Iterative Record Linkage for Cleaning and Integration

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

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

ABSTRACT

Record linkage, the problem of determining when two records refer to the same entity, has applications for both data cleaning (deduplication) and for integrating data from multiple sources. Traditional approaches use a similarity measure that compares tuples' attribute values; tuples with similarity scores above a certain threshold are declared to be matches. While this method can perform quite well in many domains, particularly domains where there is not a large amount of noise in the data, in some domains looking only at tuple values is not enough. By also examining the context of the tuple, i.e. the other tuples to which it is linked, we can come up with a more accurate linkage decision. But this additional accuracy comes at a price. In order to correctly find all duplicates, we may need to make multiple passes over the data; as linkages are discovered, they may in turn allow us to discover additional linkages. We present results that illustrate the power and feasibility of making use of join information when comparing records.

Categories and Subject Descriptors

H.2.m [**Database Management**]: Miscellaneous; H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.2.5 [**Information Systems**]: Heterogeneous Databases; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search Process, Selection Process*

General Terms

algorithms, performance

Keywords

record linkage, deduplication, distance measure, clustering

1. INTRODUCTION

One of the fundamental problems in data cleaning and information integration is determining when two tuples refer to the same real-world entity. Often times data contains errors, for example typographical errors, or data may have multiple representations, such as abbreviations, so an exact comparison does not suffice for detecting duplicates in those cases. In data cleaning, deduplication [12, 18] is important for both accurate analysis, for example determining the number of customers, and for cost-effectiveness, for example removing duplicates from direct mailing list. In information integration, determining approximate joins [5] is important for consolidating information from multiple sources; most often there will not be a unique key that can be used to join tables in distributed databases, and we must infer when two records from different databases, possibly with different structures, refer to the same entity. Traditional approaches to duplicate detection are based on approximate string matching criteria, in some cases augmented with domain specific rules. More recently, there have been adaptive approaches which make use of multiple attributes and use labeled data [24, 2, 1, 3].

Our approach also makes use of attribute similarity measures, but in addition, it takes into account the similarity of linked objects. For example, if we are comparing two census records for 'Jon Doe' and 'Jonathan Doe', we should be more likely to match them if they are both married to 'Jeannette Doe' and they both have dependents 'James Doe', 'Jason Doe' and 'June Doe'. In other words, the string similarity of the attributes is taken into account, but so too is the similarity of the people to whom the person is related. This is similar in spirit to the approach taken by [1, 3]. However in our case, we do not assume that the linked objects have already been deduplicated. In fact, in our case the links are among objects of the same type, so that when we determine that two records refer to the same individual, that may in turn allow us to make additional inferences. In other words, the deduplication process is *iterative.*

Making the process iterative has the potential benefit that we may produce more accurate results; in particular, as noted by [1], we may be able to decrease our false positive rate because we can set our string match threshold more conservatively. But it has the down-side that the process is more expensive computationally; first, as we go beyond simply comparing attributes to comparing references, the similarity computation becomes more expensive and second, as we iterate, we must continue to update the distances as new duplicates are detected.

In this paper, we formulate the problem of iterative deduplication and present efficient algorithms. Since typically we do not have ground truth to compare against, evaluation of the results of deduplication is difficult. Here we use a parameterized data generator and present a thorough investigation of our iterative deduplication algorithm, showing the performance for a range of parameter settings for both the generator and the deduplication algorithm.

2. RELATED WORK

There has been a large body of work on deduplication, record linkage, and co-reference detection.[1] Here we review some of the main work, but the review is not exhaustive. For summary reports on deduplication and record linkage, see [28, 11, 4].

Within the statistics community, the earliest work was done by Newcombe [21]. He describes methods for limiting the number of comparisons required. Fellegi and Sunter [9] define a statistical framework for predicting "match" and "not match". Roughly speaking, record linkage is viewed as classification of pairs as "matched" and "unmatched", based on certain computed features of the pairs. Fellegi and Sunter describe how to estimate the parameters of their model and how to use it for classification. More recently, Winkler [26, 27, 29] builds upon the work by Fellegi and Sunter and uses a probabilistic approach with a latent match variable which is estimated using EM.

There has been extensive work on defining approximate string matching algorithms [18, 19, 20, 7] and adaptive algorithms, for example algorithms that learn string similarity measures [23, 6, 2, 8] and use active learning [24]. An important focus is on efficient data cleaning; examples include Hernandez and Stolfo [12] and Monge and Elkan [18]. Another area of related work is the work on identity uncertainty [22], object identification [25] and co-reference resolution in natural language processing. The work on co-reference resolution [15] is typically done with unstructured text; here our focus is on structured data.

One of the domains commonly used as a testbed is the citation domain [13, 16, 22, 24]. This is also the domain that we use as motivation, however as we will see, we focus on identifying authors rather than papers. The work most closely related to ours is the work of Chaudhuri et al. [3, 1]. They also make use of join information to aid in deduplication; as mentioned in the introduction, a key difference is that they assume that the secondary tables are themselves duplicate-free. Ananthakrishna et. al. [1] use co-occurrence in dimensional hierarchies to identify duplicates. Specifically, to resolve whether two entities are duplicates, they check for co-occurrence in the children sets of the entities. We work in a more general setting where there is no hierarchy within the groups/tuples and use the entire group to check for co-occurrences. As a result, instead of the easier problem of having to match sets, we are faced with the problem of matching sets of sets. Also, we use an iterative framework for our algorithm, motivated by our recursive definition of duplicates.

3. MOTIVATING EXAMPLE

Consider the problem of trying to construct a database of papers, authors and citations, from a collection of paper references, perhaps collected by crawling the web. A well-known example of such a system is CiteSeer [10], an autonomous citation indexing engine. CiteSeer is an important resource for CS researchers, and makes searching for electronic version of papers easier. However as anyone who has used CiteSeer can attest, there are often multiple references to the same paper, citations are not always resolved and authors are not always correctly identified [24, 22].

A related, even more difficult problem, is the task of integrating several citation services. While multiple references are not as much of an issue for a curated citation index such as DBLP [14], determining when entries from different sources refer to the same entity is still a problem.

In the context of our motivating example, there are several potential references that must be resolved. The first is the paper resolution problem; this is the most commonly studied bibliographic entity resolution task. The second is the author resolution problem; this task is less commonly studied, and is the focus of this paper.

3.1 The paper resolution problem

Consider the following example from [24]:

- R. Agrawal, R. Srikant. *Fast algorithms for mining association rules in large databases.* In VLDB-94, 1994.
- Rakesh Agrawal and Ramakrishnan Srikant. *Fast Algorithms for Mining Association Rules.* In Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994.

Sometimes, the paper resolution can be done based simply on the title. We can use one of the many existing methods for string matching, perhaps even tuned to the task of title matching. There is additional relational information, in terms of the venue, the authors of the paper, and the citations made by the paper; this additional information may help add evidence to the fact that two references are the same. This type of entity resolution has been the focus of much of the work in citation matching [13, 16, 22, 24].

3.2 The author resolution problem

A more novel example of entity resolution is the case of author resolution. Suppose that we have two different papers, and we are trying to determine if there are any authors in common between them. We can also do a string similarity match between the author names, but often references to the same person vary significantly. The most common difference is the variety of ways in which the first name and middle name are specified. For an author entity "Jeffrey David Ullman", we may see references "J. D. Ullman", "Jeff Ullman", "Ullman, J. D.", and so on. For the most part, these types of transformations can be handled by specialized code that checks for common name presentation transforms. However, we are still presented with the dilemma of determining whether a first name or middle name is the same as some initial; while the case of matching "J. D. Ullman" and "Jeffrey D. Ullman" seems quite obvious, for common names such as "J. Smith" and "X. Wang" the problem is more difficult. Existing systems take name frequency into account and will give unusual names higher matching scores. But this still leaves the problem of determining when two references to "J. Smith" refer to the same individual.

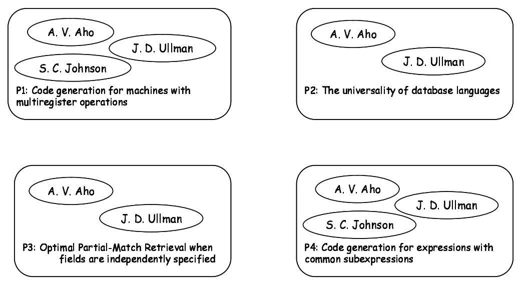
We propose to make use of additional context information in the form of coauthor relationships. If the coauthors of "J.

[1]The term deduplication is used more commonly within the database community, record linkage is the term used by statisticians and co-reference resolution is the term used more commonly by the AI and natural language processing community.

Smith" for these two papers are the same, then we should take this into account, and give the two references a higher matching score. But in order to do this we must have already determined that the other two author references refer to the same individual; thus it becomes a chicken and egg problem.

Consider the example shown in Figure 1, where we have four paper references, each with a title and author references. Figure 2 shows the final result after all the author references have been correctly resolved. We begin by examining the author references to see which ones we consider to be the same. In the first step, we might decide that all of the Aho references refer to the same individual, because Aho is an unusual last name. This corresponds to identifying r_1, r_4, r_6 and r_8 as duplicates. However, based on this name information alone, we are not quite sure whether the Ullman references (r_3, r_5, r_7 and r_{10}) are to the same individual, and we are certainly not sure about the Johnson references (r_2 and r_9). But, having decided that the Aho references are the same gives us additional information for the Ullman references. With high-confidence we consolidate the first two Ullman references; we may also consolidate the other Ullman references, although we may not be as certain that this is correct. And, at this stage, based solely on the Aho entity consolidation, we may decide we do not have enough evidence to consolidate the Johnson references. However, after having made the Ullman consolidations, we may decide that having two common coauthors for the two references is enough evidence to tip the balance, and we decide that they refer to the same Johnson.

Thus the problem of author resolution is likely to be an iterative process; as we identify common authors, this will allow us to identify additional potential co-references. We can continue in this fashion until all of the entities have been resolved.



A. V. Aho
J. D. Ullman
S. C. Johnson
P1: Code generation for machines with multiregister operations
A. V. Aho
J. D. Ullman
P2: The universality of database languages
A. V. Aho
J. D. Ullman
P3: Optimal Partial-Match Retrieval when fields are independently specified
A. V. Aho
J. D. Ullman
S. C. Johnson
P4: Code generation for expressions with common subexpressions

Figure 1: An example author/paper resolution problem. Each box represents a paper reference (in this case unique) and each oval represents an author reference.

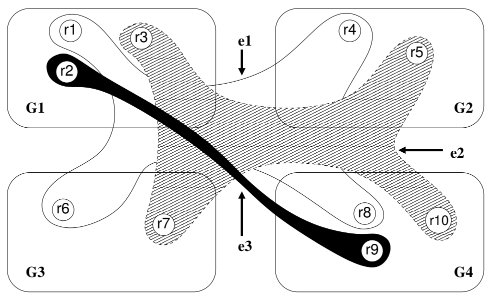


Figure 2: The single entity resolution problem corresponding to the example author/paper resolution problem.

4. FORMULATION OF THE ENTITY RESOLUTION PROBLEM

In the entity resolution problem, we have some collection of references to objects and from this set of references we would like to identify the (unique, minimal) collection of individuals or entities to which they should be mapped. In other words, we would like to find a many-to-one mapping from references to entities. But the problem is that we don't know the set of entities. Because references themselves may be many-many, such as the authorship relationship between papers and authors, we may be in the sticky situation of trying to identify more then one class of entities at the same time. Here, for simplicity, we consider the case where we have a single entity class to consolidate.

In the single entity consolidation problem, we have a collection of references $R = \{r_1, r_2, \ldots, r_n\}$. Each reference r corresponds to a unique entity, $E(r) \in \{e_1, e_2, \ldots, e_k\}$ and conversely $R(e) = \{r_i | E(r_i) = e\}$. The references may be partitioned into groups $G = \{g_1, g_2, \ldots, g_m\}$; a reference appears in only one group. Our task is, given R and G, to correctly determine both the entities (including the number of entities k) and the mapping from references to entities.

To make this more concrete, consider our earlier citation example. Figure 2 shows the correspondence between objects and variables. The references correspond to authorship relationships, such as (J. D. Ullman, Paper1). Let's call this r_3. Then $E(r_3) = e_3$, where e_3 corresponds to the entity "Jeffrey David Ullman". The groups correspond to the set of author references for each of the papers. We assume that there is one group per paper entity and that there is a one-to-one mapping between groups and papers. For example, in our earlier example, paper1 defines the group $g_1 = \{r_1, r_2, r_3\}$, where $r_1 =$ A. V. Aho, $r_2 =$ S. C. Johnson, and $r_3 =$ J. D. Ullman. Note that this is a group of references, not entities. Once we have correctly constructed the entity set for the authors, then we will be able to map from a paper to the actual authors of the paper. Thus, we want to learn that the entity sets are e_1, e_2 and e_3. The references belonging to e_1 are $\{r_1, r_4, r_6, r_8\}$, $e_2 = \{r_3, r_5, r_7, r_{10}\}$ and $e_3 = \{r_2, r_9\}$.

Not surprisingly, we do this by clustering references that are similar to each other. The key to the success of this clustering algorithm is the similarity measure (or, equivalently, the distance measure). We define a distance measure that takes into account both the attributes of the objects and the links between objects.

The attribute similarity of two references measures the similarity of the attributes of two references. For references r_i and r_j, it measures the similarity between two author names. If there are other attributes known, such as the author's institutions, then these can be factored in as well. There has been significant work on computing attribute similarity, several packages, such as [17], that implement this are available, so we assume that this is given.

In addition, to compute the similarity of the relationships that two objects participate in, we must compare their groups. But if we simply compare whether the references are the same, then we will not find any overlap — as we defined them, references correspond to authorship relations. Instead, we are interested in the authors themselves. But these are the entities that we are trying to find.

Instead, we compare two groups by looking at which references are currently known/believed to be duplicates. So clearly, this notion of similarity is bound to the current set of duplicates and will change as new duplicates are found. In the following section, we describe an iterative clustering algorithm that leverages this dynamic nature of the similarity.

5. ITERATIVE DEDUPLICATION

Our definition of duplicates is based on a distance measure between references. Our aim is to construct the relation $dup(r_i, r_j)$ over the set of references given to us. This relation is symmetric and also reflexive, meaning that every reference is its own duplicate. Our definition of duplicates is based on a distance measure between references. We define

$$dup(r_i, r_j) = true \text{ if } d(r_i, r_j) < t$$

for a given threshold t and false otherwise. This distance measure is a weighted combination of the attribute distance of the references as well as distance between their groups. We define the distance measure in the following subsection. The definition of duplicates is recursive in that the distance between references is tied to the current set of duplicates.

5.1 Distance Measure

The distance between two references is defined as

$$d(r_i, r_j) = (1 - \alpha) \times d_{attr}(r_i, r_j) + \alpha \times d_{group}(G(r_i), G(r_j))$$

where $d_{attr}()$ is the distance between the attributes and $d_{group}()$ is the distance measure between *group sets* of references and α is a weighting of the two neighbors. For the attributes of the references, we assume a vector representation that forms a metric space. This further allows us to assume numeric attribute values and an Euclidean metric for the attribute distance.

Now we define the group sets of references and the distance between them. For a reference r, $G(r)$ is all the groups that r or its duplicates occur in. $G(r) = \{g | r' \in g \text{ and } dup(r, r')\}$. There may be many possible ways to define distance between sets of sets. We build the definition of distance between group sets on the basic notion of distance between two groups of references. The similarity of two groups is defined as the ratio of the number of duplicates they share and the length of the longer group. Formally, for two groups g_1 and g_2,

$$sim(g_1, g_2) = |common(g_1, g_2)|/max(|g_1|, |g_2|) \text{ where}$$

$$common(g_1, g_2) = \{(r_1, r_2) | dup(r_1, r_2), r_1 \in g_1, r_2 \in g_2\}$$

The distance is then $d(g_1, g_2) = 1 - sim(g_1, g_2)$. Now we take the distance of a group g from a group set G to be the shortest distance from g to some group g' in G. $d(g, G) = min_{g' \in G} d(g, g')$. Finally, for the distance between two group sets G_1 and G_2, we find the mean distance of groups in G_1 to G_2 and of groups in G_2 to G_1 and take their average. This we choose to call the *group detail distance* d_{group} between two references.

We can now see the recursion in the definition of duplicates and appreciate the need for an iterative algorithm. As new duplicates are discovered, the distances between group sets of references are going to change, potentially leading to the discovery of more duplicates. We represent the current set of duplicates as clusters. We associate with each cluster the set of groups that its references occur in. This is the group set of the cluster. Formally, for a cluster c_k, $G(c_k) = \{g(r_i) | r_i \in c_k\}$. Also, with each cluster, we maintain a representative attribute value of all its references.[2] Once we have these two features for a cluster, we can easily extend the definition of distance between references to the distance between clusters. At each step, the algorithm re-evaluates the distances between clusters and merges the 'nearest' cluster-pair to represent the same entity. The iterations continue until there are no more candidates worthy of merging.

The group detail distance, as we have defined it, is computationally expensive since it involves pairwise comparison of the group sets in the two clusters. An approximate alternative that is computationally more tractable is to maintain and compare *group summaries*. The group summary g_{sum} of a reference is the set of all unique references in its group set. Defining it in terms of the cluster, it is the set of all cluster labels in the group set of the cluster. $g_{sum}(c_k) = \{c_i | c_i \in g_j, g_j \in G(c_k)\}$. Since the group summary is a set of cluster labels of references, our definition of distance between two groups carries over to distance between group summaries. This we call the *group summary distance* d_{gsum} between two clusters. If the references in the summaries are kept sorted by their cluster labels, this distance is computable in time that is linear in the group summary lengths. Finally, we define the summary distance between two clusters c_i and c_j as a weighted combination of the distance d_{attr} of the representative attributes and the group summary distance d_{gsum}.

$$d(c_i, c_j) = (1 - \alpha) \times d_{attr}(c_i, c_j) + \alpha \times d_{gsum}(c_i, c_j)$$

5.2 Iterative Group Clustering

It should be noted that at the start of the algorithm, when each reference is believed to be a distinct entity, the distance between all group summaries will be zero. Thus, in order to jump-start our group clustering algorithm, we start off by merging together references that 'obviously' correspond to the same entity, or, in other words, are separated by negligible attribute distance. Once the initial clusters have been formed, we choose the candidate cluster-pairs that are likely to be the same entity. The candidate set is chosen using a *distance threshold.* At each iterative step, the algorithm re-evaluates distances for the candidates, selects the closest pair according to the distance measure, merges the clusters and updates the attribute means and group sets and summaries. This procedure continues until the candidate set is exhausted.

6. EXPERIMENTAL EVALUATION

[2]For numeric attribute values, we use the mean. For ordinals or string attributes, we can use domain specific knowledge to decide the representative value or use the mode or medoid.

A difficulty with evaluating record linkage performance is the lack of gold standard for real-world data sets. Here we report a systematic evaluation of our iterative record linkage system on synthetic data, where we can model associations among authors, quantify the amount of noise and evaluate the recall-precision profiles for our algorithm.

6.1 Data Generator and Evaluation Measure

Since our goal is to evaluate the importance of co-occurrence information for deduplication, our data generator incorporates structure in the author domain by mimicking a real-life scenario where authors affiliated with a research group or a department in a university co-author papers. We create cliques of entities, where an entity belongs to a clique with some probability. We may imagine the cliques to represent groups of authors with similar research interests. Since any author can be associated with multiple research groups or have multiple research interests, we allow an entity to belong to multiple cliques. Each entity has fixed attribute values, corresponding to the true identity of the author. However, when it appears as an author name in any paper, it is likely to look different, which is why author identification is difficult. We mimic this phenomenon by probabilistically adding noise to the author identity when generating the author information for any particular paper. The parameters for the generator include the number of groups, cliques and authors, the degree of overlap between the cliques and the mean size of the cliques and the groups. The degree of overlap controls the extent to which entities belong to multiple cliques. Two other parameters control the noise in the attribute values of the references in each group — the error probability p_{err} and standard deviation σ_{err}. Each group g in our dataset is generated independently. First, a preference clique c is selected according to the prior probabilities of the cliques. Then the number of references for the group is chosen from a normal distribution. Each reference r is chosen by selecting an author a from the clique c according their probabilities of belonging to c. Each reference also has a small probability of being selected from any random clique c' different from c. The attribute values of the reference r, as they appears in the group, are generated by modifying by attributes of a with probability p_{err}, the magnitude of modification being determined by σ_{err}. While this procedure does not exactly capture the actual generative process for author-names in a paper, we believe ours is a simple and reasonable model that considers relationships among authors. We start off with each reference in a distinct cluster, which results in a cluster diversity of 1 but the entity dispersion is high since all references corresponding to the same entity are scattered across different clusters.

We evaluate our algorithm by measuring the quality of the clusters generated. We use two measures of cluster quality. *Entity dispersion* reflects the number of different clusters that references corresponding to the same entity are spread over. Lower dispersion is better; a perfect deduplication has dispersion 1. *Cluster diversity* quantifies the number of distinct entities that have been put in the same cluster. Lower diversity is also preferable; a perfect clustering has diversity 1. There is an inherent tradeoff between improving diversity and dispersion; an improvement in one will usually adversely affect the other. We consider the weighted average of the dispersion over entities and of the diversity over clusters as a measure of the quality of deduplication achieved.

6.2 Results

We compare the entity dispersion and cluster diversity achieved using group summary clustering against those using attribute clustering for varying data characteristics and algorithm parameters. The two algorithm parameters are the mixing weight α and the candidate selection threshold t_s. The data parameters that we experiment with are the error probability p_{err} and standard deviation σ_{err}, the mean clique size q_{mean} and the mean group size g_{mean}. As default values, we choose 1000 entities, 5000 groups with $g_{mean} = 4$, 100 cliques with $q_{mean} = 10$ and minimal overlap between cliques, $p_{err} = 0.1$ and $\sigma_{err} = 0.03$.

The dispersion-diversity plots in Figure 3 show how the performance of the two algorithms varies with α and t_s. The plots show that the best $(dispersion, diversity)$ combination achievable with group summary clustering is much better than attribute clustering. They also demonstrate the intrinsic trade-off between low dispersion and low diversity.

The results on experiments with data parameters are shown in Figures 4 and 5. It should be noted that time flows from left to right in our plots and the number of clusters decreases with each iteration. The conclusions that we draw from the plots are that the gains with group clustering are more when the mean size of the groups is larger, the cliques of authors are more in number and smaller in size and the error probability and standard deviation are higher for the attribute values.

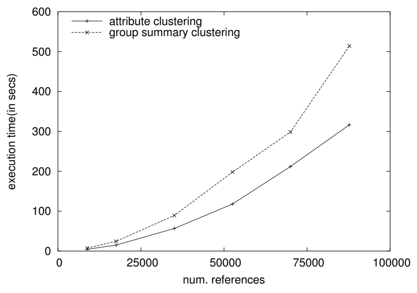


Figure 7: Execution times of attribute clustering and group summary clustering for varying data sizes

The above plots for data parameters show that group clustering is able to discover duplicates that are not easy to identify with fixed threshold attribute similarity approaches. It should be noted that our algorithm is not a substitute for attribute similarity algorithms. We present an iterative framework for combining attribute distances and group distances. While sophisticated attribute similarity measures already exist, we have presented a simple measure for quantifying distances between groups. Our algorithm should produce better results with improved attribute and group distance measures.

In Figure 6, we compare clustering using group detail distance and group summary distance against attribute clustering. We recall that group detail distance uses all the groups in a cluster for measuring distances while the group summary distance is a computationally more tractable but approximate alternative that considers a summary representation of the groups. While both of them do significantly better than attribute clustering, group detail clustering ex-

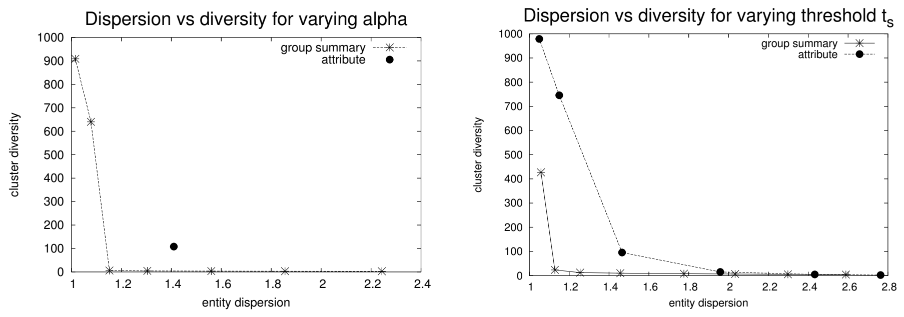


Figure 3: Dispersion-diversity plots for mixing weight and candidate selection threshold

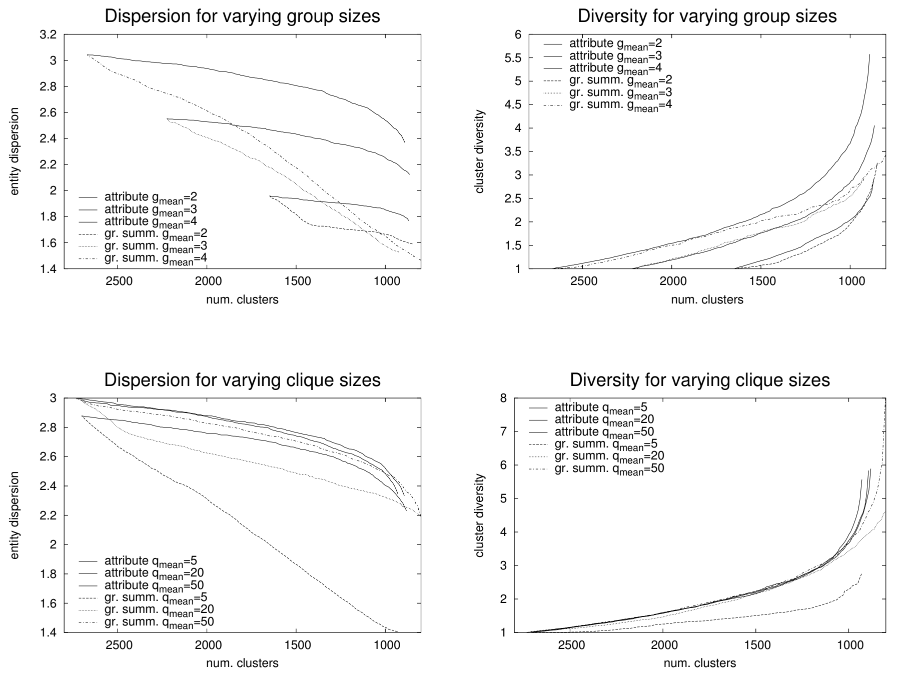


Figure 4: Entity dispersion and cluster diversity for varying group and clique sizes.

pectedly shows bigger improvements.

Figure 7 shows the execution times of group summary clustering and attribute clustering for increasing number of references. The plots show that group summary clustering scales gracefully with increasing data size. It is expectedly more costly than performing attribute similarity and for our datasets it takes roughly twice the time. However, the added cost clearly reaps bigger benefits, as shown by the performance plots. Database cleaning is not an operation that is likely to performed very frequently and the increased computation time is not expected to be too critical. There may of course be situations where this approach is not likely to prove advantageous, for example where distinctive cliques do not exist for the entities or if references for each group appear randomly. There the user has the choice of falling back on traditional attribute similarity or choosing α to set a low weight for group distances.

7. CONCLUSION

In this paper, we have defined an iterative deduplication algorithm and shown extensive evaluations on synthetically generated bibliographic data. We have found iterative deduplication to be a powerful and practical approach that performs better than attribute-based clustering. In future, we plan to work on real world data and investigate multiple entity consolidation and scaling issues.

8. ACKNOWLEDGMENTS

This study was supported by NSF Grant 0308030 and supported and monitored by the Advanced Research and Development Activity (ARDA) and the National Geospatial-Intelligence Agency (NGA) under Contract Number CG0323. The views, opinions, and findings contained in this report are those of the author(s) and should not be construed as an official Department of Defense position, policy, or decision,

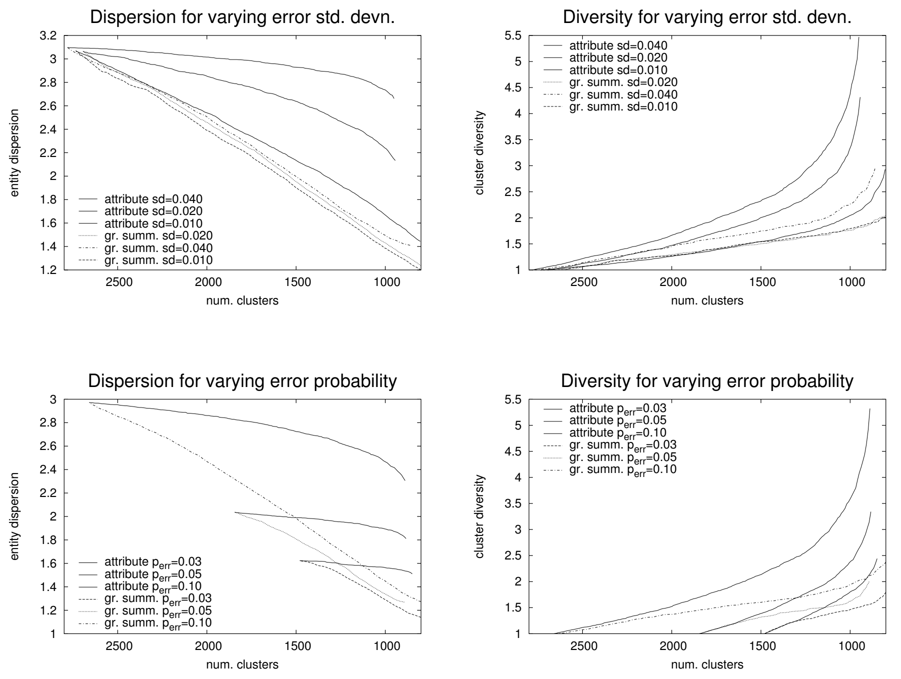


Figure 5: Entity dispersion and cluster diversity for varying error parameters.

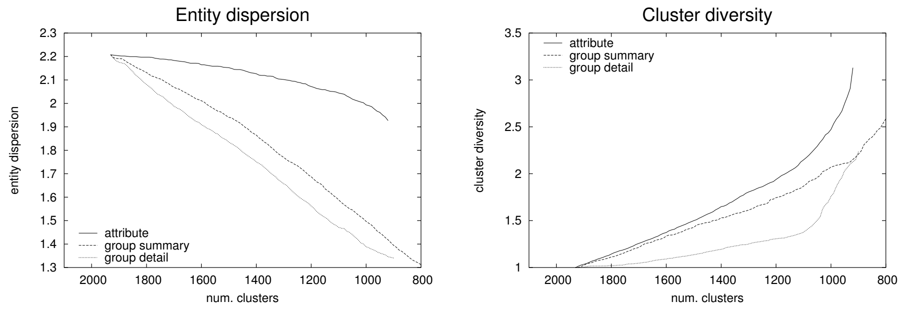


Figure 6: Comparison of group detail clustering and group summary clustering

unless so designated by other official documentation.

9. REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB-2002)*, Hong Kong, China, 2002.

[2] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, Washington, DC, 2003.

[3] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data*, pages 313–324, San Diego, CA, 2003.

[4] W. Cohen. Overview of record linkage methods. Powerpoint presentation, available at http://www-2.cs.cmu.edu/ wcohen/Matching-1.ppt.

[5] W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18:288–321, 2000.

[6] W. Cohen and J. Richman. Learning to match and cluster entity names. In *ACM SIGIR-2001 Workshop on Mathematical/Formal Methods in Information Retrieval*, New Orleans, LA, Sept. 2001.

[7] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 73–78, Acapulco, Mexico, Aug. 2003.

[8] W. W. Cohen and J. Richman. Learning to match and

cluster large high-dimensional data sets for data integration. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, 2002.

[9] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[10] C. L. Giles, K. Bollacker, and S. Lawrence. CiteSeer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, June 23–26 1998.

[11] L. Gu, R. Baxter, D. Vickers, and C. Rainsford. Record linkage: Current practice and future directions. Technical Report 03/83, CSIRO Mathematical and Information Sciences, Canberra, Australia, April 2003.

[12] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data (SIGMOD-95)*, pages 127–138, San Jose, CA, May 1995.

[13] J. A. Hylton. Identifying and merging related bibliographic records. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, 1996.

[14] M. Ley. The dblp computer science bibliography: Evolution, research issues, perspectives. In *Proceedings of the 9th International Symposium on String Processing and Information Retrieval*.

[15] A. McCallum and B. Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 79–86, Acapulco, Mexico, Aug. 2003.

[16] A. K. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the Sixth International Conference On Knowledge Discovery and Data Mining (KDD-2000)*, pages 169–178, Boston, MA, Aug. 2000.

[17] V. S. V. Mohamed G. Elfeky, Ahmed K. Elmagarmid. Tailor: A record linkage tool box. In *18th International Conference on Data Engineering (ICDE'02)*, 2002.

[18] A. E. Monge and C. P. Elkan. The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 267–270, Portland, OR, August 1996.

[19] A. E. Monge and C. P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proceedings of the SIGMOD 1997 Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 23–29, Tuscon, AZ, May 1997.

[20] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[21] H. Newcombe, J. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[22] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.

[23] E. Ristad and P. Yianilos. Learning string edit distance. *IEEE Transactions on PAMI*, 20(5):522–532, 1998.

[24] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, 2002.

[25] S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Information Systems Journal*, 26(8):635–656, 2001.

[26] W. E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods, American Statistical Association*, pages 354–359, 1990.

[27] W. E. Winkler. Improved decision rules in the fellegi-sunter model of record linkage. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 1993.

[28] W. E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 1999.

[29] W. E. Winkler. Methods for record linkage and Bayesian networks. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 2002.