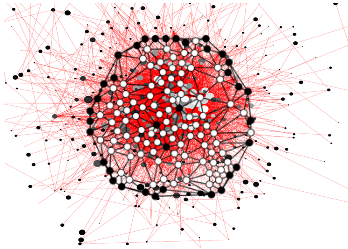
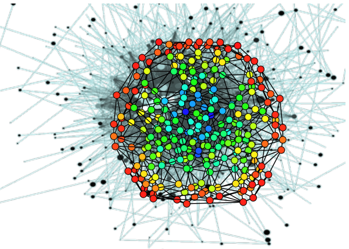


Exploiting Graph Structure for Identity Resolution

Using De-Anonymization at the Link Level



Chaz Lever* and Keith Henderson†

*Wake Forest University
chazlever@gatech.edu

† Lawrence Livermore National Laboratory
keith@llnl.gov

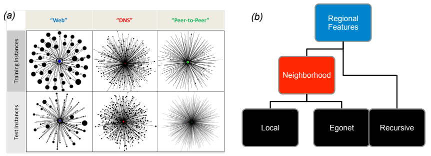
Problem & Motivation

Problem: Can the de-anonymization performance of ReFeX (Recursive Feature eXtraction) [1] be improved by evaluating a relational graph at the link level?

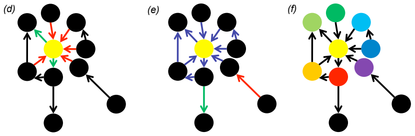
Motivation: "Anonymized" data sets are becoming more common, and often this data contains relational data (i.e., friendships in social network, connections between client/ servers in a network trace, etc.). Work done by *Henderson et al.* [1] has shown that treating relational data as a graph and exploiting its structure is often enough to reduce the uncertainty of an anonymized ID.

ReFeX (Recursive Feature Extraction)

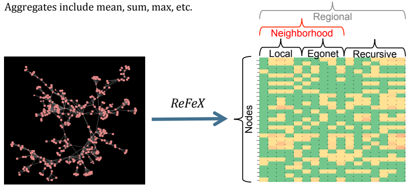
What is it? A novel algorithm that captures "behavioral" information by recursively combining local (node-based) and neighborhood (egonet-based) features.



(a) Demonstrates the intuition behind *ReFeX.* Node and edge size indicate communication volume relative to the central node in each frame. (b) Shows a summary of the different types of structural features. *ReFeX* generates regional features from local and egonet features.

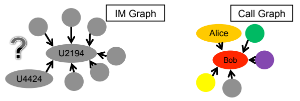


(c) Local Features encode in- and out-degree (d) *Egonet Features* encode within-, incoming-, and outgoing-*egonet* edges (e) R*ecursive Feature* are aggregates of another feature over a node's neighbors. Can aggregate Neighborhood or Recursive features. Aggregates include mean, sum, max, etc.



Methodology

Data Sets

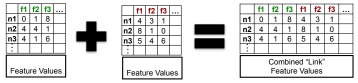


Need two data sets. One to create the *target graph* (contains the set of unlabeled nodes) and another to create the *reference graph* (contains a set of labeled nodes). This work uses pair of Yahoo! IM communication graphs collected on different days.

	yahoo-d00_message.csv	yahoo-d01_message.csv
# of nodes	50576	51937
# of edges	123496	127451

Algorithm

1) Combine node *ReFeX* features to form "link features." Used to calculate Euclidean distance between target and reference edges.



	f1	f2	f3	...
n1	0	1	8	
n2	4	4	1	
n3	4	1	6	
⋮				

Feature Values

	f1	f2	f3	...
n1	4	3	1	
n2	8	1	0	
n3	5	4	6	
⋮				

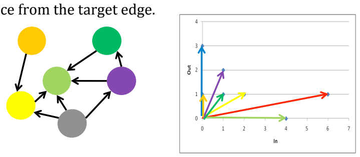
Feature Values

	f1	f2	f3	f1	f2	f3	...
n1	0	1	8	4	3	1	
n2	4	4	1	8	1	0	
n3	4	1	6	5	4	6	
⋮							

Combined "Link" Feature Values

2) For a target node that is also in the reference graph, sample a random subset of edges that a target node participates in.

3) For each randomly sampled target edge, generate a list of reference edges sorted from smallest to largest Euclidean distance from the target edge.



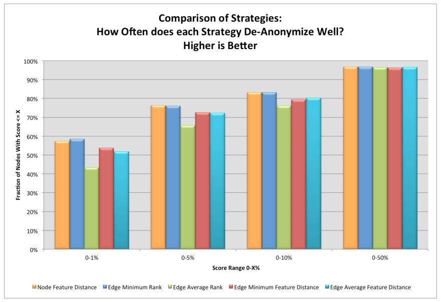
4) Collate list of sorted reference edges into node-based guesses using one of four devised collation strategies.

Collation Strategy	Description
Minimum Distance	Select node guess by minimum edge distance across edge lists.
Minimum Rank	Select node guess by minimum edge ordering across edge lists.
Average Distance	Select node guess by average edge distance across edge lists.
Average Rank	Select node guess by average edge ordering across edge lists.

5) Nodes are scored by their ordering in the final collated list (i.e., a node is scored 3 if it is correctly guessed after 3 tries). The max score is the number of vertices in the reference graph.

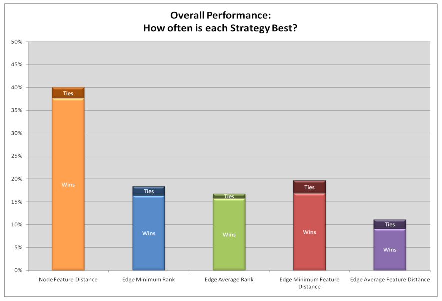
Results & Analysis

How did different collation strategies perform?



Since scores are simply the number of guesses to find a node, one method of comparing performance between different collation strategies is to calculate the fraction of target instances scoring less than a given threshold.

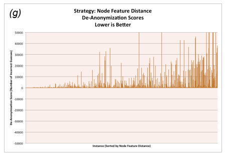
How often did a particular strategy perform better?



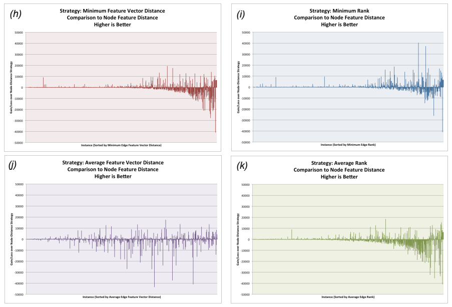
Another interesting comparison is to answer the question, "How often did a particular collation strategy perform better?" The answer to this question is equal to how often a particular strategy yielded the target node with the lowest score. This comparison is shown in the above chart.

Conclusions & Future Work

Does link-level de-anonymization improve performance?



(g) Shows the de-anonymization score for each instance using node feature distance.
(h) Gain/Loss for each node using the minimum feature vector collation strategy.
(i) Gain/Loss for each node using the minimum rank collation strategy.
(j) Gain/Loss for each node using the average feature vector collation strategy.
(k) Gain/Loss for each node using the average rank collation strategy.



As seen in Figure (a), the original node based strategy performs best on nodes with smaller node feature distance and degrades as that value decreases. Figures (h),(i),(k) show that the link-based performance is able to increase a node's score if it's collation strategy score is low. Figure (j) shows that the average edge feature distance provides the least boosting power. The average-based collation strategies appear to be less resistant to to higher scores for a particular node and tend to perform worse than the minimum counterparts.

Future Work

It has been shown that link-level de-anonymization can further reduce uncertainty on some nodes in the graph. It would be useful to devise a "super strategy" that combines node- and link-level de-anonymization strategies. To accomplish this, each strategy could be augmented by a confidence function which indicates how likely it will perform well for a given node.

References

[1] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos. It's who you know: Graph mining using recursive structural features. In *SIGKDD*, 2011.

U.S. DEPARTMENT OF ENERGY

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. DE-AC52-07NA27344.

Lawrence Livermore National Laboratory