A version of this work is accepted to ***SIAM International Conference on Data Mining (SDM) 2005***

A probabilistic model for entity disambiguation using relationships*

Dmitri V. Kalashnikov and Sharad Mehrotra

Computer Science Department
University of California, Irvine
{dvk,sharad}@ics.uci.edu

TR-RESCUE-04-12
Jun 1, 2004

Contents

*This material is based upon work supported by the National Science Foundation under Award Numbers 0331707 and 0331690. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

Abstract

Graphs representing relationships among sets of entities are of increasing focus of interest in the context of data analysis applications. These graphs are typically constructed from existing datasets from which entities and relationships are extracted. For some of the entities, values in certain attributes would refer to other entities – such references determine relationships. Often, for certain datasets such references are given in the form of (string) descriptions. Each such description alone may not uniquely identify one entity as it is supposed to, but rather can match descriptions of multiple entities. Such cases are especially common if the datasets are collected not from one but multiple heterogeneous sources. Thus the correct linking of entities via relationships can be a nontrivial challenge which, if done incorrectly, can in turn impede further graph-based analyses. To overcome this problem, standard feature-based data cleaning approaches can be employed. In this paper we argue a better solution exist which analyzes not only features but also relationships.

1 Introduction

In many application domains (biology, intelligence, commerce, marketing) data is naturally represented in the form of a graph in which nodes correspond to entities and edges to relationships among the entities. For instance, in the intelligence application a graph may represent people and organizations (entities) and the linkages between them, where a link may be a relationship such as common residence, common employer, participation in a common event, etc. Given such an entity/relationship graph, measures such as "centrality", "influence", "importance" of a (group) of nodes to the graph have been proposed to understand the evolution and structure of such graphs. For instance, in a recent paper, Smyth et al.[38] proposed the notion of relative authority where they determine how to compute the importance of nodes in a graph relative to one or more root nodes.

While research has focussed on defining graph metrics such as above and developing efficient algorithms to compute the metrics, an important question is how these graphs are constructed in the first place. Usually, knowledge about entities and the relationships among them resides in numerous documents and datasets distributed across a variety of data sources. Traditionally, analysts collect (textual) datasets/articles/reports (often written in foreign languages) and write down their subjective observations concerning entities and their relationships described in the documents. Progress in information extraction has made it possible to extract such entities and relationships automatically (at least in limited domains).

A key issue in constructing graphs from diverse sources is that of resolving the references to entities in these datasets. Consider, for instance, an article in a Golf Magazine that refers to George Bush President of USA as a member of a particular golf club. As another instance, a conference web site may refer to C. Li as an author. The question is which particular entities do the above references refer to? In the ideal world each entity would have a unique identifier and, whenever this entity is referred to, its unique identifier is specified so that there is no uncertainty. In the real world, however, entities are often referred to by descriptions that may be created by multiple persons and collected from various sources. Entities might be referred to differently (by different descriptions) and also multiple entities might end up matching the same description. For instance, a reference to President George Bush in the Golf Digest might refer to either the 41 president or his son the 43 president of USA. Similarly, C. Li on the conference web page may correspond to one of the authors: "Chen Li" or "Cheng Li" or for that matter "Chang Lee" due to a misspelling of the last name of the author on the conference web page.

We refer to the above described problem as the *entity reference resolution* or simply the *entity resolution* problem. Entity resolution is a pervasive problem in real-world data analysis in general and is of special interest in the graph based data analysis discussed earlier. The entity resolution problem has been identified and addressed in the recent literature. For instance, [30] paper has explored the entity resolution problem

in a special context of citation networks. In that paper, the authors determine whether or not two papers are the same or not using relational probability model they developed. Another body of work is the [19] where the authors refer to the problem as alias determination.

The current approaches to entity resolution explore a feature-based approach. The essential idea is to extract various properties of entities from the description that could potentially help in disambiguating. Similarity-based approach is used to determine the candidate. We refer to such approaches as *feature-based similarity* (FBS) approaches.

In this paper, we propose an approach to entity resolution that in contrast to (or rather as a complement to) the feature-based approaches attempts to resolve entities by exploiting the relationships between referring entity and the entity being referred to. Our approach, which we call Reference Disambiguation approach (RDA) determines the strength of the relationship between the two entities u and v and uses this strength to determine if a reference in the context of u is referring to v. RDA adopts a probabilistic approach to estimating the strength of the relationship. In particular, given two entities u and v in an entity/relationship graph, to determine the strength of the relationship between nodes u and v, RDA determines the probability of reaching v from u via simple paths in the graph.

The rest of the paper is developed as follows. In Section 2, we precisely formulate the problem of entity resolution and develop notation that will help explain our approach. Section 3, describes our RDA approach. The probabilistic model for computing the connection strength is presented in Section 4. The empirical results of RDA are discussed in Section 5. Section 6 contains the related work and, finally, Section 7 concludes the paper.

The main contribution of the paper is the probabilistic model for entity disambiguation.

2 Problem formulation

Notation. An undirected graph $G(V, E)$ is composed of two sets, the set of nodes V and the set of edges E. X is the set of all entities in the dataset. Node $v_i \in V$ corresponds to entity $x_i \in X$, each edge corresponds to a relationship. Notation $V(\cdot)$ as in $V(x_i)$ denotes the node corresponding to entity x_i, e.g. $V(x_i) = v_i$. The graphs in this paper are assumed to be simple, i.e. without parallel edges. Edges have weights, nodes do not have weights. Edge weights are used to reflect the degree of confidence the relationship corresponding to the edge exists. By default all weights are equal to 1. Notation "an edge label" means the same as "an edge weight" which is the same as "the probability an edge exists". A *path* is always referring to a k-short simple path, or the path that does not contain duplicate nodes and having length of no greater than k. We use $\mathcal{P}(V(x_i), V(y_j))$, or simply $\mathcal{P}(x_i, y_j)$, to denote the set of all k-short simple paths between nodes $V(x_i)$ and $V(y_j)$ in graph G.

Each entity x_i consist of a set of m_{x_i} properties $x_i.a_1, x_i.a_2, \ldots, x_i.a_{m_{x_i}}$ and contain a set of n_{x_i} references $x_i.r_1$,$x_i.r_2$,...,$x_i.r_{n_{x_i}}$, where, each reference $x_i.r_k$ is semantically referring to a single entity $d(x_i.r_k)$ in X.[1] But description provided by $x_i.r_k$ can match in general multiple entities in X. Each reference $x_i.r_k$ is essentially a description and may itself consist of one or more attributes $x_i.r_k.b_1, x_i.r_k.b_2, \ldots$.[2]

A reference $x_i.r_k$ has associated with it a non-empty set of entities $CS(x_i.r_k)$, referred to as *choice set* for the reference $x_i.r_k$. Such a 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 $x_i.r_k$.[3] 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.

[1] For instance, in the author citation graph, nodes may correspond to authors and papers. A paper node may consist of properties such as title, area, keywords, the name of the journal/conference in which the paper appeared, etc. It may also contain references to the authors who have written the paper.

[2] For instance, in the citation network, an author reference might include the name of the author as well as the organization where the author works. It could potentially include other information such as gender of the author as well.

[3] We will discuss how $CS(x_i.r_k)$ can be constructed if it is not given.

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 resolving, 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* if it is not resolved yet to any y_j and $N > 1$.

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 x_i and y_1. This edge is associated with a weight 1 to denote that the probability of the edge to exist is 1.

If $CS(x_i.r_k)$ has more than 1 elements, then graph G contains a *choice* node $Cho(x_i.r_k)$, as shown in Figure 1, to reflect the fact that $d(x_i.r_k)$ can be one of y_1,y_2,...,y_N. Node $Cho(x_i.r_k)$ is linked with node $V(x_i)$

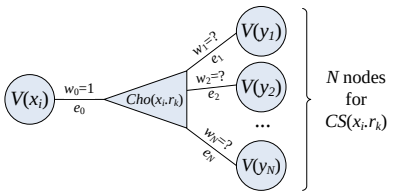

Figure 1: A choice node

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 $e_j = (Cho(x_i.r_k), V(y_j))$. Nodes $V(y_1), V(y_2), \ldots, V(y_N)$ are called the *options* of choice $Cho(x_i.r_k)$. Edges e_j between choice node $Cho(x_i.r_k)$ and its options $V(y_1), \ldots, V(y_N)$ are called the *option edges* of choice $Cho(x_i.r_k)$. The weigh of edge e_0 is one, weights of edges e_j for $j = 1, \ldots, N$ are undefined initially.

From the graph theoretic perspective, to resolve $x_i.r_k$ means to associate weights of zero with all but one edge e_j among edges e_1,e_2,...,e_N. To resolve an entity means to resolve all of its references. To resolve graph G means to resolve all entities in X (the set of all entities represented by graph G) and hence label all the option edges. Given the above notation we now can finally specify our goal in entity resolution.

The goal. Given graph $G(V, E)$, the goal is to resolve graph $G(V, E)$ as accurately as possible.

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.[4] The accuracy is defined as the fraction of references resolved correctly.

An alternative goal can be for each $y_j \in CS(x_i.r_k)$ to associate the probability that y_j is $d(x_i.r_k)$. For that goal, $Resolve(x_i.r_k)$ should associate with each $y_j \in CS(x_i.r_k)$ (and label each edge e_j with) such a probability. Those probabilities can be *interpreted* later on to achieve the first goal: for each $x_i.r_k$ try to identify $d(x_i.r_k)$ correctly. We emphases this alternative goal because the bulk of discussion of RDA approach is devoted to one approach of computing those probabilities. An interpretation of those probabilities (in order to try to identify $d(x_i.r_k)$) is a small final step for RDA as will be clear later.

3 The RDA approach

In this section we first describe relationship-based similarity, then our generic approach that utilizes both feature- and relationship-based similarity for the purpose of entity disambiguation. One implementation of this generic approach, called the Probabilistic Model (PM), is described in Section 4.

[4]The efficiency of such a procedure is not the focus of this paper.

3.1 Current feature-based approaches

There are no approaches per se directly formulated for our problem. However, existing techniques for solving related problems can be applied. Most of the them are either domain specific or domain-independent approaches which utilize *feature-based similarity (FBS)*. Since we are interested in domain independent solutions, we will concentrate on the FBS approaches.

To resolve $x_i.r_k$, FBS approaches would utilize a feature-based similarity function $FBS(x_i.r_k, y_j)$ for all $y_j \in CS(x_i.r_k)$. To simplify the notation, we will write $FBS(x_i, y_j)$ instead of $FBS(x_i.r_k, y_j)$. This function would compare properties (i.e., values in the attribute or *features*) of x_i and y_j and return the degree of similarity between $x_i.r_k$ and y_j. Note, FBS, in general, can utilize *all* attributes of x_i and y_j to resolve a *single* reference $x_i.r_k$. An FBS measure can be viewed as a distance between $x_i.r_k$ and y_j. $Resolve(x_i.r_k)$ can be computed as $Resolve(x_i.r_k) = y_j \in CS(x_i.r_k) : FBS(x_i, y_j) = \min_{y_k \in CS(x_i.r_k)} FBS(x_i, y_k)$.

Consider, for example, a reference to an author on a conference webpage, which contains author name and author affiliation. When trying to match it to one of the authors in the graph (assuming author entities have *author name* and *affiliation* attributes), FBS can use string edit distance to compare author names and normalized string edit distance to compare their affiliation, then combine those two and then output this result as its measure of similarity.

3.2 Relationship-based similarity

Similarly to FBS, it is possible to define *relationship-based similarity (RBS)*, which analyzes relationships instead of features. FBS is based on hypothesis that that if $x_i.r_k$ is referring to y_j then, there is a similarity between the description $x_i.r_k$ and y_j. RBS is based on the hypothesis that if $x_i.r_k$ is referring to y_j then, there is a high likelihood that x_i and y_j are strongly connected to each other in graph G via (chains of) relationships. $RBS(x_i, y_i)$ discovers the set of all k-short simple paths $\mathcal{P}(V(x_i), V(y_j))$ between nodes $V(x_i)$ and $V(y_j)$ in graph G. For efficiency, RBS searches only for paths of length no greater than parameter L. The set of these paths comprises a subgraph G_{ij} of graph G. Based on G_{ij} and based on how G_{ij} is connected to G, $RBS(x_i, y_j)$ outputs the degree of similarity, or what we call the *connection strength* $c(x_i, y_j)$, between x_i and y_j.

3.3 Outline of RDA

Utilizing RBS in RDA. When resolving $x_i.r_k$ for RDA the main goal is to assign weights $w_1,\ldots,w_N$ to the corresponding N edges $e_1,\ldots,e_N$. The following is an outline of the procedure that RDA utilizes for that purpose:

```
RESOLVE-REFERENCE(G, x_i.r_k)
1  PRUNE-WITH-FBS(x_i, CS(x_i.r_k))
2  if |CS(x_i.r_k)| = 1 return y_1
3  for j ← 1 to |CS(x_i.r_k)| do
4       c[j] ←CONNECTION-STRENGTH(x_i, y_j)
5  for j ← 1 to |CS(x_i.r_k)| do
6       w[j] ←ASSIGN-WEIGHT(c, j)
```

Recall, for each $x_i.r_k$ we know the set of possible matches $CS(x_i.r_k)$.[5] If $x_i.r_k$ needs to be resolved, then N must be greater than 1, where $N = |CS(x_i.r_k)|$. In general, RDA first prunes $CS(x_i.r_k)$ using an existing

[5]The best approach which we are aware of to determine such $CS(x_i.r_k)$ in the first place, is to use the best existing FBS approach with some threshold value to pick all possible candidates for $x_i.r_k$. Also domain-dependent techniques can be applied to compute that set.

FBS approach to delete from $CS(x_i.r_k)$ elements that are too dissimilar to $x_i.r_k$ based on their features. We assume this pruning is such that resulting $CS(x_i.r_k)$ has at least on element. This pruning can leave just one element y_1 in $CS(x_i.r_k)$ in which case RDA does not use RBS, but simply reports y_1 as its outcome of $Resolve(x_i.a_k)$ (labeling the original edges accordingly). The pruning step can be completely omitted, depending on the quality of choice sets provided for references $x_i.r_k$'s for a particular problem at hand.

PRUNE-WITH-FBS$(x_i, CS(x_i.r_k))$
1 **for** $j \leftarrow 1$ to $|CS(x_i.r_k)|$ **do**
2 **if** $FBS(x_i, y_j) > \lambda$ **do**
3 $CS(x_i.r_k) = CS(x_i.r_k) - y_j$
4 $N \leftarrow |CS(x_i.r_k)|$
5 rename all $y_j \in CS(x_i.r_k)$ to $y_1,\ldots,y_N$

Then RDA measures N connections strength $c_1,\ldots,c_N$ for N pairs (x_i, y_1),(x_i, y_2),..., (x_i, y_N). Based on those connection strengths RDA assigns weights $w_1,\ldots,w_N$ to edges.

Interpreting weights. Eventually, the very final step will be to interpret those weights, i.e. to choose only one of $y_1,\ldots,y_N$. This is achieved by simply choosing $y_j : w_j = \max_{l=1,\ldots,N} w_l$.

Weight assignment strategies. Recall that weights of edges correspond to the probabilities of those edges to exist. Since $x_i.r_k$ can be resolved to only one of $y_1,\ldots,y_N$ from $CS(x_i.r_k)$, the sum of edge weights $w_1,\ldots,w_N$ should add up to 1. The exact weight assignment strategy to edges $e_1,\ldots,e_N$ depends on a particular implementation of RDA. Any strategy that assigns weight such that if $c_l \geq c_j$ then $w_l \geq w_j$ is appropriate, where c_l and c_j are referring to the connection strength $c(x_i, y_l)$ and $c(x_i, y_j)$. In particular, we will assume the following strategy where, weights $w_1,\ldots,w_N$ should be proportional to their connection strength: $w_j \cdot c_l = w_l \cdot c_j$. Using this strategy weight w_j for $j = 1,\ldots,N$ can be computed as follows: if $\sum_{l=1,\ldots,N} c_l = 0$, then $w_j = \frac{1}{N}$, else $w_j = c_j / \sum_{l=1,\ldots,N} c_l$.

Computing connection strength. We use the *probabilistic model (PM)* for computing the connection strength between two nodes. This model is discussed in Section 4.

4 Probabilistic model for computing connection strength

In this section we discuss the probabilistic model for computing RBS measure between two nodes u and v. We call the RBS measure between nodes u and v the connection strength between u and v. The connection strength between u and v in the probabilistic model we use in this paper is computed as the probability to reach v from u in graph G via k-short simple paths between v and u. Section 3.2 has provided a general description for computing RBS measure. The outline of the procedure for computing connection strength is given below.

CONNECTION-STRENGTH(G, u, v, k)
1 $c \leftarrow 0$
2 $\mathcal{P}(u, v) \leftarrow$ALL-$K$-SHORT-SIMPLE-PATHS$(G, u, v, k)$
3 **for** each path $p \in \mathcal{P}(u, v)$
4 $c \leftarrow c+$PATH-CONNECTION-STRENGTH(G, p)
5 **return** c

When computing the connection strength between nodes u and v, all k-short simple paths $\mathcal{P}(u, v)$ between u and v in graph G are discovered. We use (a highly optimized version of) a depth first all k-short simple path algorithm. The optimizations are beyond the scope of the paper, some (more than 5) of those

are explained in [22].[6]

The rest of this section provides the probabilistic model for computing *connection strength* (i.e., RBS measure) $c(u, v)$ between u and v, given $\mathcal{P}(u, v)$. Namely, most of the discussion is dedicated to computing the connection strength of a single path. The section concludes with explanation how the total connection strength is computed.

To summarize the approach we are going to use: we compute $c(u, v)$ as the sum of the connection strength of each path p in $\mathcal{P}(u, v)$. In the *probabilistic model (PM)* we use in this paper, the connection strength $c(p)$ of path $p : u \overset{p}{\rightsquigarrow} v$ is the probability $\mathrm{P}_{\rightarrow}(p)$ to follow path p in graph G. In the rest of this section we will explain how we compute such probability. Please refer to Table 1 for notation.

Notation	Meaning
$\exists x$	event "x exists" for some entity x
$\nexists x$	event "x does not exist" for entity x
$\mathrm{P}_{\exists}(x)$	probability that x exists
$\mathrm{P}_{\rightarrow}(E)$	probability to follow (edge) E
$\mathcal{P}$	the path being considered
v_i	a node on the path
E_i	edge (v_i, v_{i+1}) on the path
$E_{i,j}$	edge labeled with probability $p_{i,j}$
$a_{i,j}$	if $\exists E_{i,j}$ then $a_{i,j} = 1$ else $a_{i,j} = 0$
$a_{i,0} = 1$	a dummy variable which is always 1
$p_{i,0} = 1$	a dummy variable which is always 1
$\mathbf{a}$, as a vector	$\mathbf{a} = (a_{1,0}, a_{1,1}, \ldots, a_{k,n_k})$
$\mathbf{a}$, as a set	$\mathbf{a} = \{a_{i,j} : i = 1, \ldots, k, j = 1, \ldots, n_i\}$
$\mathbf{a}$, as a variable	at each moment variable $\mathbf{a}$ is one instantiation of $\mathbf{a}$ as a vector

Table 1: Probabilistic model: Terminology

4.1 Preliminaries

Formula motivation. When computing connection strength $c(u, v)$ between nodes u and v, the set of all k-short simple paths $\mathcal{P}(u, v)$ of type $u \rightsquigarrow v$ in graph G is discovered. In Section 4.6 we will show that for the PM's formulae, discussed in the next sections, $c(u, v)$ can be computed as the connection strength of

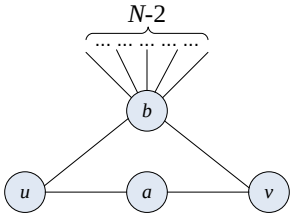

Figure 2: How to compute a path connection strength?

each individual path $p \in \mathcal{P}(u, v)$: $c(u, v) = \sum_{p \in \mathcal{P}(u,v)} c(p)$. But which factors should be taken into account when computing the connection strength of each individual path?

Figure 2 illustrates two different paths (or connections) between nodes u and v: p_1=u−b−v and p_2=u −a−v. Let us understand which connection is better, and why. Both connections have an equal length of two. One connection is going via node a, the other one via b. The intent of Figure 2 was to show that b can "connect" many things, not only u and v, whereas a is only connected to u and v. We argue since it is possible to connect u via b to many things, not only v this connection is much weaker than connection of u to v via a.[7] This weakness is captured by the probabilities to follow p_1 and p_2 which we compute as

[6] We expect to improve it even further by utilizing heuristic that, during the discovering process, would ignore certain paths which can have only marginal connection strength.

[7] For example if u and v are two authors, a is a node for a paper they co-authored and b is a node representing "Researchers", then it is not surprising it is possible to connect any two authors via "Researchers" node and such a connection should definitely have a lesser strength than a connection via the paper node.

follows. Assume all edges in Figure 2 exist with probability 1. For path p_1 we start from u and follow $u-b$ edge. From node b we could have jumped to any of the $N-1$ nodes(cannot go back to u) but we have chosen specifically v. So the probability to reach v via path p_2, for the graph in Figure 2, is $\frac{1}{N-1}$. For path p_2 we go to a at which point we have no choice but to go to v, so the probability is 1.[8]

Notation. Before we proceed let us introduce the notation used in this section, see Table 1. The meaning of this notation will become clear in the following discussion. It can be useful to refer back to Table 1 if terminology is not clear in the subsequent sections. We use $\exists x$ to denote event "x exists" for some entity (e.g., an edge) x. Similarly, we use $\nexists x$ for event "x does not exist". Notation $\mathrm{P}_{\exists}(x)$ refers to the probability that x exists. Let $\mathrm{P}_{\rightarrow}(E)$ denote the probability to follow (edge) E, usually in the context of a specific path. Notation $\mathcal{P}$ denotes the path being considered. The rest of the notations are given in the context of Figure 5. We will use v_i to refer to i^{th} node on the path. Let E_i be edge (v_i, v_i+1) on the path. Let $E_{i,j}$ denote edge labeled with probability $p_{i,j}$ in Figure 5, notice $\mathrm{P}_{\exists}(E_{i,j}) = p_{i,j}$. Notation $a_{i,j}$ is defined as follows: if $\exists E_{i,j}$ then $a_{i,j} = 1$ else $a_{i,j} = 0$, notice $\mathrm{P}(a_{i,j} = 1) = p_{i,j}$.

Introductory examples. Let us introduce PM by analyzing two examples shown in Figures 3 and 4.

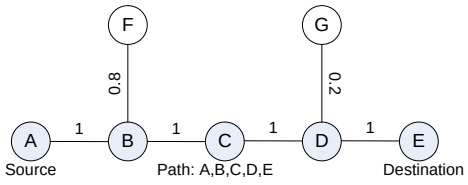

Figure 3: Toy example: independent case

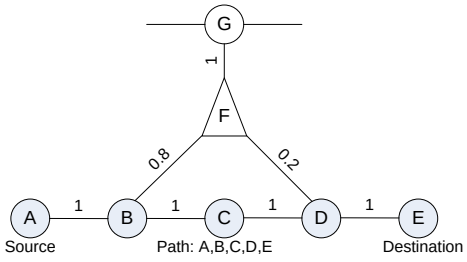

Figure 4: Toy example: dependent case

Let us consider how to compute the connection strength when the edge weight are treated as probabilities that those edges exist. Each figure show a part of a small sample graph with path $\mathcal{P}$=$A\rightarrow B\rightarrow C\rightarrow D\rightarrow E$ which will be of interest to us.

In Figure 3 we assume the events "edge BF is present" and "edge DF is present" are *independent*. The probability of the event "edge BF is present" is 0.8. The probability of the event "edge DF is present" is 0.2.

In Figure 4 node F represents a choice created to resolve a reference $x_i.r_k$ of entity x_i represented by node G. Nodes B and D are options of choice F. That is, semantically, $x_i.r_k$ can correspond to only one: either B with probability 0.8 or D with probability of 0.2. Events "edge BF exists" and "edge DF exists" are mutually exclusive (and hence strongly *dependent*): if one edge is present the other one must be absent due to the semantics of the choice node.

PM computes the connection strength of a path as the probability to reach the destination from the source by following this path. In PM computing connection strength becomes a two step process. First of all, path $\mathcal{P}$ should exist in the first place, which means each of its edges should exist. Thus the first step is to compute the probability $\mathrm{P}_{\exists}(\mathcal{P})$ that path $\mathcal{P}$ exists. For the path $\mathcal{P}$ in Figures 3 and 4, probability $\mathrm{P}_{\exists}(\mathcal{P})$ is equal to $\mathrm{P}(\exists AB \cap \exists BC \cap \exists CD \cap \exists DE)$. If the existence of each edge in the path is independent from the existence of other edges, e.g. like for the cases shown in Figure 3 and 4, then $\mathrm{P}(\exists AB \cap \exists BC \cap \exists CD \cap \exists DE) = \mathrm{P}_{\exists}(AB) \cdot \mathrm{P}_{\exists}(BC) \cdot \mathrm{P}_{\exists}(CD) \cdot \mathrm{P}_{\exists}(DE)$. Since all of the edges of path $\mathcal{P}$ are labeled with 1's in both figures, the probability that $\mathcal{P}$ exists is 1. Now the second step is to consider the

[8]Notice, another way to compute the probability to follow a path, is to apply the logic above to the source node as well. That is, with probability of $\frac{1}{2}$, and not 1 as above, the algorithm will jump to b and with probability $\frac{1}{2}$ to a, then the next step is identical to the previous method for computing the probabilities. In this formula the probability to follow p_1 is $\frac{1}{2(N-1)}$, and p_2 is $\frac{1}{2}$. Section 4.6 explains why the first formula is more preferable.

probability $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P})$ to follow path $\mathcal{P}$, given that $\mathcal{P}$ exists. Once this probability is computed, it is easy to compute our goal – the probability to follow path $\mathcal{P}$, which we use as our measure of the connection strength of path $\mathcal{P}$: $\mathrm{P}_{\rightarrow}(\mathcal{P}) = \mathrm{P}_{\exists}(\mathcal{P}) \cdot \mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P})$. The probability $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P})$ is computed differently for the cases in Figures 3 and 4. This will lead to different values for the connection strength of $\mathcal{P}$.

Case 1: The existence of each edge is independent from the existence of the other edges. In Figure 3 two events "BF exists" and "DG exists" are independent. The probability to follow path $\mathcal{P}$ is the product of probabilities to follow each of the edges on the path: $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P}) = \mathrm{P}_{\rightarrow}(AB|\exists\mathcal{P}) \cdot \mathrm{P}_{\rightarrow}(BC|\exists\mathcal{P}) \cdot \mathrm{P}_{\rightarrow}(CD|\exists\mathcal{P}) \cdot \mathrm{P}_{\rightarrow}(DE|\exists\mathcal{P})$. Given path $\mathcal{P}$ exists, the probability to follow edge AB in path $\mathcal{P}$ is one. The probability to follow edge BC is computed as follows. With probability 0.2 edge BF is absent. Then the probability to follow BC is 1. With probability 0.8 edge BF is present. Then the probability to follow BC is $\frac{1}{2}$ – because there are two links, BF and BC, that can be followed. Thus the total probability to follow BC is $0.2 \cdot 1 + 0.8 \cdot \frac{1}{2} = 0.6$. Similarly, the probability to follow CD is 1 and the probability to follow DE is $0.8 \cdot 1 + 0.2 \cdot \frac{1}{2} = 0.9$. The probability to follow path $\mathcal{P}$, given it exists, is the product of probabilities to follow each edge of the path which is equal to $1 \cdot 0.6 \cdot 1 \cdot 0.9 = 0.54$. Since for the case shown in Figure 3 path $\mathcal{P}$ exists with probability 1, the final probability to follow $\mathcal{P}$ is 0.54.

Case 2: The existence of an edge can dependent on the existence of the other edges. For the case shown in Figure 4 edges BF and DF cannot exist both at the same time. To compute $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P})$ we will consider two cases separately: $\exists BF$ and $\nexists BF$ so that we will be able to compute $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P})$ as $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P}) = \mathrm{P}_{\exists}(BF|\exists\mathcal{P}) \cdot \mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P} \cap \exists BF) + \mathrm{P}_{\nexists}(BF|\exists\mathcal{P}) \cdot \mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P} \cap \nexists BF)$.

Let us first assume $\exists BF$ (i.e., edge BF is present) and then compute $\mathrm{P}_{\exists}(BF|\exists\mathcal{P}) \cdot \mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P} \cap \exists BF)$. For the case of Figure 4, if no assumptions about the presence or absence of DF have been made yet, $\mathrm{P}_{\exists}(BF|\exists\mathcal{P})$ is simply equal to $\mathrm{P}_{\exists}(BF)$ which is equal to 0.8. If BF is present then DF is absent and the probability to follow $\mathcal{P}$ is $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P} \cap \exists BF) = 1 \cdot \frac{1}{2} \cdot 1 \cdot 1 = \frac{1}{2}$. Now let us consider the second case $\nexists BF$ (and thus $\exists DF$). The probability $\mathrm{P}_{\nexists}(BF|\exists\mathcal{P})$ is 0.2. For that case $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P} \cap \nexists BF)$ is equal to $1 \cdot 1 \cdot 1 \cdot \frac{1}{2} = \frac{1}{2}$. Thus $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P}) = 0.8 \cdot \frac{1}{2} + 0.2 \cdot \frac{1}{2} = 0.5$.

4.2 Independent edge existence

Let us now see how to compute path connection strength assuming the existence of each edge is independent from existence of the other edges in general case.

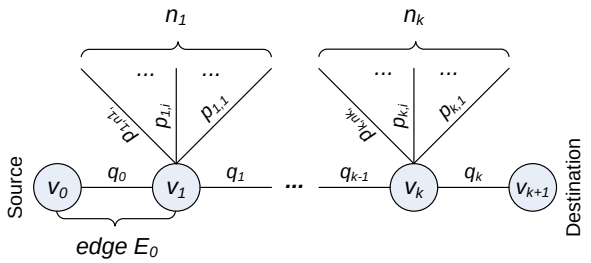

Figure 5: Independent edge existence

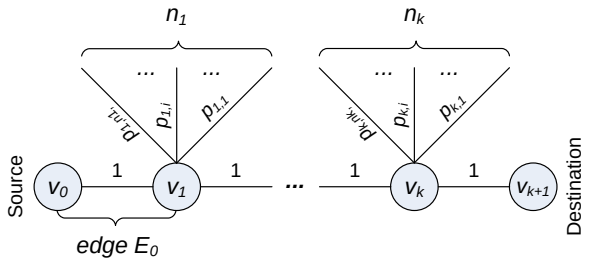

Figure 6: Assuming $\exists\mathcal{P}$.

Any non-trivial[9] path in general can be represented as shown in Figure 5. Path $\mathcal{P}$ can be viewed as a sequence of nodes $v_0, \ldots, v_{k+1}$ or as a sequence of edges $E_0, \ldots, E_k$, where $E_i = (v_i, v_{i+1})$, $\mathrm{P}_{\exists}(E_i) = q_i$.

The goal is to compute the probability to follow path $\mathcal{P}$, which is the measure of the connection strength of path $\mathcal{P}$:

$$\mathrm{P}_{\rightarrow}(\mathcal{P}) = \mathrm{P}_{\exists}(\mathcal{P}) \cdot \mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P}) \tag{1}$$

[9]A trivial path is a path of length one, i.e. the source and destination are connected via a single edge labeled with probability p. The connection strength of such a path, computed as the probability to reach the destination node from the source node via the path, is p. A path containing at least one intermediate node (i.e. of length at least 2) is called non-trivial.

The probability that $\mathcal{P}$ exists is equivalent to the probability that each of its edges exists:

$$\mathrm{P}_{\exists}(\mathcal{P}) = \mathrm{P}(\bigcap_{i=0}^{k} \exists E_i) \tag{2}$$

Given our assumption of the independence, $\mathrm{P}_{\exists}(\mathcal{P})$ can be computed as:

$$\mathrm{P}_{\exists}(\mathcal{P}) = \prod_{i=0}^{k} \mathrm{P}_{\exists}(E_i) = \prod_{i=0}^{k} q_i \tag{3}$$

Now we need to compute $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P})$. Since we assume $\exists\mathcal{P}$, we assume each edge of $\mathcal{P}$ exists and thus we are now interested in the case shown in Figure 6 (where $q_i = 1$, for $i = 1, 2, \ldots, k$).

Let us define a variable $a_{i,j} \in \{0, 1\}$ for each edge $E_{i,j}$ (labeled $p_{i,j}$) as follows: if $\exists E_{i,j}$, then $a_{i,j} = 1$, else $a_{i,j} = 0$, see Table 1. Also we define dummy variables $a_{i,0} = 1$ and $p_{i,0} = 1$ which are always 1 for $i = 0, \ldots, k$. These variables are very convenient for notation, intuitively (1) $a_{i,0} = 1$ corresponds to the fact that edge E_i exists if $\exists\mathcal{P}$; and (2) $a_{p,0} = 1$ corresponds to $\mathrm{P}_{\exists}(E_i|\exists\mathcal{P}) = 1$.

Let $\mathbf{a}$ denote the vector of all $a_{i,j}$ under consideration for path $\mathcal{P}$: $\mathbf{a} = (a_{1,0}, a_{1,1}, \ldots, a_{k,n_k})$. Let $\mathcal{A}$ denote all possible instantiations of $\mathbf{a}$, i.e. $|\mathcal{A}| = 2^{n_1+\cdots+n_k}$.

Probability $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P})$ can be computed as

$$\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P}) = \sum_{\mathbf{a}\in\mathcal{A}} \{\mathrm{P}_{\rightarrow}(\mathcal{P}|\mathbf{a} \cap \exists\mathcal{P}) \cdot \mathrm{P}(\mathbf{a}|\exists\mathcal{P})\} \tag{4}$$

where $\mathrm{P}(\mathbf{a}|\exists\mathcal{P})$ is the probability of instantiation $\mathbf{a}$ to occur while assuming $\exists\mathcal{P}$. Given our assumption of independence of probabilities, $\mathrm{P}(\mathbf{a}|\exists\mathcal{P}) = \mathrm{P}(\mathbf{a})$. Probability $\mathrm{P}(\mathbf{a})$ can be computed as

$$\mathrm{P}(\mathbf{a}|\exists\mathcal{P}) = \mathrm{P}(\mathbf{a}) = \left[\prod_{i=1,\ldots,k \text{ and } j=0,\ldots,n_i} p_{i,j}^{a_{i,j}} \cdot (1 - p_{i,j})^{1-a_{i,j}} \right]. \tag{5}$$

Probability $\mathrm{P}_{\rightarrow}(\mathcal{P}|\mathbf{a} \cap \exists\mathcal{P})$, which is the probability to go via $\mathcal{P}$ given a particular instantiation of $\mathbf{a}$ and $\mathcal{P}$ exists, can be computed as

$$\mathrm{P}_{\rightarrow}(\mathcal{P}|\mathbf{a} \cap \exists\mathcal{P}) = \left[\prod_{i=1}^{k} \frac{1}{1 + \sum_{j=1}^{n_i} a_{i,j}} \right] \equiv \left[\prod_{i=1}^{k} \frac{1}{\sum_{j=0}^{n_i} a_{i,j}} \right]. \tag{6}$$

Thus

$$\mathrm{P}_{\rightarrow}(\mathcal{P}) = \left(\prod_{i=0}^{k} q_i \right) \cdot \left(\sum_{\mathbf{a}\in\mathcal{A}} \left\{ \left[\prod_{i=1}^{k} \frac{1}{\sum_{j=0}^{n_i} a_{i,j}} \right] \cdot \left[\prod_{i,j} p_{i,j}^{a_{i,j}} \cdot (1 - p_{i,j})^{1-a_{i,j}} \right] \right\} \right) \tag{7}$$

Computing path connection strength in practice. Notice, Equation 7 iterates through all possible instantiations of $\mathbf{a}$ which is impossible to compute in practice given $|\mathcal{A}| = 2^{n_1+\cdots+n_k}$. This equation must be simplified to make the computation feasible.

To achieve the simplification, we will use our assumption of independence of probabilities which allows us to compute $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P})$ as $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P}) = \prod_{i=0}^{k} \mathrm{P}_{\rightarrow}(E_i|\exists\mathcal{P})$. In our model $\mathrm{P}_{\rightarrow}(E_0|\exists\mathcal{P})$ is always one, thus we need to specify how to compute $\mathrm{P}_{\rightarrow}(E_i|\exists\mathcal{P})$ for i greater than zero.

Let $\mathbf{a_i}$ denote vector $(a_{i,0}, a_{i,1}, \ldots, a_{i,n_i})$, i.e. $\mathbf{a} = (\mathbf{a_1}, \ldots, \mathbf{a_k})$. Let $\mathcal{A}_i$ denote all possible instantiations of $\mathbf{a_i}$, i.e. $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_k$ and $|\mathcal{A}_i| = 2^{n_i}$. Then

$$\mathrm{P}_{\rightarrow}(E_i|\exists\mathcal{P}) = \sum_{\mathbf{a_i}\in\mathcal{A}_i} \left\{ \left[\frac{1}{\sum_{j=0}^{n_i} a_{i,j}} \right] \cdot \left[\prod_{j=0}^{n_i} p_{i,j}^{a_{i,j}} \cdot (1 - p_{i,j})^{1-a_{i,j}} \right] \right\} \tag{8}$$

and $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P}) = \prod_{i=1}^{k} \mathrm{P}_{\rightarrow}(E_i|\exists\mathcal{P})$. Thus

$$\mathrm{P}_{\rightarrow}(\mathcal{P}) = \left(\prod_{i=0}^{k} q_i\right) \cdot \prod_{i=1}^{k} \left(\sum_{\mathbf{a_i} \in \mathcal{A}_i} \left\{ \left[\frac{1}{\sum_{j=0}^{n_i} a_{i,j}} \right] \cdot \left[\prod_{j=0}^{n_i} p_{i,j}^{a_{i,j}} \cdot (1-p_{i,j})^{1-a_{i,j}} \right] \right\} \right) \tag{9}$$

The effect of transformation. Notice, using Equation 7 the algorithm will need to perform $|\mathcal{A}| = 2^{n_1+\cdots+n_k}$ iterations – one per each instantiation of $\mathbf{a}$. Using Equation 7 the algorithm will need to perform $|\mathcal{A}_1| + \cdots + |\mathcal{A}_k| = 2^{n_1} + \cdots + 2^{n_k}$ iterations. Further each iteration requires less computation. These factors lead to a significant improvement.

Handling weight-1 edges. The formula in Equation 8 assumes 2^{n_i} iterations will be needed to

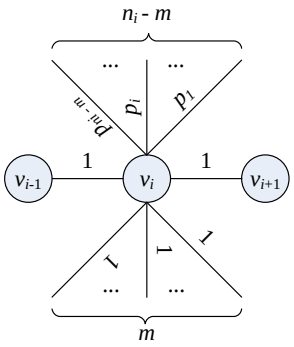

Figure 7: Probability to follow edge $E_i = (v_i, v_{i+1})$

compute $\mathrm{P}_{\rightarrow}(E_i|\exists\mathcal{P})$, thus the cost can still be quite high. This formula can be modified further to achieve even more efficient computation as follows. In practice, often some of the $p_{i,j}$'s, or even all of them, are 1's. Figure 7 shows the case where m ($m \in [0, n_i]$) edges incident to node v_i are labeled with 1. Let $\mathbf{a}'_\mathbf{i}$ denote vector $(a_{i,0}, a_{i,1}, \ldots, a_{i,n_i-m})$ and let $\mathcal{A}'_i$ be all possible instantiations of this vector. Then Equation 8 simplifies to:

$$\mathrm{P}_{\rightarrow}(E_i|\exists\mathcal{P}) = \sum_{\mathbf{a}'_\mathbf{i} \in \mathcal{A}'_i} \left\{ \left[\frac{1}{m + \sum_{j=0}^{n_i-m} a_{i,j}} \right] \cdot \left[\prod_{j=0}^{n_i-m} p_{i,j}^{a_{i,j}} \cdot (1-p_{i,j})^{1-a_{i,j}} \right] \right\} \tag{10}$$

The number of iteration is reduced from 2^{n_i} to 2^{n_i-m}.

Poisson trials. Performing 2^{n_i-m} iterations can still be expensive for the cases when $n_i - m$ is large. In this subsection we will present several solutions to deal with this issue. We will conclude our discussion with the method we have actually used to compute $\mathrm{P}_{\rightarrow}(E_i|\exists\mathcal{P})$ when 2^{n_i-m} is large in our implementation of PM.

Method 1: Do not simplify further. Even though 2^{n_i-m} cost can be expensive, it is also true that for a particular situation either (a)2^{n_i-m} is never expensive or (b) 2^{n_i-m} can be large but bearable and the cases when it is large are infrequent. In those cases further simplification might not be required.[10]

Method 2: Estimate answer using facts from Poisson trials theory. Let us denote the following sum as s_i: $s_i = \sum_{j=1}^{n_i} a_{i,j}$. The binomial distribution gives the number of successes in n independent trials where each trial is successful with the same probability p. The binomial distribution can be viewed as a sum of several *i.i.d.* Bernoulli trials. The *Poisson trails* process is similar to the binomial distribution process

[10] In our experiments cases when 2^{n_i-m} was large were infrequent, but unfortunately in those case the cost of performing 2^{n_i-m} iterations was not acceptable.

where trials are still independent but not necessarily identically distributed, i.e. the probability of success in i^{th} trial is p_i.

We can modify Equation 9 to compute $\text{P}_{\rightarrow}(E_i|\exists\mathcal{P})$ as follows:

$$\text{P}_{\rightarrow}(E_i|\exists\mathcal{P}) = \sum_{j=0}^{k} \frac{1}{1+j} \cdot \text{P}(s_i = j) \tag{11}$$

Notice, for given i we can treat $a_{i,1}, a_{i,2}, \ldots, a_{i,n_i}$ as a sequence of n_i Poisson trials with probabilities of success $\text{P}(a_{i,j} = 1) = p_{i,j}$. One would want to *estimate* $\text{P}(s_i = j)$ *quickly*, rather than compute it exactly via iterations over the corresponding cases, i.e. one straightforward (and exact) iterative method is:

$$\text{P}(s_i = l) = \sum_{\substack{\mathbf{a_i}\in\mathcal{A}_i \\ s_i = l}} \prod_{j=0}^{n_i} p_{i,j}^{a_{i,j}} \cdot (1 - p_{i,j})^{1-a_{i,j}}$$

For example, if all $p_{i,j} = p$ for all $j \in [1, n_1]$ then we have a binomial distribution case and can compute $\text{P}(s_i = l)$ quickly, and in this case exactly, as $C_{n_i}^{l} p^l (1-p)^{n_i - l}$.

In certain cases it can be possible to utilize some Poisson trials theory to estimate $\text{P}(s_i = j)$. In [22] we discuss situations when it is possible to employ Poisson distribution, Chernoff bounds, and also "Monte-Carlo like" method which derives $\text{P}(s_i = j)$ by generating several random samples.

Method 3: Use linear cost formula. In our implementation of PM we have used the following approach for computing $\text{P}_{\rightarrow}(E_i|\exists\mathcal{P})$. We have a cut-off threshold to decide if $2^{n_i - m}$ iterations is acceptable. If we decide the number of iterations is acceptable we compute $\text{P}_{\rightarrow}(E_i|\exists\mathcal{P})$ precisely, using iterations. In the very rare case when the number of iterations exceeds the threshold, we use the following linear cost formula. This formula is *not correct* in general, but can be treated as a very crude approximation. We compute the expected number of edges μ_i among n_i edges $E_{i,1}, E_{i,2}, \ldots, E_{i,n_i}$, where $\text{P}(\exists E_{i,j}) = p_{i,j}$, as follows: $\mu_i = m + \sum_{j=1}^{n_i - m} p_{i,j}$. Then we say since there are $1 + \mu_i$ possible links to follow on average, the probability to follow E_i can be estimated as:

$$\text{P}_{\rightarrow}(E_i|\exists\mathcal{P}) \approx \frac{1}{1+\mu_i} = \frac{1}{m + \sum_{j=0}^{n_i - m} p_{i,j}} \tag{12}$$

Thus in most of the cases when $2^{n_i - m}$ is small we use the exact formula, and in rare cases when the number of iterations $2^{n_i - m}$ is too large we utilize the approximate formula of Equation 12.

4.3 Dependent edge existence

In computing the connection strength between two nodes we have so far concentrated on the case where edges co-occur independently. In general that might not be the case as was illustrated in Figure 4. In this section we discuss how compute connection strength if occurance of edges is not independent. In our model, dependence between two edges arises only when those two edges are option edges of the same choice node. We next show how $\text{P}_{\rightarrow}(\mathcal{P})$ can be computed for these cases.

There are two principal cases we need to consider. The first step should be to handle all choice nodes on the path. The second step should be to handle all choice nodes such that a choice node itself is not on the path but at least two of its option nodes are on the path.

4.4 Choice nodes on the path

The first case of how to deal with choice nodes on the path is a simple one. There are two subcases in this case illustrated in Figure 8 and Figures 9.

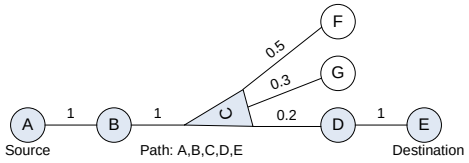

Figure 8: Choice node on the path

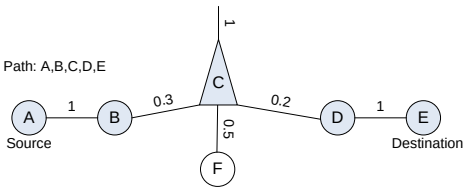

Figure 9: Choice node on the path: illegal path

Figure 8 shows a choice node C on the path which resolves some record of the entity represented by node B and which have options D,G, and F. Recall, we compute $\mathrm{P}_{\rightarrow}(\mathcal{P}) = \mathrm{P}_{\exists}(\mathcal{P}) \cdot \mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P})$. When we compute $\mathrm{P}_{\exists}(\mathcal{P})$ each edge of path $\mathcal{P}$ should exit. Thus edge CD must exist, which means edges CG and CF do not exist. Notice, this case is equivalent to the case where (a) edges CG and CF are not there (eliminated from further consideration); and (b) node C is just an average (not a choice) node connected to D via an edge (in this case the edge is labeled 0.2). If we now consider this equivalent case, then we can simply apply Equation 9 to compute the connection strength.

In general, all choice nodes on the path, can be "eliminated" from the path one by one using the procedure above. A trivial but important observation is, once a choice node is handled in such a fashion, there are no two edges left existence of which is dependent because of this choice node.

Figure 8 shows a choice node C on the path which have options B,F, and D, such that $B - C - D$ is a part of the path $\mathcal{P}$. Since BC and CD are mutually exclusive, path $\mathcal{P}$ can never exist. Such a path is said to be *illegal* and it will not be considered by RDA in computing connection strength.

4.5 Options of the same choice node on the path

Assume now we have applied the procedure from Section 4.4 and all choice nodes are "eliminated" from path $\mathcal{P}$. At this point existence of any edge does not depend on the existence of any of the edges in the path. Also probability $\mathrm{P}_{\rightarrow}(\mathcal{P})$ can be computed as $\mathrm{P}_{\rightarrow}(\mathcal{P}) = \prod_{i=0}^{k} q_i$.

Our goal becomes to create a formula similar to Equation 9, but now for the "dependent" case. Let $\mathbf{f}$ ("f" for "free") be the set of all $a_{i,j}$'s such that each corresponding event $\exists E_{i,j}$ is independent from all other events. If we treat $\mathbf{a}$ as a set, we can compute set $\mathbf{d}$ ("d" for "dependent") of all $a_{i,j}$'s for which events $\exists E_{i,j}$ are dependent: $\mathbf{d} = \mathbf{a} - \mathbf{f}$. If $\mathbf{d}$ is an empty set, then there is no dependence and solution is given by Equation 9, otherwise we proceed as follows. Similarly to $\mathbf{a_i}$ we can define $\mathbf{d_i}$ as $\mathbf{d_i} = \{a_{i,j} : a_{i,j} \in \mathbf{d}, j = 1, \ldots, n_i\}$ and $\mathbf{f_i}$ as $\mathbf{f_i} = \{a_{i,j} : a_{i,j} \notin \mathbf{d}, j = 0, \ldots, n_i\}$, i.e., $\mathbf{a_i} = \mathbf{f_i} \cup \mathbf{d_i}$ and $\mathbf{f_i} \cap \mathbf{d_i} = \emptyset$. We define $\mathcal{D}$ as the set of all possible instantiations of $\mathbf{d}$, and $\mathcal{F}_i$ as the set of all possible instantiations of $\mathbf{f_i}$. Then

$$\mathrm{P}_{\rightarrow}(\mathcal{P}) = \left(\prod_{i=0}^{k} q_i\right) \times \sum_{\mathbf{d}\in\mathcal{D}} \left\{ \left[\prod_{i=1}^{k} \left(\sum_{\mathbf{f_i}\in\mathcal{F}_i} \left\{ \left[\frac{1}{\sum_{j=0}^{n_i} a_{i,j}} \right] \cdot \left[\prod_{j:a_{i,j}\in\mathbf{f_i}} p_{i,j}^{a_{i,j}} \cdot (1-p_{i,j})^{1-a_{i,j}} \right] \right\} \right) \right] \cdot \mathrm{P}(\mathbf{d}) \right\} \quad (13)$$

Equation 13 iterates over all feasible instantiations of $\mathbf{d}$ and $\mathrm{P}(\mathbf{d})$ is the probability of a specific instance. Equation 13 contains $\sum_{\mathbf{d}\in\mathcal{D}} \{[XX] \cdot \mathrm{P}(\mathbf{d})\}$. What this achieves is that a particular instantiation of $\mathbf{d}$ "fixates" a particular combination of all "dependent" edges, and $\mathrm{P}(\mathbf{d})$ corresponds to the probability of that combination. Notice, $[XX]$ directly corresponds to $\mathrm{P}_{\rightarrow}(\mathcal{P}|\exists\mathcal{P})$ part of Equation 9. To compute $\mathrm{P}_{\rightarrow}(\mathcal{P})$ in Equation 13, we only need to specify how to compute $\mathrm{P}(\mathbf{d})$.

4.5.1 Computing P(**d**)

Recall, we now consider cases where $a_{i,j}$ is in **d** only because there is (at least one) another $a_{i',j'} \in \mathbf{d}$ such that events $\exists E_{i,j}$ and $\exists E_{i',j'}$ are dependent and both edges $E_{i,j}$ and $E_{i',j'}$ are *options* of the same *choice* node. Figure 4 is an example of such a case. For each $a_{i,j} \in \mathbf{d}$ we can find (a) choice node cho_l (cho_l is not on the path) such that edge $E_{i,j}$ is incident to cho_l (b) all other (at least one) $a_{i',j'}$'s such that $a_{i',j'} \in \mathbf{d}$ and $E_{i',j'}$ is incident to cho_l. We can put $a_{i,j}$ and all those $a_{i',j'}$'s for a given cho_l into one set C_l. Thus we can split set **d** into subsets $C_1, C_2, \ldots, C_m$.

Notice, existence of each edge $E_{i,j}$ such that $a_{i,j}$ is in one of those sets C_l depends only on existence of edges $E_{i',j'}$'s whose $a_{i',j'}$ is in C_l as well. Thus for each instantiation of **d** we have $\mathrm{P}(\mathbf{d}) = \mathrm{P}(C_1) \times \cdots \times \mathrm{P}(C_m)$. Now, to be able to compute Equation 13, we only need to specify how to compute $P(C_i)$ for $i = 1, 2, \ldots, m$.

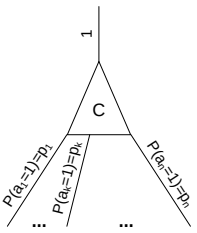

Figure 10: Intra choice dependence.

Computing $P(C_i)$. Figure 10 shows a choice node C with n options labeled with probabilities $p_1, \ldots, p_n$. Assume choice node C is one of those m choice nodes cho_l's mentioned above. As before, to specify which edge is present and which is absent, each option has a variable a_i associated with it: a_i is 1 if the corresponding edge is present and a_i is 0 of it is absent, i.e. $\mathrm{P}(a_i = 1) = p_i$, $\sum_{i=1}^{n} p_i = 1$. Let us assume, without loss of generality, that k first a_i's belong to **d**, i.e. $a_i \in \mathbf{d}$ for $i \in [1, k]$ and $a_i \notin \mathbf{d}$ for $i \in [k+1, n]$. This means first k *option* nodes of *choice* node C belong to the path being considered while the rest $n - k$ option nodes do not belong to the path.

In the context of Figure 10, computing $P(C_i)$ is equivalent to computing of the probability $\mathrm{P}(\vec{a})$ of a particular instantiation of $\vec{a} = (a_1, \ldots, a_k)$ to occur. Notice, only one of $a_1, \ldots, a_k, a_{k+1}, \ldots, a_n$ can be 1. First let us compute the probability of instantiation $\vec{a} = \vec{0} = (0, \ldots, 0)$, that is $a_i = 0$ for $i = 1, \ldots, k$. Assume $[0, 1]$ interval is divided into two intervals: $[0, \sum_{i=1}^{k} p_i)$ and $[\sum_{i=1}^{k} p_i, 1]$. Probability $\mathrm{P}(\vec{a} = \vec{0})$ is equal to the probability that a random number, generated according to $\mathrm{U}[0, 1]$ distribution, would fall into the second interval, thus $\mathrm{P}(\vec{a} = \vec{0}) = \sum_{i=k+1}^{n} p_n$. The other case that is left for consideration is when one of $a_1, \ldots, a_k$ is one. Assume $a_l = 1$, where $l \in [1, k]$, in which case $\mathrm{P}(a_l = 1) = p_l$. To summarize:

$$\mathrm{P}(\vec{a}) = \begin{cases} \sum_{i=k+1}^{n} p_n & \text{if } a_i = 0 \text{ for } i = 1, \ldots, k \\ p_l & \text{if } \exists l \in [1, k] : a_l = 1 \text{ and } a_i = 0 \text{ for } i = 1, \ldots, l-1, l+1, \ldots, k. \end{cases}$$

Now we know how to compute $P(C_i)$ for $i = 1, \ldots, m$, thus we can compute $\mathrm{P}(\mathbf{d})$. Therefore we have specified how to compute path connection strength using Equation 13.

4.6 Computing the total connection strength.

Summation operation in the computation of the connection strength. The connection strength between nodes u and v is computed as a sum of connection strengths of all simple paths between u and v: $c(u, v) = \sum_{p \in \mathcal{P}(u,v)} c(p)$. Based on this connection strength the weight of the corresponding edge will be determined. This weight will be treated as the probability of the edge to exist.

Let us give the motivation of why the *summation* of individual simple paths is performed. We associate the connection strength between two nodes u and v with probability of reaching v from u via only k-short

simple paths. Let us name those simple paths $\mathcal{P}_1,\ldots,\mathcal{P}_k$, Let us call $\mathscr{P}$ the subgraph comprised of the union of those paths: $\mathscr{P} = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_k$. Subgraph $\mathscr{P}$ is a subgraph of the complete graph $G(V, E)$, where V is the set of vertices $V = \{v_i : i = 1, \ldots, |V|\}$ and E is the set of edges $E = \{E_i : i = 1, \ldots, |E|\}$. Let us define a_i as follows: if $\exists E_i$ then $a_i = 1$ else $a_i = 0$, and let $\mathbf{a}$ denote vector $(a_1, \ldots, a_{|E|})$ and $\mathcal{A}$ is the set of all possible instantiations of $\mathbf{a}$.

We need to compute the probability to reach v from u via subgraph $\mathrm{P}_{\rightarrow}(\mathscr{P})$ which we treat as the measure of the connection strength. We can represent $\mathrm{P}_{\rightarrow}(\mathscr{P})$ as:

$$\mathrm{P}_{\rightarrow}(\mathscr{P}) = \sum_{\mathbf{a}\in\mathcal{A}} \mathrm{P}_{\rightarrow}(\mathscr{P}|\mathbf{a}) \cdot \mathrm{P}(\mathbf{a}) \tag{14}$$

Notice, when computing $\mathrm{P}_{\rightarrow}(\mathscr{P}|\mathbf{a})$ the complete knowledge of which edges are present and which are absent is available, as if all the edges were "fixed". That is, assuming one particular instantiation of $\mathbf{a}$, there is no dependence among edge existence events. That is each edge is either present with 100% probability or absent with 100% probability. Thus

$$\mathrm{P}_{\rightarrow}(\mathscr{P}|\mathbf{a}) = \mathrm{P}_{\rightarrow}(\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_k|\mathbf{a}) = \sum_{i=1}^{k} \mathrm{P}_{\rightarrow}(\mathcal{P}_k|\mathbf{a}) \tag{15}$$

and

$$\mathrm{P}_{\rightarrow}(\mathscr{P}) = \sum_{\mathbf{a}\in\mathcal{A}} \mathrm{P}_{\rightarrow}(\mathscr{P}|\mathbf{a})\cdot\mathrm{P}(\mathbf{a}) = \sum_{\mathbf{a}\in\mathcal{A}} \left[\left(\sum_{i=1}^{k} \mathrm{P}_{\rightarrow}(\mathcal{P}_k|\mathbf{a}) \right) \cdot \mathrm{P}(\mathbf{a}) \right] = \sum_{i=1}^{k} \left[\sum_{\mathbf{a}\in\mathcal{A}} \Big(\mathrm{P}_{\rightarrow}(\mathcal{P}_k|\mathbf{a}) \cdot \mathrm{P}(\mathbf{a}) \Big) \right] = \sum_{i=1}^{k} \mathrm{P}_{\rightarrow}(\mathcal{P}_k) \tag{16}$$

Equation 16 shows that the total connection strength is the sum of the connection strength of all k-short simple paths.

It is important to note a subtle difference between the semantics of "following a path" in Equation 16 and that in other equations that we have considered before, e.g. such as Equation 13. The issue arises because, in formulae we have considered before, the term for the probability to follow the first edge in the path is computed differently from that of the rest of the path edges, whereas in Equation 16 it is assumed to be computed the same way as for the rest of the edges. In other words, when computing the probability to reach v from u via p for a particular path $p : u \overset{p}{\rightsquigarrow} v$, in previous equations we have assumed that from u the algorithm cannot travel anywhere but the second node in the path. That is, at the source node u we cannot deviate from p via other links. However, it is not so for Equation 16 where formula implies that a deviation from the very first link is possible.

A question might arise "why it has been decided to compute the probability to follow the first edge differently from the others: it seems little will change if the other formula is used". Indeed, little will change. The choice of formula is exploited by an optimization to improve the efficiency of RDA. This optimization computes things slightly differently and this choice of formulae is important for it. All the optimizations are beyond the scope of this paper.

4.7 Other issues

Resolving all weights. As we have seen in Section 4, when resolving a reference $x_i.r_k$ and computing probabilities of option edges $e_1,\ldots,e_N$ of choice $Cho(x_i.r_k)$ to exist it might turn out that those probabilities are determined by the probabilities of other option edges of other choice nodes. This is because those probabilities are derived from the corresponding connection strengths which in turn are determined by the probabilities of other edges. So a change in one probability can affect other probabilities.

In essence RDA, when resolving graph G, attempts to resolve a system of equations. In this system the probability of an option edge to exist is determined by the equations specified in Section 4. We have conducted many experiments with this system including using an off-the-shelf math solver[14] and also we have implemented a naïve iterative method. Each iteration of the naïve iterative method contains a loop which simply calls RESOLVE-REFERENCE() procedure to resolve each reference one by one. Of course, this method produces only approximate solution for the system. For both iterative method and solver method, after the weights are computed, those weights are *interpreted* as discussed in Section 3.3. It turned out that the accuracy of the iterative method and of the method that uses a solver are very close.[11] Further solver is applicable only to much smaller problem sizes, and when k in "k-short simple path" is small. It is also much slower. Thus in the experimental section we study the iterative implementation of RDA.

Selecting subgraph of interest. Assume the goal for an analyst is to resolve $x_i.r_k$ using an FBS approach. Assume the analyst knows it can refer to one of the entities $y_1,\ldots,y_N$ of the same class of entities Y. In such situations the analyst usually knows that it makes no sense to use certain attributes of x_i and y_i's for that purpose, because these attributes are irrelevant for the task. Similarly, if RBS is used, it might make no sense to use certain relationships for a given task. This paper assumes the analyst first manually specifies a *subgraph of interest* G' of the overall graph $G(V,E)$ such that all the relationships in G' are relevant. In the future work we plan to address how to learn automatically which connection are irrelevant given a sample resolved set.

It is important to mention that specifying G' is not done at the instance level, which would be impossible to do. When using FBS, the analyst can specify for all entities in *class* Y not to use attribute a_1. Similarly, for RBS the analyst can specify which *types* of relationships are irrelevant for the purpose of disambiguating $x_i.r_k$.

Excluding paths when computing connection strength. When resolving $x_i.r_k$ and computing connection strength $c(x_i,y_j)$ for one of the $y_j \in CS(x_i.r_k)$, semantically RDA tests hypothesis that $d(x_i.r_k)$ is y_j, hence it assumes edge $e_j = (Cho(x_i.r_k), V(y_j))$ is present and the rest of the edges among $e_1,\ldots,e_N$ are absent. This excludes from the consideration all the paths containing those edges during discovering of all k-short simple paths by the algorithm. Path $V(x_i) - Cho(x_i.r_k) - V(y_j) = (e_0, e_j)$ is excluded from the consideration as well, because the goal is to compute the connecting strength between $V(x_i)$ and $V(y_j)$ through the rest of the graph G and then determine the probability of edge $Cho(x_i.r_k) - V(y_j)$ to exist based on that connection strength.

5 Experimental Results

In this section we experimentally study RDA using real and synthetic datasets. We have used a Pentium 2GHz machine for our experiments. We will test RDA for accuracy and efficiency. Also two more types of tests are interesting: the effect of the number/amount of relationships and of longer paths on the accuracy of RDA. The optimizations we have developed make the RDA 1–2 orders of magnitude more efficient than a naïve implementation. These optimizations are beyond the scope of this paper. When looking for k-short simple paths RDA is specified to look for paths of length less or equal to L, where L is 8 by default.

5.1 Datasets

We have considered the author matching (AM) problem: which considers papers and authors. For each paper we know the list of author names as strings. But each author name, such as "J. Doe", can in general

[11] This is because even thought the iterative method does not find the exact probabilities, those probabilities are close enough to those computed by the solver method and thus, when they are *interpreted*, both methods obtain similar results.

Figure 11: Types of entities used in all datasets

match multiple *author* entities who happen to share similar name: there can be authors called "John Doe" and "Jane Doe" in the dataset.

The exact types of entities we used in all datasets are shown in Figure 11: there are no extra attributes in those entities (e.g. *paper* entity has only id and author names attributes, but not title, keywords, paper content, etc).

Dataset1 [real]. The graph is constructed from two public-domain sources: CiteSeer[6] and HPSearch[18]. CiteSeer can be viewed as a collection of research publication, HPSearch as a collection of information about authors. From CiteSeer the following information of interest to us is extracted. Each paper is assigned a unique id and the list of its author names is extracted. From HPSearch it is possible to construct the following: for each author its unique id, its name, sometimes its affiliation: organization or university and, sometimes, its department.

There are four *classes* of entities in this example: papers, authors, organizations, departments. Each node in the graph corresponds to an entity from on of these classes. If a specific author affiliated to specific department he is connected to this department via an edge. If a department is a part of a specific organization, this department is connected to this organization via an edge. If a paper is written by a specific author, this paper is connected to this author via an edge. No other type of edges are possible.

There are 255K *paper* entities and 176K *author* entities. Dataset1 is good for testing the efficiency of RDA, but not accuracy, since the correct authors are now known. The purpose of Dataset1 and its subsets: test efficiency on real data.

Dataset2 and Dataset3 [synthetic]. A standard method of testing accuracy is to create an dirty dataset from a known dataset by introducing uncertainty in it. Thus we have created a dataset of 5000 papers and 1000 authors, which we have tried to make as reasonable as possible. Each author is affiliated with one of 25 universities each of which has 5 departments. Authors are divided into students and faculty. Most of the authors, including faculty, have (current or former) advisors. Advisors of authors who are currently students are likely to be from the same department of the same university; advisors of authors who are currently faculty are from random universities but likely from the same department type (e.g. "CS"). Correspondingly each advisor can have several generations of his students. A typical paper is written by a faculty member with other coauthors. Each coauthor can be (with different probabilities) a student of the faculty from one generation, another faculty member(student) from the same (different) department of the same (different) university.[12] The difference between Dataset2 and Dataset3 is that author full names are constructed differently in those datasets to reflect two different types of uncertainty as will be explained shortly. The purpose of Dataset2 and Dataset3: testing all aspects of RDA, however Dataset1 is better for testing efficiency.

5.2 Accuracy

In this context, the accuracy is the fraction of authors correctly resolved by RDA.

Accuracy on Dataset1. We have applied RDA to Dataset1. When matching author names with author entities, in 8% of the cases RBS could not find any paths, thus RDA was equivalent to FBS. However issues regarding the accuracy of the result are not trivial. We do not dwell any further on this issue except

[12]Notice, if there are no rules of who writes papers, i.e., each coauthor in a paper is completely random, then RBS part of RDA is not applicable and RDA is identical to FBS.

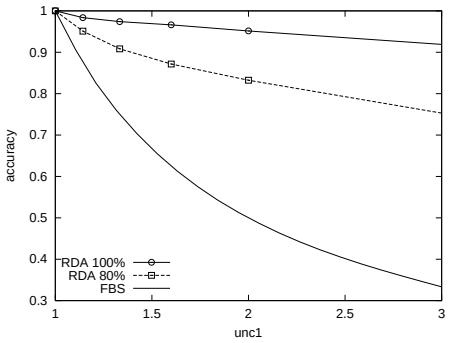

Figure 12: $acc = f(unc_1)$, $L = 8$

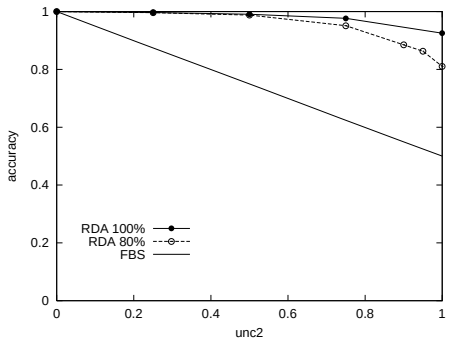

Figure 13: $acc = f(unc_2)$, $L = 5$

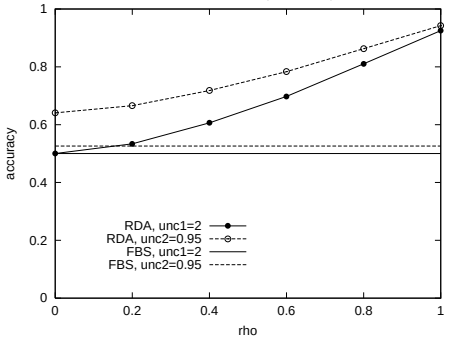

Figure 14: $acc = f(\rho)$

to observe that since the information about the real authors of the papers is not available, it is infeasible to determine the actual accuracy of the result.[13]

Accuracy on Dataset2. Figures 12 and 13 demonstrate the effect of uncertainty on accuracy of RDA and any FBS for two different kinds of uncertainty. Each figure has three curves: one for FBS and two for RDA when for 100%(80%) of authors their universities/departments are known.

Uncertainty of type 1. In the first case, shown in Figure 12, there are N_{auth} (here always $N_{auth} = 1000$) unique authors which use N_{name} different names. For ease of explanation, assume author full names are integers. The name of author with ID k, where $k = 0, \ldots, 999$, is computed as (k `mod` N_{name}) The *author name* attributes in *paper* entities always specify full author names (those integers). Parameter $uncertainty_1$

[13] We have resolved manually a random sample of authors (not in the above 8%) by going to their homepages containing their publications. The whole sample was resolved correctly but we could not test a large sample.

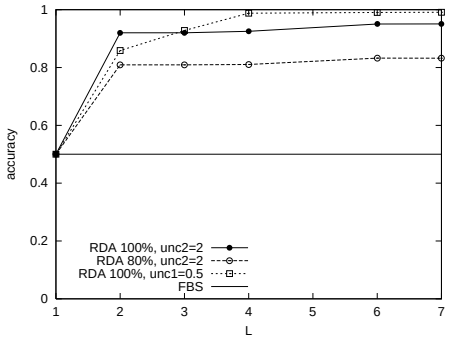

Figure 15:

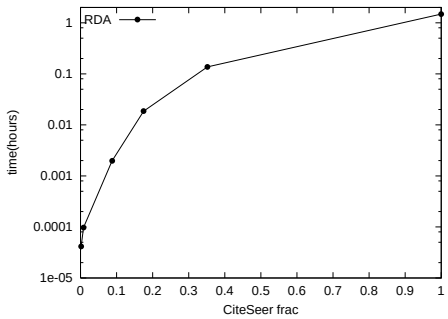

Figure 16:

in Figure 12 is $uncertainty_1 = \frac{N_{auth}}{N_{name}}$ ratio.

Example Let us see what this means if N_{name} is 750. Authors with IDs $0, \ldots, 749$ have names "0",...,"749". Authors with IDs $750, \ldots, 999$ have names "0",...,"249". Thus 500 authors (those with IDs $250, \ldots, 749$) have unique names (their names are "250",...,"749"). For each of the rest of 500 authors (those with IDs $0, \ldots, 249$ and $750, \ldots, 999$) there is another author with the identical name.

Notice, for each author whose name is not unique, one can never identify with 100% confidence any paper this author has written, thus uncertainty for such authors is very high. When $uncertainty_1$ is 1, then there is no uncertainty and all methods show accuracy of 1. As expected, accuracy monotonically decreases as uncertainty increases. When $uncertainty_1$ is 2 uncertainty is very large: for any given author there is exactly one another author with the identical name. For this case, any FBS have no choice but to guess one of the two authors, thus the accuracy of any FBS, as shown in Figures 12, is 0.5. The accuracy of RDA 100% (RDA 80%), given the uncertainty, is quite high : 94%(82%). The gap between RDA 100% and RDA 80% curves shows that for these settings RDA relies substantially on author affiliations for the disambiguations.

Accuracy on Dataset3. *Uncertainty of type 2.* Figure 13 shows the effect of different kind of uncertainty on the accuracy of RDA. For simplicity of explanation of $uncertainty_2$, we will use integers for names again. The first name of author with ID of k, where $k \in [0, 999]$, is given as "F<k>" and last name as "(k `mod` 500)". For example, author with ID of 1 has name (F1, 1), author with ID of 501 has name (F501, 1). Thus if a full name of an author is given in a paper, then this author can be uniquely identified, but if only the first initial is given for the first name, then that author name can correspond to exactly two authors, e.g. author "F. 1" can correspond to both (F1, 1) and (F501, 1). Parameter $uncertainty_2$ in Figure 13 corresponds to fraction of author names in the *paper* table specified by only the first initial and last name. If this fraction is 0, then there is no uncertainty and the accuracy of all methods is 1. Also notice that the case when $uncertainty_1$ is 2 is equivalent to the case when $uncertainty_2$ is 1. Notice, in this test there is much less uncertainty than in the previous one: each author name is unique and for each author there is a chance that for some of his/her papers he/she is known as an author with 100% confidence. The accuracy decreases as uncertainty increases, but this time the accuracy of RDA is much higher. The fact that curves for RDA 100% and RDA 80% are almost indiscernible until $uncertainty_2$ reaches 0.5, shows that RDA relies less heavily on weak author affiliation information but rather on stronger connections via papers. Thus we have empirically shown the importance of utilizing not only FBS but also RBS for the purpose of entity resolution, when information about the relationships among the entities is available.

5.3 Other experiments

Importance of relationships. Figure 14 studies what effect the amount of relationships has on the accuracy of RDA. When resolving *author name* attributes in *paper* entities there are two conceptual types of relationships: author-paper and author-affiliation. The x-axis shows the fraction of authors ρ for which their affiliation is known. If ρ is zero than the affiliation relationship is eliminated completely and RDA has to rely solely on connections via author-paper relationships. If ρ is 1, than complete knowledge of author affiliations is available. Figure 14 shows four curves for accuracy as function of ρ: FBS and RDA for the case as in Figure 12 when $uncertainty_1$ is 2 and for the case as in Figure 13 when $uncertainty_2$ is 0.95.

The accuracy increases as ρ increases showing that the current formula factors in new relationships well. The accuracy of "RDA $unc_1 = 2$" when ρ is 0 equals that of FBS: 0.5. This is because when unc_1 is 2, for each author there is exactly one author with an identical name, but there is no affiliation information that can help disambiguate between authors.

Longer paths. Figure 15 examines the effect of path limit parameter L on the accuracy. The accuracy increases as L increases. It increases rapidly as L reaches 2 since *author*→*paper*→*author* relationship can be discovered and increases slowly after that. Tests with very long relationships have not been conducted

due to slow performance of RDA for such cases. In general, larger L leads to higher accuracy. The practical usefulness of longer paths depends on the combination of other parameters. For example, in Figure 15 the difference between accuracy when L is 2 and when L is 7 is (a) 3% for RDA 80%; (b) 3% for RDA 100%; and (c) it is substantial 13.21% for RDA 100% with $uncertainty_1 = 0.5$ and when the number of universities in the model is decreased to ten.

Efficiency of RDA. Even though the efficiency is not the focus of this paper, to prove the applicability of our approach to a large dataset we have successfully applied it to Dataset1 and its subsets with L ranging from 3 up to 8. However it is not possible to measure the accuracy of the outcome as explained at the beginning of this section. Figure 16 shows the execution time of RDA as a function of the fraction of papers from CiteSeer dataset, i.e. 1 corresponds to all papers in the CiteSeer dataset.

6 Related Work

Besides works mention in the introduction, the most relevant work is in the area of data cleansing (or cleaning). The problem of data cleaning is a very important problem that has been studied extensively in the literature. It is also referred to as record linkage[29, 28, 12], merge/purge[16, 17], object identification[35], duplicate elimination[3, 1], or reference matching[25] etc. by different communities.

Probabilistic linkage technique has been used since the pioneer work of Fellegi and Sunter[12] who provide the theoretical foundation for the subsequent work. The basic idea is to view the problem of identifying matching records as classification task and use probabilistic model to decide matching and non-matching record-pairs. For the concern of the efficiency, they provide a blocking mechanism so that only records in the same block are compared.

Several methods follow the spirit of blocking mechanism of the Fellegi-Sunter theory and address the computational complexity and scalability issue. The *sorted neighborhood* method[16] use multiple keys to sort the database and compare only those records within a sliding window. The *canopy* method[25] uses an extremely inexpensive string distance metric, such as TF-IDF distance metric, to output overlapping clusters that contains possible matching records. More recently, Cohen et al. propose a scalable and adaptive methods for clustering and matching identifier names, in the sense that they can be trained to obtain better performance[11].

To improve the matching accuracy, many methods are proposed, varying by the extent to which human expertise are involved or the machine learning techniques are used. Rule-based methods require the most human involvement and are knowledge intensive. Human experts need to specify the conditions in which records are equivalent[16, 13, 24, 32]. A declarative rule language is presented in [13] to enable users to express the specifications.

Probabilistic methods are developed after the Fellegi-Sunter framework and use unsupervised machine learning methods. The powerful *expectation maximization (EM)* algorithm is employed to classify record pairs into the three classes: matched, non-matched, and possible matched based on statistical properties without any training data[40]. A domain-independent unsupervised approach is to treat the matching task as an information retrieval problem[9]. The approach uses the TF-IDF weighting scheme and employs cosine similarity in the vector space. *Database hardening* approach[8] considers the problem of inferring the most likely "hard" (precise) database from a "soft" (noisy) database constructed from heterogeneous sources and which contains inconsistencies and duplication. It uses a graph of similarity values between records to obtain the best global record matching.

Many efforts have been made to solve the linkage problem by exploiting textual similarity. There are a number of distance metrics proposed by different communities, including edit-distance metrics[26, 27], Jaro metric and its variants[20, 21, 39], TF-IDF distance metrics based on token[7, 15],and hybrid methods. A comparison of various string distance metrics are presented by [10].

The most recent work on on-line domain independent data cleaning is by Chaudhuri et. al [4]. In this publication the authors propose a new string similarity function which is based not only on edit distance but also uses IR's tf-idf concept for creating a better similarity function. In order to speedup retrieval of top-K most likely candidates for cleaning a particular tuple, [4] propose to use the error tolerant index relation. Recall, in this paper we are using the term fms from [4], but we refer to *any* string-based similarity match functions.

Recently, researchers in AI community employ supervised machine learning techniques, such as Bayesian decision model[37], Hidden Markov model[5], and Markov chain Monte Carlo[31] to make the matching decision. The methods of learning string similarity to classify matched and unmatched records are highly interested[33, 34, 36, 2].

In [23], the authors developed a method to determine the context similarity of records and identify the spurious links. A concept hierarchy is employed and association-rules mining is applied to the database to discover all associations among the attribute values, in order to identify the context attributes. The similarity of records is determined by how similar their context attributes are. This method is similar to our work in the sense of using context (relationship in our jargon), while our approach is more general and domain independent.

7 Conclusion

In this paper we have presented an entity disambiguation algorithm called RDA. Our approach determines the strength of the relationship between the two entities u and v and uses this strength to determine if a reference in the context of u is referring to v. The main contribution of this paper is the probabilistic model for computing the connection strength between two nodes based on relationships between those nodes. The empirical results have demonstrated that RDA approach achieves high disambiguation accuracy and it is applicable to a large dataset.

Our immediate research goal is to develop an approach which, given a sample resolved graph, would automatically determine which relationships are irrelevant for a particular disambiguation task.

References

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eli- minating fuzzy duplicates in data warehouses. In *Proc. of VLDB Conf.*, 2002.

[2] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of ACM SIGKDD Conf.*, 2003.

[3] D. Bitton and D. J. DeWitt. Duplicate record elimination in large data files. *ACM TODS*, 8(2):255–265, June 1983.

[4] 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.

[5] P. Christen, T. Churches, and J. X. Zhu. Probabilistic name and address cleaning and standardization. The Australasian Data Mining Workshop, 2002.

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

[7] W. Cohen. Data integration using similarity joins and a word-based information representation language. *Transactions on Information Systems*, 18(3), Jul 2000.

[8] W. Cohen, H. Kautz, and D. McAllester. Hardening soft information sources. In *Proc. of KDD Conf.*, 2000.

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

[10] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. IIWeb Workshop 2003, 2003.

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

[12] I. Fellegi and A. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.
[13] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita. Declarative data cleaning: Language, model, and algorithms. In *Proc. of VLDB Conf.*, 2001.
[14] GAMS/SNOPT solver. www.gams.com/solvers/.
[15] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB01.*
[16] M. Hernandez and S. Stolfo. The merge/purge prob- lem for large databases. In *Proc. of SIGMOD*, 1995.
[17] M. A. Hernandez and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1):9–37, January 1998.
[18] HomePageSearch. http://hpsearch.uni-trier.de.
[19] P. Hsiung. Alias detection in link data sets. CMU-RI-TR-04-22.
[20] M. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414C420, 1989.
[21] M. Jaro. Probabilistic linkage of large public health data files. *Statistics in Medicine*, 14(5C7):491C498, Mar.CApr. 1995.
[22] D. Kalashnikov and S. Mehrotra. An algorithm for entity disambiguation. University of California, Irvine, TR-RESCUE-04-10.
[23] M. Lee, W. Hsu, and V. Kothari. Cleaning the spurious links in data. *IEEE Intelligent Systems*, pages 28–33, March-April 2004.
[24] M. Lee, T.W.Ling, and W.L.Low. Intelliclean: A knowledge-based intelligent data cleaner. In *Proc. of ACM SIGKDD Conf.*, 2000.
[25] A. K. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. of ACM SIGKDD Conf.*, pages 169–178, 2000.
[26] A. E. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *Proc. of ACM SIGKDD Conf.*, 1996.
[27] A. E. Monge and C. P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proc. of SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.
[28] H. B. Newcombe and J. M. Kennedy. Record linkage making maximum use of the discriminating power of identifying information. *Communications of the ACM*, 5:563–566, 1962.
[29] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.
[30] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *Proc. of NIPS Conf.*, 2002.
[31] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In Advances in Neural Processing Systems 15, 2002.
[32] V. Raman and J. Hellerstein. Potter‘s wheel: An inte- ractive data cleaning system. In *VLDB Journal*, 2001.
[33] E. Ristad and P. Yianilos. Learning string edit distance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(5):522–532, May 1998.
[34] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proc. of ACM SIGKDD Conf.*, 2002.
[35] S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, December 2001.
[36] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proc. of ACM SIGKDD Conf.*, 2002.
[37] V. Verykios, G.V.Moustakides, and M. Elfeky. A bayesian decision model for cost optimal record matching. *The VLDB Journal*, 12:28–40, 2003.
[38] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *Proc. of ACM SIGKDD Conf.*, 2003.
[39] W. Winkler. The state of record linkage and current research problems. In *U.S. Bureau of Census, TR99.*
[40] W. E. Winkler. Advanced methods for record linkage. In *U.S. Bureau of Census*, 1994.