Object Distinction: Distinguishing Objects with Identical Names

Xiaoxin Yin
Univ. of Illinois
xyin1@uiuc.edu

Jiawei Han
Univ. of Illinois
hanj@cs.uiuc.edu

Philip S. Yu
IBM T. J. Watson Research Center
psyu@us.ibm.com

Abstract

Different people or objects may share identical names in the real world, which causes confusion in many applications. It is a nontrivial task to distinguish those objects, especially when there is only very limited information associated with each of them. In this paper, we develop a general object distinction *methodology called* DISTINCT*, which combines two complementary measures for relational similarity:* set resemblance of neighbor tuples *and* random walk probability*, and analyze subtle linkages effectively. The method takes a set of distinguishable objects in the database as the training set without seeking for manually labeled data, and apply SVM to weigh different types of linkages. Experiments show that* DISTINCT *can accurately distinguish different objects with identical names in real databases.*

1 Introduction

People retrieve information from different databases on the Web, such as DBLP, Yahoo shopping, and AllMusic . One problem that has always been disturbing is that different objects may share identical names. For example, there are 197 papers in DBLP written by at least 14 different "Wei Wang"s. Another example is that there are 72 songs and 3 albums named "Forgotten" in allmusic.com. Users are often unable to distinguish them, because the same object may appear in very different contexts, and there is often limited and noisy information associated with each appearance.

In this paper we study the problem of *Object Distinction*, *i.e.*, *Distinguishing Objects with Identical Names*. Given a database and a set of references in it referring to multiple objects with identical names, our goal is to split the references into clusters, so that each cluster corresponds to one real object. We assume that the data is stored in a relational database, and the objects to be distinguished reside in a table. A mini example is shown in Fig. 1, which contains some papers by four different "Wei Wang"s and the linkages among them.

This problem of object distinction is the opposite of a popular problem called *reference reconciliation* (or *record linkage*, *duplicate detection*) [12], which aims at merging

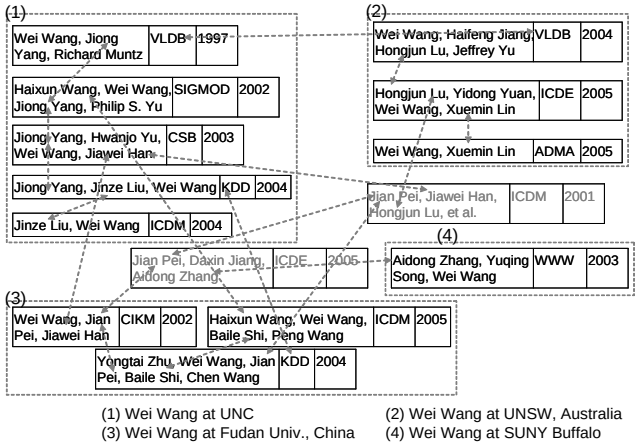

Figure 1. Papers by four different "Wei Wang"s

records with different contents referring to the same object, such as two citations referring to the same paper. There have been many record linkage approaches [1, 2, 4, 6, 9]. They usually use some efficient techniques [7] to find candidates of duplicate records (*e.g.*, pairs of objects with similar names), and then check duplication for each pair of candidates. Different approaches are used to reconcile each candidate pair, such as probabilistic models of attribute values [6, 12] and textual similarities [2, 4].

Compared with record linkage, objection distinction is a very different problem. First, because the references have identical names, textual similarity is useless. Second, each reference is usually associated with limited information, and thus it is difficult to make good judgement based on it. Third and most importantly, because different references to the same object appear in different contexts, they seldom share common or similar attribute values. Most record linkage approaches [2, 4, 6, 12] are based on the assumption that duplicate records should have equal or similar values, and thus cannot be used on this problem.

Although the references are associated with limited and possibly inconsistent information, the linkages among references and other objects still provide crucial information for grouping references. For example, in a publication database, different references to authors are connected in nu-

merous ways through authors, conferences and citations. References to the same author are often linked in certain ways, such as through their coauthors, coauthors of coauthors, and citations. These linkages provide important information, and a comprehensive analysis on them may likely disclose the identities of objects.

In this paper we develop a methodology called DISTINCT that can distinguish object identities by fusing different types of linkages with differentiating weights, and using a combination of distinct similarity measures to assess the value of each linkage. Specifically, we make the following three major contributions.

First, because the linkage information is usually sparse and intertwined, DISTINCT combines two approaches for measuring similarities between records in a relational database. The first is *set resemblance between the neighbor tuples* of two records [1] (the *neighbor tuples* of a record are the tuples linked with it); and the second is *random walk probability* between two records in the graph of relational data [9]. These two approaches are complementary: one uses the neighborhood information, and the other uses connection strength of linkages.

Second, there are many types of linkages among references, each following a join path in the database schema. Different types of linkages have very different semantic meanings and different levels of importance. DISTINCT uses Support Vector Machines (SVM) [3] to learn a model for weighing different types of linkages.

Finally, because references to the same object can be merged and considered as a whole, DISTINCT uses agglomerative hierarchical clustering [8], which repeatedly merges the most similar pairs of clusters. It combines *Average-Link* (average similarity between all objects in two clusters) and *collective similarity* (considering each cluster as a single object) to measure the similarity between two clusters, which is less vulnerable to noise.

The rest of this paper is organized as follows. Section 2 describes our similarity measure between references. Section 3 presents the approach for combining similarities on different join paths using supervised learning. The approach for clustering references is described in Section 4. Experimental results are presented in Section 5, and this study is concluded in Section 6.

2 Similarity Between References

We say a set of references are *resembling* if they have identical textual contents (*e.g.*, references to authors with identical names). Two references are *equivalent* if they refer to the same object, and *distinct* if they do not. Our goal is to group a set of resembling references into clusters so that there is a 1-to-1 mapping between the clusters and the real objects. In this section we describe our similarity measure for references. Because each reference is usually associated with very limited information, we utilize the relationships between each reference and other tuples in the database, with the following two types of information: (1) the *neighbor tuples* of each reference, and (2) the *linkages* between different references. Based on our observation, for two references, the more overlapping on their neighborhood, or the stronger linkages between them, the more likely they are equivalent.

2.1 Neighbor Tuples

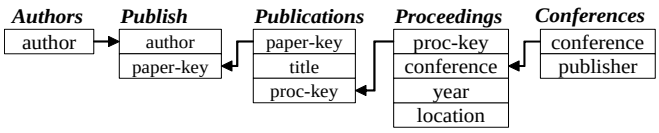

Figure 2. The schema of DBLP database

The *neighbor tuples* of a reference are the tuples joinable with it. A reference has a set of neighbor tuples along each join path starting at the relation containing references. The semantic meaning of neighbor tuples is determined by the join path. For example, in the DBLP database whose schema is shown in Fig. 2, we study the references to authors in $Publish$ relation. The neighbor tuples along join path "$Publish \bowtie Publications \bowtie Publish \bowtie Authors$" represent the authors of the paper for a reference. Because different join paths have very different semantic meanings, we treat the neighbor tuples along each join path separately, and will combine them later by supervised learning.

Definition 1 ***(Neighbor tuples)*** *Suppose the references to be resolved are stored in relation R_r. For a reference r that appears in tuple t_r, and a join path P that starts at R_r and ends at relation R_t, the neighbor tuples of r along join path P, $NB_P(r)$, are the tuples in R_t joinable with t_r along P.*

Besides neighbor tuples of each reference, the attribute values of neighbor tuples are also very useful for reconciling references. For example, two neighbor tuples in *Conferences* relation sharing the same value on *publisher* attribute indicates some relationship between these two tuples. In DISTINCT, *we consider each value of each attribute (except keys and foreign-keys) as an individual tuple*. For example, each distinct value of *publisher* attribute (ACM, Springer, *etc.*) is considered as a tuple, and the *publisher* attribute in *Proceedings* relation is considered as a foreign-key referring to those tuples. In this way we can use a single model to utilize both neighbor tuples and their attribute values.

2.2 Connection Strength

For a reference r and a join path P, the strengths of relationships between r and different tuples in $NB_P(r)$ could be very different (*e.g.*, the different relationships between an author and different coauthors). We use probability propagation [11] to model the connection strength between a reference r and its neighbor tuples $NB_P(r)$. Initially the tuple

containing r has probability 1. At each step, for each tuple t with non-zero probability, we uniformly propagate t's probability to all tuples joinable with t along the join path P. For each tuple $t \in NB_P(r)$, we compute $Prob_P(r \to t)$, the probability of reaching t from r via join path P, which is used as the *connection strength* between r and t. We also compute $Prob_P(t \to r)$, which is the probability of reaching r from t via the reverse join path of P.

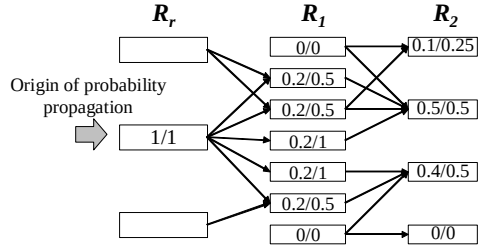

Figure 3. Propagating probabilities between tuples

The computation of both types of probabilities can be done in a depth-first traversal of all qualified join paths. Fig. 3 shows the procedure of propagating probabilities from a tuple in R_r to tuples in R_1 and R_2. The two numbers in each box are the probability of reaching this tuple and the probability of reaching the origin from this tuple.

2.3 Set Resemblance of Neighbor Tuples

Our first measure for similarity between references is the set resemblance between their neighbor tuples, which represents the similarity between the contexts of two references in a relational database. The set resemblance between neighbor tuples is defined by Jaccard coefficient [10]. Because a reference has different connection strengths to different neighbor tuples, we use such strengths as weights in Jaccard coefficient.

Definition 2 ***(Set Resemblance.)*** *The set resemblance similarity between two references r_1 and r_2 with respect to join path P is defined as*

$$Resem_P(r_1, r_2) = \frac{\sum_{t \in NB_P(r_1) \cap NB_P(r_2)} \min(Prob_P(r_1 \to t), Prob_P(r_2 \to t))}{\sum_{t \in NB_P(r_1) \cup NB_P(r_2)} \max(Prob_P(r_1 \to t), Prob_P(r_2 \to t))}$$

2.4 Random Walk Probability

Another important factor for similarity between references is the linkages between them. We use the random walk probability model used in multi-relational record linkage [9]. The total strength of the linkages between two references is defined as the probability of walking from one reference to the other within a certain number of steps. A distinguishing feature of our approach is that, we compute the random walk probability along each join path separately, so as to acknowledge the different semantics of different join paths.

It is usually expensive to compute random walk probabilities along long join paths. Since we have computed the probabilities of walking from references to their neighbor tuples, and those from neighbor tuples to references, we can easily compute the probability of walking between two references by combining such probabilities.

In general, random walk probability indicates the linkage strength between references. It is complementary to set resemblance, which indicates the context similarity of references. DISTINCT combines both measures to perform comprehensive analysis on similarities between references.

3 Supervised Learning with Automatically Constructed Training Set

In previous record linkages approaches that utilize relation information [1, 9], all join paths are treated equally. However, linkages along different join paths have very different semantic meanings, and thus should have different weights. For example, in DBLP database two references to authors being linked by the same coauthor is a strong indication of possible equivalence, whereas two references being linked by the same conference is much weaker.

DISTINCT uses supervised learning to determine the pertinence of each join path and assign a weight to it. In order to do this, a training set is needed that contains equivalent references as positive examples and distinct references as negative ones. Instead of manually creating a training set which requires much labor and expert knowledge, DISTINCT constructs the training set automatically, based on the observation that the majority of entities have distinct names in most applications. Take the problem of distinguishing persons as an example. A person's name consists of the first and last names. If a name contains a rather rare first name and a rather rare last name, this name is very likely to be unique. We can find many such names in a database and use them to construct training sets. A pair of references to an object with a unique name can be used as a positive example, and a pair of references to two different objects can be used as a negative example.

Given the training examples, we use Support Vector Machines (SVM) [3] to learn a model based on similarities via different join paths. We introduce the learning procedure for set resemblance similarities, and the same procedure is also applied on random walk probabilities. Each training example (which is a pair of references) is converted into a vector, and each dimension of the vector represents set resemblance similarity between the two references along a certain join path. Then SVM with linear kernel is applied to this training sets, and the learned model is a linear combination of similarities via different join paths. Usually some important join paths have high positive weights, whereas others have weights close to zero and can be ignored in further computation. Let $Resem(r_1, r_2)$ be the overall set re-

semblance similarity between r_1 and r_2,

$$Resem(r_1, r_2) = \sum_{P \in \mathcal{P}} w(P) \cdot Resem_P(r_1, r_2), \quad (1)$$

where $w(P)$ is the weight of join path P.

4 Clustering References

Given a set of references to the same name, DISTINCT tries to group them into clusters, so that each cluster corresponds to a real entity. The procedure of clustering references will be discussed in this section.

4.1 Clustering Strategy

The problem of clustering references has the following special features: (1) The references do not lie in a Euclidean space, (2) the number of clusters is completely unknown, and (3) equivalent references can be merged into a cluster, which still represents a single object. Therefore, agglomerative hierarchical clustering is most suitable for this problem, as it first uses each reference as a cluster, and then repeatedly merges the most similar pairs of clusters.

A most important issue is how to measure the similarity between two clusters of references. There are different measures including Single-Link, Complete-Link, and Average-Link [8]. Because references to the same object may form weakly linked partitions, Complete-Link is not appropriate. On the other hand, references to different objects may be linked, which makes Single-Link inappropriate. In comparison, Average-Link is a reasonable measure, as it captures the overall similarity between two clusters and is not affected by individual linkages which may be misleading.

Average-Link still suffers from the problem that references to the same object often form weakly linked partitions. For example, in DBLP an author may collaborate with different groups of people when she is affiliated with different institutions. When these partitions are large, the Average-Link similarity may be small even if there are many linkages between them. To address this problem, we combine Average-Link with the *collective random walk probability* between two clusters, which is the probability of walking from one cluster of references to the other cluster. In details, we adopt a composite similarity measure by combining the average set resemblance similarity with the collective random walk probability when measuring similarity between clusters. Because these two measures may have different scales, and arithmetic average will often ignore the smaller one, we use the geometric average of the two measures as the overall similarity between two clusters.

$$Sim(C_1, C_2) = \sqrt{Resem(C_1, C_2) \cdot WalkProb(C_1, C_2)}, \quad (2)$$

where $Resem(C_1, C_2)$ is the average set resemblance similarity between references in C_1 and those in C_2, and $WalkProb(C_1, C_2)$ is the collective random walk probability between them.

4.2 Computing Clusters

Initially each reference is used as a cluster, and the set resemblance similarity and random walk probability between each pair of clusters are computed. This is usually affordable because the number of references having identical names is seldom very large. At each step, the two most similar clusters C_1 and C_2 are merged into a new cluster C_3, and we need to compute the similarity between C_3 and each existing cluster C_i. When C_3 is very large, a brute-force method may consume similar amount of time as computing pair-wise similarity during initialization, and it is unaffordable to perform such computation at every step.

To address this problem, we design efficient methods that can incrementally compute the similarity between clusters as clusters are merged. One important observation for improving efficiency is that, both the average set resemblance similarity and random walk probability between C_3 and C_i can be directly computed by aggregating the similarities between C_1, C_2 and C_i.

5 Experimental Results

We report our empirical study on testing the effectiveness of the proposed approach. DISTINCT is tested on DBLP database, whose schema is shown in Fig. 2. First, authors with no more than 2 papers are removed, and there are 127,124 authors left. There are about 616K papers and 1.29M references to authors in *Publish* relation (authorship). In DBLP we focus on distinguishing references to authors with identical names.

We first build a training set using the method in Section 3, which contains 1000 positive and 1000 negative examples. Then SVM with linear kernel is applied. The whole process takes 62.1 seconds. We measure the performance of DISTINCT by precision, recall, and f-measure. Suppose the standard set of clusters is C^*, and the set of clusters by DISTINCT is C. Let TP (true positive) be the number of pairs of references that are in the same cluster in both C^* and C. Let FP (false positive) be the number of pairs of references in the same cluster in C but not in C^*, and FN (false negative) be the number of pairs of references in the same cluster in C^* but not in C.

$$precision = \frac{TP}{TP + FP}, \quad recall = \frac{TP}{TP + FN}.$$

f-measure is the harmonic mean of precision and recall.

We test DISTINCT on real names in DBLP that correspond to multiple authors. 10 such names are shown in Table 1, together with the number of authors and number of references. For each name, we manually divide the references into groups according to the authors' identities, which are determined by the authors' home pages or affiliations shown on the papers.[1]

[1]References whose author identities cannot be found (*e.g.*, no elec-

Name	#author	#ref	Name	#author	#ref
Hui Fang	3	9	Bing Liu	6	89
Ajay Gupta	4	16	Jim Smith	3	19
Joseph Hellerstein	2	151	Lei Wang	13	55
Rakesh Kumar	2	36	Wei Wang	14	141
Michael Wagner	5	29	Bin Yu	5	44

Table 1. Names corresponding to multiple authors

We use DISTINCT to distinguish references to each name, with min-sim set to 0.0005. Table 2 shows the performance of DISTINCT for each name. In general, DISTINCT successfully group references with high accuracy. There is no false positive in 7 out of 10 cases, and the average recall is 83.6%. In some cases references to one author are divided into multiple groups. For example, 18 references to "Michael Wagner" in Australia are divided into two groups, which leads to low recall.

Name	precision	recall	f-measure
Hui Fang	1.0	1.0	1.0
Ajay Gupta	1.0	1.0	1.0
Joseph Hellerstein	1.0	0.810	0.895
Rakesh Kumar	1.0	1.0	1.0
Michael Wagner	1.0	0.395	0.566
Bing Liu	1.0	0.825	0.904
Jim Smith	0.888	0.926	0.906
Lei Wang	0.920	0.932	0.926
Wei Wang	0.855	0.814	0.834
Bin Yu	1.0	0.658	0.794
average	0.966	0.836	0.883

Table 2. Accuracy for distinguishing references

We compare 6 versions: (1) DISTINCT, (2) DISTINCT without supervised learning, and (3-6) DISTINCT using each of the two similarity measures: Set-resemblance [1] and random walk probabilities [9] (with and without supervised learning). Note that supervised learning is not used in [1] and [9]. For each approach except DISTINCT, we choose the min-sim that maximizes average accuracy. Fig. 4 shows the average f-measure of each approach. DISTINCT leads by about 15% compared with the approaches in [1] and [9]. The f-measure is improved by more than 10% with supervised learning, and 3% with combined similarity measure.

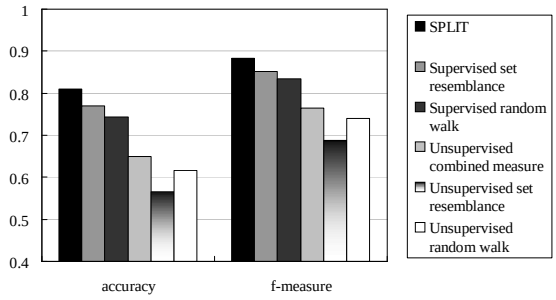

Figure 4. Accuracy and f-measure on real cases

tronic version of paper) are removed. We also remove authors with only one reference that is not related to other references by coauthors or conferences, because such references will not affect accuracy.

We visualize the results about "Wei Wang" in Fig. 5. References corresponding to each author are shown in a gray box, together with his/her current affiliation and number of references. The arrows and small blocks indicate the mistakes made by DISTINCT. It can be seen that in general DISTINCT does a very good job in distinguishing references, although it makes some mistakes because of the linkages between references to different authors.

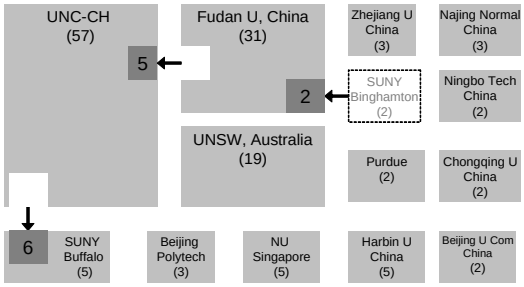

Figure 5. Groups of references of "Wei Wang"

6 Conclusions

In this paper we study the problem distinguishing references to objects with identical names. We develop a general methodology called DISTINCT for supervised composition of heterogeneous link analysis, that can fuse different types of linkages and use a combination of distinct similarity measures to assess the value of each linkage. Our experiments show that DISTINCT can accurately distinguish different objects with identical names in real databases.

References

[1] I. Bhattacharya, L. Getoor. Relational Clustering for Multi-type Entity Resolution. *MRDM workshop*, 2005.

[2] M. Bilenko, R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. *SIGKDD*, 2003.

[3] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–168, 1998.

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

[5] DBLP Bibliography. www.informatik.uni-trier.de/∼ley/db/

[6] I. Felligi, A. Sunter. A theory for record linkage. *Journal of the American Statistical Society*, 64, 1969.

[7] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, D. Srivastava. Approximate string joins in a database (almost) for free. *VLDB*, 2001.

[8] A. K. Jain, M. N. Murty, P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 1999.

[9] D. V. Kalashnikov, S. Mehrotra, Z. Chen. Exploiting Relationships for Domain-Independent Data Cleaning. *SDM*, 2005.

[10] P. N. Tan, M. Steinbach, V. Kumar. Introduction to data mining. Addison-Wesley, 2005.

[11] Y. Weiss. Correctness of Local Probability Propagation in Graphical Models with Loops. *Neural Computation* 12, 2000.

[12] W. Winkler. The state of record linkage and current research problems. Stat. Research Div., U.S. Bureau of Census, 1999.