_j$ paths for all j's, thus RelDC used only FBS and not relationships. Since we want to test the effectiveness of analyzing relationships, we remove those 8% of references from further consideration as well. We then chose a random sample of 50 papers that are still left under consideration. For this sample, we compared the reference disambiguation result produced by RelDC with the true matches. The true matches for `authorRef` references in those papers are computed manually. In this experiment, RelDC was able to resolve **all** of the 50 sample references correctly! This outcome is in reality not very surprising since in the RealPub datasets, the number of references that were ambiguous was only 13.1%. Our experiments over the synthetic datasets will show that RelDC reaches very high disambiguation accuracy when the number of uncertain references is not very high.

Ideally, we would have liked to perform further accuracy tests over RealPub by either testing on larger samples (more than 50) and/or repeating the test multiple times (in order to establish confidence levels). However, this is infeasible due to the time-consuming manual nature of this experiment. □

Experiment 2 (RealPub: accuracy of identifying author first names). We conducted another experiment over RealPub dataset to test the efficacy of RelDC in disambiguating references, which is described below.

We first remove from RealPub all the paper entities which have an `authorRef` in format "*first initial + last name*". This leaves only papers with `authorRef`'s in format "*full first name + last name*". Then we pretend we only know "*first initial + last name*" for those `authorRef`'s. Next we run FBS and RelDC and see whether or not they would disambiguate those `authorRef`'s to authors whose full first names coincide with the original full first names. In this experiment, for 82% of the `authorRef`'s the cardinality of their choice sets is 1 and there is nothing to resolve. For the rest 18% the problem is more interesting: the cardinality of their choice sets is at least 2. Figure 6(b) shows the outcome for those 18%.

Notice that the reference disambiguation problem tested in the above experiment is of a limited nature – the tasks of identifying the correct first name of the author and the correct author are not the same in general.[4] Nevertheless, the experiment allows us to test the accuracy of RelDC over the entire database and does show the strength of the approach. □

Accuracy on SynPub. The next set of experiments tests accuracy of RelDC and FBS approaches on SynPub dataset. "RelDC 100%" ("RelDC 80%") means for 100% (80%) of *author* entities the affiliation information is available. Once again, *paper* entities do not have author affiliation attributes, so FBS cannot use affiliation, see Figure 6(a). Thus, those 100% and 80% have no effect on the outcome of FBS. Notation 'L=4' means RelDC explores paths of length no greater than 4.

Experiment 3 (Accuracy on SynPub1). SynPub1 uses *uncertainty of type 1* defined as follows. There are $N_{auth} = 1000$ unique authors in SynPub1. But there are only $N_{name} \in [1, N_{auth}]$ unique `authorName`'s. We construct the `authorName` of the author with ID of k, for $k = 0, 1, \ldots, 999$, as "name" concatenated with ($k \bmod N_{name}$). Each `authorRef` specifies one of those `authorName`'s. Parameter unc_1 is $unc_1 = \frac{N_{auth}}{N_{name}}$ ratio. For instance, if N_{name} is 750, then the authors with IDs of 1 and 751 have the same `authorName`: "name1", and $unc_1 = \frac{1000}{750} = 1\frac{1}{3}$. In SynPub1 for each author whose name is not unique, one can never identify with 100% confidence any paper this author has written. Thus, the uncertainty for such authors is very high.

Figure 7 studies the effect of unc_1 on accuracy of RelDC and FBS. If $unc_1 = 1.0$, then there is no uncertainty and all methods have accuracy of 1.0. As expected, the accuracy of all methods monotonically decreases as uncertainty increases. If $unc_1 = 2.0$, the uncertainty is very large: for any given author there is exactly one another author with the identical `authorName`. For this case, any FBS have no choice but to guess one of the two authors. Therefore, the accuracy of any FBS, as shown in Figures 7, is 0.5. However, the accuracy of RelDC 100% (RelDC 80%) when $unc_1 = 2.0$ is 94%(82%). The gap between RelDC 100% and RelDC 80% curves shows that in SynPub1 RelDC relies substantially on author affiliations for the disambiguation.

Comparing the RelDC implementations. Figure 8 shows that the accuracy results of WM-Iter-RelDC, PM-Iter-RelDC, WM-Solv-RelDC implementations are comparable. □

Experiment 4 (Accuracy on SynPub2). SynPub2 uses *uncertainty of type 2*. In SynPub2, `authorName`'s (in *author* entities) are constructed such that the following holds, see Figure 6(a). If an `authorRef` reference (in a *paper* entity) is in the format "*first name + last name*" then it matches only one (correct) author. But if it is in the format "*first initial + last name*" it matches exactly two authors. Parameter unc_2 is the fraction of `authorRef`'s specified as "*first initial + last name*". If $unc_2 = 0$, then there is no uncertainty and the accuracy of all methods is 1. Also notice that the case when $unc_2 = 1.0$ is equivalent to $unc_1 = 2.0$.

[4] It is not enough to determine that 'J.' in 'J. Smith' corresponds to 'John' if there are multiple 'John Smith"s in the dataset.

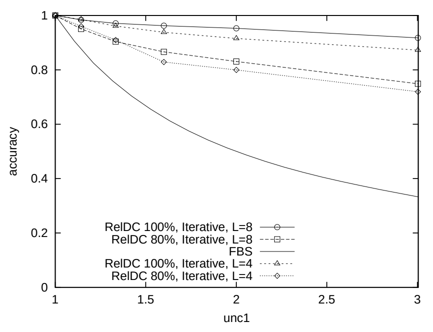


Figure 7: SynPub1: accuracy

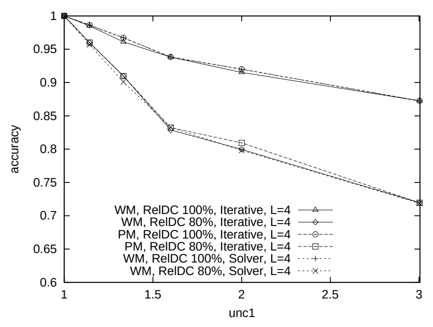


Figure 8: RelDC implementations

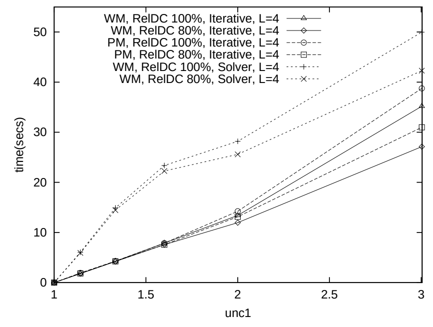


Figure 9: SynPub1: efficiency

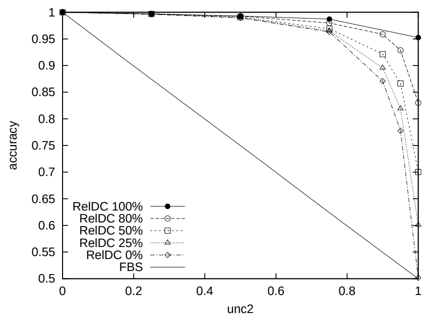


Figure 10: SynPub2: Acc. vs. unc_2

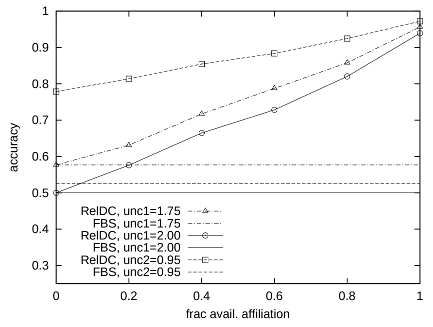


Figure 11: SynPub: affiliation

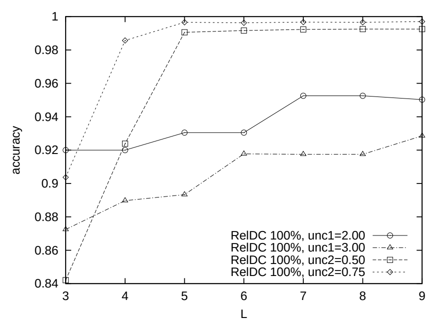


Figure 12: SynPub: Acc vs. L

There is less uncertainty in Experiment 4 then in Experiment 3. This is because for each author there is a chance that he is referenced to by his full name in some of his papers, so for these cases the *paper-author* associations are known with 100% confidence.

Figure 10 shows the effect of unc_2 on the accuracy of RelDC. As in Figure 7, in Figure 10 the accuracy decreases as uncertainty increases. However, this time the accuracy of RelDC is much higher. The fact that curves for RelDC 100% and 80% are almost indiscernible until unc_2 reaches 0.5, shows that RelDC relies less heavily on weak author affiliation relationships but rather on stronger connections via papers. □

5.1.3 Other experiments

Experiment 5 (Importance of relationships). Figure 11 studies what effect the number of relationships and the number of relationship *types* have on the accuracy of RelDC. When resolving `authorRef`'s, RelDC uses three types of relationships: (1) *paper-author*, (2) *author-department*, (3) *department-organization*.[5] The affiliation relationships (i.e., (2) and (3)) are derived from the affiliation information in *author* entities.

The affiliation information is not always available for each *author* entity in RealPub. In our synthetic datasets, we can manually vary the amount of available affiliation information. The x-axis shows the fraction ρ of *author* entities for which their affiliation is known. If $\rho = 0$, then the affiliation relationships are eliminated completely and RelDC has to rely solely on connections via *paper-author* relationships. If $\rho = 1$, then the complete knowledge of author affiliations is available. Figure 11 studies the effect of ρ on accuracy. The curves in this figure are for both SynPub1 and SynPub2: $unc_1 = 1.75$, $unc_1 = 2.00$, and $unc_2 = 0.95$. The accuracy increases as ρ increases showing that RelDC deals with newly available relationships well. □

Experiment 6 (Longer paths). Figure 12 examines the effect of path limit parameter L on the accuracy. For all the curves in the figure, the accuracy monotonically increases as L increases with the only one exception for "RelDC 100%, unc1=2" and $L = 8$. The usefulness of longer paths depends on the combination of other parameters. Typically, there is a tradeoff: larger values of L lead to higher accuracy of disambiguation but slower performance. The user running RelDC must decide the value of L based on this accuracy/performance tradeoff for the dataset being cleaned. For SynPub, $L = 7$ is a reasonable choice. □

Experiment 7 (Efficiency of RelDC). To show the applicability of RelDC to a large dataset we have successfully applied an optimized version of RelDC to clean RealPub with L ranging from 2 up to 8. Figure 13 shows the execution time of RelDC as a function of the

[5]Note, a '*type of relationship*' (e.g., *paper-author*) is different from a '*chain of relationships*' (e.g., paper1-author1-dept1-...).

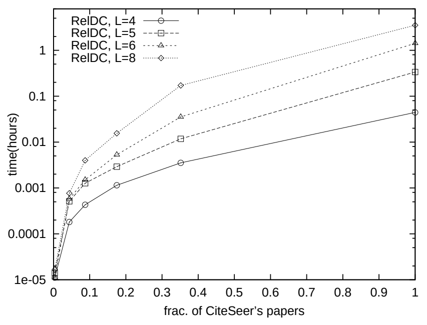


Figure 13: RealPub: efficiency

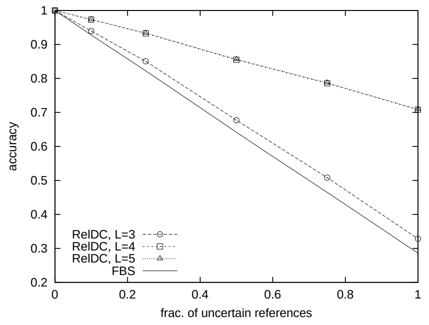


Figure 14: RealMov: *director* refs.

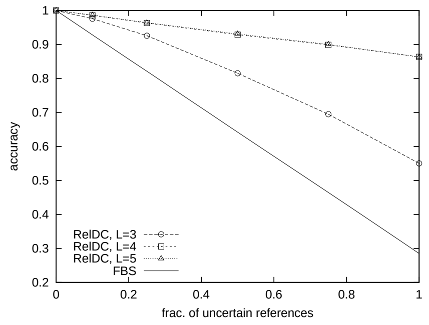


Figure 15: RealMov: *studio* refs.

fraction of papers from RealPub, e.g. 1.0 corresponds to all papers in RealPub (the whole CiteSeer) dataset. Notice, optimizations of RelDC are discussed only in [16], they are crucial to achieve 1–2 orders of magnitude of improvement in performance. □

5.2 Case Study 2: the movies dataset

5.2.1 Dataset RealMov is a real public-domain movies dataset described in [25] which has been made popular by the textbook [13]. Unlike RealPub dataset, in RealMov all the needed correct mappings are known, so it is possible to test the disambiguation accuracy of various approaches more extensively. However, RealMov dataset is much smaller, compared to the RealPub dataset. RealMov contains entities of three types: *movies* (11,453 entities), *studios* (992 entities), and *people* (22,121 entities). There are five types of relationships in the RealMov dataset: *actors*, *directors*, *producers*, *producingStudios*, and *distributingStudios*. Relationships *actors*, *directors*, and *producers* map entities of type *movies* to entities of type *people*. Relationships *producingStudios* and *distributingStudios* map *movies* to *studios*. [16] contains the sample graph for RealMov dataset as well as sample content of *people*, *movies*, *studios* and *cast* tables from which it has been derived.

5.2.2 Accuracy experiments

Experiment 8 (RealMov: Accuracy of disambiguating *director* references). In this experiment, we study the accuracy of disambiguating references from movies to directors of those movies.

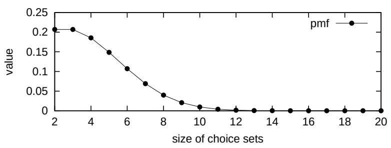


Figure 16: PMF of sizes of choice sets.

Since in RealMov each reference, including each *director* reference, already points directly to the right match, we artificially introduce ambiguity in the references manually. Similar approach to testing data cleaning algorithms have also been used by other researchers, e.g. [7]. Given the specifics of our problem, to study the accuracy of RelDC we will simulate that we used FBS to determine the choice set of each reference but FBS was uncertain in some of the cases.

To achieve that, we first choose a fraction ρ of *director* references (that will be uncertain). For each reference in this fraction we will simulate that FBS part of RelDC has done its best but still was uncertain as follows. Each *director* reference from this fraction is assigned a choice set of N people. One of those people is the true director, the rest $(N-1)$ are chosen randomly from the set of *people* entities.

Figure 14 studies the accuracy as ρ is varied from 0 to 1 and where N is distributed according to the probability mass function (pmf) shown in Figure 16, see [16] for detail. The figure shows that RelDC achieves better accuracy than FBS. The accuracy is 1.0 when $\rho = 0$, since all references are linked directly. The accuracy decreases almost linearly as ρ increases to 1. When $\rho = 1$, the cardinality of the choice set of each reference is at least 2. The larger the value of L, the better the results. The accuracy of RelDC improves significantly as L increases from 3 to 4. However, the improvement is less significant as L increases from 4 to 5. Thus the analyst must decide whether to spend more time to obtain higher accuracy with $L = 5$, or whether $L = 4$ is sufficient. □

Experiment 9 (RealMov: Accuracy of disambiguating *studio* references). This experiment is similar to Experiment 8, but now we disambiguate *producingStudio*, instead of *director*, references. Figure 15 corresponds to Figure 14. The RelDC's accuracy of disambiguating *studio* references is even higher. □

6 Related Work

Many research challenges have been explored in the context of data cleaning in the literature: dealing with missing data, handling erroneous data, record linkage, and so on. The closest to the problem of reference disambiguation addressed in this paper is the problem of record linkage. The importance of record linkage is underscored by the large number of companies, such as Trillium, Vality, FirstLogic, DataFlux, which have developed (domain-specific) record linkage solutions.

Researchers have also explored domain-independent techniques, e.g. [23, 12, 14, 5, 22]. Their work can be viewed as addressing two challenges: (1) improving similarity function, as in [6]; and (2) improving efficiency of linkage, as in [7]. Typically, two-level similarity functions are employed to compare two records. First, such a function computes attribute-level similarities by comparing values in the same attributes of two records. Next the function combines the attribute-level similarity measures to compute the overall similarity of two records. A recent trend has been to employ machine learning techniques, e.g. SVM, to learn the best similarity function for a given domain [6]. Many techniques have been proposed to address the efficiency challenge as well: e.g. using specialized indexes [7], sortings, etc.

Those domain-independent techniques deal only with attributes. To the best of our knowledge, RelDC, which was first publicly released in [15], is the first domain-independent data cleaning framework which exploits relationships for cleaning. Recently, in parallel to our work, other researchers have also proposed using relationships for cleaning. In [5] Ananthakrishna et al. employ similarity of directly linked entities, for the case of hierarchical relationships, to solve the record de-duplication challenge. In [19] Lee et al. develop an association-rules mining based method to disambiguate references using similarity of the context attributes: the proposed technique is still an FBS method, but [19] also discusses concept hierarchies which are related to relationships. Getoor et al. in DKDM04 use similarity of attributes of directly linked objects, like in [5], for the purpose of object consolidation. However, the challenge of applying that technique in practice on real-world datasets was identified as future work in that paper. In contrast to the above-described techniques, RelDC utilize the CAP principle to automatically discover and analyze relationship chains, thereby establishing a framework that employs systematic relationship analysis for data cleaning.

7 Conclusion

In this paper, we have shown that analysis of inter-object relationships is important for data cleaning and demonstrated one approach that utilizes relationships. As future work, we plan to apply similar techniques to the problem of record linkage. This paper outlines only the core of the RelDC approach, for more details the interested reader is referred to [16]. Another interesting follow-up work [18] addresses the challenge of automatically adapting RelDC to datasets at hand by learning how to weigh different connections directly from the data. Solving this challenge, in general, not only makes the approach to be a plug-and-play solution but also can improve the accuracy as well as efficiency of the approach as discussed in [18].

References

[1] CiteSeer. `http://citeseer.nj.nec.com/cs`.

[2] HomePageSearch. `http://hpsearch.uni-trier.de`.

[3] Knowledge Discovery. `http://www.kdnuggets.com/polls/2003/data_preparation.htm`.

[4] SNOPT solver. `http://www.gams.com/solvers/`.

[5] Ananthakrishna, Chaudhuri, and Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, 2002.

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

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

[8] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proc. ACM SIGMOD Conf.*, 2003.

[9] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *IEEE TKDE*, 16(9), Sept. 2004.

[10] R. Cheng, S. Prabhakar, and D. Kalashnikov. Querying imprecise data in moving object environments. In *Proc. IEEE ICDE Conf.*, 2003.

[11] C. Faloutsos, K. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *SIGKDD*, 2004.

[12] I. Fellegi and A. Sunter. A theory for record linkage. *J. of Amer. Stat. Assoc.*, 64(328):1183–1210, 1969.

[13] H. Garcia-Molina, J. Ullman, and J. Widom. *Database systems: the complete book*. Prentice Hall, 2002.

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

[15] D. Kalashnikov and Mehrotra. Exploiting relationships for data cleaning. *TR-RESCUE-03-02*, Nov. 2003.

[16] D. Kalashnikov and S. Mehrotra. Exploiting relationships for domain-independent data cleaning. *Extended Version of SIAM Data Mining 2005 publication*, `http://www.ics.uci.edu/~dvk/pub/sdm05.pdf`.

[17] D. V. Kalashnikov and S. Mehrotra. RelDC project. `http://www.ics.uci.edu/~dvk/RelDC/`.

[18] D. V. Kalashnikov and S. Mehrotra. Learning importance of relationships for reference disambiguation. *Submitted for Publication*, Dec. 2004. `http://www.ics.uci.edu/~dvk/RelDC/TR/TR-RESCUE-04-23.pdf`.

[19] M. Lee, W. Hsu, and V. Kothari. Cleaning the spurious links in data. *IEEE Intelligent Systems*, Mar-Apr 2004.

[20] R. Little and D. Rubin. *Statistical Analysis with Missing Data*. John Wiley and Sons, 1986.

[21] J. Maletic and A. Marcus. Data cleansing: Beyond integrity checking. In *Conf. on Inf. Quality*, 2000.

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

[23] Newcombe, Kennedy, Axford, and James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[24] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *SIGKDD*, 2003.

[25] G. Wiederhold. `www-db.stanford.edu/pub/movies/`.