Appeared in: International Conference on Database Systems for Advanced Applications (DASFAA), April 9-12, 2007.

Self-tuning in Graph-based Reference Disambiguation*

Rabia Nuray-Turan Dmitri V. Kalashnikov Sharad Mehrotra
{rnuray,dvk,sharad}@ics.uci.edu

Computer Science Department
University of California, Irvine

Abstract. Nowadays many data mining/analysis applications use the graph analysis techniques for decision making. Many of these techniques are based on the importance of relationships among the interacting units. A number of models and measures that analyze the relationship importance (link structure) have been proposed (e.g., centrality, importance and page rank) and they are generally based on *intuition*, where the analyst intuitively decides a reasonable model that fits the underlying data. In this paper, we address the problem of *learning such models directly from training data*. Specifically, we study a way to calibrate a *connection strength* measure from training data in the context of *reference disambiguation* problem. Experimental evaluation demonstrates that the proposed model surpasses the best model used for reference disambiguation in the past, leading to better quality of reference disambiguation.

1 Introduction

Many modern data mining and data analysis applications employ decision making capabilities that view the underlying dataset as a graph and then compute the relationship/link importance using various link analysis measures/models including node importance, centrality [32], and page rank [5]. Many of these models are intuition-based and depend on the underlying dataset. In general, since the importance measures are data-driven, a domain analyst decides which measure fits the data best. In the absence of domain analyst, an arbitrary model can be used; however, the results might not be optimal. But, what if there is training data available wherein given any two nodes in the graph it is known which node should be more central/important/etc. Can one design measures that are not purely intuition-based but also take into account such information?

In this paper we provide an answer to that question for one of the graph link analysis measures, called *connection strength* (CS). Given any two nodes u and v in the graph G, the connection strength $c(u, v)$ returns how strongly

* 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.

u and v are interconnected to each other in G. We study this measure in the context of reference disambiguation problem. In [8, 16–18, 21] a methodology that successfully applies the CS measure to better the disambiguation quality has been proposed.

Reference disambiguation often comes up when entities in a real world dataset contain references to other entities. Frequently, entities are represented using properties/descriptions that may not uniquely identify them leading to ambiguity. For instance, a dataset may store information about two distinct individuals 'John Smith' and 'Jane Smith', both of whom are referred to as 'J. Smith' ambiguously. References may also be uncertain due to differences in the representations of the same entity and errors in data entries (e.g., 'John Smith' misspelled as 'Jon Smith'). The **goal** of reference disambiguation is for each reference to correctly identify the unique entity it refers to.

It is crucial to preprocess and clean the dataset before applying any data mining/analysis applications; because the quality of the output depends on the quality of the input data. Consequently, a large number of database and machine learning approaches have been proposed for solving the reference disambiguation and related disambiguation challenges, such as entity resolution and record linkage [1, 4, 7, 9, 10, 19, 20, 23, 24, 27, 28, 31].

Recently, some domain-independent data cleaning approaches for reference disambiguation has been proposed [16, 25], that systematically exploits features and relationships among entities for the purpose of disambiguation. The approach in [16], which we employ to test our adaptive solution, views the dataset as a graph of entities that are linked to each other via relationships. The model first utilizes a feature based method to identify a set of candidate entities for a reference. 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 analysis is based on the CAP principle:

> **Context Attraction Principle(CAP):** *If reference r made in the context of entity x refers to and entity y_j, whereas the description provided by r matches multiple entities $y_1, y_2, \ldots, y_N$, then x and y_j are likely to be more strongly connected to each other via chains of relationships than x and y_ℓ $(\ell = 1, 2, \ldots, N; \ell \neq j)$*

To illustrate the CAP, consider a simple publication scenario, where 'authors' write 'papers', and 'authors' are affiliated with some 'organizations'. For instance, some paper P_1 might mention 'J. Smith' as its author. The dataset might contain only two people who have similar names: John and Jane Smith. Then $r =$ 'J. Smith', $x = P_1$, $y_1 =$ 'John Smith', and $y_2 =$ 'Jane Smith'. To decide if the 'J. Smith' is Jane or John, the CAP proposes to compare two sets of paths in the entity-relationship graph that exist between x and y_1 and between x and y_2.

The **main contribution** of this paper is a supervised learning algorithm that learns the importance of relationships, or CS, among the classified entities and makes the approach self-tunable to any underlying domain so that the participation of the domain analyst is minimized significantly.

The rest of this paper is organized as follows. Section 2 covers related work. Section 3 defines the problem of reference disambiguation and the essence of the disambiguation approach we use. An adaptive model for CS is discussed in Section 4. The empirical evaluation of the proposed solution is covered in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

In this section we give a brief overview of the existing connection strength models (Section 2.1) and the reference disambiguation techniques (Section 2.2).

2.1 Connection Strength Models

The connection strength $c(u, v)$ between two nodes u and v reflects how strongly these nodes are related to each other via relationships in the graph. Generally, a domain expert decides a mathematical model to compute $c(u, v)$, which describes the underlying dataset best. Various research communities have proposed measures that are directly related to $c(u, v)$. Below we summarize some of the principal models.

Diffusion kernels on graphs in kernel methodology [29] is directly related to connection strength. Diffusion kernel methods view the underlying dataset as a graph $G = (V, E)$, where V is a set of entities and E is a set of edges which define the base similarities between entities. The base similarity for entities $x, y \in V$ represents the degree of attraction between x and y. Moreover, the base similarities are used to compute indirect similarities by combining the direct similarities in a particular way, see [29] for details.

Another model of measuring CS is *random walks* in graphs. It has been extensively studied, including our previous work [16, 17]. The model uses the probability of reaching node v from node u by random walks in G to compute $c(u, v)$. *Relevant importance in graphs* [33] is a generalized version of page rank algorithm [5]. It studies the relevant importance of a set of nodes with respect to a set of root nodes. The connection strength in this study is the importance of node t given node r (i.e., $I(t|r)$). *Electric circuit model* is also a CS model which uses the electric networks principles to find the connection subgraphs between two nodes u and v [11]. That model views the graph as an electric circuit consisting of resistors, and compute $c(u, v)$ as the amount of electric current that goes from u to v.

2.2 Disambiguation

Reference disambiguation problem is related to the *record de-duplication*, *record linkage*, and *object consolidation* problems [7,15,22] and often arises when different information sources are merged to create a single database. The differences between record linkage and reference disambiguation 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 in a database point to the correct entities. In the reference disambiguation problem, for each reference, a set of possible candidates is given and the task is to pinpoint the correct entity in this set. On the other hand, the object consolidation problem aims to correctly group/cluster the references that refer to the same object without knowing the possible entities in the dataset.

The traditional approach to these problems is to analyze the textual similarities among the object features to make a disambiguation decision. Such approaches are called feature-based similarity (FBS) techniques [12, 13, 26]. Recently, a number of techniques have been proposed that go beyond the traditional approach [1,3,8,10,16–19,23,27,30]. Ananthakrishna et al [1] presented relational deduplication in data warehouses where there is dimensional hierarchy over the relations. Bhattacharya and Getoor introduced a method which requires that social groups function as cliques [3]. This model expects that there are strong correlations between pairs or sets of entities, such that they often co-occur in information sources. Bekkerman and McCallum studied the disambiguation of name references in a linked environment [2]. Their model utilizes the hyperlinks and distance between the pages where ambiguous names are found. Minkov et al [25] introduced extended similarity metrics for documents and other objects embedded in graphs, facilitated by a lazy graph walk. They also introduced a learning algorithm which adjusts the ranking of possible candidates based on the edges in the paths traversed.

In this paper, we employ the algorithm presented in [16,17] to test our adaptive connection strength model. The algorithm uses a graphical methodology; the disambiguation decisions are made not only based on object features like in the traditional approach, but also based on the inter-object relationships, including indirect ones that exist among objects. The essence of the adaptive model is to be able to learn the importance of various connections on past data in the context of reference disambiguation.

3 Problem Definition

We now formally define the reference disambiguation problem. Assume dataset $\mathcal{D}$ contains a set of entities X. Each entity $x \in X$ itself consists of one or more attributes $x.a_1, x.a_2, \ldots$, and it might also contain several references $x.r_1, x.r_2, \ldots$ to other entities in X. Let R be the set of all references. Each reference $r \in R$ is essentially a description and may itself contain one or more attributes. For each reference $r \in R$ the **option set** S_r of that reference is known. It contains all entities in X to which r might potentially refer: $S_r = \{y_{r1}, y_{r2}, \ldots, y_{rn_r}\}$. For r its S_r is initially determined either by ad hoc techniques, domain knowledge, or by choosing all entities whose feature-based similarity exceed a certain threshold. The true (unknown to the algorithm) entity to which r refers to is denoted as r^*. Then the goal of reference disambiguation is to pick the right y_{rj} (i.e., r^*) from S_r to which r really refers to.

We denote the entity in the context of which reference r is made as x_r. The employed reference disambiguation approach resolves each reference $r \in R$ by analyzing direct and indirect relationships that exist between x_r and each member of S_r. For that, it views the dataset $\mathcal{D}$ as an undirected entity-relationship graph G, where node represent entities and edges represent relationships. In essence, G can be viewed as an instantiation of the E/R diagram for $\mathcal{D}$. The approach relies on the CAP principle (Section 1), which can be reformulated in terms of connection strength as: for r its r^* is likely to be such element y_{rj} from S_r that $c(x_r, y_{rj}) \geq c(x_r, y_{r\ell})$ for all $\ell \neq j$.

To make the definition clear, let us assume that we use the publication dataset for reference disambiguation. In the publication domain 'authors' write 'papers'. We might have a paper P_1 that mentions 'J. Smith' as its author. Dataset $\mathcal{D}$ might contain two authors who match that description: John Smith and Jane Smith, where the actual author of P_1 is John. Then r = 'J. Smith', $x_r = P_1$, r^* = 'John Smith', and $S_r = \{$'John Smith', 'Jane Smith'$\}$.

4 Solution

The core of the approach in [16,17] that we employ to test our adaptive solution is a connection strength model, called WM. It is a *fixed* mathematical model and based on some intuitive assumptions which are true for many datasets. In this section we first describe how an *adaptive* CS model can be created (Section 4.1). Then we give an example adaptive CS model (Section 4.2) which is used in this paper. Finally, we discuss the self-tuning algorithm (Section 4.3).

4.1 Adaptive Connection Strength Model

Assume that we can classify each path that the disambiguation algorithm finds in graph G into a finite set of path types $S_T = \{T_1, T_2, ..., T_n\}$. Namely, there is a function $T(p, G) \rightarrow S_T$ such that for any given path p and graph G, maps it to one of those path types. If any two paths p_1 and p_2 are of the same type T_j, then they are treated as identical by the algorithm. Then, for any two nodes u and v we can characterize the connections among them with a path-type count vector $Tuv = (c_1, c_2, ..., c_n)$, where each c_i is the number of paths of type T_i that exist between u and v. If we assume that there is a way to associate weight w_i with each path type T_i, then CS model computes $c(u, v)$ as:

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

The existing CS models differ in classification of path types and in the way of assigning weights to path types. Furthermore, these are generally chosen by the algorithm designer.

4.2 Path Type Model

To classify the paths we use a model which we refer to as Path Type Model (PTM). It classifies paths by looking at the types of edges the path is comprised of. Namely, PTM views each path as a sequence of edges $\langle e_1, e_2, \ldots, e_k \rangle$, where each edge has a type associated with it. This sequence of edge types ($\langle E_1, E_2, \ldots, E_k \rangle$) are treated as a string and PTM assigns different weights to each string. For example, in the publications domain authors write papers and are affiliated with organizations. Hence there are two types of edges that correspond to the two types of relationships: E_1 for 'writes' and E_2 for 'is affiliated with'.

4.3 Learning Algorithm

The CAP principle in the reference disambiguation problem allows us to calibrate a CS model directly from data and apply it in the context of reference disambiguation. The principle states that for a reference r it is likely that

$$c(x_r, y_{rj}) \geq c(x_r, y_{r\ell}) \text{ for any } r, \ell \neq j \text{ where } y_{rj} = r^*. \tag{2}$$

Because of the 'likely' part in the CAP, many of the inequalities in the system (2) should hold, but some of them might not. That is, system (2) might be overconstrained and might not have a solution. To address the 'likely' part, we add a slack to the system and then require it be minimized:

$$\begin{cases} \text{Constraints:} \\ c(x_r, y_{rj}) + \xi_{r\ell} \geq c(x_r, y_{r\ell}) \text{ for any } r, \ell \neq j, y_{rj} = r^* \\ \xi_{r\ell} \geq 0 \\ \\ \text{Objective:} \\ \textit{Minimize } \sum_{r\ell} \xi_{r\ell}. \end{cases} \tag{3}$$

The employed reference disambiguation approach also states that for reference r the connection strength 'evidence' for the right option $y_{rj} = r^*$ should visibly outweigh that for the wrong ones $y_{r\ell}, \ell \neq j$. Thus, in addition to the objective in (3), the value of $[c(x_r, y_{rj}) - c(x_r, y_{r\ell})]$ should be maximized for all $r, \ell \neq j$, which translates into maximizing $\sum_{r\ell} [c(x_r, y_{rj}) - c(x_r, y_{r\ell})]$. After combining the first and second objectives, we have:

$$\begin{cases} \text{Constraints:} \\ c(x_r, y_{rj}) + \xi_{r\ell} \geq c(x_r, y_{r\ell}) \text{ for any } r, \ell \neq j, y_{rj} = r^* \\ \xi_{r\ell} \geq 0 \\ \\ \text{Objective:} \\ \textit{Minimize } \alpha \sum_{r\ell} \xi_{r\ell} + (1 - \alpha) \sum_{r\ell} [c(x_r, y_{r\ell}) - c(x_r, y_{rj})] \end{cases} \tag{4}$$

Here α is a parameter that allows to vary the contribution of the two different objectives. It is a real number between 0 and 1, whose optimal value can be

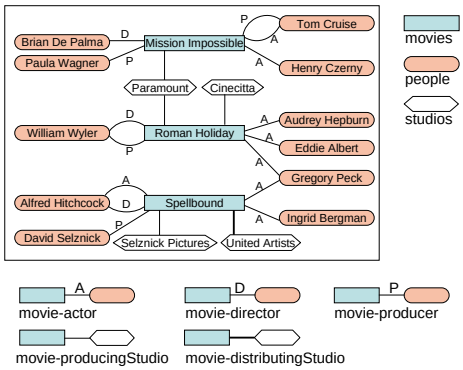

Fig. 1. Movies Dataset: Sample entity-relationship graph.

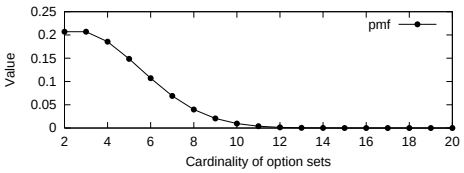

Fig. 2. PMF of sizes of option sets.

determined by varying α on training data and observing the effect on the quality of the disambiguation. System (4) essentially converts the learning task into solving the corresponding linear programming problem, and linear programming, in general, is known to have efficient solutions [14]. All $c(u, v)$ in (4) should be computed according to (1) and adding a normalization constraint that all weights should be in $[0, 1]$ domain: $0 \leq w_i \leq 1$, for all i. The task becomes to compute the best combination of weights $w_1, w_2, \ldots, w_n$ that minimizes the objective, e.g. using any off-the-shelf linear programming solver.

5 Experimental Results

We experimentally study our method using real and synthetic datasets taken from two domains: Movies (Section 5.1) and Publications (Section 5.2). We compare the learning approach (PTM) against the best existing model used for disambiguation so far: the random walk model (WM) [17], which we will refer to as RandomWalk.

RandomWalk model computes $c(u, v)$ as the probability to reach node v from node u via random walks in graph G, such that the probability to follow an edge is proportional to the weight of the edge. Accordingly, $c(u, v)$ is computed as the sum of the connection strength $c(p)$ of each path p from $P_L(u, v)$, where $c(p)$ is the probability of following path p in G, i.e.

$$c(u, v) = \sum_{p \in P_L(u,v)}^{n} c(p) \tag{5}$$

We report the results in terms of **accuracy**[1], which is defined as the fraction of correctly resolved references.

[1] For the reference disambiguation problem we solve, the accuracy and F-measure are known to be the same.

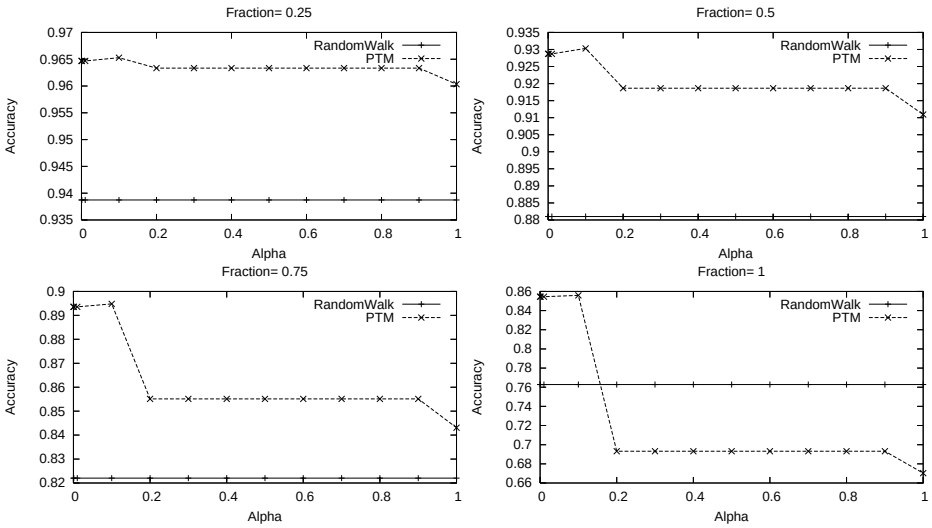

Fig. 3. Accuracy vs. Alpha: where $c = 2$.

5.1 Experiments on the Movies Domain

We use the Stanford Movies Dataset[2]. A sample entity-relationship graph for this dataset is illustrated in Figure 1. The dataset contains three different entity types: *movies* (11,453 entities), *studios* (992 entities) and *people* (22,121 entities) and there are five types of relationships: *actors, directors, producers, producing studios and distributing studios.*

When studying the accuracy of disambiguation, we use a method of testing commonly employed by many practitioners, including the recent KDD CUP. We introduce uncertainty in the dataset manually in a controlled fashion and then analyze the resulting accuracy of various methods. Specifically, we disambiguate references from movies to directors, by making them uncertain. First, a fraction $f : 0 \leq f \leq 1$ of all director references is chosen to be made uncertain, while the rest remain certain. Each to-be-uncertain director reference r is made ambiguous by modifying it such that it either points to two directors instead of one (i.e., $c = |S_r| = 2$) or points to c directors where c is distributed according to the PMF in Figure 2 (i.e., $c = |S_r| \sim pmf$). Here c stands for the **cardinality** of S_r. Training and testing is performed for the same values of f and c, but the director references chosen to be ambiguous are different in training and test data.

Figures 3 and 4 study the effect of the parameter α, that controls the contribution of various objectives in system (4) from Section 4.3, on the accuracy of various approaches, for different combinations of f and c. We performed the experiments for two different cardinalities ($c = 2$ and $c \sim pmf$) and four different fractions of ambiguous entities ($f = \{0.25, 0.5, 0.75, 1\}$). In these experiments,

[2] http://www-db.stanford.edu/pub/movies/

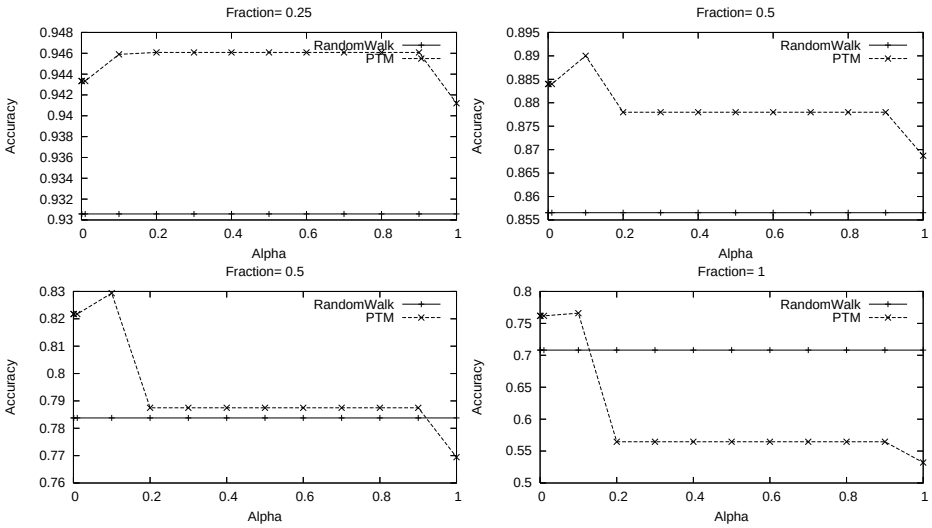

Fig. 4. Accuracy vs. Alpha: where $c \sim \ pmf$.

the optimal α was found to be approximately 0.10. When α is set to its best value, PTM visibly outperforms the state-of-the-art model, RandomWalk. As the number of ambiguous references increases, the improvement with the PTM against the RandomWalk method becomes more significant. For example, the improvement with PTM for $f = 0.25$ and $c = 2$ is 2.7%, whereas it is 9.18%, when $f = 1$ and $c = 2$.

5.2 Experiments on the Publications Domain

Dataset. We now present the results on **SynPub** dataset, which is from [16] and emulates CiteSeer dataset. It contains four different types of entities: *author*, *paper*, *department*, and *organization* and three types of relationships: *author-paper*, *author-department*, and *department-organization*.

We generated *five* different sets of datasets. Each set contains a training and *ten* different testing datasets, the parameters are same for all datasets; however, the authors to be disambiguated are different. Each dataset has different types and levels of uncertainty (see [17]) and contains 5000 papers, 1000 authors, 25 organizations, and 125 departments. The least ambigious datasets are in set 4, while the most ambiguous ones are either in set 5 or set 1, see Table 1.

Results. For each training dataset, we selected the optimal α value, which is 0.10 for datasets 1, 2, and 5 and 0.01 for dataset 3 and 4. Then these optimal values were used in testing. The average accuracy of different testings are reported in Table 1. Since the results of PTM and RandomWalk are essentially identical, we performed another experiment with a different path classification model, **hybrid** model. This model is the combination of PTM with RandomWalk, such

that it takes into account the node degrees in addition to the edge types in a path. The connection strength of this model is computed as:

$$c(u,v) = \sum_{p \in P_L(u,v)}^{n} c(p) w_{T_i}, \text{ where } T_i = T(p,G) \tag{6}$$

Accuracy results with the hybrid model is the same as the other two models. So we can conclude that RandomWalk model is a good model for this specific setting. However, it may not work ideally for every instance of the publications domain. To show that, we performed some additional experiments. Our intuition

Table 1. Publications dataset results

Dataset	PTM	RandomWalk	Hybrid	FBS
1	93.2%	93.1%	93.1%	50.0%
2	94.9%	94.9%	94.9%	55.0%
3	74.6%	74.7%	74.7%	55.0%
4	98.4%	98.4%	98.4%	74.8%
5	64.9%	64.9%	64.9%	50.0%

in these experiments is that when creating the SynPub dataset, the analyst has chosen to project from CiteSeer relationships of only a few carefully chosen types that wouold work well with RandomWalk, i.e. the three types discussed above, while purposefully pruning away relationship types that are less important for disambiguation and would confuse RandomWalk model. In other words, the analyst has contributed his intelligence to that unintelligent model.

We gradually added random noise to one of the datasets, namely dataset 5, by introducing relationships of a new type – that represent random meaningless relationships. The random relationships were added to the 'false' cases only. That is, the added relationships are between the reference r and the candidates $(y_{rj}) \in \{S_r - r^*\}$. Figure 5 examines the effect of this noise on the accuracy of RandomWalk and PTM techniques. It shows that both of the techniques obtain very high accuracy compared to the standard approach, shown as 'FBS', which does not use relationships for disambiguation. Initially, RandomWalk and PTM has the same accuracy. But as the level of noise increases, the accuracy of RandomWalk drastically drops below that of PTM and FBS. PTM is an intelligent technique that learns the importance of various relationships and can easily handle noise – its curve stays virtually flat. Notice, since FBS does not use any relationships, including the random noise, its curve stays flat as well.

6 Discussions and Conclusion

Our results show that adaptive connection strength model always outperforms the state-of-the-art RandomWalk model. There are many advantages of self-tunable CS model in the context of reference disambiguation. First of all, it

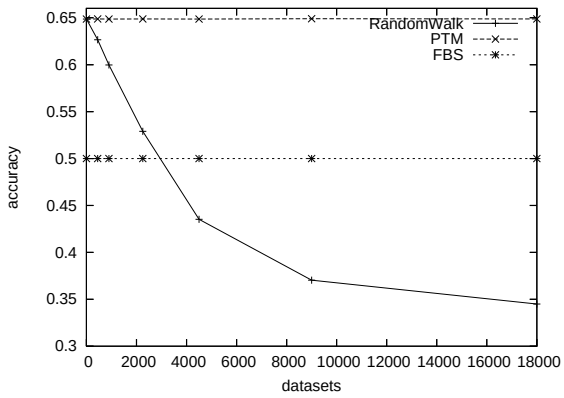

Fig. 5. Accuracy vs. Number of random relationships(noise).

minimizes the analyst participation, which is important since nowadays various data-integration solutions are incorporated in real Database Management Systems (DBMS), such as Microsoft SQL Server DBMS [6]. Having a less analyst-dependent technique makes that operation of wide applicability, so that non-expert users can apply it to their datasets. The second advantage of such a CS model is that it expects to increase the quality of the disambiguation technique. There are also less obvious advantages. For example, the technique is able to detect which path types are marginal in their importance. Thus, the algorithm that discovers paths when computing $c(u, v)$ can be sped up, since the path search space can be reduced by searching only for important paths. Speeding up the algorithm that discovers paths is important since it is the bottleneck of the overall disambiguation approach [16, 17].

References

1. R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, 2002.
2. R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *WWW*, 2005.
3. I. Bhattacharya and L. Getoor. Relational clustering for multi-type entity resolution. In *MRDM Workshop*, 2005.
4. M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD*, 2003.
5. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proc of International World Wide Web Conference*, 1998.
6. S. Chaudhuri, K. Ganjam, V. Ganti, R. Kapoor, V. Narasayya, and T. Vassilakis. Data cleaning in Microsoft SQL Server 2005. In *SIGMOD*, 2005.
7. S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD*, 2003.
8. Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Exploiting relationships for object consolidation. In *ACM IQIS*, 2005.

9. W. Cohen, H. Kautz, and D. McAllester. Hardening soft information sources. In *SIGKDD*, 2000.
10. X. Dong, A. Y. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.
11. C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *SIGKDD*, 2004.
12. I. Fellegi and A. Sunter. A theory for record linkage. *Journal of Amer. Statistical Association*, 64(328):1183–1210, 1969.
13. M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *SIGMOD*, 1995.
14. F. Hillier and G. Lieberman. *Introduction to operations research.* McGraw-Hill, 2001.
15. L. Jin, C. Li, and S. Mehrotra. Efficient record linkage in large data sets. In *DASFAA*, 2003.
16. D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM TODS*, 31(2), 2006.
17. D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM SDM*, 2005.
18. D. V. Kalashnikov, S. Mehrotra, Z. Chen, R. Nuray-Turan, and N. Ashish. Disambiguation algorithm for people search on the web. *ICDE, to appear*, 2007.
19. M. Lee, W. Hsu, and V. Kothari. Cleaning the spurious links in data. *IEEE Intelligent Systems*, Mar-Apr 2004.
20. X. Li, P. Morie, and D. Roth. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *AAAI*, 2004.
21. B. Malin. Unsupervised name disambiguation via social network similarity. In *Workshop on Link Analysis, Counterterrorism, and Security*, 2005.
22. A. McCallum and B. Wellner. Object consolidation by graph partitioning with a conditionally-trained distance metric. In *KDD Workshop on Data Cleaning, Record Linkage and Object Consolidation*, 2003.
23. A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS*, 2004.
24. A. K. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *ACM SIGKDD*, 2000.
25. E. Minkov, W. W. Cohen, and A. Ng. Contextual search and name disambiguation in email using graphs. In *SIGIR*, 2006.
26. H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.
27. H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS Conference*, 2002.
28. S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *SIGKDD*, 2002.
29. J. Shawe-Taylor and N. Cristianni. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.
30. P. Singla and P. Domingos. Multi-relational record linkage. In *MRDM Workshop*, 2004.
31. S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string tranformation weights for high accuracy object identification. In *SIGKDD*, 2002.
32. S. Wasserman and K. Faust. *Social Network Analysis Methods and Applications.* Cambridge University Press, 1994.
33. S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *SIGKDD*, 2003.