Future link regression using supervised learning on graph topology

Adam Goldberg Jonathan Hung Mark Ulrich

I. Introduction and Problem Statement

Link prediction provides useful information for a variety of graph models, including communication, biochemical, and social networks. The goal of link prediction is usually to predict novel interactions (modeled as links/edges) between previously unconnected nodes in a graph. Link prediction is used on social networks to suggest future friends and in protein networks to suggest possible undiscovered pairwise interactions. Link prediction does not model repeat interactions or make any predictions about the number of interactions. To do this we need to predict the weight of links. We call this this problem link regression.

Link regression is useful in graphs where each edge has a weight. Based on a snapshot of the graph at a given time, we predict how the weight of all possible links between observed edges will change by some future time. We use supervised regression techniques using graph topology features such as shortest path and random walks. We also explore Natural Language Processing inspired features, such as sentiment and email categorization.

For experimental evaluation, we use the Enron email dataset because it is the largest publicly available email dataset and a common method for link prediction. A model based on additional graph topology features gets an average R^2 of 0.459, a model based on NLP features gets an R^2 of 0.454, and a model based on all the best features gets an average R^2 of 0.486.

II. Relevant Literature

Many researchers have explored the topic of link prediction. Although new interactions between previously unconnected nodes may occur for different reasons than interactions between connected nodes, our hypothesis was that the similar features used to predict missing links could also be used in link regression. The two primary types of features useful in missing link prediction are graph topology features (node degree, distance between two nodes) and domain-specific attributes of nodes (such as age, gender, or salary), and links (for example, time of day, sentiment, or length).

Some early work in missing link prediction was directly inspired by the small-world model of Watts and Strogatz. Goldberg and Roth [13] look at predictions on a graph modeled by small world based upon edge data, as well as goodness-of-fit to the distribution to consider potential new edges, validating our prediction problem in a general sense. Furthermore, Liben-Nowell and Kleinberg [14] use graph topology alone to make new link predictions on social networks.

Xu and Chen's research on [12] link prediction was concerned with modeling terrorist networks that had very few observed links. They theorized that unobserved links were more likely to exist between nodes that had many common neighbors (i.e. mutual friends) and between nodes with high degrees. Adding these links maintained important properties common to many real world social networks, including small-world (short average path length between two randomly chosen nodes) and scale-free (network degree distribution follows the power-law, presumably because of the preferential attachment where high degree nodes are more likely to form new links). While the

exact methodology is not aligned with ours, they validate common neighbors and similar features as strong candidates.

Kim and Leskovec [8] use node attributes to predict the probability that two nodes will form a link. In particular, they use a "Multiplicative Attribute Graph" model that contains information regarding each node's attributes and the tendency for attributes to cause links to form. Gong [4] tackles a similar problem, using a model called the "social-attribute network", in which a standard social network is augmented with attribute nodes. These attribute nodes form edges with the social nodes that are related to the attribute, and the authors show that running standard algorithms on this augmented network leads to better link prediction results.

Lictenwalter [3] takes a different approach to link prediction. The authors used supervised learning on general node features, such as (in- and out-) degree, as well as features between two nodes such as max flow, shortest path, or PropFlow. The LPmade [3] library was very useful for calculating many of our graph features. We will utilize a combination of features used by the above researchers and [6], combining their node-based features and pair-based features, for a more holistic approach, similarly to what is seen in [9].

All of these papers corroborate our choice of using graph topology, node attributes, and parametric supervised learning to perform link regression and also serve as inspiration for our feature engineering.

III. Data

I. Collection and Preprocessing

We explored our problem via the Enron Email Dataset, which is a conventional email dataset [7]. It was collected as a subpoena of the computers of 150 Enron employees, and therefore is a collection of emails sent to and from those people during this time period. Key fields, such as the sender, recipient, and date of each message were parsed out using regular expressions and then converted into a directed network, with people represented as nodes, and communications between people treated as edges. As in [2], we weight our edges by the number of messages sent in that direction, and annotate the edges with the dates of the contacts.

After inspecting our dataset, we eliminated incomplete and unparsable data. In the end, our dataset has 7475 nodes and 50098 edges. It appears our graph has a preferential attachment with $x_{\min} = 7$ and $\alpha = 2.1006$. Huang [2] mentions that we should not eliminate statistical outliers so as to not bias the data, so we also chose to not.

We also computed the distribution of emails over time. This is a reference to the windows of time that will be most accurate in predicting number of emails sent, depending on the density or sparsity of the data in that timeframe.

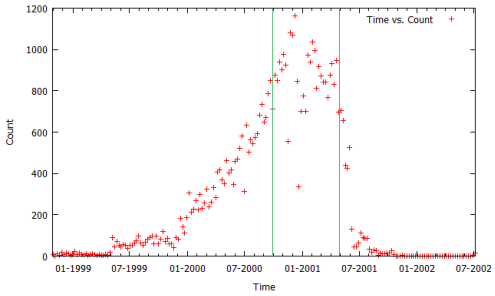

Figure 1: *E-mails sent over time*

Figure 1 shows where the density of e-mails is highest. With a fixed time window, we want to choose dates where the density is highest. From the figure, we investigate e-mails between October 2000 and May 2001.

II. Summary Network Statistics

The assortativity coefficient is a measure of the likelihood that two nodes have an edge based on their node degrees. A high assortativity coefficient (close to 1) indicates that nodes with similar degree are likely to have a link. The assortativity coefficient for this network is 0.0195, indicating that nodes having similar degree has little effect on whether or not they have a link. In this network, that means if two people send a similar number of emails, this reflects little

on the chance that they send emails to each other.

The average clustering coefficient is measured to indicate how likely it is for an email address's recipients to email each other. In this graph the average clustering coefficient is 0.111164.

Mean degree gives a rough estimate of how many emails each person sent/received. In this network the mean in/out degree is 6.702970.

The in-degree distribution is not included because the data does not represent the full set of emails sent between all employees. From the way the data was collected, the recipient of each email is 1 of 150 people, meaning the number of people with in-degree greater than 0 is 150. With this sample size, it is difficult to deduce any pattern in the data.

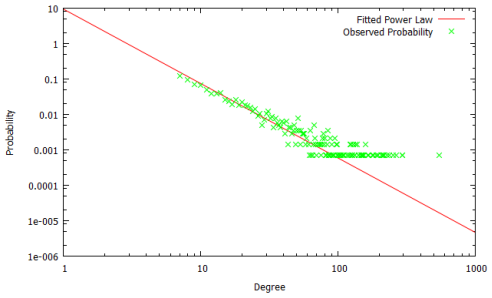

Figure 2: *Out-Degree Distribution*

The out-degree distribution in Figure 2 gives us an idea of how many people send how many emails. It fits a power law with $\alpha = 2.100595$ and $x_{\min} = 7$.

1. Median in-degree: 0
2. Median out-degree: 2
3. Number of SCCs: 7394
4. Largest SCC: 80 nodes
5. Largest SCC diameter: 7

In particular, as a sanity check, we note that the maximal strongly connected component size is 80, which is reasonably close to the 150 email addresses subpoenaed.

To begin our analysis of how links are formed, we look at patterns that link attachments follow. We use the formula from [10],

$$p_e(d) = \frac{\sum_t [e_t = (u, v) \wedge d_{t-1}(v) = d]}{\sum_t |\{u : d_{t-1}(u) = d\}|},$$

to generate Figure 3. In this equation, t is each time step that a new edge is added, and the numerator counts once for each edge that attaches to a node of degree d. This is then normalized by the number of nodes of degree d at each time step so that $p_e(d)$ becomes the probability a destination node of degree d was chosen over all the possible nodes.

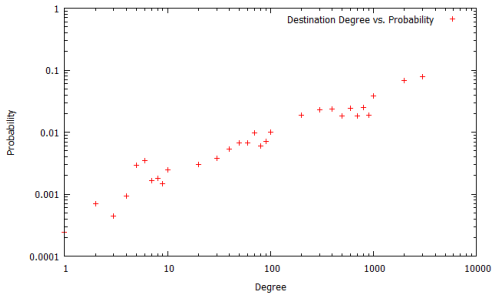

Figure 3: *Edge Attachment*

In Figure 3, statistics for each degree are aggregated depending on the degree size. For example, degrees 1000-2000 are aggregated into one point. This is done to clean the graph but preserve the trend of degree vs. probability.

The probability an edge forms to a node of degree d is roughly increasing with d. This indicates that e-mails tend to be sent to those who receive lots of e-mails. In this context, it makes sense, since there may be latent factors such as an employee's role that would cause them to receive many e-mails both in the past and the future.

We also looked at the tendency for edges to form locally. For each pair of nodes that form an edge, we look at the path length d between the two nodes before the edge is formed. We then aggregate statistics based on the number of edges formed between nodes with distance d.

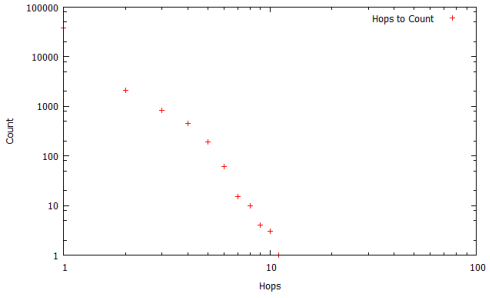

Figure 4: *Edge Locality*

From Figure 4, we see that edges tend to form between nodes that are already close to each other. The e-mails are sent mostly between those that are already close in the network, which may indicate closeness in the organization itself.

IV. Problem Statement

Define two time intervals Δ_x and Δ_y and a discrete time t. We view our problem as creating a regression function $f(X)$ where X is a pair of nodes n_1, n_2, and features f_1 on n_1, f_2 on n_2 and f_3 on n_1 and n_2, all of which are over windows of size Δ_x, starting at t. The target of $f(X)$ is the number of expected emails, within Δ_y timesteps later than $t + \Delta_x$. For concreteness, we consider Δ_y to be one month. Later we use our feature distributions to determine what the minimal reasonable window of Δ_x is that gives us a good chance of predicting $f(X)$.

V. Features

We wanted to select a robust feature set so we could perform ablations and iteratively remove feature to massage out our stronger signals.

I. Single Node Attributes

I.1 Page Rank

Page Rank of each node is the standard formulation for page rank with random teleportation. It is computed by initializing each node's score to be $1/N$, where N is the number of nodes. Then each node's score is updated to be

$$s_i = \sum_{i \to j} d \frac{s_j}{|N(s_j)|} + (1 - d)\frac{1}{N},$$

where s_i is the score of node i, $N(s_i)$ is the set of nodes that i points to, and $1 - d$ is the probability of teleporting to a random node.

I.2 Rooted Page Rank

In addition to page rank, we compute rooted page rank for each node. Rooted page rank is the same as page rank with random restarts. In other words, we do a random walk starting at the source node. However, instead of teleporting to a random node with probability $1 - d$, we teleport to the original node. Otherwise we follow an outgoing edge uniformly at random.

I.3 Node Degree

For each node, we compute its undirected degree as a feature.

I.4 Volume

Volume of n is defined to be the sum of weights of the nodes that n points to.

II. Node Pair Attributes

II.1 Prop Flow

Prop flow is a metric introduced in [3]. It is similar to page rank with a few exceptions:

- The random walk ends when reaching the target node
- The random walk ends when revisiting any node
- Paths are restricted to a certain length

The pseudocode from [3] is included in Algorithm 1.

II.2 Adamic/Adar

Adamic/Adar is a measure of how important similar neighbors are in predicting a link between two nodes. From [11], the similarity is computed as

$$sim(A, B) = \sum_{n \in N(A) \cap N(B)} \frac{1}{\log(inDeg(n))}$$

where $N(A)$ is the set of out-neighbors of node A. Intuitively, this weights nodes that are unique to both A and B higher than nodes that many nodes point to.

Data: network $G = (V, E)$, node v_s, max length l
Result: Score S_{sd} for all $n \leq l$-degree neighbors v_d of v_s
insert v_s into *Found*;
push v_s onto *NewSearch*;
insert $(v_s, 1)$ into S;
for *CurrentDegree* $\leftarrow 0$ *to* l **do**
  OldSearch $\leftarrow$ *NewSearch*;
  empty *NewSearch*;
  while *OldSearch is not empty* **do**
    pop v_i from *OldSearch*;
    find *NodeInput* using v_i in S;
    SumOutput $\leftarrow 0$;
    for *each* v_j *in neighbors of* v_i **do**
      add weight of e_{ij} to *SumOutput*;
    end
    Flow $\leftarrow 0$;
    for *each* v_j *in neighbors of* v_i **do**
      $w_{ij} \leftarrow$ weight of $e_i j$;
      Flow $\leftarrow$ *NodeInput* $\times \frac{w_{ij}}{SumOutput}$;
      insert or sum $(v_i, Flow)$ into S;
    end
    if v_i *is not in Found* **then**
      insert v_i into *Found*;
      push v_i onto *NewSearch*;
    end
  end
end

Algorithm 1: PropFlow Predictor

II.3 Common Neighbors

Common neighbors is simply a count of the number of nodes pointed to by both n_1 and n_2.

II.4 Jaccard Coefficient

Jaccard Coefficient (number of common neighbors divided by total neighbors) is a measure of similarity between two nodes. It is defined as $\frac{N(n_1) \cap N(n_2)}{N(n_1) \cup N(n_2)}$. This yields a number between 0 and 1, being closer to 1 if many of n_1 and n_2's neighbors are shared between them.

II.5 Target Value

Target value is the number of emails sent between two nodes during the training period.

We tried the features as predictors in isolation and tested their effects using univariate linear regression tests. Our exact methodology was to take a single linear regressor formed from a single variable, and orthogonalize it and the data with respect to constant regressors. Then we compute the cross-correlation between each such regressor and the data. This yields our F-scores, below.

III. NLP Features

In order to prevent signal loss, we supplemented the feature set with additional features on each email edge. We also will attempt to use email metadata, for example the length of the message, as well as the sentiment of its contents. Carvalho [1] had success predicting leadership roles from emails by classifying emails into five categories: requests, deliveries, proposals, commitments, and meetings, Including Carvalho's work label classifications from [1] will give us a better glimpse into the meaning of each interaction, which is often indicative of future email patterns. Requests correspond to email inquiries, and commit refers to commitments. Deliver refers to people delivering opinions, while DeliverData refers to people delivering powerpoints or other attachments. Finally, Propose indicates a proposal was made, and Meet is applied to emails relating to a meeting (e.g. saying you will be late or the meeting is in Gates 300).

1. Sentiment Score of Email Body between $[-1, 1]$. Positive values are positive messages, while negative values indicate negative messages.
2. Email Length measured in number of lines
3. Indicator on Request in $\{0, 1\}$.
4. Indicator on Commit in $\{0, 1\}$.
5. Indicator on Deliver in $\{0, 1\}$.
6. Indicator on DeliverData in $\{0, 1\}$.
7. Indicator on Propose in $\{0, 1\}$.

8. Indicator on Meet in $\{0, 1\}$.

We found 22.5% of all emails were Requests, 74.7% were Deliver, 1.1% were Commits, 0.2% were Proposals, 0.7% were Meet, and 31.2% were DeliverData.

Now we have to figure out how to combine these email-based features to fit our problem statement. We add the propose the following features to supplement our original feature vectors for our regression problem. When are trying to predict emails sent from person i to person j from time t to $t + \Delta_y$, these refer to two features each, one for emails sent from i to j, and one for those sent from j to i.

1. Average email length
2. Average email sentiment
3. Sum email length
4. Sum email sentiment
5. Percent of emails that were Request
6. Percent of emails that were Commit
7. Percent of emails that were Deliver
8. Percent of emails that were DeliverData
9. Percent of emails that were Propose
10. Percent of emails that were Meet

It is difficult to discern how to combine these email-level features to pairs of nodes over a time period. In particular, sums may be highly correlated to number of emails, amongst other features, and averages may fail to account for magnitude. Hence we evaluate our candidate features using the F-score, below. We utilize these scores when computing our best feature combinations.

F-score is relative to the set of features it is run on. As you can see from both F-score tables, EdgeWeight has a very high F-score which makes sense because people tend to follow similar email patterns over time. All of the NLP sum features are highly correlated with the EdgeWeight which gives them a very high F-score as well. We can see that the most important graph topology feature is PropFlow followed by RootedPageRank.

Feature	F-Score (average)
EdgeWeight	2.936E+06
DeliverSum	2.825E+06
DeliverDataSum	1.839E+06
LinesSum	1.758E+06
RequestSum	1.246E+06
SentSum	8.330E+05
MeetSum	2.504E+05
PropFlow	9.644E+04
RootedPageRank	6.592E+04
DeliverMean	3.946E+04
LinesMean	3.455E+04
SentMean	3.451E+04
CommitSum	1.487E+04
JPageRank	6.858E+03
ProposeSum	4.490E+03
IVolume	7.980E+02
CommonNeighbor	7.764E+02
JDegree	5.595E+02
JaccardCoefficient	2.818E+02
IDegree	1.032E+02
JVolume	7.603E+01
ProposeMean	0.000E-05
DeliverDataMean	9.915E+03
RequestMean	3.335E+03
MeetMean	6.198E+02
CommitMean	1.959E+02
IPageRank	1.520E+01

Huang [2] mentioned that predictors on link regression problems that work on one dataset do not necessarily generalize well to another dataset. We can also see this behavior here, noting that only 11 of our 26 features received F-scores above 1e+04. To observe how the r^2 value changes as we add more features in order of increasing F-score in our forward search.

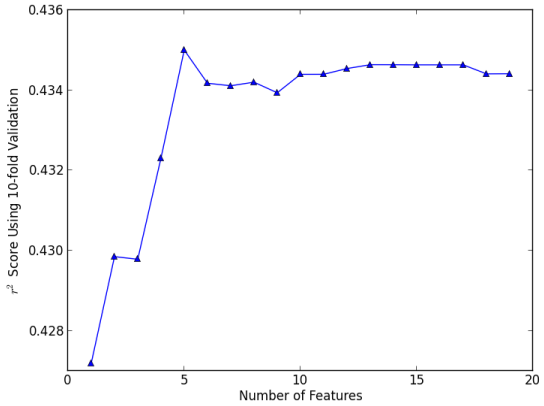

Figure 5: *r^2 for a 30-day window starting on 2000-10-01 depending on top-N features used.*

IV. Machine learning and tuning

Using the above f-scores, we decided to proceed without features with an f-score below 10000. This left us with 11 features. We define our regression over examples for a given window and their target values and then run k-fold cross-validation on individual windows, with $k = 10$, to try to avoid overfitting. We chose $k = 10$ by convention and for speed. Because our dataset is so large, We would like to be window-agnostic and compare arbitrary examples across windows, but currently we cannot process the millions of examples (ranging between 1000000 and 5000000) that exist across all of our windows. After doing preprocessing on our data and recentering it around 0, we tried a variety of regression algorithms, but found linear ridge regression to have the best base performance before tuning and hence used it for our feature engineering.

VI. Results

First we consider two baseline systems, one which always guesses 0 emails for any pair of people over any time, and one which guesses the mean number of emails per person over the training window. The zero baseline received an average R^2 of -0.00019, and the mean baseline scored an average R^2 of -0.00018. This makes sense according to the definition of R^2 being a benchmark of prediction against the mean of the data and that the mean is likely very close to zero as less than 1 out of 100 pairs has a nonzero target value.

As our problem is a regression, we use a traditional R^2 coefficient of determination to measure our accuracy. In particular, our definition allows for negative values, so if our regression fit is worse than the mean of the data, the coefficient will be negative.

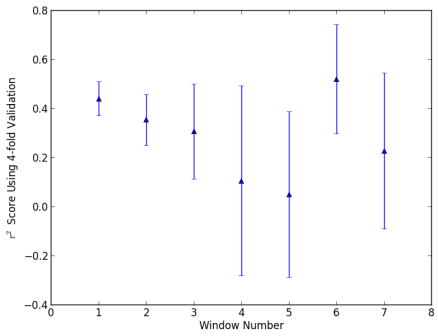

Figure 6: *Predictive Power Varies Accross Time*

As we can see in the above figure, there are several instances of negative R^2 coefficients, which indicates the function we are trying to learn is possibly nonlinear.

In summary, we created several baseline systems, non-parameteric, and parametric. We extracted graph topology and NLP features, and computed the errors for each of those feature sets along with past email edge weights. Finally, we made an optimal selection of features and tuning parameters to give our final result. Via the aforementioned F-score methodology, we found our best subset of features to be the top 6 features by F-score.

Method	R^2 (avg.)	R^2 (σ^2)
MeanBaseline	-0.0002	0.0002
ZeroBaseline	-0.0002	0.0001
RidgeBaseline	0.4163	0.1439
RidgeWithOnlyNLP	0.4538	0.0681
RidgeWithOnlyGraph	0.4596	0.0982
RidgeWithBest	0.4856	0.04175

Figure 6 was calculated using feature and label windows of 100 days starting on 05/18/1999 using simple linear regression and

all base features achieved a mean R^2 of 0.287. The accuracy is highly dependant on the time period, possibly because changes in company structure could render certain patterns obsolete accross large time windows. This corroborates our earlier insight that we could consider examples independently of their windows to create a more general model that is not specific to a certain time period. On the other hand, as we see in Figure 7, using feature and label windows of about 30 days works best. If you predict more than 100 days into the future things become more difficult, while when using less than 10 days the data is too noisy. Using an extra large feature window seems to yield particularly poor results if you are not predicting far into the future.

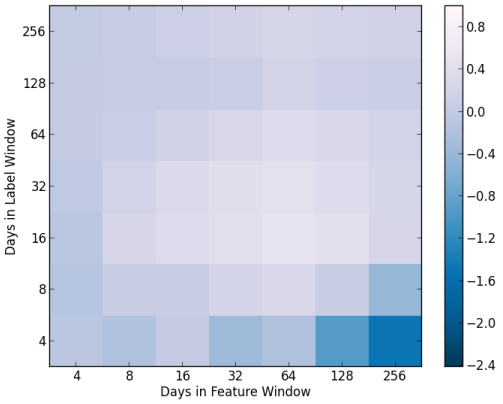

Figure 7: *R^2 Results Predicting Emails Sent in the First Y Days of Year 2000 Using Data From the Last X Days in the End of 1999*

It is clear we have problems with high error variance and that our feature set is likely suboptimal. Hence we investigate features motivated by Natural Language Processing.

VII. Conclusions and Future Work

We see that our approach is reasonable and that both NLP and graph topology features contribute to the overall result, and work best in unison. However, there are many additional considerations for further improvement. Other potential additional features include measures of burstiness (i.e. the groupings) of messages sent between nodes, which we might model using fractal dimension, or even various Fourier transform coefficients of the emails between people when viewed as a time-series. Additionally, more could be done to tune our models, or we could explore deep learning models or ensemble learning. Furthermore, a variety of dimensionality reduction techniques such as principal component analysis might be in handling our high dimensional feature space.

VIII. Contributions

1. Adam: Data extraction, parsing of emails, creation of network edges and nodes, annotation of nodes with metadata, feature selection, statistics on feature sets, regression algorithm framework (training and testing and metrics), regressions, data preprocessing, some graph summary stats, problem statement, worked on windowing with Mark, worked with Jonathan on identifying degree distribution and parameter fitting, NLP features and their feature engineering, email categorization and classification into types, baseline systems.
2. Mark: Created data pipeline, created windowing logic to take appropriate snapshots of the network, researched link prediction features and chose link regression problem, extracted graph topology features. Helped Jonathan with creating network snapshots, baseline systems. Helped Adam with regression algorithm framework, feature selection, and baseline statistics.
3. Jonathan: Converted between internal labels used for feature library to external labels (network snapshots), graph analysis (degree distribution, e-mail distribution over time), degree, clustering coefficient, edge attachment, edge locality.

References

[1] *Discovering Leadership Roles in Email Workgroups*. Vitor R. Carvalho, Wen Wu and William W. Cohen, 2007.

[2] *Link prediction based on graph topology: The predictive value of the generalized clustering coefficient*. Huang, Z. In Workshop on Link Analysis (KDD). August 2006.

[3] *New perspectives and methods in link prediction*. Lichtenwalter, R. N., Lussier, J. T., and Chawla, N. V. (2010, July). In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 243-252). ACM.

[4] *Joint Link Prediction and Attribute Inference Using a Social-Attribute Network*. Gong, N. R., Talwalkar, A., Mackey, L., Huang, L., Shin, E. C. R., Stefanov, E., et. al, ACM Transactions on Intelligent Systems and Technology (TIST), v.5 n.2, p.1-20, April 2014

[5] *Predicting the growth of new links by new preferential attachment similarity indicies* Hu, K., Xiang, J., Xu X.K., Li, H.J., Pramana Journal of Physics, v.82 n.3, March 2014

[6] *Modeling Networks with Auxiliary Information* Kim, M., Dissertation submitted to the Department of Electrical Engineering at Stanford University, August 2014

[7] *Exploration of Communication Networks from the Enron Email Corpus* Diesner, J., Proceedings of Workshop on Link Analysis, Counterterrorism and Security SIAM International Conference on Data Mining 2005 (2005), pp. 3-14

[8] *Modeling Social Networks with Node Attributes using the Multiplicative Attribute Graph Model* Kim, M., Leskovec, J., Internet Mathematics 2011

[9] *Multiplicative Attribute Graph Model of Real-World Networks* Kim, M., Leskovec, J., CoRR 2011

[10] *Dynamics of Large Networks* Leskovec, J., Dissertation submitted to Carnegie Mellon University, 2008

[11] *Friends and Neighbors on the Web* Adamic, L., Adar, E., Social Networks 2003 pp. 211-230

[12] *The Topology of Dark Networks* Xu, J., Chen, H., ACM 2008

[13] *Assessing experimentally derived interactions in a small world* Goldberg, D., Roth, F., PNAS 2003

[14] *The Link-Prediction Problem for Social Networks* Liben-Nowell, D., Kleinberg, J., Journal of American Society for Information Science and Technology. 1019-1031. 2007.