

Query-Time Entity Resolution

Indrajit Bhattacharya
University of Maryland,
College Park
MD, USA 20742
indrajit@cs.umd.edu

Lise Getoor
University of Maryland,
College Park
MD, USA 20742
getoor@cs.umd.edu

Louis Licamele
University of Maryland,
College Park
MD, USA 20742
licamele@cs.umd.edu

ABSTRACT

The goal of entity resolution is to reconcile database references corresponding to the same real-world entities. Given the abundance of publicly available databases where entities are not resolved, we motivate the problem of quickly processing queries that require resolved entities from such ‘unclean’ databases. We propose a two-stage collective resolution strategy for processing queries. We then show how it can be performed on-the-fly by adaptively extracting and resolving those database references that are the most helpful for resolving the query. We validate our approach on two large real-world publication databases where we show the usefulness of collective resolution and at the same time demonstrate the need for adaptive strategies for query processing. We then show how the same queries can be answered in real time using our adaptive approach while preserving the gains of collective resolution.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval — Clustering, Query formulation

General Terms: Algorithms

Keywords: entity resolution, relations, query, adaptive

1. INTRODUCTION

Entity resolution is a practical problem that comes up in data mining applications in a variety of ways. It is studied as the data cleaning problem of ‘deduplication’, where the goal is to identify and consolidate pairs of records or references within the same relational table that are duplicates of each other. It is also important in data integration as the ‘fuzzy match’ problem, where tuples from two heterogeneous databases with different keys, and possibly different schemas, need to be matched and consolidated.

In spite of the widespread research interest and the practical nature of the problem, many publicly accessible databases remain unresolved, or partially resolved, at best. The popular publication databases, CiteSeer and PubMed, are rep-

resentative examples. CiteSeer contains several records for the same paper or author, while author names in PubMed are not resolved at all. This is due to a variety of reasons, ranging from rapid and often uncontrolled growth of the databases and the computational and other expenses involved. Yet, millions of users access and query such databases everyday, mostly seeking information that, implicitly or explicitly, requires knowledge of the resolved entities. The information gathered from such databases would be significantly more useful or accurate if the entities were resolved.

The abundance of such important and unresolved public databases motivates us to formulate the problem of query-time entity resolution. The goal is to enable users to query an unresolved or partially resolved database and resolve the *relevant* entities on the fly. A user may access several databases everyday and he does not want to clean every database that he queries. He only needs to resolve those entities that matter for his query. For instance, when looking for all books by ‘Stuart Russell’ in CiteSeer, it is not useful to resolve all other author references in CiteSeer. Also, the resolution needs to be quick, even if it is not entirely accurate.

Though entity resolution queries have not been addressed in the literature, there has been significant progress on the general entity resolution problem. Recent research has focused on the use of additional relational information between database references to improve resolution accuracy [2, 15, 6, 1, 10]. This improvement is made possible by resolving related references or records jointly, rather than independently. Intuitively, this corresponds to the notion that figuring out that two records refer to the same underlying entity may in turn give us useful information for resolving other record pairs that are related. While it has been shown that collective resolution significantly improves entity resolution accuracy, the added improvement comes at a considerable computation cost arising from the dependencies. This added computational expense makes its application in query-time resolution challenging. Due to its inter-dependent nature, the set of references that influence collective resolution of a query may be very large. In this paper, we present adaptive algorithms for extracting the most relevant references for a query that enable us to resolve entities at query-time, while preserving the gains of collective resolution.

Our specific contributions in this paper are as follows. First, we motivate and formulate the problem of query-time entity resolution. Our entity resolution approach is based on a relational clustering algorithm. To the best of our knowledge, clustering based on queries in the presence of relations

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD’06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.
Copyright 2006 ACM 1-59593-339-5/06/0008 ...\$5.00.

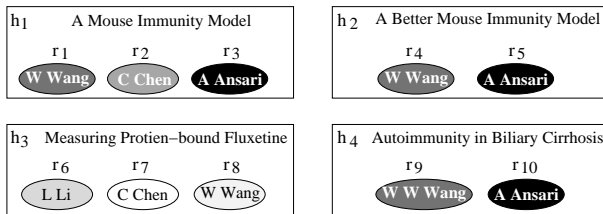


Figure 1: An example set of papers represented as references connected by hyper-edges. References are shaded according to their entities.

has received little attention. We also formulate query-time clustering as a resource-constrained problem and propose adaptive strategies for constructing the set of references that influence a query. Finally, we present experimental results on large real-world datasets where our strategy enables collective resolution in seconds with minimal loss in accuracy.

The rest of the paper is organized as follows. In Section 2, we formalize the relational entity resolution problem and the concept of entity resolution queries. In Section 3, we briefly review the relational clustering algorithm that we employ for collective entity resolution. Next, in Section 4, we describe an unconstrained strategy for extracting the references relevant for collectively resolving a query, and in Section 5 we present our adaptive algorithm for resolving queries under resource constraints. We present experimental results in Section 6, review related work in Section 7 and finally conclude in Section 8.

2. ENTITY RESOLUTION: FORMULATION

In the simplest formulation, we have a collection of references, $\mathcal{R} = \{r_i\}$, with attributes $\{\mathcal{R}.A_1, \dots, \mathcal{R}.A_k\}$. Let $\mathcal{E} = \{e_j\}$ be the unobserved domain entities. For any particular reference r_i , we denote the entity to which it maps as $E(r_i)$. We will say that two references r_i and r_j are *co-referent* if they correspond to the same entity, $E(r_i) = E(r_j)$. Note however that the database is unresolved, i.e. the mapping $E(r_i)$ is *not provided*. Further, the domain entities \mathcal{E} and even the number of such entities is not known. However, we may have information about relationships between the references. To model relationships in a generic way, we use a hyper-edge set \mathcal{H} with possible attributes $\{\mathcal{H}.A_1 \dots \mathcal{H}.A_l\}$. Each hyper-edge connects multiple references. To capture this, we associate a set of references $\mathcal{H}.R$ with each hyper-edge. Note that each reference may be associated with zero or more hyper-edges.

Let us now look at a sample domain to see how it can be represented in our framework. Consider a database of academic publications similar to DBLP, CiteSeer or PubMed. Each publication in the database has a set of author names, each of which is a reference r_i in \mathcal{R} . For each reference, $r_i.Name$ records the observed name of the author in the publication. In addition, we can have attributes such as $\mathcal{R}.Email$ to record other information for each author reference that may be available in the paper. Also, each publication represents a co-author relationship among the references in it. So we have a hyper-edge $h_i \in \mathcal{H}$ for each publication and $r_j \in h_i.R$ for each reference r_j in the publication. If publications have information such as title, keywords, etc, they are represented as attributes of \mathcal{H} .

To illustrate, consider the following four papers, which we will use as a running example:

1. W. Wang, C. Chen, A. Ansari, “A mouse immunity model”
2. W. Wang, A. Ansari, “A better mouse immunity model”
3. L. Li, C. Chen, W. Wang, “Measuring protein-bound fluxetine”
4. W. W. Wang, A. Ansari, “Autoimmunity in biliary cirrhosis”

To represent them in our notation, we have 10 references $\{r_1, \dots, r_{10}\}$ in \mathcal{R} , where $r_1.Name = \text{‘W Wang’}$, etc. There are 4 hyper-edges $\{h_1, \dots, h_4\}$ in \mathcal{H} for the four papers. This is represented pictorially in Figure 1.

Given this formulation, the **entity resolution task** is defined as the partitioning or clustering of the references according to the underlying entity-reference mapping $E(r)$. To illustrate, assume that we have six underlying entities. This is illustrated in Figure 1 using a different shading for each entity. For example, the ‘Wang’s of papers 1, 2 and 4 are the same individual but that from paper 3 is a different person. Also, the ‘Chen’s from papers 1 and 3 are different individuals. Then, the correct resolution for our example database with 10 references returns 6 entity clusters: $\{\{r_1, r_4, r_9\}, \{r_8\}, \{r_2\}, \{r_7\}, \{r_3, r_5, r_{10}\}, \{r_6\}\}$. The first two clusters correspond to ‘Wang’, the next two to ‘Chen’, the fifth to ‘Ansari’ and the last to ‘Li’.

Instead of clustering *all* database references, in many applications, users are interested in just a few of the clusters. For example, we may want to retrieve all papers written by some person named ‘W Wang’. We will call this an **entity resolution query** on ‘W Wang’, since answering it involves knowing the underlying entities. We will assume that queries are specified using $\mathcal{R}.Name$, which is a noisy identifier for entities. Since names are ambiguous, treating them as identifiers leads to undesirable results. For example, it would be incorrect to return the set $\{r_1, r_4, r_8\}$ of all references with name ‘W Wang’ as the answer to our query. This answer does not indicate that r_8 is not the same person as the other two. Also, the answer should include the paper by ‘W W Wang’ (r_9), who is the same entity as the author of the first paper. Therefore, the correct answer to the entity resolution query on ‘W Wang’ should be the partition $\{\{r_1, r_4, r_9\}, \{r_8\}\}$.

Different approaches employed for the general entity resolution problem can also be used for entity resolution queries. In traditional **attribute-based entity resolution**, similarity is computed for each pair of references based on their attributes and only those pairs that have similarity above some threshold are considered to be co-referent. This often runs into problems. In our example, it is hard to infer with just attributes that references r_1 and r_8 are not co-referent although they have the same name, while r_1 and r_9 are co-referent although their names are different. When relations between references are available, the **naive relational entity resolution** approach additionally considers the attributes of the related references when computing similarity between pairs of references. For example, when computing the similarity between ‘W. Wang’ and ‘W. W. Wang’, it takes into account that both have co-authors with name ‘A. Ansari’. This also can be misled in many cases. For instance, the two ‘W. Wang’ references r_1 and r_8 are not co-referent, though they both have co-authors with name ‘C. Chen’. The correct evidence to use here is that the ‘Chen’s are not co-referent either. Therefore, in order to resolve the ‘W. Wang’ references, it is necessary to *resolve* the ‘C. Chen’ references as well, and not just consider their attributes. This is the

goal of **collective entity resolution**, which improves accuracy but is harder to solve because the references cannot be clustered independently. Instead, any resolution decision is affected by other resolutions through hyper-edges.

3. RELATIONAL CLUSTERING

Given that the goal of entity resolution is to cluster the database references according to their entities, we have developed a relational clustering algorithm for entity resolution (**RC-ER**) [2]. Given a current set of reference clusters $\mathcal{C} = \{c_i\}$, it iteratively merges the pair of clusters that are the most similar. We associate a cluster label $r.C$ with each reference to denote its current cluster membership. Recall that for the attribute-based approach, the similarity measure only considers the attributes of references. For the naive relational approach, it additionally considers the attributes of related references. The collective approach, in contrast, considers the cluster labels of the related references and the similarity of two clusters c_i and c_j is defined as

$$sim(c_i, c_j) = (1 - \alpha) \times sim_A(c_i, c_j) + \alpha \times sim_R(c_i, c_j)$$

where sim_A is the attribute similarity and sim_R the relational similarity between the references in the two clusters with combination weight α ($0 \leq \alpha \leq 1$). The interesting aspect of the collective approach is the dynamic nature of the similarity. The similarity between two references depends on their *current* cluster labels and therefore changes with the labels. In our example, the similarity of the two references ‘W. Wang’ and ‘W. W. Wang’ increases once the ‘Ansari’ references are given the same cluster label. Let us now see how the two components of the similarity are computed.

Attribute Similarity: For each reference attribute, we assume the existence of some basic similarity measure that takes two reference attributes and returns a value between 0 and 1 that indicates the degree of similarity between them. In addition, if hyper-edges have attributes, then the attribute similarity of two references can also take into account the attributes of the hyper-edges with which they are associated. Several sophisticated similarity measures have been developed for names, and popular TF-IDF schemes may be used for other textual attributes such as keywords. The measure that works best for each attribute may be plugged in. Finally, a weighted combination of the similarities over the different attributes yields the combined attribute similarity between two reference clusters.

Relational Similarity: For collective entity resolution, relational similarity considers the cluster labels of the references that each cluster is connected to via the hyper-edges. There are many possible ways to define this similarity; we use a variant that we have proposed earlier [2, 3].

The hyper-edges relevant for a cluster are the hyper-edges for all its references. Recall that each reference r is associated with one or more hyper-edges in \mathcal{H} . Therefore, the hyper-edge set $c.H$ for an entity cluster c is defined as

$$c.H = \bigcup_{r \in \mathcal{R} \wedge r.C=c} \{hid \mid (hid, rid) \in \mathcal{H} \wedge r.id = rid\}$$

This set defines the hyper-edges that connect a cluster c to other clusters, and are the ones that relational similarity needs to consider. For instance, when all the references in our running example have been correctly clustered as in Figure 1(b), the hyper-edge set for the larger ‘Wang’ cluster is

$\{h_1, h_2, h_4\}$, which are the hyper-edges associated with the references r_1 , r_4 and r_9 in that cluster.

The different clusters to which any entity cluster c is connected via its hyper-edge set is called the neighborhood $Nbr(c)$ of that cluster c .

$$Nbr(c) = \bigcup_{h \in c.H, r \in h} \{c_j \mid c_j = r.C\}$$

For our example ‘Wang’ cluster, its neighborhood consist of the ‘Ansari’ cluster and one of the ‘Chen’ clusters, which are connected by its edge-set. Now, for the relational similarity measure between two entity clusters, their neighborhoods are compared using a set similarity measure, such as Jaccard similarity:

$$sim_R(c_i, c_j) = Jaccard(Nbr(c_i), Nbr(c_j))$$

Recall that for two sets A and B , their Jaccard similarity is defined as $Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$. The similarity can be computed and updated very efficiently, in time that is linear in the average number of neighbors per cluster.

Clustering Algorithm: Given the similarity measure for a pair of clusters, a greedy agglomerative clustering algorithm is used for collective entity resolution. The algorithm bootstraps the clusters, identifies the candidate set of potential duplicates and iterates over the following steps. At each step, it identifies the currently ‘closest pair’ of clusters (c_i, c_j) from the candidate set and merges them to create a new cluster c_{ij} . It identifies new candidate pairs and updates the similarity measures for the ‘related’ cluster pairs. All of these tasks are performed efficiently using an indexed priority queue. The algorithm terminates when the similarity for the closest pair falls below a threshold.

4. ENTITY RESOLUTION QUERIES

As we have seen, for entity resolution queries, the answer includes only a few of the entity clusters for the database. Then it is clearly unnecessary to resolve all database references for any query. However, for collective resolution, correctly resolving the relevant entities for a query may involve resolving neighboring entities as well. To address this, we propose a two-phase query processing strategy consisting of an *extraction phase* followed by a *resolution phase*. In the extraction phase, the goal is to extract the relevant set of references $Rel(Q)$ for answering the query Q accurately and then, in the resolution phase, we perform collective resolution on $Rel(Q)$.

We introduce two expansion operators for constructing the relevant set for an entity resolution query $Q(n)$. The first operator is the **name expansion operator** X_N or *n-expansion* for short. For a name n , $X_N(n)$ returns all references whose names exactly match that name or are ‘similar’ to it. Similar names can be determined by blocking techniques [12]. For a query $Q(n)$, we first need to find all references that can potentially be included in the answer. This base level of references can be retrieved by expanding the name n as $Rel^0(Q(n)) = X_N(n)$. The first step in Figure 2 shows *n-expansion* on ‘W Wang’ in our example.

The second operator is **hyper-edge expansion** X_h , or *h-expansion*. For any reference r , $X_h(r)$ returns all references that share a hyper-edge with it. For collective entity resolution, we need to consider all related references for each references. Therefore, we need to perform *h-expansion* on

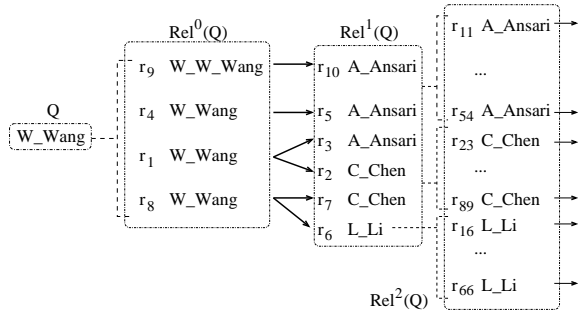


Figure 2: Relevant set for query ‘W. Wang’ using h-expansion and n-expansion alternately

the references in $Rel^0(Q(n))$. Figure 2 illustrates this operation in our example. The interesting aspect of collective resolution is that we cannot stop here. We will need to *resolve* the references that we so obtained, and this requires n-expanding these new references. This suggests a recursive growth of the relevant set. Formally, for a query $Q(n)$, the expansion process alternates between n-expansion and h-expansion:

$$Rel^i(Q(n)) = \begin{cases} X_N(n) & \text{for } i = 0 \\ X_H(Rel^{i-1}(Q(n))) & \text{for odd } i \\ X_N(Rel^{i-1}(Q(n))) & \text{for even } i \end{cases}$$

The improvement in resolution accuracy for $Q(n)$ falls off quickly with expansion depth, so we can terminate the expansion process at some cut-off depth d^* : $Rel(Q) = \bigcup_{i=0}^{d^*} Rel^i(Q)$. Also, the size of the relevant set can be significantly reduced by restricting name expansion to **exact n-expansion** X_N^e that only considers references with exactly the same name. Interestingly, we can show that the restricted strategy that alternates between exact n-expansion and h-expansion does not affect recall significantly.

5. ADAPTIVE QUERY EXPANSION

The query expansion strategy from the previous section is unconstrained in that it blindly expands all references in the current relevant set and also includes all new references generated by an expansion operation. However, for many domains the size of the relevant set resulting from such unconstrained expansion is prohibitive for query-time resolution even for small expansion depths. Given the limited time to process a query, our solution is to include the references that are most helpful for resolving the query. To illustrate using our example from Figure 2, observe that ‘Chen’ and ‘Li’ are significantly more common or ‘ambiguous’ names than ‘Ansari’ — even different ‘W. Wang’ entities are likely to have collaborators named ‘Chen’ or ‘Li’. Therefore, when h-expanding $Rel^0(Q)$ for ‘W. Wang’, ‘Ansari’ is more informative than ‘Chen’ or ‘Li’. Similarly, when n-expanding $Rel^1(Q)$, we can choose not to expand the name ‘A. Ansari’ any further, since two ‘A. Ansari’ references are very likely to be coreferent. But we need more evidence for the ‘Chen’s and ‘Li’s. To describe this formally, the ambiguity of a name n is the probability that any two references r_i and r_j in the database that have this name ($r_i.Name = r_j.Name = n$) are *not* coreferent: $Amb(n) = P(E(r_i) \neq E(r_j))$. The goal of adaptive expansion is to add less ambiguous references to the relevant set and, of the references currently in the

relevant set, expand the most ambiguous ones.

For **adaptive hyper-edge expansion**, we set an upper-bound h_{max} on the number of new references that h-expansion at a particular level can generate. Formally, we want $|X_H(Rel^i(Q))| \leq h_{max}|Rel^i(Q)|$. The value of h_{max} may depend on depth i but it is small enough to rule out full h-expansion of the current relevant set. Then, given h_{max} , our strategy is to choose the least ambiguous references from $X_H(Rel^i(Q))$, since they provide the most informative evidence for resolving the references in $Rel^i(Q)$. We sort the h-expanded references in increasing order of ambiguity and select the first k from them, where $k = h_{max}|Rel^i(Q)|$.

$$Rel_A^i(Q, h_{max}) = LeastAmb(k, X_H(Rel^{i-1}(Q))) \quad (1)$$

The setting for **adaptive name expansion** is very similar. For some positive number n_{max} , exact n-expansion of $Rel^i(Q)$ is allowed to include at most $n_{max}|Rel^i(Q)|$ references. Note that now the selection preference needs to be flipped — more ambiguous names need more evidence, so they are expanded first. So we can sort $X_N^e(Rel^i(Q))$ in decreasing order of ambiguity and select the first k from the sorted list, where $k = n_{max}|Rel^i(Q)|$. But this could potentially retrieve only references for the most ambiguous name, totally ignoring references with any other name. To avoid this, we choose the top k ambiguous references from $Rel^i(Q)$ before expansion, and then expand the references so chosen.

$$Rel_A^i(Q, n_{max}) = X_N^e(MostAmb(k, Rel^i(Q))) \quad (2)$$

Though this cannot directly control the number of new references added, $\mu_r \times k$ is a reasonable estimate, where μ_r is the average number of references per name.

The adaptive expansion scheme proposed in this section is crucially dependent on the estimates of name ambiguity. We now describe one possible scheme that worked quite well. Recall that we want to estimate the probability that two randomly picked references with $Name = n$ correspond to different entities. For any single valued attribute $\mathcal{R}.A$ of the underlying entity, a naive unsupervised estimate of $Amb_A(n)$ is the fraction of references having $A = n$. This estimate is clearly not good since the number of references with a certain $Name$ does not always match the number of different entities for that $Name$. However, we can do much better if we have an additional attribute $\mathcal{R}.A_1$ that also records a single valued attribute of the underlying entities. Given A_1 , ambiguity of $A = n$ can be estimated as $Amb_A(n) = |A_1|_n^A / |A_1|_*^A$ where $|A_1|_x^A$ is the number of different values observed for A_1 in references r with $r.A = x$. For example, we can estimate the ambiguity of a last name by counting the different first names observed for it, since last name and first name are both single valued attributes of the underlying people, and the two are not correlated. In fact, when multiple such uncorrelated single valued attributes $\mathcal{R}.A_i$ are available, this approach can be generalized to obtain even better estimates of ambiguity.

6. EXPERIMENTAL RESULTS

We experimented on two datasets to evaluate our query-time resolution strategy. The first dataset, **arXiv**, contains papers from high energy physics and was used in KDD Cup 2003¹. It has 58,515 references to 9,200 authors, contained in 29,555 publications. Our second dataset is the **Elsevier**

¹<http://www.cs.cornell.edu/projects/kddcup/index.html>

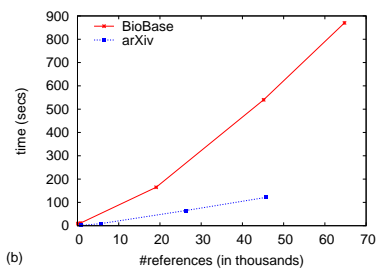


Figure 3: Execution time of RC-ER for increasing number of references.

BioBase database² of publications from biology used in the recent IBM KDD-Challenge competition. It contains 156,156 publications with 831,991 author references. Apart from the size difference, the average number of author names per paper is 5.3 for BioBase, as compared against 1.9 for arXiv. Also, unlike arXiv, BioBase includes keywords, topic classification, language, country of correspondence and affiliation of the corresponding author as attributes of the each paper, which we use as attributes for resolution in addition to author names.

For entity resolution queries in arXiv, we selected all ambiguous names that correspond to more than one author entity. This gave us 75 queries with the number of true entities for each varying from 2 to 11 (average 2.4). For BioBase, we query the top 100 author names with the highest number of references. The average number of references for each of these 100 names is 106. The number of entities for each name ranges from 1 to 100 (average 32), thereby providing a wide variety of entity resolution settings over the queries.

We first explore the growth rate of the relevant set over expansion depth for a sample query in each of the two datasets. The growth rate for the arXiv query is moderate. The number of relevant references is 7 at depth 0, and grows to 7,500 at depth 7. In contrast, for BioBase the growth is quite dramatic. The relevant set size grows from 84 at depth 0 to 586,000 by depth 5 for name similarity expansion and to 384,000 for exact expansion. The growth rates for these two samples from arXiv and BioBase are typical for all of our queries in these two datasets.

Next, Figure 3(b) shows how the relational clustering algorithm **RC-ER** scales with number of references. All execution times are reported on a Dell Precision 870 server with 3.2GHz Intel Xeon processor and 3GB of memory. The plot shows that the algorithm scales linearly with increasing references, but the gradient is different for the two datasets mainly due to the difference in the average number of references per hyperlink. This suggests that **RC-ER** is well-suited for query-time resolution for arXiv. But for BioBase, it would require up to 600 secs for 40,000 references.

In our next experiment, we evaluate several algorithms for entity resolution queries. We compare entity resolution accuracy of the pair-wise co-reference decisions using the F1 measure (which is the harmonic mean of precision and recall). For a fair comparison, we consider the best F1 for each of these algorithms over all possible thresholds for determining duplicates. For the algorithms, we compare *attribute-based entity resolution (A)*, *naive relational entity resolution*

that uses *attributes* of related references (**A+N**), and our relational clustering algorithm (**RC-ER**) for *collective entity resolution* using unconstrained expansion up to depth 3. We also consider transitive closures over the pair-wise decisions for the first two approaches (**A*** and **A+N***). For attribute similarity, we use the *Soft TF-IDF* with Jaro-Winkler similarity for names, which has been shown to perform the best for name-based resolution [4], and TF-IDF similarity for the other textual attributes.

Table 1: Entity resolution accuracy (F1) for different algorithms over 75 arXiv queries and 100 BioBase queries

| | arXiv | BioBase |
|----------------------|-------|---------|
| A | 0.721 | 0.701 |
| A* | 0.778 | 0.687 |
| A+N | 0.956 | 0.710 |
| A+N* | 0.952 | 0.753 |
| RC-ER Depth-1 | 0.964 | 0.813 |
| RC-ER Depth-3 | 0.970 | 0.820 |

The average F1 scores over all queries are plotted in Table 1 for each algorithm in the two datasets. It shows that **RC-ER** improves accuracy significantly over the baselines. For example in BioBase, the improvement is 21% over **A** and **A+N**, 25% over **A*** and 13% over **A+N***. This validates the potential benefits of collective resolution, as shown by recent research [2, 15, 6, 13] in the context of offline cleaning, and motivates its application for query-time entity resolution. Significantly, most of the accuracy improvement comes from the depth-1 relevant references. For 56 out of the 100 BioBase queries accuracy does not improve beyond the depth-1 relevant references and for the remaining the average improvement is 2%. However, for 8 of the most ambiguous queries, accuracy improves by more than 5%, the biggest improvement being as high as 27% (from 0.67 to 0.85 F1). Such instances are fewer for arXiv, but the biggest improvement is 37.5% (from 0.727 to 1.0). This suggests that while there are potential benefits to looking at greater depths, the benefits fall off quite quickly on average beyond depth 1.

The first set of experiments show the benefits of **RC-ER**. Next, we measure the processing times over unconstrained relevant sets up to depth 3 for all queries in the two datasets. For arXiv, the average processing time of 1.6 secs (with 406 references in the relevant set on average) is quite acceptable. However, it is more than 10 minutes for BioBase (avg. relevant set size is 44,129), which clearly necessitates adaptive strategies for relevant set construction.

Finally, we investigate the effectiveness of our adaptive expansion strategy on BioBase. For estimating ambiguity of references, we use last names with first initial as the secondary attribute. This resulted in very good estimates of ambiguity — the ambiguity estimate for a name is strongly correlated (correlation coeff. 0.8) with the number of entities for that name. For each of the 100 queries, we construct the relevant set $Rel(Q)$ with $d^* = 3$ using adaptive h-expansion and adaptive exact n-expansion. Since most of the improvement from collective resolution comes from depth-1 references, we consider two different experiments. In the first, we use adaptive expansion only at depths 2 and

²<http://help.sciencedirect.com/robo/projects/sdhelp/about.Biobase.htm>

beyond (AX-2) and unconstrained h-expansion at depth 1. In the second(AX-1), we use adaptive h-expansion even at depth 1, with $h_{max} = 6$. For both of them, we use adaptive expansion at higher depths 2 and 3 with parameters $h_{max} = 3$ at 3 and $n_{max} = 0.2$ at 2.

Table 2: Comparison between unconstrained and adaptive expansion for 100 BioBase queries

| | Unconstr. | AX-2 | AX-1 |
|---------------|-----------|----------|----------|
| relv-set size | 44,129.5 | 5,510.52 | 3,743.52 |
| time (secs) | 606.98 | 43.44 | 31.28 |
| accuracy (F1) | 0.821 | 0.818 | 0.820 |

In Table 2, we compare the two adaptive schemes against unconstrained expansion with $d^* = 3$ over all queries. Clearly, accuracy remains almost unaffected for both schemes. First, we note that AX-2 matches the accuracy of unconstrained expansion and shows almost the same improvement over depth 1 even though it n-expands a small fraction of $Rel^1(Q)$ — the average size of the relevant set reduces to 5,500 from 44,000. More significantly, AX-1 also matches this improvement even without including many depth-1 references. This reduction in the size of the relevant set has an immense impact on the query processing time. The average processing time drops from more than 600 secs for unconstrained expansion to 43 secs for AX-2 and further to just 31 secs for AX-1, thus making it possible to use collective entity resolution for query-time resolution.

As a further improvement, we investigate setting expansion depth d^* adaptively. In a simple setup, we set d^* to 1 for queries where the number of different first initials for a last name is less than 10 (out of 26), and explore depth 2 only for more ambiguous queries. This reduced the average processing time for 18 of the 100 queries by 35% to 11.5 secs from 17.7 secs with no reduction in accuracy. In a more general setting, where a bigger fraction of queries have lower ambiguity, the impact is expected to be even more significant.

7. RELATED WORK

The entity resolution problem has been studied in many different areas under different names. Much of the work has focused on traditional attribute-based entity resolution. Extensive research has been done on defining approximate string similarity measures [14, 4] that may be used for unsupervised entity resolution. Efficiency has been an important issue in data cleaning [9, 12] and probabilistic techniques have also been proposed for efficiently looking up candidate matches for incoming tuples [5].

Many recently proposed approaches take relations into account for data integration [2, 3, 6, 10, 1]. Dong et al.[6] collectively resolve entities of multiple types by propagating relational evidences in a dependency graph, while Ananthakrishna et al.[1] leverage on a dimensional hierarchy over the relations. Some of these approaches have been shown to be scalable, but the focus has not been on query-time cleaning.

Probabilistic models for collective entity resolution have been applied to named entity recognition and for citation matching [13, 11, 15, 16]. While these perform well, they have mostly been useful for small datasets and probabilistic

inference for relational data is not known to be scalable in practice. Approaches have been proposed for localized evaluation of Bayesian networks [7], but not for clustering problems, which is our approach for entity resolution. Adaptive machine learning approaches have been proposed for data integration [17], where active learning requires the user to label informative examples. As we do, Fuxman et al. [8] motivate the problem of querying databases that violate integrity constraints. However, the relational aspect of the problem does not come up in their setting.

8. CONCLUSIONS

In this paper, we have motivated the problem of query-time entity resolution for accessing unresolved third-party databases. The biggest issue in query-time resolution of entities is reducing the computational expense of collective resolution while maintaining its benefits in terms of resolution accuracy. We propose an adaptive strategy for extracting the set of most relevant references for collectively resolving a query. We demonstrate that this adaptive strategy preserves the accuracy of unconstrained expansion while dramatically reducing the number of relevant references, thereby enabling query-time collective resolution. While we have presented results for bibliographic data, the techniques are applicable in other relational domains. Future research directions include exploring stronger coupling between the extraction and resolution phases of query processing and investigating localized resolution for offline data cleaning as well.

9. REFERENCES

- [1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, 2002.
- [2] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *SIGMOD Workshop on Data Mining and Knowledge Discovery*, 2004.
- [3] I. Bhattacharya and L. Getoor. *Entity Resolution in Graphs*, chapter Entity Resolution in Graphs. Wiley, 2006.
- [4] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5), 2003.
- [5] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD*, 2003.
- [6] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.
- [7] D. Draper and S. Hanks. Localized partial evaluation of belief networks. In *UAI*, 1994.
- [8] A. Fuxman, E. Fazli, and R. Miller. Conquer: Efficient management of inconsistent databases. In *SIGMOD*, 2005.
- [9] M. Hernández and S. Stolfo. The merge/purge problem for large databases. In *SIGMOD*, 1995.
- [10] D. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM SDM*, 2005.
- [11] X. Li, P. Morie, and D. Roth. Semantic integration in text: From ambiguous names to identifiable entities. *AI Magazine. Special Issue on Semantic Integration*, 2005.
- [12] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, 2000.
- [13] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS*, 2004.
- [14] A. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *KDD*, 1996.
- [15] Parag and P. Domingos. Multi-relational record linkage. In *KDD Workshop on Multi-Relational Data Mining*, 2004.
- [16] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS*, 2003.
- [17] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *KDD*, 2002.