Entity Resolution In Graphs

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

ABSTRACT

The goal of entity resolution is to reconcile data references corresponding to the same real world entity. Here we introduce the problem of entity resolution in graphs, where the nodes are the references in the data and the hyper-edges represent the relations that are observed to hold between the references. The goal then is to reconstruct a 'cleaned' entity graph that captures the relations among the true underlying entities from the reference graph. This is an important first step in any graph mining process; mining an unresolved graph will be inefficient and result in inaccurate conclusions. We also motivate collective entity resolution in graphs where references sharing hyper-edges are resolved jointly, as opposed to independent pair-wise resolution of the references. We illustrate the problem of graph-based entity resolution in bibliographic datasets. We discuss several interesting issues such as multiple entity types, local and global resolution and different kinds of graph-based evidence. We formulate the graph-based entity resolution problem as an unsupervised clustering task, where each cluster represents references that map to the same entity, and the similarity measure between two clusters incorporates the similarity of the references attributes and, more interestingly, the similarity between their relations. We explore two different measures of relational similarity. One approach, which we call 'edge detail similarity', explicitly compares the individual edges that each cluster participates in, but is expensive to compute. A less computationally intensive alternative is measuring 'neighborhood similarity', which only compares the multi-set of neighboring clusters for each cluster. We perform an extensive empirical evaluation of the two relational similarity measures for author resolution using co-author relations in two real bibliographic datasets. We show that both similarity measures improve performance over unsupervised algorithms that consider only reference attributes. We also describe an efficient implementation and show that these algorithms scale gracefully with increasing size of the data.

1. INTRODUCTION

In many applications, there are a variety of ways of referring to the same underlying real-world entity. For example, "J. Doe", "Jonathan Doe" and "Jon Doe" may all refer to the same person. In addition, entity references may be linked or grouped together. For example "Jonathan Doe" may be married to "Jeanette Doe" and may have dependents"James Doe", "Jason Doe" and "Jacqueline Doe" and "Jon Doe" may be married to "Jean Doe" and "J. Doe" may have dependents "Jim Doe", "Jason Doe" and "Jackie Doe". Given such data, we can build a graph from the entity references, where the nodes are the entity references and edges (or often hyper-edges) in the graph indicate links among the references.

However, the problem is that for any real-world entity, there may well be more than one node in the graph that refers to that entity. In the example above, we may have three nodes all referring to the individual "Jonathan Doe", two nodes referring to "Jeanette Doe", two nodes referring to each of "James Doe", "Jason Doe" and "Jacqueline Doe". Further, because the edges are defined over entity references, rather than entities themselves, the graph does not accurately reflect the relationships between entities. For example, until we realize that "Jon Doe" refers to the same person as "Jonathan Doe", we may not think that "Jon Doe" has any children, and until we realize that "J. Doe" refers to the same person as "Jonathan Doe", we will not realize that he is married.

Thus an important first step in any graph mining algorithm is transforming such a **reference graph**, where nodes are entity references and edges are among entity references, into an **entity graph**, where nodes are the entities themselves and edges are among entities. Given a collection of references to entities, we would like to a) determine the collection of 'true' underlying entities b) correctly map the entity references in the collection to these entities and c) correctly map the entity reference relationships (edges in the reference graph) to entity relationships (edges in the entity graph).

This problem comes up in many guises throughout computer science. Examples include computer vision, where we need to figure out when regions in two different images refer to the same underlying object (the correspondence problem); natural language processing when we would like to determine which noun phrases refer to the same underlying entity (co-reference resolution); and databases, where, when merging two databases or cleaning a database, we would like to determine when two tuple records are referring to the

same real world object (deduplication and/or record linkage).

Why do these ambiguities in entity references occur? Often times data may have data entry errors, such as typographical errors. Or multiple representations are possible, such as abbreviations, alternate representations, and so on. Or in a database, we may have different keys — one person database may use social security numbers while another uses name and address. Regardless, an exact comparison does not suffice for resolving entity references in such cases.

In data cleaning, deduplication [18, 27] is important for both accurate analysis, for example determining the number of customers, and for cost-effectiveness, for example removing duplicates from mailing lists. In information integration, determining approximate joins [9] 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 entity resolution and deduplication are based on approximate string matching criteria. These work well for correcting typographical errors and other types of noisy references, but do not make use of domain knowledge, such as common abbreviations, synonyms or nicknames, and do not learn mappings between values. More sophisticated approaches can make use of domain specific attribute similarity functions and mapping functions, and consider the reference not just as a string, but as a more structured object, such as a person entity which has first name, middle name, last name, address, and so on.

More recent approaches make use of attribute similarity measures, but in addition, take graph (i.e. relational) similarity into account. 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.

The problem becomes even more interesting when we do not assume that the related entities have already been resolved. In fact, when the relations are among entities of the same type, determining that two references refer to the same individual may in turn allow us to make additional inferences. In other words, the resolution process becomes *iterative*.

As mentioned earlier, the problem may be viewed in terms of a graph where the nodes represent the entity references that need to be resolved and the edges correspond to the observed relations among them. We will call this the *reference graph* capturing relations among the entity references. Our census example is shown in Figure 1(a). Before this graph can be mined for potential patterns or features, it needs to be 'cleaned'. Many nodes in this graph are duplicates in that they refer to the same underlying entity. The task is to identify which references correspond to the same entity and then consolidate them to create the *entity graph*. Figure 1(b) shows the entity graph after the references in the reference graph have been resolved. First, note that even in this simple graph, the entity graph is much smaller than the reference graph. In addition to the reduction in graph size that comes with resolving references, resolution is necessary for discovering the true patterns in the entity graph as well. Reference graphs are often collections of disconnected subgraphs. Unless they are resolved, the edges involving the same entity will be dispersed over its many references. Models built from such a graph will be inaccurate. In the example, the connections from the "Jeanette Doe" entity to the "Jacqueline Doe" entity can only be seen in the resolved entity graph.

Graph-based approaches for entity resolution take the edges into account as well for resolving references. In the above example, we may decide that the two "Jason Doe" references are the same, based on the fact that there is an exact string match and their fathers' have similar last names (though in general, we wouldn't want to always do this; certainly two "J. Doe"s don't necessary refer to the same person). Having done this, we may make use of the fact that both "J. Doe" and "Jonathan Doe" have a common dependent, to merge them.

Using the relational evidence provided in the graph has the potential benefit that we may produce more accurate results than if we use only attribute similarity measures. In particular, we may be able to decrease the 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 edges and subgraphs, the similarity computation becomes more expensive and secondly, as we iterate, we must continue to update the similarities as new resolutions are made.

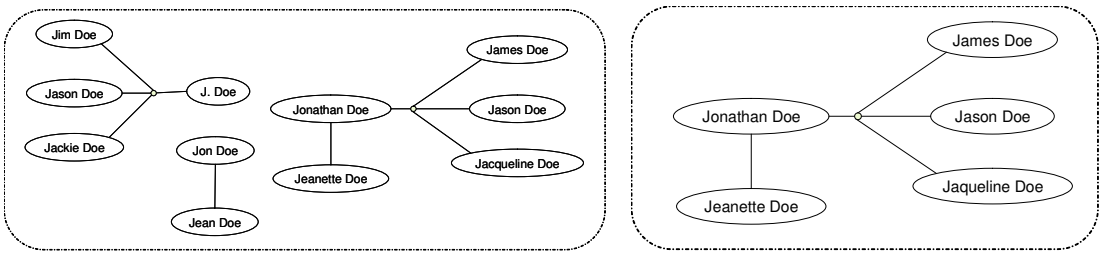
In the next section, we review related work on entity resolution; most of the work that we describe does not take a graph-based approach. In Section 3, we introduce another more realistic motivating example for graph-based entity resolution. In Section 4, we formalize the graph-based entity resolution problem. in Section 5, we define several similarity measures appropriate for entity resolution in graphs and in Section 6 we describe a clustering algorithm which uses them to perform entity resolution. In Section 7, we describe some experimental results using the different similarity measures on two real-world datasets.

2. RELATED WORK

There has been a large body of work on deduplication, record linkage, and co-reference resolution. Here we review some of the main work, but the review is not exhaustive. For a nice summary report, see [40].

2.1 String Similarity

The traditional approach to entity resolution looks at textual similarity in the descriptions of the entities. For example, whether or not two citations refer to the same paper depends on the similarity measure such as edit distance between the two citation strings. There has been extensive work on defining approximate string similarity measures [27, 29, 11, 8] that may be used for unsupervised entity resolution. The other approach is to use adaptive supervised algorithms that learn string similarity measures from labeled data [36, 6, 12, 39]. One of the difficulties in using a supervised method for resolution is constructing a good training set that includes a representative collection of positive and negative examples. One approach which avoids the problem of training set construction is active learning [37, 39],



(a) (b)

Figure 1: Example of (a) a reference graph for simple example given in the text and (b) the resolved entity graph.

where the user is asked to label ambiguous examples by the learner.

2.2 Theoretical Bounds

Cohen et al. [10] studies the theoretical problem of 'hardening a soft database' that has many co-referent entries. Hardening refers to the task of figuring out which pairs of soft identifiers refer to the same real world object. Given the likelihood of being co-referent for each soft pair and a probability distribution of possible hard databases, hardening is defined as the optimization problem of finding the most likely hard model given the soft facts. A cost is associated with each hard tuple that is added to the database and for each co-reference decision made. They show that this optimization problem is NP-hard even under strong restrictions. They propose a greedy agglomerative clustering approach for an approximate solution. This algorithm's complexity is linear in the number of entries in the soft database.

2.3 Efficiency

Given that solving the entity resolution problem optimally is computationally expensive, an important focus is on efficiency issues in data cleaning, where the goal is to come up with inexpensive algorithms for finding approximate solutions to the problem. The key mechanisms for doing this involve computing the matches efficiently and employing techniques commonly called 'blocking' to quickly find potential duplicates and eliminate non-duplicates from consideration [18, 28, 25, 19]. The merge/purge problem was posed by Hernandez and Stolfo [18] with efficient schemes to retrieve potential duplicates without resorting to quadratic complexity. They use a 'sorted neighborhood method' where an appropriate key is chosen for matching. Records are then sorted or grouped according to that key and potential matches are identified using a sliding window technique. However, some keys may be badly distorted so that their matches cannot be spanned by the window and such cases will not be retrieved. The solution proposed is a multi-pass method over different keys and then merging the results using transitive closure. Monge and Elkan [28] combine the union find algorithm with a priority queue lookup to find connected components in an undirected graph. McCallum et al. [25] propose the use of canopies to first partition the data into overlapping clusters using a cheap distance metric and then use a more accurate and expensive distance metric for those data pairs that lie within the same canopy. Gravano et al. [17] propose a sampling approach to quickly compute cosine similarity between tuples for fast text-joins within an SQL framework. Chaudhuri et al. [8] use an error tolerant index for data warehousing applications for probabilistically looking up a small set of candidate reference tuples for matching against an incoming tuple. This is considered 'probabilistically safe' since the closest tuples in the database will be retrieved with high probability. This is also efficient since only a small number of matches needs to be performed.

2.4 Probabilistic Modeling

The groundwork for posing entity resolution as a probabilistic classification problem was done by Fellegi and Sunter [14], who extend the ideas of Newcombe [31] for labeling pairs of records from two different files to be merged as "match" or "non-match" on the basis of agreement among their different fields. They estimate the conditional probabilities of these field agreement values given that the pair is really from the match class or the non-match class. They show that if the agreement values for the different fields are conditionally independent given the class, then these probabilities can be estimated in an unsupervised fashion. Winkler [41] builds upon this work for cases when the conditional independence assumption cannot be made and uses a generalized expectation maximization algorithm for estimating parameters to separate matches and non-matches. More recently, hierarchical graphical models have been proposed [35] that use a separate match variable for each attribute and an overall match variable that depends on all of these lower level matches.

2.5 Graph-based Approaches

Approaches that take into account relational structure of the entities for data integration have been proposed [21, 1, 13, 30, 4, 20]. Ananthakrishna et al. [1] introduce relational deduplication in data warehouse applications where there is a dimensional hierarchy over the relations. They augment the string similarity measure between two tuples with the similarity between their foreign key relations across the hierarchy which they call children sets. To avoid comparison between all pairs of tuples in a relation, they propose a

grouping strategy that makes uses of the relational hierarchy as well.

Kalashnikov et al. [21] enhance feature-based similarity between an ambiguous reference and the many entity choices for it with relationship analysis between the entities, like affiliation and co-authorship. They propose a 'content attraction principle' hypothesizing that an ambiguous reference will be more strongly connected via such relationships to its true entity compared to other entity choices for it. They translate this principle to a set of non-linear equations that relate all the connection strengths in the entity graph and those between a reference and its choice entities. A solution to this nonlinear optimization problem yields the connection strengths and the strongest connection determines the entity choice for each reference.

Neville et al. [30] explore different graph partition schemes for clustering in graphs where the edge weights reflect attribute similarity between nodes. By varying the edge weights and edge existence probabilities conditioned on the cluster labels, they compare algorithms that consider only attributes and those that combine attribute and relational evidence. They report that spectral techniques for partitioning [38] work better that other min-cut and k-clustering approaches but combining attribute and relational information proves detrimental for clustering.

The SUBDUE system proposed by Jonyer et al. (see [20] and Chapter 8 of this book) is a scheme for conceptual clustering of structured data. In addition to partitioning the data, conceptual clustering also summarizes the clusters with conceptual descriptions of objects contained in them. SUBDUE generates a hierarchical conceptual clustering by discovering substructres in the data using the minimum description length principle. This helps to compress the graph and represent conceptual structure as well.

Doan et al. [13] explore a profiler-based approach for tying up disjoint attributes for sanity checks using domain knowledge. For example, on merging two objects, (9, John Smith) and (John Smith, 120k) from two tables with schemas (age, name) and (name, salary), we get a person whose age is 9 years and whose salary is 120K. This would be deemed an unlikely match by a profiler.

In earlier work of our own [4], we propose different measures for relational similarity in graphs and show how this can be combined with attribute similarity for improved entity resolution in collaboration graphs. We also relate the problem of graph-based entity resolution to discovering groups of collaborating entities in graphs [3] and suggest that the two tasks may be performed jointly so that a better solution for one of these tasks leads to improvements in the other as well.

2.6 Probabilistic Inference In Graphs

Probabilistic models that take into account interaction between different entity resolution decisions have been proposed for named entity recognition in natural language processing and for citation matching. McCallum et al. [24] use conditional random fields for noun coreference and use clique templates with tied parameters to capture repeated relational structure. They do not directly model explicit links among entities.

Li et al. [23] address the problem of disambiguating "entity mentions", potentially of multiple types, in the context of unstructured textual documents. They propose a probabilistic generative model that captures a joint distribution over pairs of entities in terms of co-mentions in documents. In addition, they include an appearance model that transforms mentions from the original form. They evaluate a discriminative pairwise classifier for the same task which is shown to perform well. However, they show both empirically and theoretically that direct clustering over the pairwise decisions can hurt performance for the mention matching task when the number of entity clusters is more than two.

Parag et al. [32] use the idea of merging evidence to allow the flow of reasoning between different pair-wise decisions over multiple entity types. They are able to achieve significant benefit from generalizing the mapping of attribute matches to multiple references, for example being able to generalize from one match of the venue "Proc. of SIGMOD" with "Proceedings of the International Conference on Management of Data" to other instances.

Pasula et al. [33] propose a generic probabilistic relational model framework for the citation matching problem. Because of the intractability of performing exact probabilistic inference, they propose sampling algorithms for reasoning over the unknown set of entities. Milch et al. [26] propose a more general approach to the identity uncertainty problem. They present a formal generative language for defining probability distribution over worlds with unknown objects and identity uncertainty. This can be seen as a probability distribution over first order model structures with varying number of objects. They show that the inference problem is decidable for a large class of these models and propose a rejection sampling algorithm for estimating probabilities.

In other work of our own [5], we have adapted the Latent Dirichlet Allocation model for documents and topics and extended it to propose a generative group model for joint entity resolution. Instead of performing a pair-wise comparison task, we use a latent group variable for each reference, which is inferred from observed collaborative patterns among references in addition to attribute similarity to predict the entity label for each reference.

2.7 Tools

A number of frameworks and tools have been developed. Galhardas et al. [15] propose a framework for declarative data cleaning by extending SQL with specialized operators for matching, clustering and merging. The WHIRL system [9] integrates a logical query language for doing 'soft' text joins in databases with efficient query processing. Potter's Wheel [34], Active Atlas [39] and D-Dupe [7] are some other data cleaning frameworks that involve user interaction.

2.8 Application Domains

Data cleaning and reference disambiguation approaches have been applied and evaluated in a number of domains. The earliest application is on medical data [31]. Census data is an area where detection of duplicates poses a significant challenge and Winkler [41] has successfully applied his research and other baselines to this domain. A great deal of work has been done making use of bibliographic data [19, 22, 25, 37, 33, 4]. Almost without exception, the focus has been on the matching of citations. Work in coreference resolution and disambiguating entity mentions in natural language processing [24, 23] has been applied to text cor-

pora and newswire articles like the TREC corpus. For detailed evaluation of algorithm performance, researchers have also resorted to synthetic [4, 30] and semi-synthetic [1, 8] datasets where various features of the data can be varied in a controlled fashion.

2.9 Evaluation Metrics

As has been pointed out by Sarawagi et al. [37], choice of a good evaluation metric is an issue for entity resolution tasks. Mostly, resolution has been evaluated as a pair-wise classification problem. Accuracy may not be the best metric to use since datasets tend to be highly skewed in their distribution over duplicate and non-duplicate pairs; often less than 1% of all pairs are duplicates. In such a scenario, a trivial classifier that labels all pairs as non-duplicates would have 99% accuracy. Though accuracy has been used by some researchers [41, 27, 8, 30], most have used precision over the duplicate prediction and recall over the entire set of duplicates. Observe that a classifier that indiscriminately labels all pairs as non-duplicates will have high precision but zero recall. The two measures are usually combined into one number by taking their harmonic mean. This is the so-called F1 measure. Another option that has been explored is weighted accuracy but this may report high accuracy values even when precision is poor. Cohen et al. [11] rank all candidate pairs by distance and evaluate the ranking. In addition to the maximum F1 measure of the ranking, they consider the non-interpolated average precision and interpolated precision at specific recall levels.

Some other approaches to this problem have posed it as a clustering task, where references that correspond to the same entity are associated with the same cluster. Performance measures that evaluate the qualities of the clusters generated compared to the true clusters are more relevant in such cases. Monge and Elkan [28] use a notion of cluster purity for evaluation. Each of the generated clusters may either match a true cluster, be a subset of a true cluster or include references from more than one cluster. They treat the first two cases as pure clusters while the third category of clusters is deemed impure. They use the number of pure and impure clusters generated as the evaluation metric. We have proposed an alternative evaluation metric for this clustering task where we measure the diversity of each constructed cluster of entity references in terms of the number of references to different real entities that it contains [4] and the dispersion of each entity over the number of different clusters. We show that dispersion-diversity plots capture the quality of the clusters directly and can be used to evaluate the trade-off in a fashion similar to precision-recall curves.

3. MOTIVATING EXAMPLE FOR GRAPH-BASED ENTITY RESOLUTION

Throughout the rest of this chapter, we will motivate the problem of entity resolution in graphs using an illustrative entity resolution task from the bibliographic domain. 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 [16], an autonomous citation indexing engine. CiteSeer is an important resource for CS researchers, and makes searching for electronic versions 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 [37, 33].

3.1 An Example

The most commonly studied bibliographic entity resolution task is resolving paper citations. Consider the following example from [37]:

- 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.

These very different strings are citations of the same paper and clearly string edit distance alone will not work. However, if we extract the different fields or attributes of the paper, we may have better luck. Sometimes, 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. However, 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 provide further evidence to the fact that two references are the same.

3.2 Issues in Graph-based Entity Resolution

3.2.1 Multi-Type Entity Resolution

In the above example and in the earlier census example, we had one type of entity (e.g. papers, people) and we were trying to resolve them. We refer to this as *single-type entity* resolution. But note that while the above two citation strings are used to motivate the paper resolution problem, it is more interesting to see that they present an illustrative example of multi-type entity resolution, the problem of resolving entity references when there are different types of entities to be resolved. The citations refer to papers, but in addition to the paper title, they contain references to other types of entities, which themselves may be ambiguous. In particular, the strings refer to *author* and *venue* entities in addition to *paper* entities. This brings up a scenario where multiple types of entities need to be resolved simultaneously. Assuming that the strings have been correctly parsed to separate out the different fields — which is a difficult problem by itself — the first citation string mentions 'R. Agrawal' and 'R. Srikant' as the authors and 'VLDB-94, 1994' as the venue, while the second has 'Rakesh Agrawal' and 'Ramakrishnan Srikant' as the authors and 'Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994' as the venue reference. Not all of these pairs are easy to disambiguate individually. While it may not be too difficult to resolve 'R. Srikant' and 'Ramakrishnan Srikant', 'Agrawal' is an extremely common Indian last name and it is certainly not obvious that 'R. Agrawal' refers to the same author entity as 'Rakesh Agrawal'. As for the venue references, unless one knows that for two resolved papers their venues must be the same, it is very difficult, if not impossible, to resolve the two venue references without specialized domain knowledge.

3.2.2 Collective Entity Resolution

For our example, it is not hard to observe the dependence among the different resolution decisions across multiple classes. We can make the resolution decisions *depend* on each other in cases where they are *related*. When the resolution decisions are made *collectively*, they can be much easier to make. In the citation domain the relevant entities are *papers*, *authors* and *venues* and the relevant relationships are *write* and *published in*, i.e. Authors *write* papers, which get *published in* venues. An author is a *co-author* of another author if they write the same paper. We can use these relations to make one resolution decision lead to another. We may begin by resolving 'R. Srikant' and 'Ramakrishnan Srikant', of which we are most confident. This may lead us to believe 'R. Agrawal' and 'Rakesh Agrawal' are the same author, since they are *co-authors* with the resolved author entity 'Srikant'. Now we can go back to the paper citations which, in addition to having very similar titles, have now been determined to be *written by* the same author entities. This makes us more confident that the two paper references map to the same paper entity. Following the same thread, two identical papers citations must have identical venue citations. So we may resolve the apparently disparate venue references.

3.2.3 Graph-based Evidence for Entity Resolution

It is now easier to see how graphs help in representing these dependencies. The reference graph has nodes for each entity reference. If the references correspond to multiple entity classes, then we have multiple node types. Edges represent the relations that hold between the references. We may also have more than one type of edge to represent the different types of relations that exist. The co-author relation illustrates the need for hyper-edges that may involve more than two references. Since all observed authors of a paper are co-authors, this relation is naturally captured as a hyper-edge that spans all the author references from a paper. Note that this may alternatively be captured using a quadratic number of binary edges but this leads to a dramatic increase in the graph size.

3.2.4 Local Versus Global Resolution

When we resolve two references, such as the resolution of two venue strings 'Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994' and 'VLDB-94, 1994', we should probably resolve all other occurrences of these venue references. In other words, once we have decided the two references are the same, we should also resolve any other venue references that exactly match these two references. We call this type of resolution *global resolution*. For certain entity references it makes the most sense, and can speed things up significantly. However, it may not always be appropriate. In the case of names for example, 'R. Agrawal' may refer to the 'Rakesh Agrawal' entity in one case, while some other instance of 'R. Agrawal' might refer to a different entity 'Rajeev Agrawal'. We refer to this latter resolution strategy as *local resolution*.

3.2.5 Additional Sources of Relational Evidence

Other bibliographic relations can also potentially be used when available. Papers are *cited by* other papers and two papers are *co-cited* if they are cited by the same paper entity. If two similar paper references are co-cited by the same paper entity, that may serve as additional evidence that they are the same. However, graph-based evidence may also be *negative* in some cases. For example, two paper references that are *cited by* the same paper entity are unlikely to be duplicates, as are two author references that are *co-authors* of the same paper entity.

3.3 Author Resolution in Graphs

Within the context of entity resolution in bibliographic data, we first look at the problem of resolving author references in graphs leveraging co-author relationships. Suppose that we have two different papers, and we are trying to determine if there are any authors in common between them. We can 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" or "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 whether a reference to "J. Smith" refers to "James Smith" or "Jessica Smith".

As mentioned before, additional context information can be used in the form of coauthor relationships. If the coauthors of "J. 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 2(a), where we have four paper references, each with a title and author references. In order to resolve these references, we may 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 author entity because Aho is an unusual last name. This corresponds to resolving all of the Aho references into a single entity. However, suppose based on name information alone, we are not quite sure that the Ullman references are to the same author, and we are certainly not sure about the Johnson references, since Johnson is a very common name. But, deciding that the Aho references correspond to the same entity gives us additional information for the Ullman references. We know that the references share a common co-author. Now with higher confidence we can consolidate the Ullman references. Based solely on the Aho entity consolidation, we do not have enough evidence to consolidate the Johnson references. However, after making the Ullman consolidations, we may decide that having two co-authors in common is enough evidence to tip the balance, and that all the Johnson references correspond to the same Johnson entity. Figure 2(b) shows the final result after all the author references have been correctly resolved, where references to the same entity

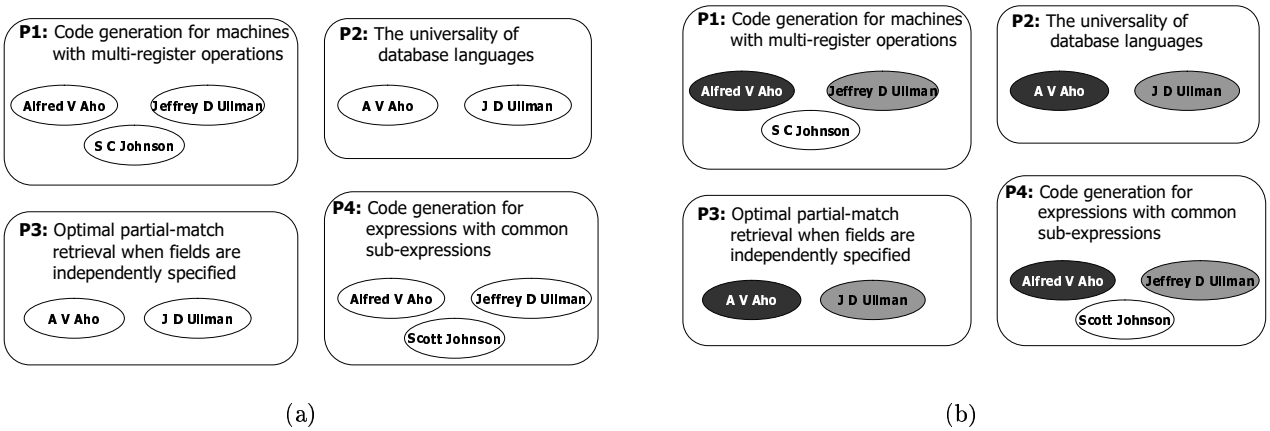


Figure 2: (a) An example author/paper resolution problem. Each box represents a paper reference (in this case unique) and each oval represents an author reference. (b) The resolved entities corresponding to the example author/paper resolution problem.

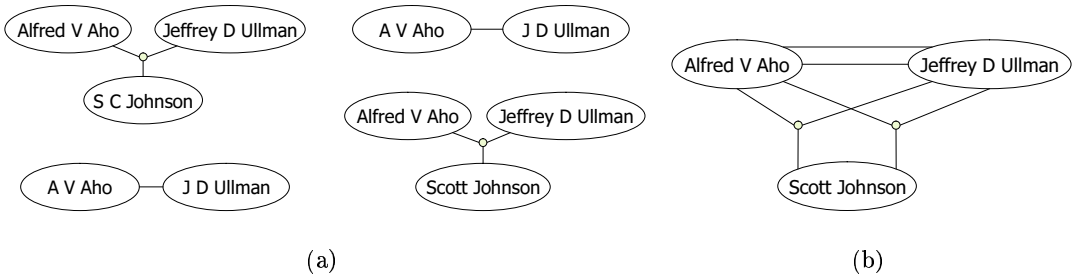


Figure 3: (a) The reference graph and (b) the entity graph for the author resolution example.

are shaded accordingly.

As illustrated in the above example, the problem of graph-based author resolution is likely to be an iterative process: as we identify co-author/collaborator relationships among author entities, this will allow us to identify additional potential co-references. We can continue in this fashion until all of the entities have been resolved. Figure 3(a) shows the reference graph for this example and Figure 3(b) shows the resulting entity graph.

4. GRAPH-BASED ENTITY RESOLUTION: PROBLEM FORMULATION

In the graph-based entity resolution problem, we have a reference graph — a graph over some collection of references to entities — and from this graph we would like to identify the (unique, minimal) collection of individuals or entities to which they should be mapped, and the induced entity graph. In other words, we would like to find a many-to-one mapping from references to entities. Figure 3(a) shows the reference graph for the author resolution example and Figure 3(b) shows the resulting entity graph.

In what follows, lower case characters e and r denote entities and references and qualified upper case letters like $e.A$ and $r.E$ denote attributes of the variables, and we will use $e.a$ and $r.e$ to denote particular values of variables (shorthand for $e.A = a$). In the single entity resolution problem, we are given a set of references $\mathcal{R} = \{r_i\}$, where each reference r has its attributes $r.A$. The references correspond to entities $\mathcal{E} = \{e_i\}$ so that each reference r has a hidden entity label $r.E$. Each entity e also has its own attributes $e.A$, but the entities are not directly observed. What we observe are the attributes $r.A$ of individual references. We can imagine $r.A$ to be generated by some distortion process from the attributes of the corresponding entity $r.E$. Obviously, the entity labels of the references are not observed. The problem is to recover the hidden set of entities $\mathcal{E} = \{e_i\}$ and the entity labels $r.E$ of individual references given the observed attributes of the references.

We use relational information among references to help us in collective entity resolution. We will assume that the references are observed, not individually, but as members of hyper-edges. We are given a set of hyper-edges $\mathcal{H} = \{h_i\}$ and the membership of a reference in a hyper-edge is captured by its hyper-edge label $r.H$. In general, it is possible for a reference to belong to multiple hyper-edges and all of our approaches can be extended to handle this. However

in this chapter, we will consider a simpler scenario where each reference occurs in a single hyper-edge. If reference r occurs in hyper-edge h, then $r.H = h$. Note that unlike the entity labels, we *know* the association of hyper-edges and references. The hyper-edges can help us make better predictions if we assume that they are indicative of associative patterns among the entities. In other words, the entity labels of references that occur in the same hyper-edge are related to each other. Now the resolution decisions are not independent. Instead of finding the entity labels for each reference individually, our task is to predict the entity labels of the references collectively, where the entity label $r.E$ of any reference r is directly influenced by the choice of entity label $r'.E$ for another reference r' if they are associated with the same hyper-edge, i.e., $r.H = r'.H$.

To make this more concrete, consider our earlier example. Figure 4(a) shows the references and hyper-edges. Each observed author name corresponds to a reference, so there are ten references r_1 through r_{10}. In this case, the attributes are the names themselves, so for example, $r_1.A$ is "Alfred Aho", $r_2.A$ is "S.C. Johnson" and $r_3.A$ is "Jeffrey Ullman". The set of true entities $\mathcal{E}$ is $\{e_1, e_2, e_3\}$ as shown in Figure 4(b), where $e_1.A$ = "Alfred V. Aho", $e_2.A$ = "S.C. Johnson" and $e_3.A$ = "Jeffrey D. Ullman". Clearly, r_1, r_4, r_6 and r_8 correspond to "Alfred V. Aho", so that $r_1.E = r_4.E = r_6.E = r_8.E = e_1$. Similarly, $r_3.E = r_5.E = r_7.E = r_{10}.E = e_2$ and $r_2.E = r_9.E = e_3$. There are also the hyper-edges, which correspond to each set of authors for a paper. There are four papers, so that $\mathcal{H} = \{h_1, h_2, h_3, h_4\}$. The references r_1 through r_3 are associated with hyper-edge h_1, since they are the observed author references in the first paper. This is represented as $r_1.H = r_2.H = r_3.H = h_1$. We may similarly capture the hyper-edge associations of the other references. The problem here is to figure out from the attributes $r.A$ and the hyper-edge labels $r.h$ of the references that there are three distinct entities such that r_1, r_4, r_6 and r_8 correspond to one entity, r_3, r_5, r_7 and r_{10} correspond to the second and r_2 and r_9 correspond to the third. Here we have introduced the notation assuming that all references correspond to the same class of entities. It may be extended to handle multiple entity classes and multiple types of hyper-edges between references. In the rest of this chapter, we will use the term edge as a substitute for hyper-edge. It will be understood that an edge may involve more than two nodes unless explicitly stated otherwise.

4.1 Entity Resolution as a Clustering Problem

Alternatively, the task of collective entity resolution may be viewed as a graph-based clustering problem where the goal is to cluster the references so that those that have identical entity labels are in the same cluster. One approach to finding this mapping is to use a greedy agglomerative clustering approach. At any stage of the clustering algorithm, the set of *entity clusters* reflect our current beliefs about the underlying entities. In other words, each constructed entity cluster should correspond to one underlying entity and all references in that cluster should be to that entity. To start off, each reference belongs to a separate cluster and at each step the pair of clusters (or entities) that are most likely to refer to the same entity are merged. The key to the success of a clustering algorithm is the similarity measure that is employed. In graph-based clustering for entity resolution, we want to use a similarity measure that takes into account the similarity in the attributes of the references in the two clusters as well as the relational similarity. In addition, the measure should take into account the *related* resolution decisions that have been made previously. Accordingly, similarity measures are extended to consider both reference attributes and edge-based patterns.

The *attribute similarity component* of the similarity measure takes into account the similarity of the attributes $r.A$ of the references in the two clusters. In the author resolution case, it measures the similarity between two observed reference names. In addition, the *graph-based similarity component* takes into account the similarity of the relations that the two entities or clusters participate in. Each cluster is associated with a set references and through these references, the cluster is associated with a set of edges to other references. But what is it about these other references that we want to consider? We want to look not at their attributes but at the resolution decisions that have been taken on them. Specifically, we want to look at the entity cluster labels of these references, or which clusters they currently belong to. To illustrate this using our example from Figure 2, suppose we have already resolved the 'Aho' and 'Ullman' references in clusters c_1 and c_2 respectively and we are looking at the current similarity of the two 'Johnson's which are yet to be resolved and still belong to separate entity clusters, say c_{3a} having r_2 and c_{3b} having r_9. The two clusters are associated with one edge each, h_1 and h_4 respectively. We want to factor into the similarity measure *not the names* of the other references in the two edges, but the fact that both of them are associated with the same resolved entity clusters c_1 and c_2, which is what makes them similar.

An issue that is brought out by the above discussion is the dynamic nature of the graph-based similarity component. Initially, when all references belong to distinct entity clusters, the two 'Johnson' clusters will not be considered similar enough. But their graph-based similarity goes up in stages as first the 'Aho' references and then the 'Ullman' references are resolved. In the following section, we describe an iterative clustering algorithm that leverages this dynamic nature of graph-based similarity.

5. SIMILARITY MEASURES FOR ENTITY RESOLUTION

In this section, we define the similarity measure between two entity clusters as a weighted combination of the attribute similarity and graph-based similarity between them and highlight the computational and other algorithmic issues that are involved.

For two entity clusters c_i and c_j, their similarity may be defined as

$$sim(c_i, c_j) = (1 - \alpha) \times sim_{attr}(c_i, c_j) + \alpha \times sim_{graph}(c_i, c_j) \qquad 0 \leq \alpha \leq 1$$

where $sim_{attr}()$ is the similarity of the attributes and $sim_{graph}()$ is the graph-based similarity between the two entity clusters and they are linearly combined with weights α and $1-\alpha$. In the following two subsections, we discuss the two similarity components in detail.

5.1 Attribute Similarity

We assume the existence of some basic similarity measure that takes two reference attributes and returns a value be-

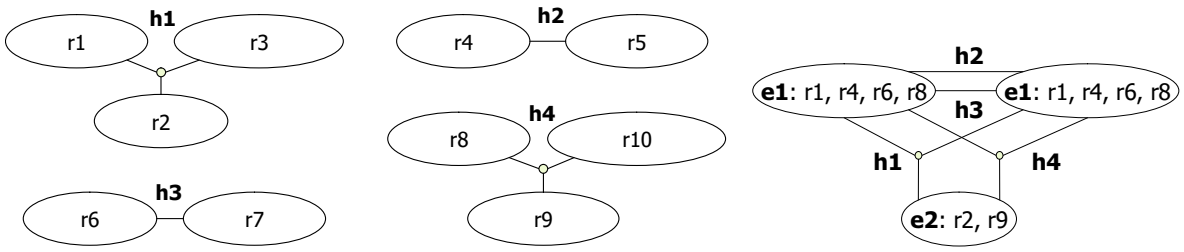


Figure 4: (a) A more abstract representation of the reference graph for the author resolution example; the r's are references and the h's are hyper-edges. (b) An abstract representation for the entity graph for the author resolution example; the e's are entities, the references they correspond to are listed, and the h's are hyper-edges.

tween 0 and 1 that indicates the degree of similarity between them. A higher value indicates greater similarity between the attributes. We are not making any other assumptions about the attribute similarity measure. Any measure that satisfies these assumptions can be used. This is particularly helpful since such similarity measures are often tuned for specific domains and may be available as library functions. Depending on the domain, it is possible to adapt a different attribute measure and tie it in with our algorithm seamlessly.

However, we need $sim_{attr}()$ to define the similarity of attributes between two entity clusters. Each entity cluster is a collection of references with their own attributes. So we need to use the similarity measure that takes two attributes to define the similarity between two clusters of attributes. This is similar to finding the aggregate distance between two clusters given a pairwise distance measure. Many approaches like *single link*, *average link* and *complete link* have been proposed [2] where some aggregation operation is used to combine the pair-wise distances between the two clusters. The duplicate relation is typically transitive: if references r_i are r_j are duplicates, then all other duplicates of r_i will also be duplicates of r_j. So the single link measure that takes the minimum pair-wise distance (or the maximum pairwise similarity) between two clusters is the most relevant for our purposes.

The first issue with a cluster linkage metric is that it is computationally intensive. The number of attribute similarity computations involved in finding the similarity between two clusters is quadratic in the average number of references in each cluster. So pair-wise comparison is a problem for larger clusters. Further, as clusters merge and new references are added to a cluster, similarities need to be recomputed. This repeated similarity computation adds to the complexity. There are a number of ways this problem may be addressed. First, updating the cluster similarities is not computationally challenging when using the single link metric. They may be incrementally modified as clusters merge. Specifically, when two clusters c_i and c_j are merged to create a new cluster c_{ij}, the similarity to a third cluster c_k may be updated as $sim_{attr}(c_{ij}, c_k) = max(sim_{attr}(c_i, c_k), sim_{attr}(c_j, c_k))$. Secondly, when we are dealing with attributes like names, though a cluster may contain hundreds of references, there will be very few distinct names. So for two clusters c_i and c_j, computing attribute similarity involves $|distinct(c_i)| \times |distinct(c_j|$ similarity computations, rather than $|c_i| \times |c_j|$ computations, where $distinct(c)$ is the number of unique reference attribute values in cluster c. Finally, if we have some way of finding an 'average' attribute value for each of the clusters, the problem is much simpler.

This last point also touches on the issue of the semantics of the attribute similarity in the case where we are clustering in order to perform entity resolution. By design, each cluster corresponds to one resolved entity e. This entity may have been referred to using many different attributes, as is evident from the attributes $r.A$ of the references belonging to that cluster. But if we assume this attribute to be single valued, then there really is one true value of the entity attribute $e.A$. For instance, the reference attributes may have been 'A. Aho', 'Alfred V. Aho', 'A.V Aho' and so on, but the true entity attribute is 'Alfred V. Aho'. We may define this *representative cluster attribute* as the most likely one given the reference attributes in that cluster. So, when computing the attribute similarity of the two entity clusters, it makes more sense to look at the similarity of the two representative attributes for the clusters, rather than considering all of the attributes in each cluster. This makes a difference in situations where we have detected differences which may be due to typographical errors in reference attributes within an entity cluster but have still been able to resolve them correctly. When computing similarity values for this entity cluster we possibly do not want these noisy reference attributes to play a role.

5.2 Graph-based Similarity

Next, we address the graph-based similarity measure between two entity clusters considering the entities that they are related to via the observed edges. There are many possible ways to define this similarity. We explore some possibilities here and focus on relevant issues.

5.2.1 Edge Detail Similarity

Just as in the case of the attributes, each entity cluster is associated with a set of edges to which the references contained in it belong. Recall that each reference r is associated with an observed hyper-edge $r.H$. Then

the edge set $c.H$ for an entity cluster c may be defined as $c.H = \{h \mid r.H = h,\ r \in c\}$. To ground this in terms of our example in Figure 4, after we have resolved all the entities so that cluster c_1 refers to the 'Aho' entity, cluster c_2 to the 'Ullman' entity and cluster c_3 to the 'Johnson' entity, then the edge set for the 'Aho' cluster is $c_1.H = \{h_1, h_2, h_3, h_4\}$ having the edges corresponding to the four papers written by 'Aho'. The edge set $c_2.H$ for 'Ullman' is identical while that for 'Johnson' is $c_3.H = \{h_1, h_4\}$.

We define a similarity measure for a pair of edges, so that given this pair-wise similarity measure, we can again use a linkage metric like single link to measure the relational similarity between two clusters. First we define a pairwise similarity measure between two hyper-edges. We have already noted that what we want to consider for relational similarity are the cluster labels of the references in each edge, and not their attributes. So for an edge, we consider the multi-set of entity labels, one for each reference associated with it.[1] We will employ the commonly-used Jaccard similarity measure over these multi-sets of entity labels. Specifically, let $label(h_i)$ be the set of entity labels for an edge h. Then for a pair of edges h_i and h_j, we define their similarity as

$$sim(h_i, h_j) = \frac{|label(h_i) \bigcap label(h_j)|}{|label(h_i) \bigcup label(h_j)|}$$

Given this pairwise similarity measure for edges, we can use an aggregation operation like max to calculate the graph-based similarity between two entity clusters as follows:

$$sim_{graph}(c_i, c_j) = max_{(h_i, h_j)}\{sim(h_i, h_j)\}$$
$$h_i \in c_i.H, h_j \in c_j.H$$

This we will call the *edge detail similarity* between two clusters since it explicitly considers every edge associated with each entity cluster. Observe that computing the similarity between two edges is linear in the average number of references in an edge, while computing the edge detail similarity of two clusters is quadratic in the average number of edges associated with each cluster. An issue with using max as the aggregation function is that a high similarity between a single pair of edges may not always be a strong evidence for determining duplicates in domains where the edges can be noisy as well. In the absence of noise however, it is very likely that two authors with similar names are duplicates if both are observed to write just one paper with the same set of collaborating authors. Another advanage of using max, as compared with other functions like avg, is that it allows cluster similarities to be updated efficiently when related clusters merge.

5.2.2 Neighborhood Similarity

Clearly, an issue with edge detail similarity is the computational complexity. The solutions discussed in the context of attribute similarity cannot be extended for edges. It is not trivial to incrementally update edge detail similarity when clusters merge. Also, the reduction due to repeating entity patterns in edges is not expected to be as significant as for attributes.

A second and more pertinent issue is whether the detailed pair-wise similarity computation for edges is really necessary for the task at hand. While it may make sense for some applications, it may not be necessary to look at the structure of each edge separately for the task of graph-based entity resolution. Using a more concrete example, for two author entities e_1 and e_1' to be considered similar, it is not necessary for both of them to co-author a paper with entities e_2, e_3 and e_4 together. Instead, if e_1 participates in an edge $\{e_1, e_2, e_3, e_4\}$ and e_1' participates in three separate edges $\{e_1', e_2\}$, $\{e_1', e_3\}$ and $\{e_1', e_4\}$, then that also should count as significant graph-based evidence for their being the same author entity (if they have similar attributes as well). In other words, whether or not two entity clusters with similar attributes have the same edge structures, if they have the *same neighbor clusters* to which they have edges, that is sufficient graph-based evidence for bibliographic entity resolution.

We will now formalize the notion of neighborhood clusters for an entity cluster. Recall that we have defined $label(l)$ as the multi-set of entity labels for an edge h and for an entity cluster c, $c.H$ is the set of edges associated with it. Then we can formally define the neighborhood multi-set $c.N$ for c as follows:

$$c.N = \bigcup_m label(h_i) h_i \in c.H$$

where $\bigcup_m$ is the multi-set union operator. Intuitively, we collapse the edge structure and just look at how many times c has participated in the same edge with another entity cluster. Note that we do not include c itself among its neighbors. Returning to our example from Figure 2, once the entities have been resolved and clusters c_1, c_2 and c_3 correspond to entities 'Aho', 'Ullman' and 'Johnson' respectively, the labels of the edges for cluster c_1 are $\{c_1, c_2, c_3\}$ for h_1, $\{c_1, c_2\}$ for h_2, $\{c_1, c_2\}$ for h_3 and $\{c_1, c_2, c_3\}$ for h_4. Then the neighborhood multi-set $c_1.N$ is the collection of all cluster labels occurring in these four edges: $c_1.N = \{c_2, c_2, c_2, c_2, c_3, c_3\}$. Now, for the graph-based similarity measure between two entity clusters, we take the Jaccard similarity between their neighborhood multi-sets.

$$sim_{graph}(c_i, c_j) = \frac{|c_i.N \bigcap c_j.N|}{|c_i.N \bigcup c_j.N|}$$

We will call this the *neighborhood similarity* between the two entity clusters. Computing and updating neighborhood similarity is significantly cheaper computationally compared to the *edge detail similarity*. It is linear in the average number of neighbors per entity. Note that the neighborhood of an entity cluster has a natural interpretation for author entities and collaborator relationships. The neighborhood is the multi-set of collaborators for an author entity. Also, by avoiding looking at individual links, neighborhood similarity is less susceptible to noisy edges than edge detail similarity.

Observe that when all edges are binary, the similarity measure between two edges becomes boolean, as does the edge detail similarity between two clusters. Also, the edges can be mapped directly to the neighborhood set. Accordingly using the edge detail similarity is not appropriate any more and neighborhood similarity is sufficient.

5.3 Negative Evidence from Graphs

So far, we have considered graph structure as additional evidence for two author references actually referring to the same underlying author entity. However, graph-based ev-

[1] Though in general we can have a multi-set of entity labels for an edge, all the labels are likely to be distinct when the edges represent co-authorship relations.

idence can be negative as well. A 'soft' aspect of negative evidence is directly captured by the combined similarity measure. Imagine two references with identical names. If we only consider attributes, their similarity would be very high. However, if they do not have any similarity in their edge structures, then we are less inclined to believe that they correspond to the same entity. This is reflected by the drop in the their overall similarity when the graph-based similarity measure is factored in as well.

We may also imagine stronger graph-based constraints for clustering. In many relational domains, there is the constraint that no two references appearing in the same edge can be duplicates of each other. To take a real bibliographic example, if a paper is observed to be co-authored by 'M. Faloutsos', 'P. Faloutsos' and 'C. Faloutsos' then probably these references correspond to distinct author entities. We have such constraints for every edge that has more than one reference. This can be taken into account by the graph-based similarity measure. The similarity between two cluster pairs is zero if merging them violates any relational constraint.

6. GRAPH-BASED CLUSTERING FOR ENTITY RESOLUTION

Given the similarity measure for a pair of entity clusters, we can use a greedy agglomerative clustering algorithm that finds the closest cluster pair at each step and merges them. Here we discuss several implementation and performance issues regarding graph-based clustering algorithms for entity resolution (**GBC-ER**).

6.1 Blocking to find Potential Resolution Candidates

Initially, each reference belongs to a distinct entity cluster and the algorithm iteratively looks to find the closest pair of clusters. Unless the datasets are small, it is impractical to consider all possible pairs as potential candidates for merging. Apart from the scaling issue, most pairs checked by an $O(n^2)$ approach will be rejected since usually about 1% of all pairs are true duplicates. Blocking techniques are usually employed to rule out pairs which are certain to be non-duplicates. Bucketing algorithms may be used to create groups of similar reference attributes and only references within the same bucket are considered as potential duplicates.

The algorithm inserts all of these potential duplicate pairs into a priority queue considering their similarities. Then it iteratively picks the pair with the highest similarity and merges them. The algorithm terminates when the similarity for the closest pair falls below a threshold.

6.2 Graph-based Bootstrapping for Entity Clusters

The graph-based clustering algorithm begins with each reference assigned to a separate entity cluster. So to start with, the clusters are disconnected and there is no graph-based evidence to make use of between clusters. As a result, all initial merges occur based solely on attribute similarity, but we want to do this in a relatively conservative fashion. One option is to assign the same initial cluster to any two references that have attributes v_1 and v_2, where either v_1 is identical to v_2, or v_1 is an initialed form of v_2. For example, we may merge 'Alfred Aho' references with other 'Alfred Aho' references or with 'A. Aho' references. However, for domains where last names repeat very frequently, like Chinese, Japanese or Indian names, this can affect precision quite adversely. For the case of such common last names[2], the same author label can be assigned to pairs only when they have document co-authors with identical names as well. For example, two 'X. Wang' references will be merged when they are co-authors with 'Y. Li'. (We will also merge the 'Y. Li' references.) This should improve bootstrap precision significantly under the assumption that while it may be common for different authors to have the same (initialed) name, it is extremely unlikely that they will collaborate with the same author, or with two other authors with identical names.

In addition to using a secondary source for determining common names, a data-driven approach may also be employed. A [last name, first initial] combination in the data is ambiguous, if there exist multiple first names with that initial for the last name. For example, though 'Zabrinsky' is not a common last name, 'K. Zabrinsky' will be considered ambiguous if 'Ken Zabrinsky' and 'Karen Zabrinsky' occur as author references in the data. Ambiguous references or references with common last names are not bootstrapped in the absence of relational evidence in the form of co-authorships, as described above.

6.3 Finding the Closest Entity Clusters

Once potential duplicate entity clusters have been identified and clusters have been bootstrapped, the algorithm 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 removes from the candidate set all pairs that involve either c_i or c_j and inserts relevant pairs for c_{ij}. It also updates the similarity measures for the 'related' cluster pairs. All of these tasks need to be performed efficiently to make the algorithm scalable.

For efficient extraction of the closest pairs and updating of the similarities, an indexed max-heap data structure can be used. In addition to the actual similarity value, each entry in the max-heap maintains pointers to the two clusters whose similarity it corresponds to. Also, each cluster c_i indexes all the entries in the max-heap that stores similarities between c_i and some other cluster c_j. The maximum similarity entry can be extracted from the heap in $O(1)$ time. For each entry whose similarity changed because of the merge operation, the heap can be updated in $O(\log n)$ steps, where n is the number of entries in the heap. The requirement therefore is to be able to efficiently locate the entries affected by the merge. First, the set includes all entries that involve c_i or c_j. These are easily retrieved by the indexes maintained by c_i and c_j and then one of the entries is replaced by c_{ij} and similarities are recomputed. The other set of entries whose similarities are affected are those that are *related* to c_i or c_j as neighbors. These may be retrieved efficiently as well if each cluster maintains indexes to its neighbors. For the initial set of clusters, the neighbors are those that correspond to references in the same edge. As clusters are merged, the new cluster inherits the neighbors from both of its parents. It also inherits the heap indexes from its two parents.

[2]A list of common last names is available at http://en.wikipedia.org/wiki/List_of_most_popular_family_names

6.4 Evaluation Measures

Most approaches to entity resolution have viewed the problem as a pairwise classification task. However, the duplicate relation semantically defines an equivalence partitioning on the references and our approach is to find clusters of duplicate references. From this perspective, it would be preferable to evaluate the clusters directly instead of an evaluation of the implied binary duplicate predictions.

Given the author entity label $l_i \in L$ for each reference in the evaluation, if the resolution is perfect, then each generated cluster will exactly correspond to one author entity. When this is not the case, we may imagine two different scenarios — either a cluster includes references that have different entity labels, or it fails to capture all references that have a particular entity label.

The first measure is defined over generated clusters $c_i \in C$. Ideally, each cluster should be homogeneous — all references in an cluster should have the same entity label. The greater the number of references to different entities in a cluster, the worse the quality of the cluster. Rather than naively counting the different entity labels in a cluster, we can measure the entropy of the distribution over entity labels in a cluster. This we call *cluster diversity*. As an example, suppose there are ten references in a generated entity cluster; five of them correspond to one entity, three to a second entity and the remaining two to a third entity. Then the probability distribution over entity labels for the references in this cluster is $\langle 0.5, 0.3, 0.2 \rangle$. In general, if this distribution be $\langle p_1, p_2, \ldots, p_k \rangle$, then the entropy is defined as $\sum_k -p_i \log p_i$. Note that the entropy is 0 when all references have the same real entity label and the label distribution is $\langle 1.0 \rangle$. The entropy is highest for a uniform distribution. Also, a uniform distribution over $k+1$ labels has higher entropy than that over k labels. The weighted average of the diversity of clusters as considered as the first quality measure, where the entropy of each cluster is weighted by the number of references in it. To be more precise, the combined diversity over a clustering C is:

$$Div(C) = \sum_{c_i \in C} \frac{|c_i|}{N} div(c_i)$$

where N is the total number of references, and $div(c_i)$ is the entropy of the entity label distribution in cluster c_i.

However, just minimizing the cluster diversity is not enough since zero diversity can be achieved by assigning each reference to a distinct cluster. Obviously this is not a good clustering since references that correspond to the same entity are dispersed over many different clusters. So a second measure of cluster quality is *entity dispersion* defined over each entity label l_i. Here for each entity, we consider the distribution over different cluster labels assigned to the references for that entity. As for cluster diversity, the entropy is measured for the distribution over cluster labels for each entity. We look at the weighted average of the entity dispersions, where the weight of an entity label is the number of references that have that label:

$$Disp(C) = \sum_{l_i \in L} \frac{|l_i|}{N} disp(l_i),$$

where $disp(l_i)$ is entropy of the cluster label distribution for the references having entity label l_i.

Note that lower values are preferred for dispersion and diversity. A perfect clustering will have zero cluster diversity and zero entity dispersion. This is practically not achievable in most cases and a decrease in one will usually mean an increase in the other. We can plot the dispersion-diversity curve for each set of generated clusters. This shows the value of diversity achieved for any value of dispersion. Also, we can observe how the dispersion and diversity change as clusters are merged iteratively.

7. EXPERIMENTAL EVALUATION

To illustrate the power of graph-based entity resolution, we present evaluations on two citation datasets from different research areas and compare an implementation of the graph-based entity resolution algorithm (**GBC-ER**) with others based solely on attributes.

The first of the citation datasets is the CiteSeer dataset containing citations to papers from four different areas in machine learning, originally created by Giles et al.[16]. This has 2,892 references to 1,165 authors, contained in 1,504 documents. The second dataset is significantly larger; arXiv (HEP) contains papers from high energy physics used in KDD Cup 2003[3]. This has 58,515 references to 9,200 authors, contained in 29,555 papers. The authors for both datasets have been hand-labeled.[4]

To evaluate the algorithms, we measure the performance of the algorithms for detecting duplicates in terms of the traditional precision, recall and F1 on pairwise duplicate decisions in addition to our proposed dispersion-diversity measure for clusters. It is practically infeasible to consider all pairs, particularly for HEP, so a 'blocking' approach is employed to extract the potential duplicates. This approach retains $\sim$99% of the true duplicates for both datasets. The number of potential duplicate pairs of author references after blocking is 13,667 for CiteSeer and 1,534,661 for HEP.

As our baseline (**ATTR**), we compare with the hybrid *SoftTF-IDF* measure [11] that has been shown to outperform other unsupervised approaches for text-based entity resolution. Essentially, it augments the TF-IDF similarity for matching token sets with approximate token matching using a secondary string similarity measure. Jaro-Winkler is reported to be the best secondary similarity measure for *SoftTF-IDF*. We also experiment with the Jaro and the Scaled Levenstein measures. However, directly using an off-the-shelf string similarity measure for matching names results in very poor recall. From domain knowledge about names, we know that first and middle names may be initialed or dropped. A black-box string similarity measure would unfairly penalize such cases. To deal with this, **ATTR** uses string similarity only for last names and *retained* first and middle names. In addition, it uses drop probabilities p_{DropF} and p_{DropM} for dropped first and middle names, initial probabilities p_{FI} and p_{MI} for correct initials and p_{FIr} and p_{MIr} for incorrect initials. The probabilities we used are 0.75, 0.001 and 0.001 for correctly initialing, incorrectly initialing and dropping the first name, while the values for the middle name are 0.25, 0.7 and 0.002. We arrived at these values by observing the true values in the datasets and then

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

[4]We would like to thank Aron Culotta and Andrew McCallum for providing the author labels for the CiteSeer dataset and David Jensen for providing the author labels for the HEP dataset. We performed additional cleaning for both.

hand-tuning them for performance. Our observation is that baseline resolution performance does not vary significantly as these values are varied over reasonable ranges.

ATTR only reports pairwise match decisions. Since the duplicate relation is transitive, we also evaluate **ATTR*** which removes inconsistencies in the pairwise match decisions in **ATTR** by taking a transitive closure. Note that this issue does not arise with **GBC-ER**; it does not make pairwise decisions. All of these unsupervised approaches **ATTR**, **ATTR*** and **GBC-ER** need a similarity threshold for deciding duplicates. We consider the best $F1$ that can be achieved over all thresholds.

Table 1: Performance of ATTR, ATTR* and GBC using neighborhood and edge detail similarity in terms of F1 using various secondary similarity measures with SoftTF-IDF. The measures compared are Scaled Levenstein (SL), Jaro (JA), and Jaro Winkler (JW).

	CiteSeer			HEP		
	SL	JA	JW	SL	JA	JW
ATTR	0.980	**0.981**	0.980	**0.976**	0.976	0.972
ATTR*	0.989	**0.991**	0.990	**0.971**	0.968	0.965
GBC(Nbr)	**0.994**	**0.994**	**0.994**	0.979	**0.981**	0.981
GBC(Edge)	**0.995**	**0.995**	**0.995**	0.982	**0.983**	0.982

Table 2: Best F1 and corresponding precision and recall for ATTR, ATTR* and GBC-ER with neighborhood and edge detail similarity for CiteSeer and HEP datasets.

	CiteSeer			HEP		
	P	R	F1	P	R	F1
ATTR	0.990	0.971	0.981	0.987	0.965	0.976
ATTR*	0.992	0.988	0.991	0.976	0.965	0.971
GBC(Nbr)	0.998	0.991	**0.994**	0.990	0.972	**0.981**
GBC(Edge)	0.997	0.993	**0.995**	0.992	0.974	**0.983**

Table 1 records F1 achieved by the four algorithms with various string similarity measures coupled with SoftTF-IDF while Table 2 shows the best F1 and the corresponding precision and recall for the four algorithms for each dataset over all secondary similarity measures. The recall includes blocking, so that the highest recall achievable is 0.993 for CiteSeer and 0.991 for HEP.

The best baseline performance is with Jaro as secondary string similarity for CiteSeer and Scaled Levenstein for HEP. It is also worth noting that a baseline without initial and drop probabilities scores below 0.5 F1 using Jaro and Jaro-Winkler for both datasets. It is higher with Scaled Levenstein (0.7) but still significantly below the augmented baseline. Transitive closure affects the baseline differently in the two datasets. While it adversely affects precision for HEP reducing the F1 measure as a result, it improves recall for CiteSeer and thereby improves F1 as well.

GBC-ER outperforms both forms of the baseline for both datasets. Also, for each secondary similarity measure **GBC-ER** with neighborhood similarity outperforms the baselines with that measure and is in turn outperformed by **GBC-ER** using edge detail similarity. For CiteSeer, **GBC-ER** gets close to the highest possible recall with very high accuracy. Improvement over the baseline is greater for HEP. While the improvement may not appear large in terms of F1, note that **GBC-ER** reduces error rate over the baseline by 44% for CiteSeer (from 0.009 to 0.005) and by 29% for HEP (from 0.024 to 0.017). Also, HEP has more than $64,6000$ true duplicate pairs, so that a 1% improvement in F1 translates to more than $6,400$ correct pairs.

Looking more closely at the resolution decisions from CiteSeer, we were able to identify some interesting combinations of decisions by **GBC-ER** that would be difficult or impossible for an attribute-only model. There are instances in the dataset where reference pairs are very similar but correspond to different author entities. Examples include *(liu j, lu j)* and *(chang c, chiang c)*. **GBC-ER** correctly predicts that these are not duplicates. At the same time, there are other pairs that are not any more similar in terms of attributes than the examples above and yet are duplicates. These are also correctly predicted by **GBC-ER** using the same similarity threshold by leveraging common collaboration patterns. The following are examples: *(john m f, john m st)*, *(reisbech c, reisbeck c k)*, *(shortliffe e h, shortcliffe e h)*, *(tawaratumida s, tawaratsumida sukoya)*, *(elliott g, elliot g l)*, *(mahedevan s, mahadevan sridhar)*, *(livezey b, livezy b)*, *(brajinik g, brajnik g)*, *(kaelbing l p, kaelbling leslie pack)*, *(littmann michael l, littman m)*, *(sondergaard h, sndergaard h)* and *(dubnick cezary, dubnicki c)*. An example of a particularly pathological case is *(minton s, minton andrew b)*, which is the result of a parse error. The attribute-only baselines cannot make the right prediction for both these sets of examples simultaneously, whatever the decision threshold, since they consider names alone.

Figures 5 and 6 show how performance varies for **GBC-ER** for the two datasets with varying combination weight α for attribute and graph-based similarity. Recall that when α is 0, the similarity is based only on attributes and when α is 1 it is wholly graph-based. The plots show that **GBC-ER** with both neighborhood and edge detail similarity outperform the baselines over *all* values of α. Note that **GBC-ER** takes advantage of graph-based bootstrapping in these experiments which explains why it is better than the baseline even when α is 0. The best performance for CiteSeer is around 0.5 while for HEP performance peaks around 0.1 and then trails off. It can also be observed that edge detail similarity is more stable in performance over varying α than neighborhood similarity. Significantly, once clusters have been bootstrapped using attribute and graph-based evidence, **GBC-ER** outperforms the baselines even when α is 1, which means that attributes are being overlooked altogether and clusters are merged using graph-based evidence alone.

The first row of plots in Figure 5 use single link criterion for attribute similarity while those in the second measure similarity between representative cluster attributes. The differences in the two plots are observable for low values of α when attribute similarity plays a more dominant role. It can be seen that the single link approach performs better than the cluster representative approach for all secondary similarity measures. The curves become identical for higher values of α when attributes matter less. A similar trend was observed for HEP as well.

Figure 7 shows performance of **GBC-ER** without using

graph-based bootstrapping. When α is 0, **GBC-ER** is identical to **ATTR*** which is verified by the results. As α increases from 0, performance improves over the baseline and then drops again. For HEP, performance falls sharply with higher α with neighborhood similarity. Edge detail similarity however still performs surprisingly well. Even when α is 1, it does better than the baseline for CiteSeer and is able to achieve close to 0.9 F1 for HEP. This suggests that edge detail is a reliable indicator of identity even without considering attributes. It should however be noted that these results include blocking, which uses attributes to find potential duplicates. This suggests that given people with similar names, it is possible to identify duplicates with a high degree of reliability using edge detail similarity alone.

Figure 8 shows the precision recall characteristics for the four algorithms with Jaro as secondary similarity. Plot (a) shows that all algorithms perform well for CiteSeer. Plot (b) concentrates on the region of difference between the algorithms. Still the curves for edge detail and neighborhood similarity are almost identical, as is the case for HEP in plot (c). But they consistently remain over the baselines for both datasets. Observe that **ATTR*** dominates **ATTR** for CiteSeer but the roles reverse for HEP. Note that **GBC-ER** starts above 90% recall. This is by virtue of graph-based bootstrapping. The corresponding precision is high as well validating the effectiveness of our graph-based bootstrapping scheme. The characteristics with other secondary similarity measures are similar.

Figure 9 illustrates how dispersion-diversity measures may be used. We show the results only for HEP with Jaro similarity. Other plots are similar. Recall that while values closer to 1 are preferred for precision and recall, values closer to 0 are better for dispersion and diversity so that points on the dispersion-diversity curves in plot (a) that are closer to $(0, 0)$ indicate better performance. Plot (a) shows that dispersion-diversity curves are very similar to precision-recall curves but they evaluate the clusters directly. Note that since **ATTR** makes pair-wise decisions and does not generate clusters of duplicate references, it cannot be evaluated using this approach. Plots (b-d) plots diversity and dispersion against number of clusters. Again, observe how graph-based bootstrapping significantly lowers the initial number of clusters for **GBC-ER** thereby starting off with much lower dispersion. The initial number of clusters is lowered by 83% for HEP and by 58% for CiteSeer. That it is accurate can be inferred from the significantly lower diversity at the same number of clusters compared to **ATTR***. The number of real author entities for HEP is 8967, which is shown using a vertical line in plots (b-d). Plot (b) highlights the difference in dispersion between the three algorithms and we can see that **GBC-ER** improves dispersion over **ATTR*** at the correct number of entity clusters and that edge detail performs better than neighborhood similarity.

Finally, we look at the execution times of the algorithms. All experiments were run on a 1.1GHz Dell PowerEdge 2500 Pentium III server. Table 3 records the execution times in CPU seconds of the baselines and different versions of **GBC-ER** on the CiteSeer and HEP datasets. **GBC-ER** expectedly takes more time than the baselines. But it is quite fast for CiteSeer taking only twice as much time as **ATTR***. It takes longer for HEP; about 7 times as long compared to the baseline. While using edge detail is more expensive than neighborhood similarity, it does not take significantly longer for either dataset. The complexity of edge detail depends on a number of factors. It grows quadratically with the average number of edges per entity and linearly with the average number of references in each edge. While the average edge size is the same for both datasets, the average number of edges per entity is 2.5 for CiteSeer and 6.36 for HEP which explains the difference in execution times. In contrast, complexity of neighborhood similarity is linear in the average number of neighbors per entity, which is 2.15 for CiteSeer and 4.5 for HEP. Separately, we expected single link clustering to be more expensive than using representative attributes for clusters. While this is true for HEP, the trend reverses for CiteSeer. This may be explained by taking into account the added overhead of updating the cluster representative every time new references are added to a cluster. Since CiteSeer only has an average of 2.5 references per entity, the cost of a naive updation scheme for cluster representatives overshadows the small gain in computing attribute similarity for clusters.

Table 3: Execution time of GBC-ER, ATTR and ATTR* in CPU seconds for CiteSeer and HEP datasets.

	CiteSeer	HEP
ATTR	1.88	162.13
ATTR*	2.30	217.37
GBC(Nbr w/ Single Link)	3.14	543.83
GBC(Edge w/ Single Link)	3.18	690.44
GBC(Nbr w/ Cluster Rep.)	3.65	402.68
GBC(Edge w/ Cluster Rep.)	3.75	583.58

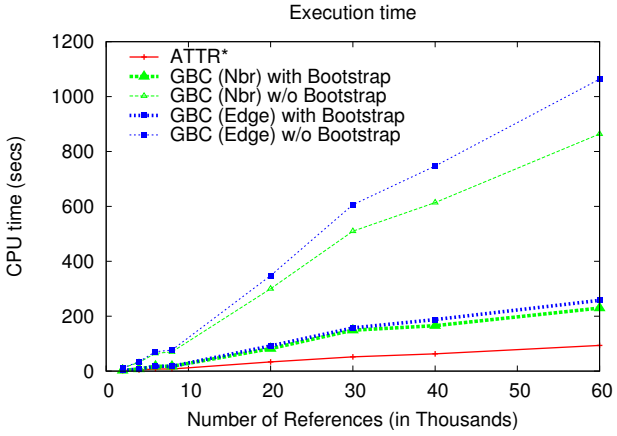


Figure 10: Execution time for ATTR* and GBC-ER with neighborhood and edge detail similarity over varying number of references in synthetic datasets.

To see how the algorithms scale with increasing number of references in the dataset, we used a synthetic generator for ambiguous data. We preserved features of the real datasets wherever possible, like the average number of references and edges per entity, the degree of neighborhood for each entity and the average number of references per edge. The execu-

tion times of **ATTR*** and different versions of **GBC-ER** are plotted in Figure 10 against varying number of references in the dataset. We would like to stress that the execution time depends on other factors as well like the number of potential duplicate pairs in the data, which we were not able to control directly. So these numbers should not be compared with the execution times on the real datasets but instead should serve only for a comparative study of the different algorithms. The curves confirm that **GBC-ER** takes longer than the baseline but they also show that the trend is roughly linear in the number of references for all versions of it. The plots also show the significant speedup that is achieved with graph-based bootstrapping in addition to the performance benefits that it provides.

8. CONCLUSION

In this chapter, we have seen how graph-based entity resolution may be posed as a clustering problem that augments a general category of attribute similarity measures with graph-based similarity among the entities to be resolved. We looked at two different similarity measures based on graphs and a graph-based clustering algorithm (**GBC-ER**) that shows performance improvements over attribute-based entity resolution in two real citation datasets. The combined similarity measure allows a smooth transition from an attribute-only measure at one extreme to one that is based just on graph-based similarity at the other. Depending on the reliability of the attributes or the relations in the application domain or the particular dataset, it is possible to attach higher weights to either of them.

The graph-based similarity measures are intended to capture the dependencies between resolution decisions. They do so by looking at the current entity labels of related entities. We presented two different graph-based similarity measures that consider relations between entities at different levels of detail. Edge-detail similarity explicitly considers each edge in which an entity participates. Neighborhood similarity reduces the computational complexity involved by collapsing the edge structure and just looking at the set of neighborhood entities for each entity cluster. The execution times for the similarity measures depends on the graph structure in the data. In general, edge detail takes more time than neighborhood similarity, but is a more reliable indicator of identity. Even without considering attributes at all in the similarity measure, edge detail is able to achieve high accuracy in determining duplicates. In fact, in one of our datasets, even without using attributes at all, it does better than attribute-only baselines. This suggests that in domains where attributes are extremely unreliable or perhaps unavailable, it may still be possible to discover identity going by the graph patterns alone.

We consider two different options for measuring attribute similarity between entity clusters. The first is the single linkage criterion that looks at all pairwise attribute similarities between two clusters and chooses the highest one. This reduces to transitive closure over pairwise attribute decisions that is appropriate for entity resolution. This measure is expected to be computationally intensive and therefore we propose an alternative measure that constructs the representative attribute for each entity cluster and then measures similarities between these representatives only. However, our experiments show that the single linkage measure performs better in practice. Also, unless the representatives are recomputed quickly as clusters expand, this approach does not come with significant improvements in execution time either.

We evaluate a graph-based bootstrapping approach for initializing the entity clusters quickly and accurately. Experimental results show that bootstrapping reduces the initial number of clusters by 83% on our larger dataset without significantly compromising on precision. We show that graph-based bootstrapping also improves performance by uncovering patterns quickly that the graph-based similarity measures can leverage.

Since **GBC-ER** considers relational similarities which are expensive to compute and updates similarities iteratively, it is expectedly more costly than performing attribute similarity. However, our experiments on synthetic data show that both graph-based similarity measures scale gracefully over increasing number of references in the dataset. Also, 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 edge appear randomly. There the user has the choice of falling back on traditional attribute similarity or setting α to assign lower weights for graph-based similarity.

In summary, entity resolution is an area that has been attracting growing attention to address the influx of structured and semi-structured data from a multitude of heterogeneous sources. Accurate resolution is important for a variety of reasons ranging from cost-effectiveness and reduction in data volume to accurate analysis for critical applications. In the case of graph data, it is especially important to look at entity resolution from a graph-based perspective. We have found graph-based entity resolution to be a powerful and promising approach that combines attribute similarity with relational evidence and shows improved performance over traditional approaches.

9. ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. 0308030 and 0438866.

10. 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] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, 2002.

[3] I. Bhattacharya and L. Getoor. Deduplication and group detection using links. In *Proceedings of the 10th ACM SIGKDD Workshop on Link Analysis and Group Detection (LinkKDD-04)*, August 2004.

[4] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *Proceedings of the SIGMOD 2004 Workshop on Research Issues on Data Mining and Knowledge Discovery*, June 2004.

[5] I. Bhattacharya and L. Getoor. A latent dirichlet model for entity resolution. Technical report, University of Maryland, College Park, 2005.

[6] 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.

[7] M. Bilgic, L. Licamele, L. Getoor, and B. Shneiderman. D-dupe: An interactive tool for entity resolution in social networks. In *The 13th Interantional Symposium on Graph Drawing (Poster)*, Limerick, Ireland, September 2005.

[8] 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.

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

[10] W. W. Cohen, H. Kautz, and D. McAllester. Hardening soft information sources. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD-2000)*, pages 255–259, Boston, MA, August 2000.

[11] 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.

[12] 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.

[13] A. Doan, Y. Lu, Y. Lee, and J. Han. Object matching for data integration: A profile-based approach. In *Proceedings of the IJCAI Workshop on Information Integration on the W eb*, Acapulco, MX, August 2003.

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

[15] D. Florescu, D. Florescuand, E. Simon, and D. Shasha. An extensible framework for data cleaning. In *ICDE '00: Proceedings of the 16th International Conference on Data Engine ering*, page 312. IEEE Computer Society, 2000.

[16] 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.

[17] L. Gravano, P. Ipeirotis, N. Koudas, and D. Srivastava. Text joins for data cleansing and integration in an rdbms. In *19th IEEE International Conference on Data Engineering*, 2003.

[18] 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.

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

[20] I. Jonyer, L. B. Holder, and D. J. Cook. Graph-based hierarchical conceptual clustering. *Journal of Machine Learning Research*, 2(1-2):19–43, 2001.

[21] D. V. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM International Conference on Data Mining (SIAM SDM)*, Newport Beach, CA, USA, April 21–23 2005.

[22] S. Lawrence, K. Bollacker, and C. L. Giles. Autonomous citation matching. In *Proceedings of the Third International Conference on Autonomous Agents*, New York, NY, May 1999. ACM Press.

[23] 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. to appear.

[24] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS*, 2004.

[25] 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.

[26] B. Milch, B. Marthi, D. Sontag, S. Russell, D. L. Ong, and A. Kolobov. Blog: Probabilistic models with unknown objects. In *Proc. IJCAI*, 2005.

[27] 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.

[28] 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.

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

[30] J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. In *Proceedings of the Text Mining and Link Analysis Workshop, Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

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

[32] Parag and P. Domingos. Multi-relational record linkage. In *Proceedings of 3rd Workshop on Multi-Relational Data Mining at ACM SI GKDD*, Seattle, WA, August 2004.

[33] 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.

[34] V. Raman and J. M. Hellerstein. Potter's wheel: An interactive data cleaning system. In *Proc. VLDB*, 2001.

[35] P. Ravikumar and W. W. Cohen. A hierarchical graphical model for record linkage. In *UAI 2004*, Banff, CA, July 2004.

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

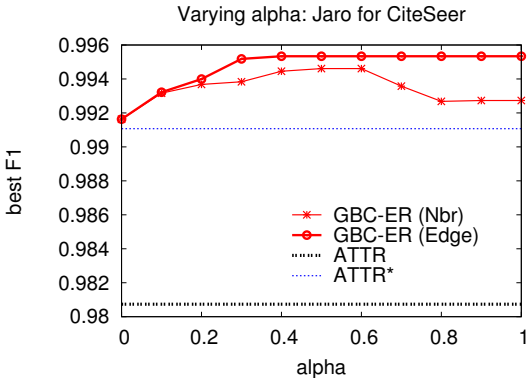
[37] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, 2002.

[38] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

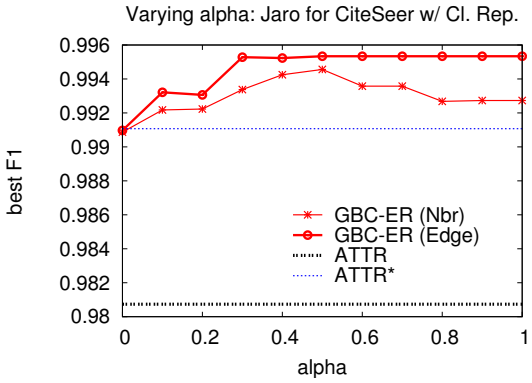
[39] S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Information Systems Journal*, 26(8):635–656, 2001.

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

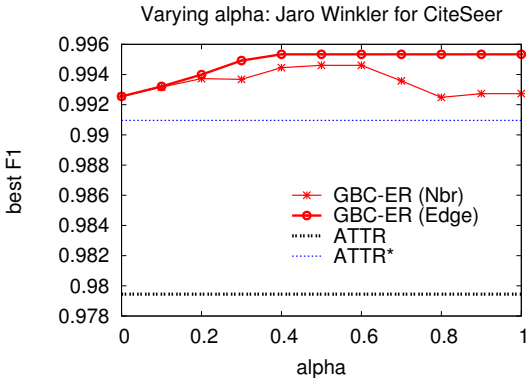
[41] W. E. Winkler. Methods for record linkage and Bayesian networks. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 2002.



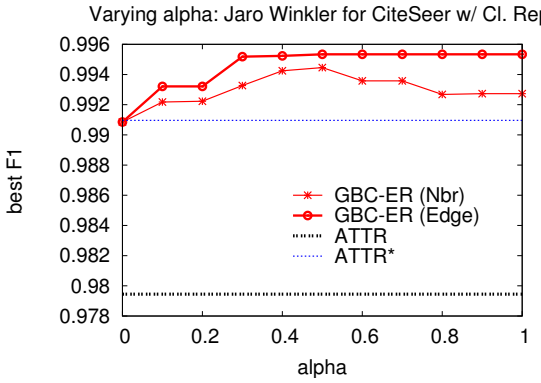
(a)



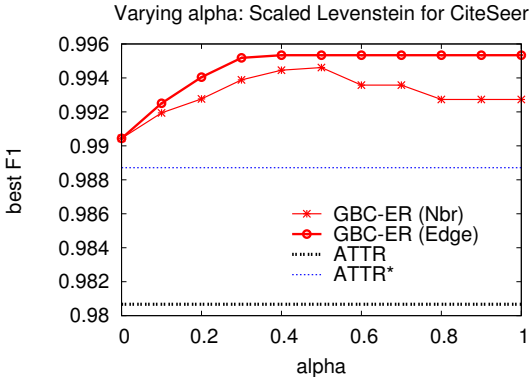
(b)



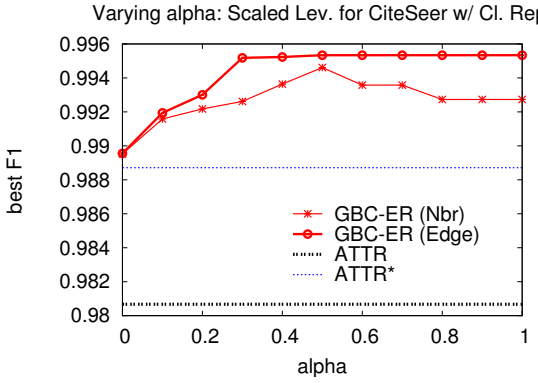
(c)



(d)

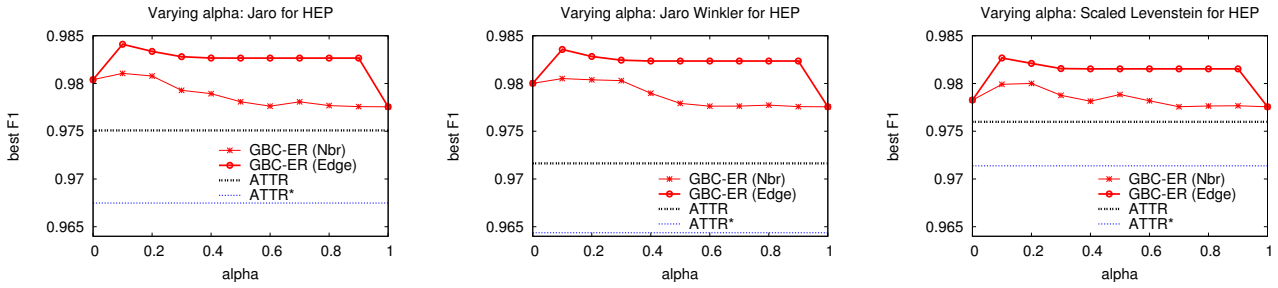


(e)



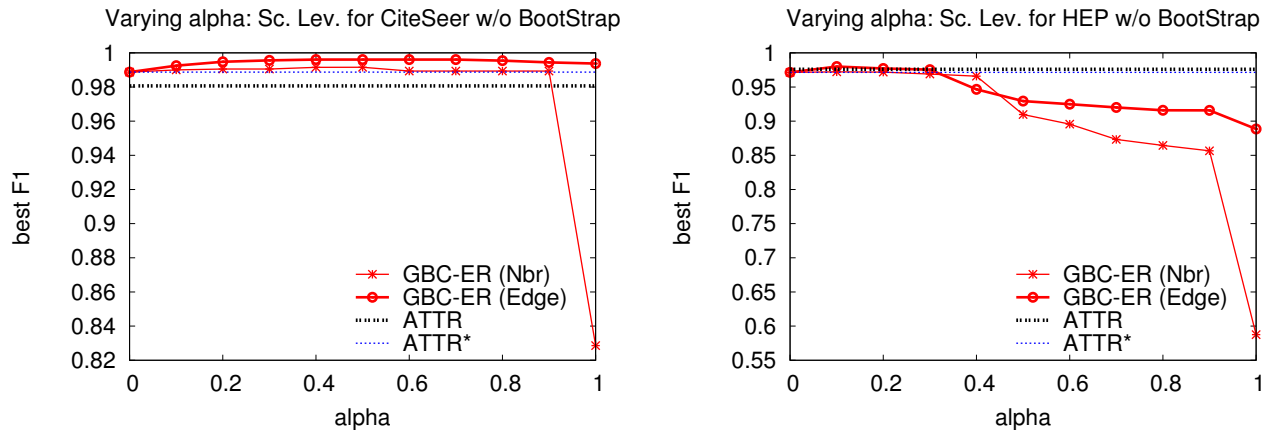
(f)

Figure 5: Best F1 measures achieved by GBC-ER with neighborhood and edge detail similarities over varying combination weight α for CiteSeer. Plots (a,c,e) use single link for attribute similarity with Jaro, Jaro-Winkler and Scaled Levenstein respectively as secondary similarity while plots (b,d,f) use representative attributes for clusters with the same three secondary similarity measures.



(a) (b) (c)

Figure 6: The best F1 measures achieved by GBC-ER with neighborhood and edge detail similarities over varying combination weight α for HEP. Plots (a-c) are with Jaro, Jaro-Winkler and Scaled Levenstein respectively as secondary similarity for attributes.



(a) (b)

Figure 7: The effect of not using bootstrapping for initializing the entity clusters for (a) CiteSeer and (b) HEP using Scaled Levenstein as secondary similarity.

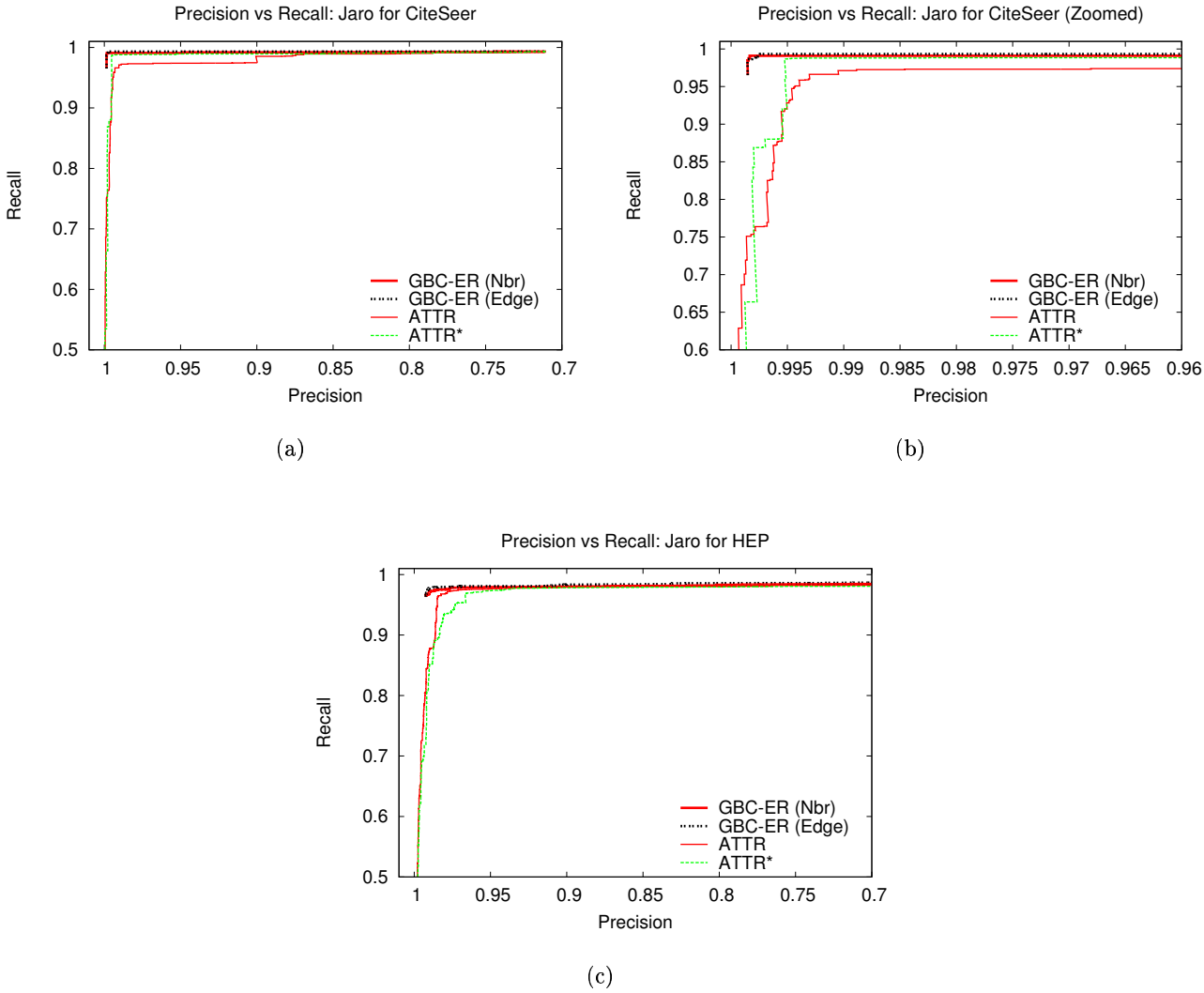
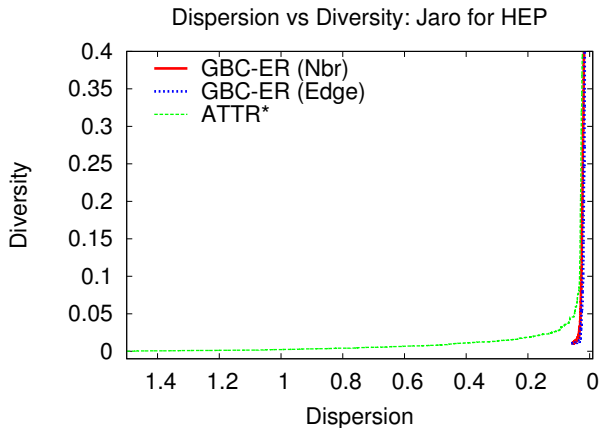
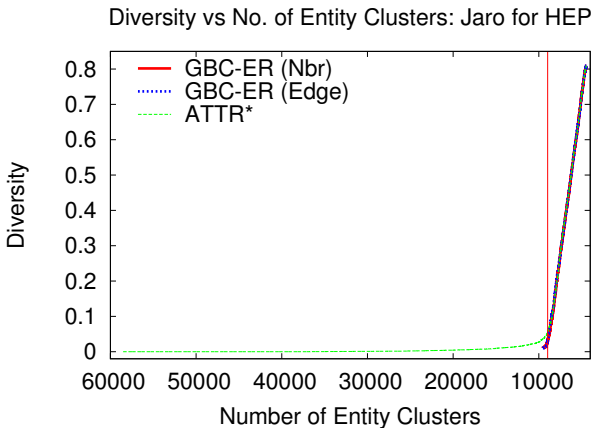


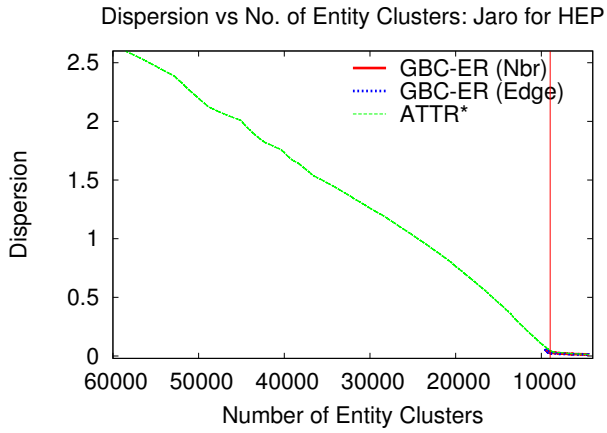
Figure 8: Precision-recall characteristics for (a-b) CiteSeer and (c) HEP using Jaro similarity. Plot (b) highlights the difference between the algorithms from plot (a).



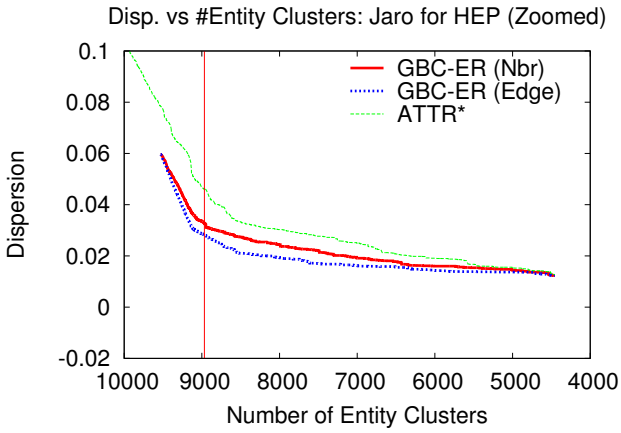
(a)



(b)



(c)



(d)

Figure 9: Performance measure using dispersion and diversity for HEP with Jaro similarity. The plots show (a) dispersion against diversity, (b) diversity and (c-d) dispersion over changing number of entity clusters.