Collective Entity Resolution in Relational Data

I. Bhattacharya, L. Getoor
University of Maryland

Presented by: Srikar Pyda, Brett Walenz
CS590.01 - Duke University

Parts of this presentation from:
http://www.norc.org/PDFs/May%202011%20Personal%20Validation%20and%20Entity%20Resolution%20Conference/GetoorCollectiveEntityResolution052311.pdf

High-level Overview

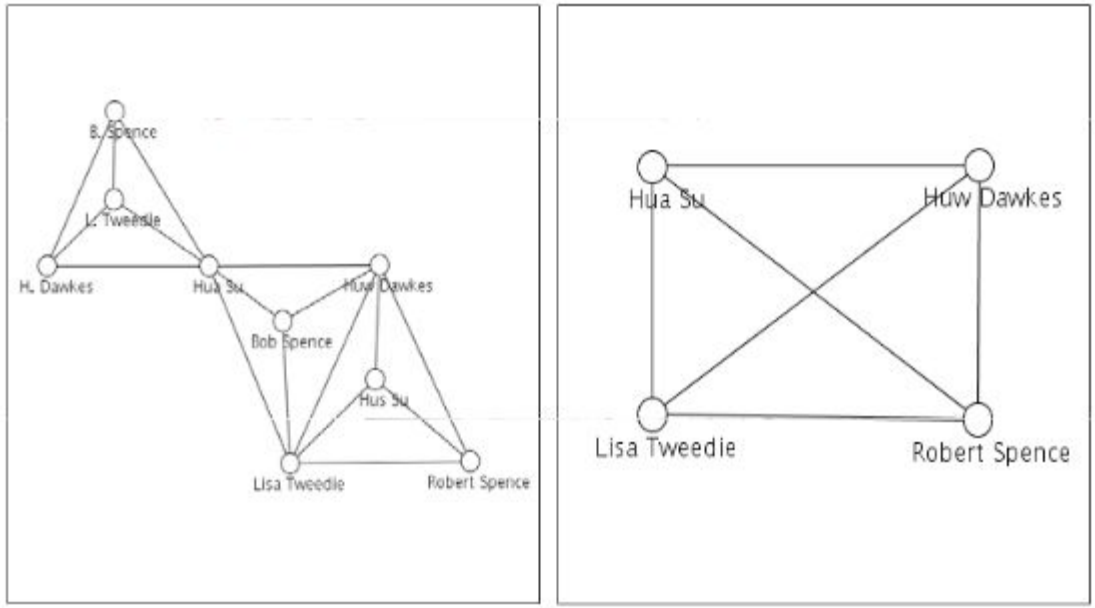
Traditional ER problem: resolve authors in a publications database - discover which authors refer to the same underlying author entity.

Solution: Rather than individual-to-individual mappings and attribute-to-attribute comparisons, use cluster-to-cluster mappings and consider relationships between entity references to resolve matchings

Results: Improvement over baseline approaches, marginal improvement on LDA-ER but massive runtime improvement

Introduction

InfoVis Co-Author Network Fragment



before

after

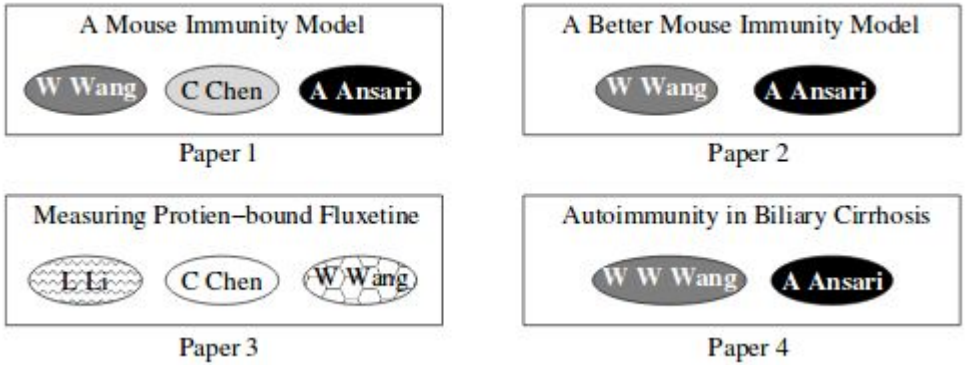
Introduction

References that have similar attributes are more likely to be the same entity if their references have high overlap

Example:

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

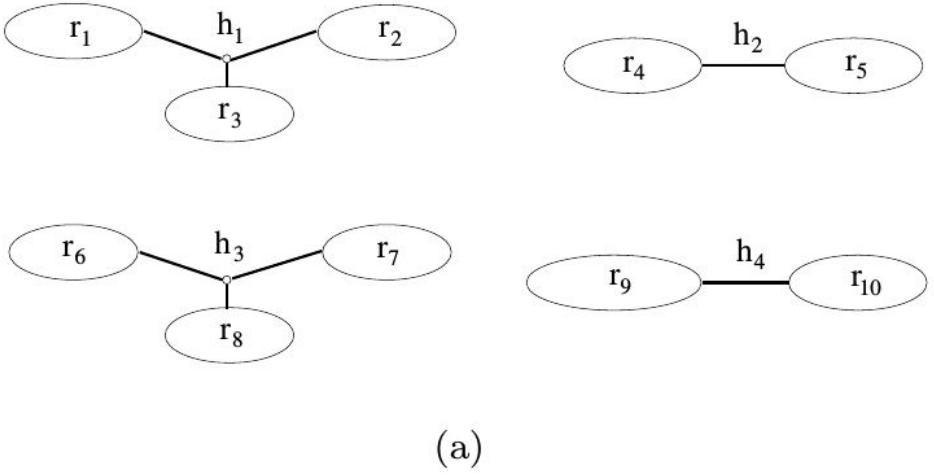
Introduction



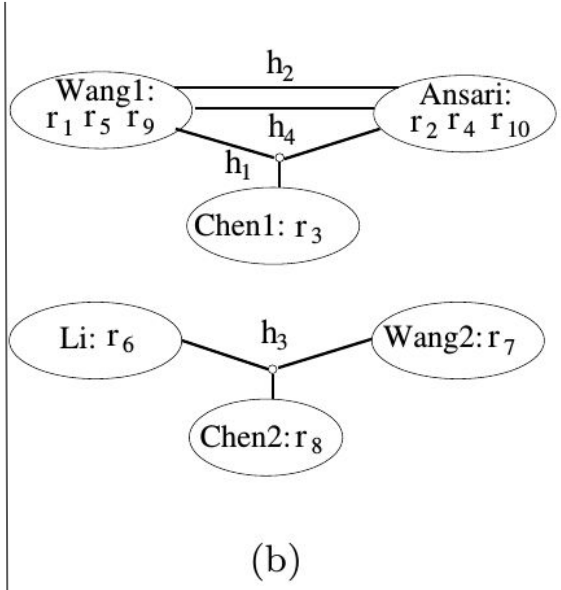
Problem Formulation

1. Set of references ***R*** where each reference *r* has attributes r.A1, r.A2, r.A3, etc.
2. Corresponds to set of unknown entities E = {e1, e2, … en}.
3. References co-occur with each other by a set of hyper-edges ***H***, which may also have attributes H.A1…
4. r.H is set of hyper edges a reference belongs to. Is {0, 1} in this paper.

Problem Formulation



Reference graph (start)



Entity graph (end)

Baseline Approaches

1. Attribute-based Entity Resolution
    a. Fellegi Sunter
    b. Sophisticated string similarity metrics (Jaro-Winkler, Levenstein, etc)
2. Naive Relational Entity Resolution
    a. Treat related references as additional attributes
    b. Define a hyper-edge similarity measure: best pair-wise attribute match between hyper-edges
    c. Take a simple linear combination of attributes and hyper-edge similarities
    d. Improvement over attribute-based, but can still cause errors (see Slide 5)

Collective Entity Resolution

Key idea: Resolutions not made independently, resolution decisions affects other resolutions via hyper-edges.

Solution: Relational (Greedy, Agglomerative) clustering algorithm.

1. *References* placed in clusters representing underlying entity.
2. *Iteratively* pick the highest similarity clusters in queue above a threshold
    a. Use both attribute and reference graph in similarity
3. Merge clusters, remove clusters from queue
4. Recalculate similarity to others using merged cluster
5. Update all neighbor similarities (those that share hyper edges)
6. Repeat until no similarities above threshold.

Collective Entity Resolution

First Step

Define a similarity function $sim(c_i, c_j)$

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

A simple linear combination between attribute similarity and reference similarity.
But what is reference similarity?

Collective Entity Resolution

Reference Similarity - compare references, handle ambiguities, and enforce constraints

1. **Common Neighbors**
2. **Jaccard Coefficient**
3. **Adar Similarity**
4. Using Ambiguities in Data
5. Negative Constraints

Neighborhood Similarity Measures
Common Neighbors

The greater the number of common entities, the higher the possibility that the references refer to the same entity

Set Model: $$CommonNbrScore(c_i, c_j) = \frac{1}{K} \times |Nbr(c_i) \cap Nbr(c_j)|$$

If we want to take into account the frequencies of co-references:

Bag Model: $$CommonNbrScore(c_i, c_j) + Fr(c_i, c_j) = \frac{1}{K} \times |Nbr_B(c_i) \cap Nbr_B(c_j)|$$

Neighborhood Similarity Measures
Jaccard Coefficient

For large values of K, Common Neighbors can ‘overfit’ certain hyper-edges.
Define similarity in terms of *relative* size of neighborhoods.

$$JaccardCoeff(c_i, c_j) = \frac{|Nbr(c_i) \cap Nbr(c_j)|}{|Nbr(c_i) \cup Nbr(c_j)|}$$

Neighborhood Similarity Measures
Adar Similarity

Consider the significance of being similar to another cluster. Make ‘popular’ clusters less impactful than unique clusters.

$$Adar(c_i, c_j) = \frac{\sum_{c \in Nbr(c_i) \cap Nbr(c_j)} u(c)}{\sum_{c \in Nbr(c_i) \cup Nbr(c_j)} u(c)}$$

Jaccard is a special case of Adar when all clusters have the same value of *u.*

Neighborhood Similarity Measures
Adar Similarity with Ambiguity Estimate

Define the uniqueness in terms of how ambiguous (common) a name is.

$$u(c) = \frac{1}{Avg_{(}r \in c)(Amb(r.Name))}$$

Amb(r.Name) is the proportion of references with r.Name divided by the size of all references.

$$Amb(r.A_1) = \frac{|\sigma R.A_1 = r.A_1(R)|}{|R|}$$

Can do better if we have more attributes - can estimate ambiguity of last name by counting number of different first names observed for it.

Negative Constraints from Relationships

Handle hard negative constraints by setting similarity of two clusters to zero if any negative constraints are violated.

In many relational domains, two references appearing in the same hyper-edge cannot refer to the same entity.

Example: C. Faloutsos, M. Faloutsos, P. Faloutsos - must be different entities if in same hyper-edge (paper).

Relational Clustering Algorithm

Find similar references using blocking
Initialize clusters using bootstrapping

For clusters c_i, c_j such that similar(c_i, c_j)
    Insert $\langle sim(c_i, c_j), c_j, c_j \rangle$ into priority queue

While priority queue not empty
    Extract $\langle sim(c_i, c_j), c_i, c_j \rangle$ from queue
    If $sim(c_i, c_j)$ less than threshold, then stop
    Merge c_i and c_j to new cluster c_{ij}
    Remove entries for c_i and c_j from queue
    For each cluster c_k such that similar(c_{ij}, c_k)
        Insert $\langle sim(c_{ij}, c_k), c_{ij}, c_k \rangle$ into queue
    For each cluster c_n neighbor of c_{ij}
        For c_k such that similar(c_k, c_n)
            Update $sim(c_k, c_n)$ in queue

Blocking

Any blocking technique can be used as a black box.

Paper implementation:

1. Make a single pass, assign reference to bucket(s) using an attribute similarity measure.
2. Each bucket maintains a *representative* which an incoming reference is scored against.
3. Any reference scoring above a threshold on *sim(r, rep)* is put into the bucket.

Bootstrapping

If each reference started in a distinct cluster, we could only use attribute-similarity for merges, which may be very inaccurate.

Bootstrap clusters with a high precision pass that only merges exact matches.

Use naive relational approach.

1. Blocking
2. Determine if pair(c1, c2) is a bootstrap candidate
    a. Exact attribute matches with low ambiguity
    b. Ambiguous reference matches with high overlap in hyper-edges
3. Put all bootstrap candidates into clusters

Merging Clusters and Updating Similarities

Maintain a series of indexes for each cluster for fast merging and updating

1. Maintain list of similar clusters for each cluster
2. Track all neighboring clusters
3. Maintain list to all entries involving cluster in priority queue

As two clusters merge, use 1, 2, 3 to quickly merge and create merged indexes

Updates are performed using indexes

Complexity
Blocking

Worst Case: Bootstrapping does not reduce the number of clusters. Therefore, every pair of references within a bucket must be compared.

Suppose *n* references are assigned to *b* buckets, with each reference being assigned to a maximum of *f* buckets.

Comparisons per Bucket: $O((nf/b)^2)$
Total Number of Comparisons: $O(n^2f^2/b)$

Assumptions:

1. Number of buckets is proportional to the number of references (*b* is O(n)).
2. *f* is a small constant independent of *n.*

The total number of computations is around *O(n)*. A bad bucketing algorithm would assign *O(n)* references to each bucket leading to a total of $O(n^2)$ comparisons.

Complexity
Iteration

Time taken for each iteration of the algorithm: Assume that for each bucket that is affected by a merge operation, all $O((nf/b)^2)$ comparisons per buckets have to be redone.

Definition: Two buckets are connected if any hyper-edge connects two references in the two buckets.

If any bucket is connected to *k* other buckets, each merge operation would lead to $O(\mathbf{k}(nf/b)^2)$ update/insert operations. REMINDER: Still reduces to ***O(k)*** because of our previous assumptions about *f* and *b*.

Binary-heap implementation for the priority queue (pq): Assuming q is the number of entries in the pq, the insert/update and extract-max operations take $O(\log q)$ time.

Total cost of each iteration of the algorithm is ***O(k log q)***

Complexity

Total number of Iterations

Worst case: the algorithm may exhaust the pq before the similarity falls below the threshold.

Number of merge operations required to exhaust a pq of size *q*:

1. Best case: Merge tree is perfectly balanced; size of each cluster is doubled by each merge operation. *O(log q)* merges are required.
2. Worse case: merge tree is *q* deep and requires *O(q)* merges.

Therefore, total cost of the iterative process is ***O(qk log q)*.**

Complexity
Priority Queue

Worst case: bootstrapping does not reduce the number of initial clusters.

Bound of $O(n^2f^2/b)$ on the original size q of the PQ, which reduces to O(n).

Therefore, total cost of the algorithm can be bounded by *O(nk log n).*

(Worst case analysis of the total number of iterations accounts for the cost of bootstrapping because bootstrapping can be considered as a sequence of cluster merge operations which do not require updates/inserts to the pq)

Complexity Baseline

Both attribute and naive relational baselines must take a decision for each pair of references in a bucket:

Worst case: *O(n)*

Similarity computation is more expensive for the naive-relational approach because it requires a pair-wise match to be computed between two hyper-edges.

Experimental Evaluation
Real-World: Datasets

Key Idea: Use co-author relationships in papers to help discover the author entities in the domain and map the author references to entities.

CiteSeer: Contains 1,504 machine learning documents with 2,892 author references to 1,165 author entities. The only attribute information available is author name. Author's full last name always provided, while first and middle names are sometimes provided as initials.

arXiv: Contains 29,555 high energy physics publications with 58,515 author references to 9,200 author entities. The only attribute information available is author name. Author's full last name always provided, while first and middle names are sometimes provided as initials.

BioBase: Contains 156,156 biology publications with 831,991 author references. All author first and middle names are abbreviated. Entity labels are available only for top 100 author names with the highest number of references. The BioBase dataset has a variety of other attributes which the authors use for resolution: the evaluate entity resolution performance over 10,595 references that have these 100 names.

Experimental Evaluation
Real-World: Uncertainty

Ambiguity: The disambiguation aspect of resolution. A name (last name and first initial) is ambiguous if multiple entities share that name.

Dispersion: The identification aspect of resolution. The dispersion for an entity is the number of distinct observed names for each entity.

CiteSeer: Only 3 out of 1185 names are ambiguous, and the average number of entities per ambiguous name is 2.33 while the maximum is 3 (**low ambiguity)**. 202 out of 1164 entities have multiple recorded names, the average and maximum dispersion are 1.24 and 8.
arXiv: 374 of 8737 names are ambiguous, and the average number of entities per ambiguous name is 2.41 while the maximum is 11. 3083 out of 8967 entities for arXIV are dispersed over multiple names, and the average dispersion is 1.44 while the maximum is 10 (**high dispersion)**.
BioBase: 84 of 100 names are ambiguous (**high ambiguity**). The number of entities for each name ranges from 1 to 100 with an average of 32. Cannot determine dispersion because we do not have complete ground truth. **Interesting hyper-edge structure** because the number of author references per publication ranges from 1 to 100 with an average of 5.3 (CiteSeer and arXiv range from 1 to 10).

Experimental Evaluation
Real World: Evaluation

Measure performance of algorithms based on the correctness of the pair-wise co-reference decisions over all references using the F1 measure: harmonic mean of precision and recall.

Comparison between:

- Attribute-based entity resolution **(A)**
- Naive resolutional entity resolution **(NR)**
- A* and NR* are variants of the above two algorithms which perform transitive closures over pair-wise decisions.
- Collective Relationational Entity Resolution (CR)

Experimental Evaluation
Real World: Similarity Measures

1. Soft TF-IDF: Augments TF-IDF schema of matching token sets with approximate token matching using a secondary string similarity measure. Performs well for name-based entity resolution.
2. TF-IDF: Utilized for the BioBase dataset because it contains attributes other than names.

Secondary String Similarity Measures:

1. Scaled-Levenstein: Edit-distance family of similarity measures which assigns unit cost to each edit operation and normalizes the result.
2. Jaro: Measures the number and order of common characters between strings.
3. Jaro-Wrinkler: Variant of Jaro that also considers the longest common prefix. Both are well suited for short strings such as personal names.

Experimental Evaluation
Real World: Experimental Details

Blocking is employed in both CiteSeer and arXiv because it is infeasible to consider all pairs as potential matches: retains ~99% of true duplicates for both datasets by allowing at most two character transpositions for last names.

Bootstrapping is employed over all three datasets. The authors implemented bootstrapping with a value of k=1 for low ambiguity domains CiteSeer and arXiv while they utilized a value of k=2 for BioBase.

The authors measured the performance of NR, NR*, and CR at 20 different values of combination weight **α** and report the best performance.

The authors estimated ambiguity of references for AdarName based on last names with first initial as the secondary attribute.

Experimental Evaluation Real World: Results

Table I. Performance of different algorithms on the CiteSeer, arXiv and BioBase datasets. We report the mean and the standard deviations (within parenthesis) of the F1 scores obtained using Scaled Levenstein, Jaro and Jaro-Winkler as secondary similarity measure within Soft TF-IDF.

	CiteSeer	arXiv	BioBase
A	0.980 (0.001)	0.974 (0.002)	0.568 (0)
A*	0.990 (0.001)	0.967 (0.003)	0.559 (0)
NR	0.981 (0.006)	0.975 (0.016)	0.710 (0)
NR*	0.991 (0.002)	0.972 (0.039)	0.753 (0)
Bootstrap H-Amb	0.217 (0)	0.119 (0)	0.452 (0)
Bootstrap L-Amb	0.942 (0)	0.977 (0)	0.317 (0)
CR	0.995 (0)	0.985 (0)	0.819 (0)

Standard deviation measures sensitivity of entity resolution performance in terms of similarity measure used for names.

The results are not very sensitive to secondary string metric choice. For CR and the BioBase dataset, the choice is irrelevant.

Scaled Levenstein measure was most often the best.

Experimental Evaluation Real World: Results

Table I. Performance of different algorithms on the CiteSeer, arXiv and BioBase datasets. We report the mean and the standard deviations (within parenthesis) of the F1 scores obtained using Scaled Levenstein, Jaro and Jaro-Winkler as secondary similarity measure within Soft TF-IDF.

	CiteSeer	arXiv	BioBase
A	0.980 (0.001)	0.974 (0.002)	0.568 (0)
A*	0.990 (0.001)	0.967 (0.003)	0.559 (0)
NR	0.981 (0.006)	0.975 (0.016)	0.710 (0)
NR*	0.991 (0.002)	0.972 (0.039)	0.753 (0)
Bootstrap H-Amb	0.217 (0)	0.119 (0)	0.452 (0)
Bootstrap L-Amb	0.942 (0)	0.977 (0)	0.317 (0)
CR	0.995 (0)	0.985 (0)	0.819 (0)

Attribute-based entity resolution performs well for CiteSeer and arXiv because both datasets have relatively low levels of ambiguity as compared to BioBase. This correlates with a decrease in performance for A in the BioBase dataset.

When data-sets are not ambiguous, all dispersed entities can be successfully identified by raising the discrimination threshold for determining duplicates without generating false positives.

Bootstrap L-Amb performs almost as well as the attribute baseline for CiteSeer and arXIV while it performs many incorrect matches in BioBase because of its higher ambiguity.
Bootstrap H-amb performs poorly for CiteSeer and arXiv due to low recall but improves performance over Boostrap L-Amb for BioBase by increasing precision

Experimental Evaluation Real World: Results

Table I. Performance of different algorithms on the CiteSeer, arXiv and BioBase datasets. We report the mean and the standard deviations (within parenthesis) of the F1 scores obtained using Scaled Levenstein, Jaro and Jaro-Winkler as secondary similarity measure within Soft TF-IDF.

	CiteSeer	arXiv	BioBase
A	0.980 (0.001)	0.974 (0.002)	0.568 (0)
A*	0.990 (0.001)	0.967 (0.003)	0.559 (0)
NR	0.981 (0.006)	0.975 (0.016)	0.710 (0)
NR*	0.991 (0.002)	0.972 (0.039)	0.753 (0)
Bootstrap H-Amb	0.217 (0)	0.119 (0)	0.452 (0)
Bootstrap L-Amb	0.942 (0)	0.977 (0)	0.317 (0)
CR	0.995 (0)	0.985 (0)	0.819 (0)

Attributes of related entries can help in disambiguation in domains with high ambiguity, there may not be much improvement for less ambiguous domains.

- The naive relational entity resolution algorithm only performs marginally better than A for CiteSeer and arXIV, while the improvement is significant for Biobase.

Transitive closures improve performance in data-sets with low ambiguity, but it may result in false identifications in datasets with higher ambiguity:

- Improves performance for both A and NR for CiteSeer and arXiv, but in BioBase it helps NR but not A.

Experimental Evaluation Real World: Results

Table I. Performance of different algorithms on the CiteSeer, arXiv and BioBase datasets. We report the mean and the standard deviations (within parenthesis) of the F1 scores obtained using Scaled Levenstein, Jaro and Jaro-Winkler as secondary similarity measure within Soft TF-IDF.

	CiteSeer	arXiv	BioBase
A	0.980 (0.001)	0.974 (0.002)	0.568 (0)
A*	0.990 (0.001)	0.967 (0.003)	0.559 (0)
NR	0.981 (0.006)	0.975 (0.016)	0.710 (0)
NR*	0.991 (0.002)	0.972 (0.039)	0.753 (0)
Bootstrap H-Amb	0.217 (0)	0.119 (0)	0.452 (0)
Bootstrap L-Amb	0.942 (0)	0.977 (0)	0.317 (0)
CR	0.995 (0)	0.985 (0)	0.819 (0)

The collective relational entity resolution algorithm (CR) performs the best across all three data-sets.

- The performance benefits are modest for low ambiguous domains while they are highlighted in high ambiguous domains.
- Robust because performance is independent of choice of attribute similarity used.
- **How much better??**

The performance improvements of CR over NR is attributed to the consideration of the identities of related references rather than just their attributes.

Experimental Evaluation
Real World: Results

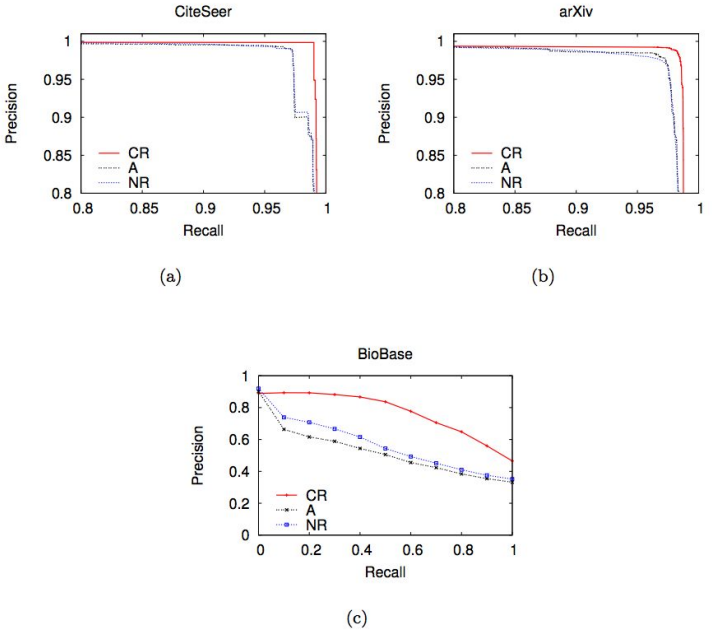


Fig. 6. Precision Vs Recall for (a) CiteSeer, (b) arXiv and (c) BioBase

Precision-Recall curve using Jaro similarity measure for names: **Precision** is the fraction of retrieved instances which are relevant while **recall** is the fraction of relevant instances which are retried.

- **Benefits of CR are significantly larger in domains with high ambiguity such as BioBase (the authors had to zoom into the other two graphs to highlight the differences).**

Experimental Evaluation
Real World: Results

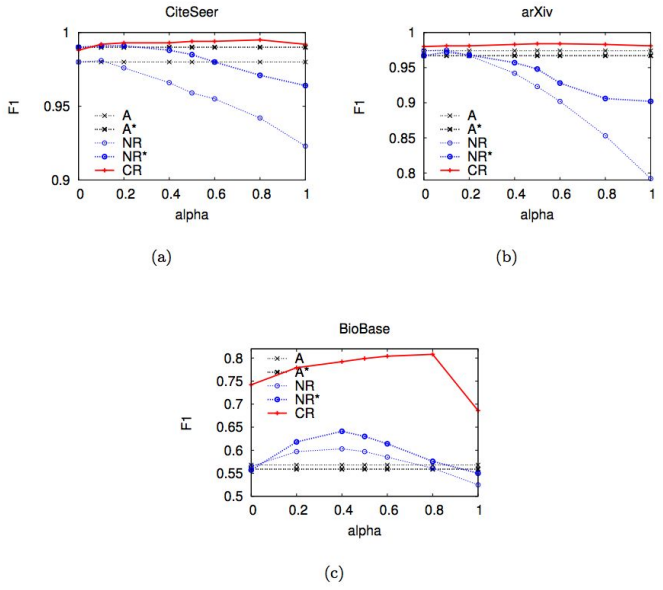


Fig. 7. Entity resolution performance at different values of α for (a) CiteSeer, (b) arXiv and (c) BioBase

CR, NR, NR* require a weighing parameter **α** for combining attribute and relational similarity. A and A* do not depend on **α.**

1. **CR consistently outperforms NR and NR* for all values of α.**
2. **NR outperformance A only marginally for small values of α but performance drops significantly at higher values. Although more stable for BioBase, performance still drops below A for higher values of α.**

Difference between CR and A performance at α=0 comes from the bootstrapping phase.

Experimental Evaluation
Real World: Results

Table II. F1 performance for collective relational entity resolution using different neighborhood similarity measures in the three bibliographic datasets.

	CiteSeer	arXiv	BioBase
Common	0.994	0.984	0.814
Common+Fr	0.994	0.984	0.816
Jaccard	0.994	0.985	0.818
Jaccard+Fr	0.995	0.985	0.818
AdarNbr	0.994	0.984	0.815
AdarNbr+Fr	0.994	0.984	0.816
AdarName	0.994	0.985	0.819
AdarName+Fr	0.995	0.984	0.817
Path3Sim	0.994	0.984	0.812

There is little difference in performance between graph-based similarity measures on the CiteSeer and arXiv datasets while there is more impact on BioBase

- There is not enough evidence to validate the use of frequencies (+Fr)

Experimental Evaluation
Real World: Results

Table II. F1 performance for collective relational entity resolution using different neighborhood similarity measures in the three bibliographic datasets.

	CiteSeer	arXiv	BioBase
Common	0.994	0.984	0.814
Common+Fr	0.994	0.984	0.816
Jaccard	0.994	0.985	0.818
Jaccard+Fr	0.995	0.985	0.818
AdarNbr	0.994	0.984	0.815
AdarNbr+Fr	0.994	0.984	0.816
AdarName	0.994	0.985	0.819
AdarName+Fr	0.995	0.984	0.817
Path3Sim	0.994	0.984	0.812

1. Jaccard similarity improves performance over Common neighbors. This highlights the importance of considering the size of the common neighborhood as a fraction of the entire neighborhood.
2. AdarNbr performs worse than Jaccard. This highlights that the connectedness of a shared neighbor is not a reliable indicator because the graph is consolidated over iterations and new hyper edges are added to each cluster.
3. **AdarName performs the best over all the graph-based similarity measures. It attempts to capture the uniqueness of a cluster of references.**
4. Path3Sim has the lowest performance of all the graph-based similarity measures. This suggests that in dense collaboration graphs with many ambiguous entities, going beyond immediate neighborhood can hurt entity resolution performance.

Experimental Evaluation Real World: Execution Time

Table III. Execution time of different algorithms in CPU seconds

	CiteSeer	arXiv	BioBase
A	0.1	11.3	3.9
NR	0.1	11.5	19.1
CR	2.7	299.0	45.6

The use of collective relational entity resolution is more expensive computationally.

- 9 times as long as the baseline for CiteSeer and 17 times for arXiv.

Differences in degree of relational connectivity explains difference in execution times; the average number of neighbors per entity for CiteSeer is 2.15 and 4.5 for arXiv.

- Difference in execution time between CR and the baselines is much smaller for BioBase possibly because it has many attributes that A must take into account.
- NR is significantly more expensive than A: average number of authors per publication is 5.3 for BioBase as compared to 1.9 for the other two data-sets.

Experimental Evaluation: Scalability

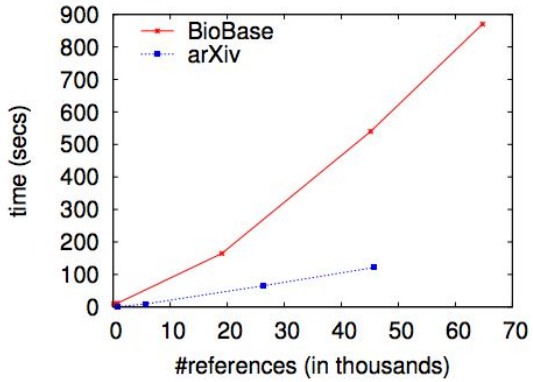


Fig. 9. Execution time of CR with increasing number of references

Complexity of CR is *O(nk log n)*, for *n* input references where *k* represents the degree of connectivity among the references.

Experimental Evaluation: Real World: Comparison

Table IV. Comparison of CR with LDA-ER

	CiteSeer		arXiv	
	F1	secs	F1	secs
CR	0.995	2.7	0.985	299
LDA-ER	0.993	240	0.981	36,000

LDA-ER is a probabilistic generative model for collective entity resolution which uses a non-parametric approach to resolve entities by discovering underlying groups of entities from observed relationships. It runs in O(t(ng+e) time for n references, where t is the number of iterations to converge, g is the number of groups, and e is the number of entities discovered.

- CR is superior in terms of performance and execution time but requires a similarity threshold to be specified
- LDA-ER does not require such a threshold and automatically figures out the most likely number of entities.

Experimental Evaluation: Synthetic Data

- Examine how different structural properties of datasets affect the performance of CR:
  - Fraction of references which are ambiguous
  - Number of neighbors per entity

Experimental Evaluation: Synthetic Data: Generator

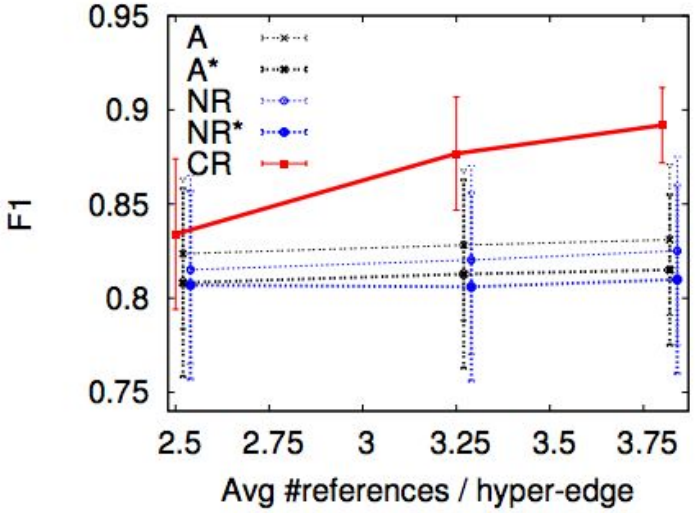
```
    Creation Stage
1.  Repeat N times
2.      Create random attribute x with ambiguity p_a
3.      Create entity e with attribute x
4.  Repeat M times
5.      Choose entities e_i and e_j randomly
6.      Set e_i = Nbr(e_j) and e_j = Nbr(e_i)

    Generation Stage
7.  Repeat R times
8.      Randomly choose entity e
9.      Generate reference r using N(e.x, 1)
10.     Initialize hyper-edge h = ⟨r⟩
11.     Repeat with probability p_c
12.         Randomly choose e_j from Nbr(e) without replacement
13.         Generate reference r_j using N(e_j.x, 1)
14.         Add r_j hyper-edge h
15.     Output hyper-edge h
```

Fig. 10. High-level description of synthetic data generation algorithm

First stage: domain entities and their relationships are created. Create *N* entities and add *M* binary relationships between them.

Second Stage: generation of publications and their co-authors. *R* hyper-edges are generated, each with its own references. Each hyper-reference $\langle r_i, r_{i1}, \ldots, r_{ik} \rangle$ is generated by first sampling an entity *e* and generating a reference r_i from it. Each instance r_{ij} comes from randomly sampling a neighbor *e* without replacement.

Experimental Evaluation:
Synthetic Data: Evaluation



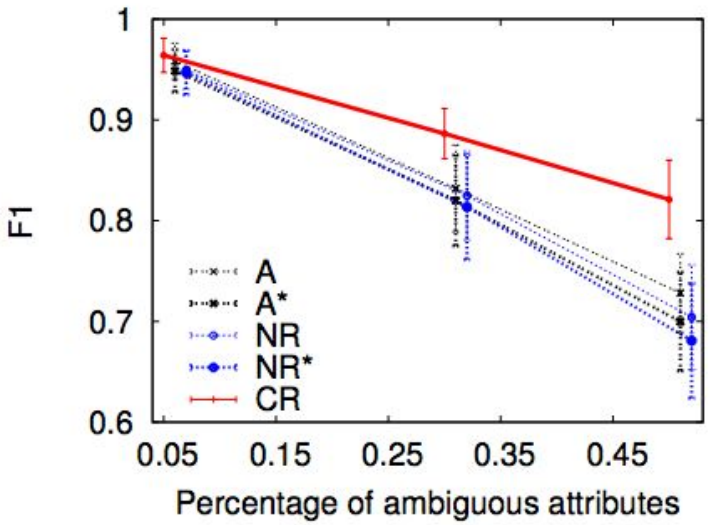
(a)

First experiment: studied the effect of the number of references in each hyper-edge

- CR is expected to benefit from larger hyper-edge sizes.
- Constructed an entity graph by creating 100 entities and 200 binary relationships. Then created different reference datasets, each with 500 hyper-edges.

The performance of CR improves with increasing number of references per hyper-edge in stark contrast with NR's degrade.

Experimental Evaluation:
Synthetic Data: Evaluation



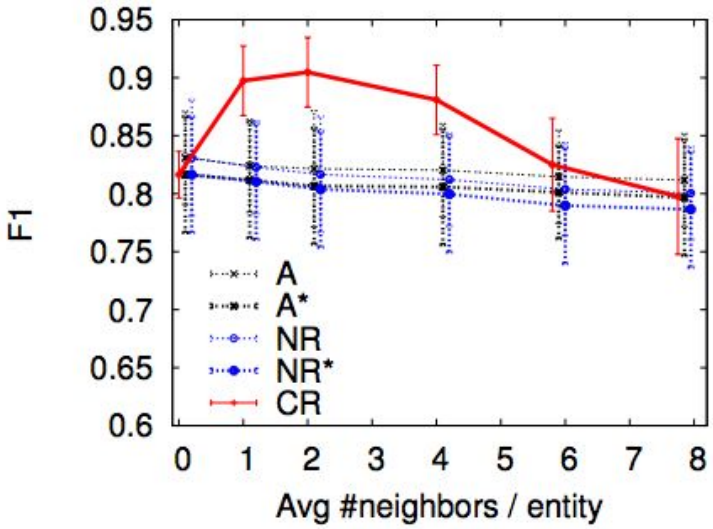
(b)

Second experiment: varied the number of ambiguous references in the data.

- CR is expected to show larger improvements over the baseline for more ambiguous data. **Disambiguation cannot be done using attributes.**
- Created five sets of datasets, each with 100 entities, but with different ambiguous attribute probability. Then, they added 200 binary relations between these entities and generated 500 hyper-edges with an average of 2 references per hyper-edge.

The performance drop for CR is significantly slower than those for the baselines because entity relationships help make the algorithm robust.

Experimental Evaluation:
Synthetic Data: Evaluation



(c)

Third experiment: studied the impact of varying the number of relationships between underlying entities.

- Created a set of 100 entities. Created different entity graph structures by adding different number of relations between entities. Generated 500 hyper-edges with an average of 2 references per hyper-edge from each of the different entity graphs.

A does not change with respect to neighborhood size because performance is independent
NR degrades slightly with higher neighborhood sizes.

Increasing the number of relationships does not always help CR; performance increases initially but peaks when average number of neighbors per entity is around 2.

Experimental Evaluation: Limitations

1. Similarity measures require a weighing parameter **α** for combining attribute and relational similarity. The authors argue that CR shows significant improvements in performance over all values of **α.**
2. Determination of the termination threshold is an issue. This is a problem for any clustering algorithm.

Conclusion

- CR significantly outperformed the baseline algorithm overall three data-sets; the degree of improvement depended on the ambiguity of the dataset.
- Benefits of relational clustering over attribute-based approaches are greatest when there are many references per hyper-edge and when there are ambiguous references in the data.
- CR performance improves initially in relation to the number of relations between underlying entities increase although the later diminish as the patterns become less informative.

Future Work

1. All of their data-sets were bibliographic; it would be interesting to study their algorithms on different types of relational data such as consumer data, social network data, and biological data
2. Integrate the collective resolution process with the extraction process; their work starts from data in which the references have already been extracted
3. They viewed entity resolution as an offline data-cleaning process while they have begun to investigate the notion of query-time entity resolution: only the data relevant to a query is resolved.