Relational Clustering for Entity Resolution Queries

Indrajit Bhattacharya INDRAJIT@CS.UMD.EDU
Louis Licamele LICAMELE@CS.UMD.EDU
Lise Getoor GETOOR@CS.UMD.EDU
University of Maryland, College Park, USA

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 first propose a cut-based relational clustering formulation for collective entity resolution. 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.

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.

Presented at the ICML Workshop on Open Problems in Statistical Relational Learning, Pittsburgh, PA, 2006.

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 important 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. Clearly, 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 will 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 books by 'Stuart Russell' in CiteSeer, it is not useful to resolve all author references in there. 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 (Bhattacharya & Getoor, 2004; Singla & Domingos, 2004; Dong et al., 2005). This performance improvement is made possible by resolving related references jointly, rather than independently. Intuitively, this corresponds to the notion that figuring out that two references refer to the same underlying entity may in turn give us useful information

for resolving other 'related' reference pairs. Our first contribution in this paper is a cut-based formulation for entity resolution as a relational clustering problem. This can be optimized using algorithms that we have earlier proposed (Bhattacharya & Getoor, 2004; Bhattacharya & Getoor, 2006) and we demonstrate, as others have done, that collective resolution significantly improves entity resolution accuracy. However, the added improvement comes at a considerable computation cost arising from the dependencies. Due to this added computational expense, its application in query-time resolution is challenging. In this paper, we motivate and formulate the problem of relational clustering based on queries. We also 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.

The rest of the paper is organized as follows. In Section 2, we formalize the relational entity resolution problem. In Section 3, we formulate a cut-based objective function for relational clustering and present a greedy clustering algorithm for collective entity resolution. Next, in Section 4, we describe entity resolution queries and an unconstrained strategy for extracting the references relevant for collectively resolving a query, and in Section 5 we present our adaptive algorithm. 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 $\{\mathcal{R}.A_1, \ldots, \mathcal{R}.A_k\}$. 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 — 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 $\{\mathcal{H}.A_1 \ldots \mathcal{H}.A_l\}$. Each hyper-edge connects multiple references. To capture this, we associate a set of references $\mathcal{H}.R$ with each hyper-edge. Each reference may be associated with zero or more hyper-edges.

Let us now look at a sample domain to see how it can 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, 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, these are represented as attributes of $\mathcal{H}$.

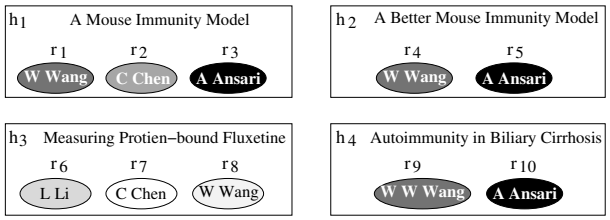

Figure 1. An example set of papers represented as references connected by hyper-edges. References are shaded according to their entities.

To illustrate, consider the following four papers, which we will also 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 $\{r_1, \ldots, r_{10}\}$ in $\mathcal{R}$, where $r_1.Name =$'W Wang', etc. There are 4 hyper-edges $\{h_1, \ldots, h_4\}$ in $\mathcal{H}$ for the four papers. This is represented pictorially in Figure 1.

Given this formulation, the **entity resolution task** can be 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'.

Different approaches may be used for the entity resolution task. In the traditional **attribute-based entity resolution** approach, 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 considers the attributes of the related references when computing similarity between pairs of references. In our running example, when computing the similarity between 'W. Wang' and 'W. W. Wang', it would take into account that both have co-authors with name 'A. Ansari'. This also can be misled when many entities have the same name and the relationship graph is dense. In our example, 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. In such a setting, 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**, where resolutions are not made independently, but instead one resolution decision affects other resolutions via hyperedges.

3. A Cut-based Objective Function

In this section, we propose a cut-based objective function for clustering references according to their entities. Cut-based formulations for clustering are not new. In the traditional clustering problem, we have edges between vertex pairs and the weights represent their similarity or 'closeness'. Then the clustering task of grouping similar vertices together reduces to minimizing the cost of the 'cut', which is the summed weight of all edges that cut across cluster boundaries (Shi & Malik, 2000).

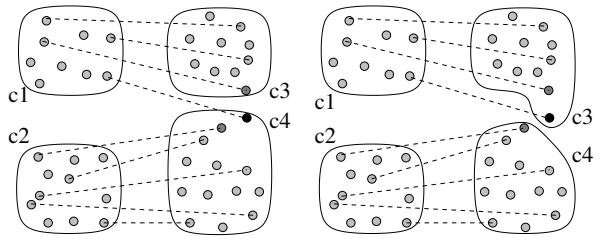

Figure 2. A set of vertices with relational edges and two different ways to partition them into four clusters.

We now motivate a similar cut-based objective function for the general relational clustering problem. As in the traditional case, we have weighted edges between vertex pairs that capture their similarity in terms of attributes. We will call these the attribute edges. In addition, we have relational edges (or, hyperedges) to represent the relationships observed in the data. Figure 2 depicts an example set of vertices. For simplicity, the attribute edges are not shown explicitly. Assume that vertices that are closer in the two-dimensional euclidean space are more similar in terms of their attributes and, accordingly, have higher weights on their attribute edges. The relational edges have been shown explicitly using dashed lines. Suppose that the task is to partition these vertices into four clusters in a way that respects the weights on attribute edges as well as the relational edges. Figure 2 shows two different ways to cluster them. Note they differ only in the cluster membership of the black vertex. The first partitioning is faithful to the attribute edges. The second, in comparison, produces a simpler relational pattern between the clusters. It moves the black vertex to cluster $c3$ from $c4$, since all the other vertices with relational edges to those in cluster $c1$ belong to cluster $c3$.

How do we capture this distinction between these two cases using a cut-based objective function? Counting the total number of cross-cluster relational edges does not work, because it is the same in both cases. The difference is in the *number of cluster pairs* that share relational edges. In the first case, we have three such pairs, $(c1, c3)$, $(c2, c4)$ and $(c1, c4)$. In the second, it is reduced to two by eliminating the $(c1, c4)$ relational connection. Deciding which of these two cases represents a better clustering depends on the attribute penalty of reassigning that one vertex in comparison to the gain in terms of relations.

With this motivation, we propose a cut-based objective function for relational clustering. For any pair of clusters c_i, c_j, let $sim_A(c_i, c_j)$ denote the cross-cluster attribute-edge weight and $\delta_R(c_i, c_j)$ be 1 when c_i and c_j share a relational edge, and 0 otherwise. Then the 'cost' of a particular assignment of the vertices to clusters $\{c_1, \ldots, c_k\}$ is given by

$$\sum_i \sum_j w_A \times sim_A(c_i, c_j) + w_R \times \delta(c_i, c_j) \qquad (1)$$

where w_A and w_R are weights representing the relative importance of attributes and relations respectively. The lower the value of the objective function, the better is the cluster assignment.

Let us now see what this cost function means for the task of relational entity resolution. As in traditional clustering, two author references are more likely to belong to the same entity cluster if they are similar in

their attributes. When we have co-author information available, they can be represented as relational edges. Then, if two authors with reasonably similar names are observed to collaborate with the same other author, we are more inclined to believe that they are the same person. This is exactly what this cost function captures. It finds the minimal number of author-author collaborations to explain the co-author relations and the attribute similarities. However, relational entity resolution enforces some additional constraints. If a co-author relationship exists between two references, they *cannot* be clustered as the same person. A simple modification to the objective function enables us to capture this. Adding an incompatibility $f(c_i, c_j)$ for any two clusters, we get our new objective function:

$$\sum_i \sum_j w_A \times sim_A(c_i, c_j) + w_R \times \delta(c_i, c_j) \times f(c_i, c_j) \quad (2)$$

In this case, we set $f(c_i, c_j)$ to ∞ when $c_i = c_j$ and to 1 otherwise. Note that in general, more complex incompatibilities can be defined. Also, for clustering with relational hyper-edges, Eq. (2) may be generalized to capture more complex relationships across clusters. Instead of counting only related cluster pairs, or 2-cliques, we can also count higher-order cliques. For example, $\delta_R(c_i, c_j, c_k)$ can check if three clusters c_i, c_j and c_k share a relational edge.

For this paper, we focus on an algorithm that minimizes the objective function in Eq. (2). For the entity resolution problem, the number of clusters or entities is hard to specify as a parameter. So we look at an agglomerative clustering algorithm that starts from an initial clustering and iteratively merges clusters pairs in a greedy fashion in terms of reducing the value of the objective function. The iterations continue until the improvement, or the 'gain', falls below a threshold. For any pair of clusters, the gain on merging them can be represented as

$$\Delta(c_i, c_j) = w_A \times \Delta_A(c_i, c_j) + w_R \times \Delta_R(c_i, c_j) \quad (3)$$

The attribute gain Δ_A is essentially $sim_A(c_i, c_j)$. For each attribute in the reference table $\mathcal{R}$, we assume the existence of a $[0, 1]$ similarity measure. If the hyper-edges have attributes, those can also be taken into account. Several sophisticated similarity measures exist for matching names, and popular TF-IDF schemes may be used for other textual attributes like keywords.

The relational gain $\Delta_R(c_i, c_j)$ is the common cluster neighborhood of c_i and c_j. In other words, if $N_R(c_i)$ is the set of all clusters that share relational edges with c_i, then $\Delta_R(c_i, c_j)$ is the intersection of the two sets $N_R(c_i)$ and $N_R(c_j)$. An interesting aspect is the dynamic nature of Δ_R. For any pair of clusters, it changes as their cluster neighborhoods evolve. We have earlier proposed a relational cluster algorithm for collective entity resolution (**RC-ER**) (Bhattacharya & Getoor, 2004; Bhattacharya & Getoor, 2006) where cluster similarity measure looks identical to Eq. (3). The cut-based objective function simply provides a theoretical justification for such an approach. We experimented with different measures for Δ_R and found that Jaccard similarity of the cluster neighborhoods works significantly better than the set intersection size.

4. Entity Resolution Queries

Recall that the entity resolution task involves partitioning all the database references according to their entities. However, in many applications, users are interested in just a few of the clusters rather than all of them. 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 requires knowledge of 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 $\{\{r_1, r_4, r_9\}, \{r_8\}\}$.

Observe that the answer above includes only two out of all the entity clusters in our toy database. It is clearly a waste to resolve the entire database in this case. However, for collective resolution, correctly resolving the relevant entities for a query may involve resolving neighboring entities as well. In general, 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 will introduce two expansion operators which will help us construct 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 determined by blocking techniques (McCallum et al., 2000). For a query $Q(n)$,

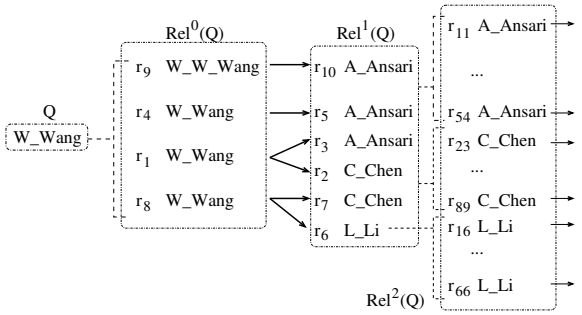

Figure 3. Relevant set for the query 'W. Wang' constructed using h-expansion and n-expansion alternately.

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 3 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 the references in $Rel^0(Q(n))$. Figure 3 illustrates this operation in our example. The interesting aspect about 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, a solution is to include the references that are most helpful for resolving the query. To illustrate using out example from Figure 3, 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, to expand the more 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 these 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 $k = h_{max}|Rel^i(Q)|$.

$$Rel_A^i(Q, h_{max}) = LeastAmb(k, X_H(Rel_A^{i-1}(Q))) \quad (4)$$

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.

$$Rel^i_A(Q, n_{max}) \quad = \quad X^e_N(MostAmb(k, Rel^i(Q))) \quad (5)$$

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 can be estimated as $Amb_A(n) = |A_1|^A_n / |A_1|^A_*$ where $|A_1|^A_x$ 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 strategies. The first dataset, **arXiv**, contains papers from high energy physics and was used in KDD Cup 2003[1]. It has 58,515 references to 9,200 authors, contained in 29,555 publications. Our second dataset is the **Elsevier BioBase** database[2] 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, 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.

[1] http://www.cs.cornell.edu/projects/kddcup/index.html

[2] http://help.sciencedirect.com/robo/projects/sdhelp/about_biobase_db.htm

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.

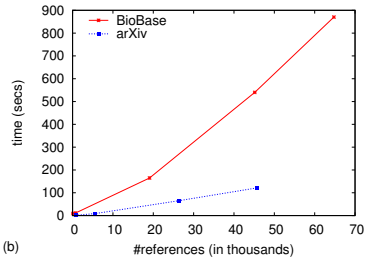

Figure 4. Execution time of RC-ER for increasing number of references.

We first explore the growth rate of the relevant set for sample queries over expansion depth in 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, in Figure 4(b), we observe 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. 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 re-

lational clustering algorithm (**RC-ER**) for *collective entity resolution with relations* 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 comparing attributes, we use the *Soft TF-IDF* with Jaro-Winkler similarity for names, which has been shown to perform the best for name-based resolution (Bilenko et al., 2003), 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 (Bhattacharya & Getoor, 2004; Singla & Domingos, 2004; Dong et al., 2005; McCallum & Wellner, 2004) 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 measured 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 mins 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 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 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 a 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.

7. Related Work

The entity resolution problem has been studied in many different areas under different names. A lot of work has focused on traditional attribute-based entity resolution. Extensive research has been done on defining approximate string similarity measures (Monge & Elkan, 1996; Bilenko et al., 2003) that may be used for

unsupervised entity resolution.

Many recently proposed approaches take relations into account for data integration. In earlier work, we have proposed different measures for relational similarity and a relational clustering algorithm for collective author resolution (Bhattacharya & Getoor, 2004; Bhattacharya & Getoor, 2006). Dong et al. (2005) collectively resolve entities of multiple types by propagating relational evidences in a dependency graph, and demonstrate the benefits of collective resolution in real datasets. While some of these approaches have been shown to be scalable, the focus has not been on query-time cleaning. Very recently, Long et al. (2006) have proposed a model for multi-type relational clustering using matrix factorization. Neville et al. (2003) have also shown how relations may be combined with attributes for clustering. Among cut-based approaches, Dhillon (2001) has proposed an objective function for the co-clustering. However, this applies only to bipartite graphs and constrains each cluster to relate to exactly one other cluster.

Probabilistic models for collective entity resolution have been applied to named entity recognition and for citation matching (McCallum & Wellner, 2004; Li et al., 2005; Singla & Domingos, 2004; Pasula et al., 2003). 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 (Draper & Hanks, 1994), but not for clustering problems, which is our approach for entity resolution. As we do, Fuxman et al. (2005) 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 first propose a cut-based relational clustering formulation for collective entity resolution. We motivate its application for query-time entity resolution and 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 real time collective resolution. While we have presented results for bibliographic data, the techniques are applicable in other relational domains. Interesting directions of future research include application of spectral techniques for optimizing the objective function, exploring stronger coupling between the extraction and resolution phases of query processing and investigating localized resolution for offline data cleaning as well.

References

Bhattacharya, I., & Getoor, L. (2004). Iterative record linkage for cleaning and integration. *SIGMOD Workshop on Data Mining and Knowledge Discovery*.

Bhattacharya, I., & Getoor, L. (2006). *Entity resolution in graphs*, chapter Entity Resolution in Graphs. Wiley.

Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., & Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*, *18*.

Dhillon, I. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. *SIGKDD*.

Dong, X., Halevy, A., & Madhavan, J. (2005). Reference reconciliation in complex information spaces. *SIGMOD*.

Draper, D., & Hanks, S. (1994). Localized partial evaluation of belief networks. *UAI*.

Fuxman, A., Fazli, E., & Miller, R. (2005). Conquer: Efficient management of inconsistent databases. *SIGMOD*.

Li, X., Morie, P., & Roth, D. (2005). Semantic integration in text: From ambiguous names to identifiable entities. *AI Magazine. Special Issue on Semantic Integration*.

Long, B., Zhang, M., Wu, X., & Yu, P. (2006). Spectral clustering for multi-type relational data. *ICML*.

McCallum, A., Nigam, K., & Ungar, L. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. *KDD*.

McCallum, A., & Wellner, B. (2004). Conditional models of identity uncertainty with application to noun coreference. *NIPS*.

Monge, A., & Elkan, C. (1996). The field matching problem: Algorithms and applications. *KDD*.

Neville, J., Adler, M., & Jensen, D. (2003). Clustering relational data using attribute and link information. *Text Mining and Link Analysis Workshop, IJCAI*.

Pasula, H., Marthi, B., Milch, B., Russell, S., & Shpitser, I. (2003). Identity uncertainty and citation matching. *NIPS*.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE PAMI*, *22*, 888–905.

Singla, P., & Domingos, P. (2004). Multi-relational record linkage. *KDD Workshop on Multi-Relational Data Mining*.