LLNL-CONF-479453

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

It's Who You Know: Graph Mining Using Recursive Structural Features

K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, C. Faloutsos

April 7, 2011

ACM SIGKDD Conference on Knowledge Discovery and Data Mining
San Diego, CA, United States
August 21, 2011 through August 24, 2011

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

It's Who You Know: Graph Mining Using Recursive Structural Features

Keith Henderson
Lawrence Livermore Lab
keith@llnl.gov

Brian Gallagher
Lawrence Livermore Lab
bgallagher@llnl.gov

Lei Li & Leman Akoglu
Carnegie Mellon University
{leili,lakoglu}@cs.cmu.edu

Tina Eliassi-Rad
Rutgers University
eliassi@cs.rutgers.edu

Hanghang Tong
IBM Watson
htong@us.ibm.com

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

ABSTRACT

Given a graph, how can we extract good features for the nodes? For example, given two large graphs from the same domain, how can we use information in one to do classification in the other (i.e., perform across-network classification or transfer learning on graphs)? Also, if one of the graphs is anonymized, how can we use information in one to de-anonymize the other? The key step in all such graph mining tasks is to find effective node features. We propose ReFeX (Recursive Feature eXtraction), a novel algorithm, that recursively combines local (node-based) features with neighborhood (egonet-based) features; and outputs regional features – capturing "behavioral" information. We demonstrate how these powerful regional features can be used in within-network and across-network classification and de-anonymization tasks – without relying on homophily, or the availability of class labels. The contributions of our work are as follows: (a) ReFeX is scalable and (b) it is effective, capturing regional ("behavioral") information in large graphs. We report experiments on real graphs from various domains with over 1M edges, where ReFeX outperforms its competitors on typical graph mining tasks like network classification and de-anonymization.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; E.1 [**Data Structures**]: Graphs and networks

General Terms

Algorithms, Design, Performance, Experimentation.

Keywords

Graph mining, feature extraction, network classification, identity resolution

1. INTRODUCTION

Extracting effective features for nodes of a given graph is a key step for many data mining tasks, such as outlier detection (i.e., nodes with strange feature combinations), or mining across different graphs from the same domain. For example, given IP communication graphs from different days or different enterprise networks, can we train a classifier on one day (say Monday) and use the same classifier to accurately predict traffic on another day (say Thursday) without any labels on the latter graph? Here, the key step is to extract effective and transferrable features from each node that would best capture node characteristics, so that we can distinguish and classify the nodes (or edges). Another example is the following: given an anonymized social network (say Twitter's who-follows-whom graph) and a non-anonymized communication graph (say Twitter's who-mentions-whom graph), can we find node features that will help de-anonymize (or re-identify) the people in the social network?

Here, we propose a novel solution, *ReFeX* (Recursive Feature eXtraction) to these graph mining tasks. *ReFeX* recursively combines local (node-based) features, with neighborhood (egonet-based) features, and outputs regional features that capture "behavioral" information. These regional features represent the *kind* of nodes to which a given node is connected (e.g., connected to rich people), as opposed to the *identity* of those nodes (e.g., connected to Bill Gates) – i.e., it is who you know at the type-level that matters in mining across different graphs. In empirical studies, we demonstrate how these powerful regional features can be used in within- and across-network classification and de-anonymization tasks – without relying on homophily, or the availability of class labels.

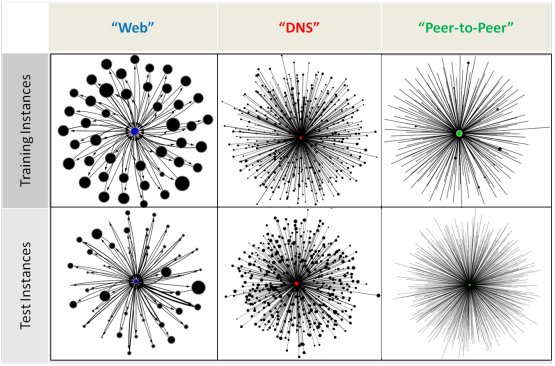
Thus, the problem is defined as follows. Given a graph G, compute a node-feature matrix F with the following properties:

- *Structural*: The construction of F should not require additional attribute information on nodes or links.
- *Effective*: Good node-features should (1) help us predict node attributes, when such attributes are available (as in the case of IP traffic that we discuss next), and (2) be transferable across graphs (e.g., when the graph changes over time).

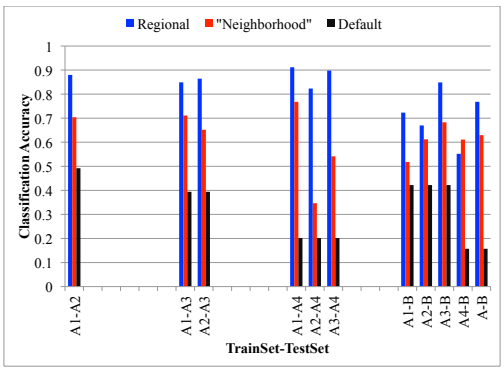
The ideal features should help with data mining tasks. Typical tasks include node classification (after we are given some labels), de-anonymization of the nodes of a graph, and transfer learning.

Figure 1 gives a quick overview of (a) the intuition behind *ReFeX* and (b) a preview of its classification accuracy. In (a), we show egonets of nodes from different days in an IP network, where node size and edge size are proportional to the traffic. The main point is that the characteristics of the neighbors are valuable and help us characterize the center node (manually labeled as 'web', 'dns-server' and 'p2p', respectively for each column). Figure 1(b) shows the classification accuracy of *ReFeX* (in blue bars) vs competitors - higher is better; *ReFeX* wins consistently.

Copyright 2010 Association for Computing Machinery. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the U.S. Government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.
KDD'11, August 21–24, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0813-7/11/08 ...$10.00.



(a) Intuition (neighbors matter)



(b) Performance preview (blue bars: highest - win)

Figure 1: (a) Intuition behind *ReFeX*: Six IP hosts from different days of an enterprise network trace and (manually) labeled by their primary traffic type. Node and edge size indicate communication volume relative to the central node in each frame. Regional structure, i.e. the *types* of neighbors that a given host connects to, are vital. (b) classification accuracy of *ReFeX* with respect to transfer learning, in blue bars (higher is better) - see Figure (4) for more details.

The contributions of our work are as follows:

- *Novel Design*: We propose *ReFeX*, a scalable algorithm that computes regional features capturing "behavioral" information on large graphs.
- *Effectiveness*: *ReFeX*'s regional features perform well for several graph mining tasks, like transfer learning (across-network classification) and node de-anonymization on large graphs.

The rest of the paper is organized as follows: the proposed strategies are presented in Section 2; the experiments are presented in Sections 3 and 4. In Section 5, we review the related work; and we conclude the paper in Section 6.

2. PROPOSED ALGORITHM

Our algorithm *ReFeX* aggregates existing feature values of a node and uses them to generate new *recursive* features. The initial set of features used to seed the recursive feature generation can be structural information from the network or attributes from an external source. Here, we focus on tasks where only structural information is available. We separate structural attributes into three types: *local*, *egonet*, and *recursive* features. Local and egonet features together are called *neighborhood* features, and all three together are called *regional* features.

2.1 Neighborhood Features

The base features that seed the recursive *ReFeX* process are *local* and *egonet* features. These can be computed quickly for a given node. We call the set of local and egonet features together *neighborhood* features.

Local features are all essentially measures of the node degree. If the graph is directed, they include in- and out-degree as well as total degree. For weighted graphs, they contain weighted versions of each local feature.

Egonet features are computed for each node based on the node's ego network (a.k.a. egonet). The egonet includes the node, its neighbors, and any edges in the induced subgraph on these nodes. Egonet features include the number of within-egonet edges, as well as the number of edges entering and leaving the egonet. Strictly speaking, the latter are not in the egonet but they can be counted without looking at non-egonet nodes. As with local features, we compute directed and/or weighted versions of these features if the edges are directed and/or weighted.

2.2 Recursive Features

We broadly define a recursive feature as any aggregate computed over a feature value among a node's neighbors.

2.2.1 Generating Recursive Features

Currently *ReFeX* collects two types of recursive features: means and sums.[1] As a typical example, one recursive feature is defined as the mean value of the feature *unweighted degree* among all neighbors of a node. The features that can be aggregated are not restricted to neighborbood features, or even to structural features. The aggregates can be computed over any real-valued feature (including other recursive features). We compute the means and sums of all feature values. Moreover, when applicable, we compute these for incoming and outgoing edges separately.

2.2.2 Pruning Recursive Features

Clearly, the number of possible recursive features is infinite and grows exponentially with each recursive iteration. To reduce the number of generated features, a variety of pruning techniques can be employed. A simple example is to look for pairs of features that are highly correlated. In this example case, the pruning strategy is to eliminate one of the features whenever two features are correlated above a user-defined threshold.

For computational reasons, *ReFeX* uses a simplified version of this approach. Specifically, feature values are mapped to small integers via *vertical logarithmic binning*, then *ReFeX* looks for pairs of features whose values never disagree by more than a threshold. For details on the threshold, see Section 2.3 below.

First, each feature's values are transformed into *vertical logarithmic bins* of size p (where $0 < p < 1$). The process is as

[1]We selected sum and mean as aggregate functions heuristically. These simple measures capture the dominant trends among a node's neighbors w.r.t. each feature. Other functions, such as maximum, minimum, and variance could easily be added to *ReFeX*. In our experiments, sum and mean were sufficient to provide good empirical performance on data mining tasks with a reasonable runtime.

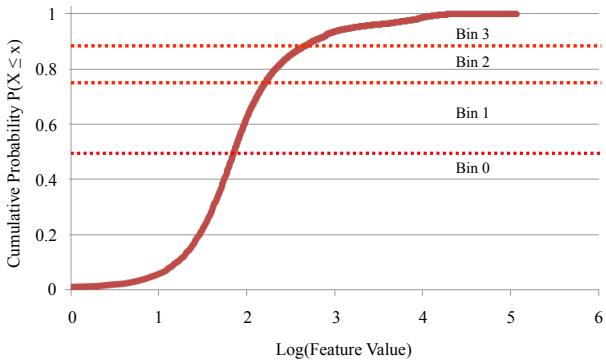


Figure 2: Vertical logarithmic binning of a feature value.

follows. For feature f_i, the $p\,|V|$ nodes with the lowest f_i value are reassigned f_i value 0. If there are ties, it may be necessary to include more than $p\,|V|$ nodes. Next, p fraction of the remaining nodes are assigned f_i value 1, and p of the remaining nodes after this are assigned value 2. This is repeated until all f_i values have been replaced by integer values between 0 and $log_{p^{-1}}(|V|)$ (see Figure 2).

We choose logarithmic binning for all features based on the observation that many graph properties exhibit power law distributions [1]. In particular, logarithmic binning always places most of the discriminatory power among sets of nodes with large feature values. This is reasonable, given that we expect to be able to make better predictions about active nodes for which we have many observations than nodes for which we only have a few.

Once a set of features has been generated and binned, *ReFeX* looks for pairs of features that do *not* disagree at any vertex by more than a threshold s. We call such a pair of features s-friends. To eliminate redundant features, we construct a *feature-graph*, whose nodes are features and whose links are s-friend relations. Each connected component of this graph is replaced by a single feature. When possible, we retain "simpler" features, i.e. features generated using fewer recursive iterations.[2]

If a recursive iteration results in no retained features, *ReFeX* halts and reports the retained feature values from each of the previous iterations. Note that a feature retained in iteration k may not be retained in iteration $k+1$, due to recursive features connecting them in the feature-graph. In this case, we still record and output the feature because it was retained at some iteration.

2.3 Parameters

ReFeX requires two parameters: p, which is the fraction of nodes placed in each logarithmic bin, and s, which is the feature similarity threshold.

The parameter p takes a value between 0 and 1, inclusive. Increasing p too close to 1 reduces the number of bins and increases the effective pruning aggressiveness, which can lead to a loss of discriminatory power. Decreasing p to near 0 can generate many bins and retain many features during pruning, which can increase runtime significantly. In our experiments, we found $p = 0.5$ to be a sensible choice – with each bin containing the bottom half of the remaining nodes. We also found that the results were not sensitive to the value of p as long as its value was not near 0 or 1.

For s, *ReFeX* uses *relaxation* at each iteration. For small graphs ($\leq 100K$ nodes), *ReFeX* uses $s = 0$ for the initial iteration (to generate neighborhood features). This effectively retains any feature that does not totally agree with another feature in logarithmic bin values. For larger graphs ($> 100K$ nodes), the initial value of s may be increased if computational resources are insufficient to generate the full set. On each subsequent iteration, *ReFeX* increased s by 1. This ensures that the process will halt after no more than $log_{p^{-1}}(|V|)$ iterations, since the maximum value of any feature is s at that point.

2.4 Computational Complexity

Let n be the number of nodes, m be number of edges, M = maximum degree, f = number of features, and d_i = degree of node i. Computational complexity of *ReFeX* can be divided into two steps: (1) computation of neighborhood features and (2) computation at each subsequent iteration. Computation of neighborhood features is expected to take $O(n)$ for real-world graphs. See Lemma 1 for details. At each subsequent iteration, *ReFeX* takes $O(f(m+nf))$ time where $f \ll n$. The space requirement is $O(m+nf)$.

Lemma 1. The computation of neighborhood features takes $O(nM^{\epsilon})$.

Proof. For brevity, we only give a sketch of the proof: $\sum_{(u\to v)\in E} degree(u) \approx \int_1^M n * d_i^{(\epsilon-1)} \partial d_i \approx nM^{\epsilon}$. $\epsilon = 3-\alpha$ for real-world graph with power-law degree distributions with exponent α. However, since α is typically in the range $2 < \alpha < 3$ for real-world graphs [3], $0 < \epsilon < 1$. ■

3. FEATURE EFFECTIVENESS ON NETWORK CLASSIFICATION

We describe experiments on within- and across-network classification using features from *ReFeX*.

3.1 Data

IP-A and IP-B are real network-trace data sets collected roughly one year apart on separate enterprise networks. The nodes are IP addresses and the links are communications between the IPs. The IP-A trace begins at midnight on day 1 and continues up to 12pm on day 5. The IP-B trace begins at midnight on day 1 and continues up to ≈5pm on day 6.

For days 1-4 of the IP-A dataset (IP-A1 to IP-A4), we extract flows in the period from 12pm-1pm. We exclude day 5 because the trace ended at 12pm. For IP-B, we extract flows from 12pm-1pm for day 3 only. We then label all flows using a payload signature-based classification tool. Once network flows are labeled, we transfer labels to hosts by selecting the most frequent class labels from among the host's flows. The payload classifier can distinguish between over 15 classes of traffic (e.g., Web, DNS, SMTP, P2P). However, since we found that 3 classes (namely, Web, DNS, and P2P) made up the dominant traffic type for over 90% of the labeled hosts, we remove all other labels and focus on the 3-class classification problem. Table 1 summarizes the data that we extracted.

3.2 Classifiers

To test the predictive ability of *ReFeX*'s features, we use the logForest model described by Gallagher et al. [11]. The logForest is a bagged model, composed of a set of logistic regression (LR) clas-

[2]It may be the case that two features are not similar enough to be joined in the feature graph, but reside in the same component because of a chain of similar features between them. We examined feature graphs for a number of data sets and found that this does occur. However, any other choice of criterion for joining features (cliques, community discovery algorithms, etc.) would be similarly heuristic and probably include outside cases that are unsatisfactory. Any of these criteria could be used in place of the connected-components criterion, however; we use connectedness for its simplicity and ease of computation.

	IP-A1	IP-A2	IP-A3	IP-A4	IP-B
Nodes	81450	57415	154103	206704	181267
(labeled)	29868	16112	30955	67944	27649
Links	968138	432797	1266341	1756082	1945215
(unique)	206112	137822	358851	465869	397925
Web	32%	38%	38%	18%	42%
DNS	36%	49%	39%	20%	42%
P2P	32%	12%	23%	62%	16%

Table 1: Extracted data

sifiers, where each is given a subset of $\log(f) + 1$ of the f total features. For our experiments, we use a logForest of 500 LR classifiers. Like Gallagher et al. [11], we found that the overall performance of logForest was superior to standard logistic regression. In addition, as a baseline, we include a standard homophily-based relational neighbor classifier: wvRN (short for weighted-vote Relational Neighbor) [23]. The wvRN classifier has been shown to have excellent performance across a range of tasks. The classifiers we compare are:

- *wnRN+RL* - a relational neighbor model, which uses wvRN and relaxation labeling for collective classification
- *Neighborhood* - a logForest model, which uses the neighborhood features only
- *Regional* - a logForest model, which uses the regional features (i.e., neighborhood + recursive)

3.3 Feature Effectiveness on Within-Network Classification

3.3.1 Methodology

Each data set contains a set of core nodes for which we have ground-truth (i.e., we know the true class labels). In all cases, classifiers have access to the entire data graph during both training and testing. However, not all of the core nodes are labeled. We vary the proportion of labeled core nodes from 10% to 90%. Classifiers are trained on all labeled core nodes and evaluated on all unlabeled core nodes.

Our methodology is as follows. For each proportion of core nodes labeled, we run 10 trials and report the average performance. For each trial and proportion labeled, we choose a class-stratified random-sample containing (1.0 - proportion labeled)% of the core instances as the test set and the remaining core instances become the training set. Note that proportion labeled less than 0.9 (or greater than 10 trials) means that a single instance will necessarily appear in multiple test sets. The test sets cannot be made to be independent because of this overlap. However, we carefully choose the test sets to ensure that each instance in our data set occurs in the same number of test sets over the course of our experiments. This ensures that each instance carries the same weight in the overall evaluation regardless of the proportion labeled. Labels are kept on the training instances and removed from the test instances. We use identical train/test splits for each classifier. Our experimental framework sits on top of the open source Weka system [28]. We implement our own network data representation and experimental code, which handles tasks such as splitting the data into training and test sets, labeling and unlabeling of data, and converting network fragments into a Weka-compatible form. We rely on Weka for the implementation of logistic regression and for measuring classifier performance on individual training/test trials.

3.3.2 Results

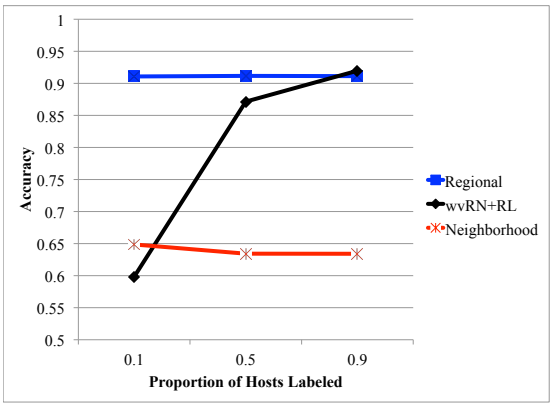


Figure 3: Within-network classification with wvRN+RL, neighborhood, and regional classifiers. The regional classifier dominates when labels are sparse.

		Test Graph			
		IP-A2	IP-A3	IP-A4	IP-B
Train Graph	IP-A1	✓	✓	✓	✓
	IP-A2		✓	✓	✓
	IP-A3			✓	✓
	IP-A4				✓
	IP-A1 to IP-A4				✓

Table 2: Across-network experiments performed

Figure 3 demonstrates the performance of the *wnRN+RL*, *Neighborhood* and *Regional* classifiers on a within-network classification task on the IP-A3 data set. We repeated this task on each of the IP-A data sets with essentially equivalent results. The within-network classification setting is the sweet spot for homophily-based models like wvRN. So, it is no surprise that wvRN+RL performs well on this task with 90% of nodes in the network labeled. However, the performance of wvRN+RL degrades quickly as labels become more sparse. The Regional and Neighborhood classifiers are less sensitive to the availability of labeled data since they do not rely on labeled neighbors to make accurate classifications. As a result, the Regional classifier outperforms wvRN+RL when labeled data are sparse. The dramatic difference in performance between Neighborhood and Regional demonstrates that the recursive feature generation process leads to more expressive features that are able to represent important concepts that cannot be captured by the neighborhood features alone.

3.4 Feature Effectiveness on Across-Network Transfer Learning

3.4.1 Methodology

For each experiment, the training graph has all known labels available. The test graph is *completely* unlabeled. Each classifier is evaluated on all known ground truth labels in the test graph. We use an identical set of features for all data sets. This set of features comes from running *ReFeX* on the IP-A1 data set. Table 2 summarizes the across-network experiments:

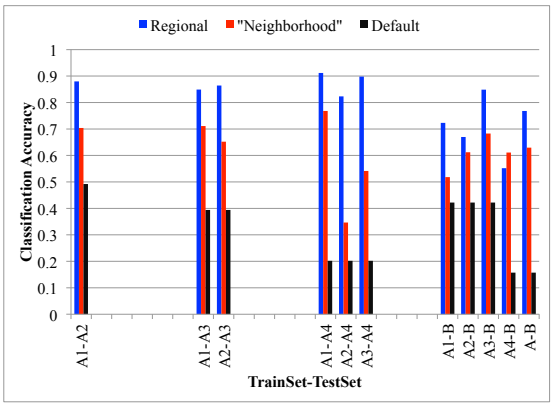


Figure 4: Across-network transfer learning with neighborhood and regional features. Regional demonstrates consistently high accuracy on difficult transfer learning tasks.

3.4.2 Results

Figure 4 demonstrates the performance of *Neighborhood* and *Regional* on a series of across-network transfer learning tasks. Here, we train on one network, where all known labels are available, and test on a separate network that is completely unlabeled. We emphasize the difficulty of these tasks given the (sometime extreme) differences in class distributions between data sets (see Table 1). The performance of the *Default* classifier is a good indicator of the difficulty of each task since this model makes predictions based solely on the most frequent class from the training set. We also note that *wnRN+RL* is not applicable for these tasks since it relies on the availability of some known class labels to seed the inference process.

As in the within-network setting, the Regional classifier is the best overall performer on the across-network tasks, achieving 82% - 91% accuracy training and testing on separate days of IP-A and 77% accuracy training on all days of IP-A and testing on IP-B. The performance of Regional applied to IP-A4 is particularly impressive, given the extreme differences in class distribution between IP-A4 and the other data sets (see Table 1). We note that Regional is somewhat less successful training on IP-A4. In fact, training on IP-A4 and testing on IP-B is the one case where Regional underperforms Neighborhood. However, the difference in performance is small (¡5%). Finally, and not surprisingly, we see a benefit to training on a number of diverse data sets instead of a single data set. Specifically, we achieve 77% training on all of the IP-A data sets and testing on IP-B, whereas we see a lot of variation (55% - 85%) training on individual days from IP-A.

4. FEATURE EFFECTIVENESS ON IDENTITY RESOLUTION

To demonstrate that regional features capture meaningful and informative behaviors of nodes, we present a collection of identity resolution tasks. In each task, we compute a set of regional features on a pair of networks whose node-sets overlap. Our hypothesis is that a node's feature values will be similar across graphs. We present an experimental framework that allows us to test this empirically. In our experiments, we will demonstrate how this method can be used to perform "de-anonymization"on social network datasets when external non-anonymized data is available.

4.1 Problem Statement

For a pair of graphs whose node-sets overlap, but whose edge-sets can be distinct (or even represent a totally different type of observation), can we use network structure alone to map nodes in one network to nodes in the other? More realistically, can we reduce the entropy associated with each node in one graph, with respect to its possible identity among nodes in the second graph? For a given method, we will measure success at this task by counting how many "incorrect" nodes the method guesses before it finds the correct node in the second graph.

4.2 Methodology

We are given two graphs, G_{target} and $G_{reference}$, and a vertex v_{test} which exists in both graphs. To test a given identity resolution strategy, we allow the strategy to guess reference vertices $\langle v_1^{guess}, v_2^{guess}, \ldots, v_k^{guess} \rangle$ until it correctly guesses v_{test}. The score associated with this strategy is k, the number of guesses required to find the node. The baseline method is to guess at random; for this strategy we assume the expected score $|V_{reference}|/2$.

The strategies we test experimentally use structural features to compute guesses. We present results for (1) Local features only, (2) Neighborhood features only, and (3) Regional features. The features are computed using *ReFeX* on G_{target}. The same features are then computed on $G_{reference}$. For a given strategy, the guesses are generated in order of increasing Euclidian distance from V_{target} in feature space. Our hypothesis is that Regional will score lower (i.e. better) than Local or Neighborhood.

To compare the overall performances of strategies, we compute scores across a set $S_{overlap}$ of all vertices that exist in both graphs. When it is not computationally feasible to analyze every node in $S_{overlap}$, we select a set of vertices $S_{test} \subset S_{overlap}$ and report all scores for nodes in S_{test}. In these experiments, we select S_{test} by taking the 1000 vertices in V_{target} with the highest degree, and keeping only those vertices that are also in $S_{overlap}$.

There are a number of ways to compare performance on a given test set. For example, the mean score across all target instances is a measure of success, with lower mean scores indicating better performance. We can also compute the fraction of target instances that score less than a given threshold; here a larger fraction is better. For example, we can report the fraction of target vertices whose score is less than 1% of $|V_{reference}|$.[3]

4.3 Data

Table 3 outlines the data sets used in this set of experiments. The first is a pair of Twitter networks from 2008, including a who-follows-whom network and a who-mentions-whom network. The second is an SMS communication network. The third is a collection of 28 days of Yahoo IM events. The fourth is IP traffic on two separate enterprise networks, observed at several different times. We described the IP network in details in Section 3.1.

Yahoo! IM Networks. Each graph here is a collection of IM events taken from one of 28 days of observation.[4] Each node is an IM user, and each link is a communication event on a given

[3]If we combine the latter across all thresholds, we can treat it as the cumulative distribution function of $p(score(x))$, the empirical probability distribution over scores for this strategy. This presentation is harder to interpret, however, since one strategy may perform better in some regions and worse in others, and we are generally concerned with the lowest possible scores.

[4]http://sandbox.yahoo.com/

Data Graph	Source	# Nodes	# Edges	Weighted?	Directed?	# Local Features	# Nbrhood Features	# Regional Features	# Recursive Iterations
Twitter (T)	Who-follows-whom	465K	845K	Yes	No	3	8	45	6
Twitter (R)	Who-mentions-whom-1st	840K	1.4M	Yes	No	3	8	45	–
SMS (T)	04/01/2008	144K	580K	Yes	Yes	7	21	53	3
SMS (R)	04/02/2008	128K	505K	Yes	Yes	7	21	53	–
Yahoo! IM (T)	Day 1	50K	123K	Yes	Yes	3	12	79	5
Yahoo! IM (R)	Day 2-28	29K-100K	80K-280K	Yes	Yes	3	12	79	–
IP Comm. (T)	IP-A1	81K	206K	Yes	Yes	7	22	373	4
IP Comm. (R)	IP-A2, -A3, -A4, IP-B	57K-206K	137K-466K	Yes	Yes	7	22	373	–

Table 3: Data sets used in the identity resolution task. (R) refers to the Reference Graph. (T) refers to the Target Graph. The features are generated on the Target Graph are used to compute feature values in the Reference Graph.

day. In some cases, multiple events are reported for a given pair of users on a given day; when this happens the links are given weights according to how many events are reported.

We use the IMs from day 1 as G_{target}, and compute scores for the other days (2 - 28) as $G_{reference}$. The task here is to take an IM user from day 1 and locate him in later days. Figure 5 shows that the similarity in node-sets and edge-sets with G_{target} is low for all $G_{reference}$ graphs. However, in most cases there is a significant overlap with the 1000 nodes in the test set S_{test}.

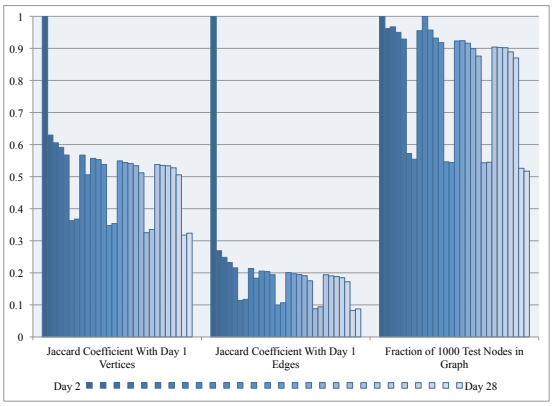


Figure 5: Yahoo! IM graphs across 28 days. The similarity between G_{target} on Day 1 and $G_{reference}$ on Days 2 to 28 is low but there is significant overlap in most cases with the test nodes.

Enterprise Network Traces. We describe these IP networks in detail in Section 3.1. The target graph comes from IP-A1. There are four reference graphs total: IP-A2, IP-A3, IP-A4, and IP-B. The task is to identify an external IP address from one day to the next or one IP network to the next. A potential application of this task is to de-anonymize a network trace where the IP addresses are hidden; one could observe a non-anonymized enterprise trace and use structural information to guess the identities of the anonymized IPs.

Twitter Relationships. This dataset consists of two graphs taken from Twitter in 2008.[5] One is a social "who-follows-whom" network that was generated by starting with a few seed users (all famous people with verified accounts) and crawling "follows" links for a few iterations. The other is taken from actual tweets observed during a year. The users whose tweets are observed are the same as the users in the social network.

To construct a graph from tweets, we generate "mentions-first" edges. That is, if a tweet by $user1$ contains a username (e.g. $@user2$), we add a link between $user1$ and $user2$ in the network, but only if $user2$ is the first user mentioned in the tweet. We do not include self-mentions.

For this experiment, we use the social network as G_{target} and the mentioning network as $G_{reference}$. Success at this task indicates that one could de-anonymize a social network by using publicly available text data, so long as usernames can be parsed from the text. This has important implications with respect to privacy of users in published "anonymized" social networks.

SMS Messages. SMS-Message dataset is constructed from short messages in a mobile phone operator at Asia. Each node corresponds to a mobile phone client. The reference and the target graphs encode the number of short messages among those nodes, extracted at two different days. We also eliminate those pairs of less active users with only one message exchange between them.

4.4 Feature Effectiveness Results

We present results for each of the four datasets. The first three demonstrate the effectiveness of regional features for identity resolution, and the fourth serves as a practitioner's note regarding test-set selection.

Yahoo! IM Networks. Figure 6 shows the average score for each of the 27 reference graphs, as a percentage of the expected score in the baseline strategy (recall that the baseline strategy scores $\frac{|V_{reference}|}{2}$ on average). All feature-based strategies outperform the baseline strategy, but performance of all strategies suffers on weekends when the graphs are significantly different. In general, expected scores are much better (lower) for Neighborhood than Local, and somewhat better for Regional than Neighborhood.

Figure 7 provides another view of the performance. Each data point here shows the fraction of test nodes that scored better than 1% of the maximum possible score $|V_{reference}|$. A higher fraction here indicates better performance, as the distribution of scores is higher near the minimum score 1. Again, Regional outperforms the other strategies in most instances. This improvement decreases on the weekends, and in one case (Day 27) Regional does no better than Local.

Poor performance on the weekends is not unexpected, and strengthens our belief that structural features are capturing behavior. Recall from Figure 5 that the set of active users is significantly different on the weekends, as is the set of observed communications. Intuitively, it is an obvious fact that many users behave differently both in general and specifically with respect to IM communication on

[5]http://www.public.asu.edu/ mdechoud/datasets.html

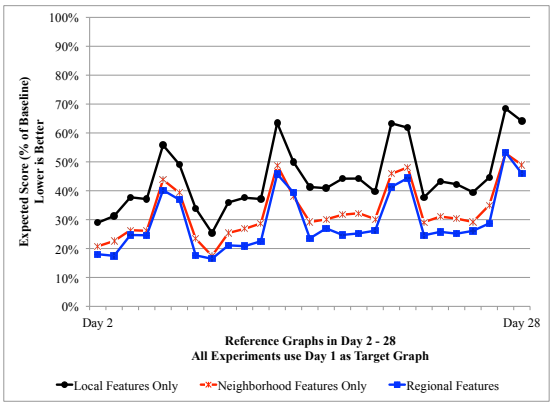


Figure 6: Yahoo! IM: Regional is consistently better (lower average score) at tracking test nodes across several weeks of IM traffic.

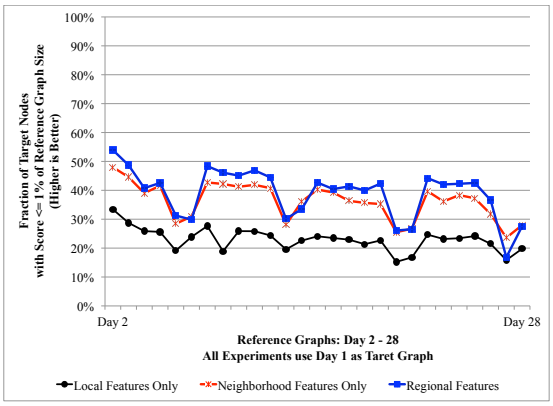


Figure 7: Yahoo! IM: Regional is consistently able to identify more test nodes within the first 1% of guesses (higher is better). It dips in performance corresponding to weekends, when the graph structure is very different.

the weekend than during the weekdays. Regional is especially susceptible to errors in this case, because these changes in behavior are amplified during recursive feature generation.

Enterprise Network Traces. Regional is very effective at tracking external IP addresses over time, as seen Figure 8. It dominates the performance of Local and Neighborhood in all tests, with over 45% of S_{test} scoring in the top 1% of possible scores even a year after the communications in G_{target} are observed. Note that S_{test} is significantly smaller after a year due to decreased overlap; the reported results are fractions of $|S_{test}|$. Even though the traces were collected from separate networks, common external nodes (e.g., google.com) will be included in both.

In this task, average scores were inconsistent, with Local outperforming Regional in some instances. Analysis of the distributions

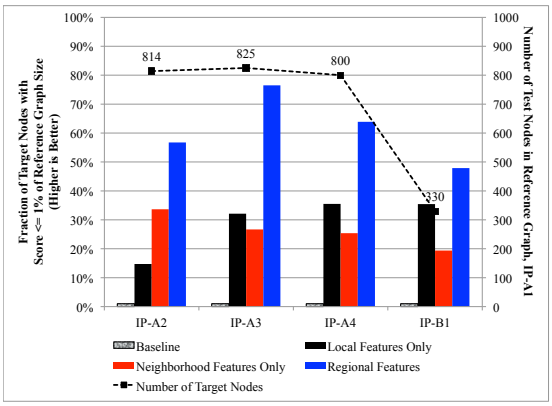


Figure 8: IP Communication: Regional is the best strategy for tracking IP addresses (higher is better). Even in the IP-B1 graph, which was observed on a different enterprise network, Regional locates nearly 50% of test nodes in the first 1% of its guesses.

of scores here shows that Regional outperforms Local for a majority of the test instances, but there is a small subset of test nodes for which Regional performs extremely poorly (near baseline).

Twitter Relationships. This experiment is the "hardest", in that G_{target} and $G_{reference}$ are generated by different processes altogether. G_{target} is a social network of users following other users, while $G_{reference}$ is generated by users mentioning each other in tweets. Surprisingly, our method is still able to do well at resolving some users here.

For this experiment, we applied each strategy to several different test sets, rather than just taking the top 1000 nodes by degree. In particular, we tested separately on each vertical logarithmic bin (see Section 2.2.2). For bins with over 1000 users, we sampled 1000 users uniformly at random.

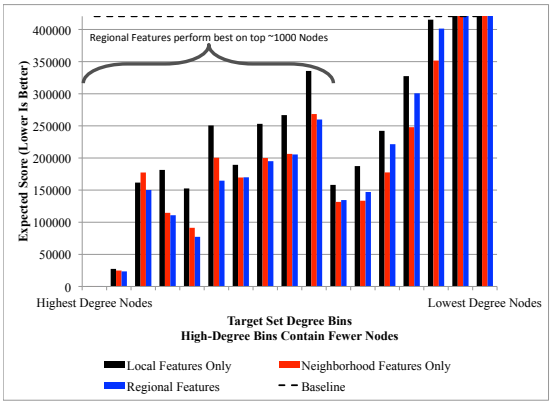


Figure 9: Twitter: Regional features are able to de-anonymize more active nodes (lower is better).

Figure 9 shows the expected scores of each strategy. The ten highest-degree bins, taken together, correspond roughly to the top 1000 nodes by degree. Regional features outperform other strategies in this range, although their performance drops off in the lower degree bins.

All three strategies perform less well on these graphs than on the other datasets, but they are still able to reduce the entropy significantly for the most active users. For example, the highest degree user is BarackObama, and Regional achieved a score of 5 on that instance (the four wrong guesses were AddToAny, MrTweet, tferriss, and THE_REAL_SHAQ). Local and Neighborhood each scored 24 for BarackObama.

For the lowest-degree nodes, all methods perform worse than baseline. This drop-off is expected; less active nodes are harder to identify because (1) they have fewer observed behaviors to leverage and (2) there are more nodes that are very similar to them.

SMS Messages. This dataset serves to point out a case in which Regional does not perform best, but also a practitioner's note that often allows for improved results when performing identity resolution outside of an experimental setting.

Figure 10 shows that the performance with respect to nodes scoring better than 1% of $|V_{reference}|$ can depend on the elements of S_{test}. When S_{test} consists of the top 1000 nodes by degree, each method scores better than 1% on $60 - 65\%$ of the test nodes, and Regional performs worst. However, if each strategy is allowed to "hand-pick" a test set by taking the top 1000 nodes by feature-vector magnitude (in that strategy's feature space), performance improves across the board. Regional improves the most, and is the top performer in this case.

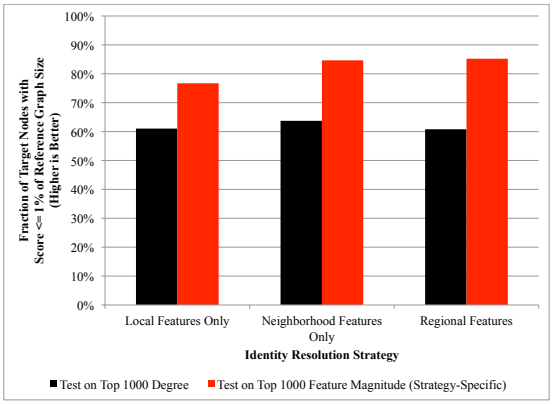


Figure 10: SMS: When allowed to suggest test nodes, Neighborhood and Regional locate near 85% of test nodes in the top 1% of guesses. Without test set selection, all strategies perform equally well.

By allowing each strategy to select its test set, we leverage the fact that a high-degree node may be less distinctive than a low-degree node that has other large feature values. However, as an experiment to compare strategies this approach is not "fair" because the set of test instances is different for each strategy.

We tested the other datasets to determine whether feature magnitude is always better than degree for selecting test nodes. While this is true in general, there are cases in which performance degrades when using feature magnitudes. This is often due to different sizes in the overlap between the 1000 proposed test nodes and $V_{reference}$.

With respect to runtime, our largest graph (Twitter Mentions graph with 840K nodes and 1.4M edges) ran in about 5 hours on a commodity processor. Graphs with fewer than 100K nodes run in an hour or less.

5. RELATED WORK

Feature Engineering in Graph data. There has been comprehensive work studying both global and local structural features of real graphs. For example, at global scale, graph diameter has been observed to be small [2] and to shrink over time [20]. It has also been reported that the number of edges in a time-varying graph grows super-linearly with the number of nodes, following a power-law relation [20]. Moreover, the principal (largest) eigenvalue of the graph is shown to be informative as a vulnerability measure of the graph [27]. Other local and community-level observations show that degree distributions in many graphs follow power-laws [5, 7, 17, 24] and that graphs exhibit modular structure, with nodes forming groups, and groups within groups [9, 14]. This body of work has informed our selection of a set of neighborhood features, upon which we then build recursive features.

Feature Extraction for Data Mining. There also exists related work which exploits feature extraction from graphs for several data mining tasks. One study approaches link prediction as a supervised learning problem [21]. They extract topological features for node pairs and show that the features boost prediction performance and that the supervised approach outperforms unsupervised methods. Another recent study [16] develops a multi-level framework to detect anomalies in time-varying graphs based on graph, subgraph, and node level features. The algorithms rely on extracting graph-level (global) features and tracking these metrics over time. The time points identified as interesting or suspicious are passed to finer levels where more complex tools inspect node and community features to spot regions of interests and flag abnormal nodes. Another study extracts egonet features and patterns in order to detect anomalous nodes in weighted graphs[1]. There has also been work on using local and global structural features to improve the performance of network classifiers[10]. In our work, we introduce methods for *recursive feature extraction*. Our results show that recursive features yield better performance in several data mining tasks, such as transfer learning and identity resolution, compared to their non-recursive counterparts.

Another body of related work uses frequent sub-graphs as structural features for classification of graphs [18, 8] and anomaly detection [26, 22]. These methods, however, assume that the nodes in given graphs have labels. On the other hand, our feature extraction procedures exploit the structure of the graphs, without any assumption on the availability of labels.

Transfer Learning. Transfer learning (i.e., domain adaption) has been a very active research area in recent years. Representative work includes multi-label text classification [30, 13], cross-domain sentiment prediction [4, 6, 15], intrusion detection [12], verb argument classification [19], cross-lingual classification [25], and cross-domain relation extraction [29]. In all of these scenarios, the features are *given* as the input of their algorithms (e.g. the word frequency of the documents) and the goal is to leverage the given features to boost performance in the target domain. In this paper, we aim to answer an orthogonal question: *what kind of features are effective in transferring the knowledge from the source domain to the target domain?* Our case studies show that the proposed recursive features are indeed effective for across-network classification and identity resolution, especially when there are few or no

labels in the target domain, or homophily among class labels does not hold. We expect the proposed recursive features to have wide applicability for transfer learning tasks.

6. CONCLUSIONS

We described a novel algorithm *ReFeX*, which extracts regional features from nodes based on their neighborhood connectivity. These regional features capture behavior in terms of the the kind of nodes to which a given node is connected as opposed to the identity of those nodes. We showed that *ReFeX* was scalable and effective in various graph mining tasks including within- and across-network classification and identity resolution tasks.

7. ACKNOWLEDGEMENTS

We thank Danai Koutra for all her help on this paper. 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 and partially supported by Army Research Laboratory (ARL) under Cooperative Agreement No. W911NF-09-2-0053. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ARL or the U.S. Government.

8. REFERENCES

[1] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *PAKDD*, pages 410–421, 2010.

[2] R. Albert, H. Jeong, and A.-L. Barabasi. Diameter of the world wide web. *Nature*, (401):130–131, 1999.

[3] A.-L. Barabasi, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(1-4):69 – 77, 2000.

[4] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, 2007.

[5] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. *SIAM Int. Conf. on Data Mining*, Apr. 2004.

[6] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *ICML*, pages 193–200, 2007.

[7] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM*, pages 251–262, Aug-Sept. 1999.

[8] H. Fei and J. Huan. Structure feature selection for graph classification. In *CIKM*, pages 991–1000, 2008.

[9] G. Flake, S. Lawrence, C. L. Giles, and F. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35(3), Mar. 2002.

[10] B. Gallagher and T. Eliassi-Rad. Leveraging label-independent features for classification in sparsely labeled networks: An empirical study. *Lecture Notes in Computer Science*, 5498:1–19, 2010.

[11] B. Gallagher, H. Tong, T. Eliassi-Rad, and C. Faloutsos. Using ghost edges for classification in sparsely labeled networks. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 256–264, Las Vegas, Nevada, USA, 2008. ACM. http://doi.acm.org/10.1145/1401890.1401925.

[12] J. Gao, W. Fan, J. Jiang, and J. Han. Knowledge transfer via multiple model local structure mapping. In *KDD*, pages 283–291, 2008.

[13] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *CIKM*, pages 195–200, 2005.

[14] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99:7821, 2002.

[15] J. He, Y. Liu, and R. D. Lawrence. Graph-based transfer learning. In *CIKM*, pages 937–946, 2009.

[16] K. Henderson, T. Eliassi-Rad, C. Faloutsos, L. Akoglu, L. Li, K. Maruhashi, B. A. Prakash, and H. Tong. Metric forensics: a multi-level approach for mining volatile graphs. In *KDD*, pages 163–172, 2010.

[17] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The Web as a graph: Measurements, models and methods. *Lecture Notes in Computer Science*, 1627:1–17, 1999.

[18] X. Kong and P. S. Yu. Multi-label feature selection for graph classification. *Data Mining, IEEE International Conference on*, 0:274–283, 2010.

[19] S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *ICML*, pages 489–496, 2007.

[20] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proc. of ACM SIGKDD*, pages 177–187, Chicago, Illinois, USA, 2005. ACM Press.

[21] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *KDD*, pages 243–252, 2010.

[22] C. Liu, X. Yan, H. Yu, J. Han, and P. S. Yu. Mining behavior graphs for backtrace of noncrashing bugs. In *SDM*, 2005.

[23] S. A. Macskassy and F. Provost. A simple relational classifier. In *Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003*, pages 64–76, 2003.

[24] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46:323–351, 2005.

[25] X. Ni, J.-T. Sun, J. Hu, and Z. Chen. Cross lingual text classification by mining multilingual topics from wikipedia. In *WSDM*, pages 375–384, 2011.

[26] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *KDD*, pages 631–636, 2003.

[27] H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. On the vulnerability of large graphs. In *ICDM*, pages 1091–1096, 2010.

[28] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[29] L. Yao, S. Riedel, and A. McCallum. Collective cross-document relation extraction without labelled data. In *EMNLP*, pages 1013–1023, 2010.

[30] J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *NIPS*, 2005.