Graph-based Features for Supervised Link Prediction

William Cukierski, Benjamin Hamner, Bo Yang

***Abstract*— The growing ubiquity of social networks has spurred research in link prediction, which aims to predict new connections based on existing ones in the network. The 2011 IJCNN Social Network challenge asked participants to separate real edges from fake in a set of 8960 edges sampled from an anonymized, directed graph depicting a subset of relationships on Flickr. Our method incorporates 94 distinct graph features, used as input for classification with Random Forests. We present a three-pronged approach to the link prediction task, along with several novel variations on established similarity metrics. We discuss the challenges of processing a graph with more than a million nodes. We found that the best classification results were achieved through the combination of a large number of features that model different aspects of the graph structure. Our method achieved an area under the receiver-operator characteristic (ROC) curve of 0.9695, the 2nd best overall score in the competition and the best score which did not de-anonymize the dataset.**

I. Introduction

Directed graphs encapsulate relationships in social networks, with nodes representing members of the network and edges signifying the relations between them. Link prediction, the task of forecasting new connections based on existing ones, is a topic of growing importance as digital networks grow in size and ubiquity [1], [2], [3], [4]. The study of network dynamics has numerous applications. Marketers would like to recommend products or services based on existing preferences or contacts. Social networking websites would like to customize suggestions for new friends and groups. Financial corporations would like to monitor transaction networks for fraudulent activity. In recognition of the growing demand for accurate link prediction, the 2011 IJCNN Social Network challenge asked participants to separate real edges from fake edges in a set of 8960 edges sampled from an anonymized, directed graph depicting a subset of relationships on Flickr.

The graph representation of social networks allows existing and powerful methods of graph theory to be applied to the task of link prediction. We refer the interested reader to Liben-Nowell and Kleinberg's paper [2] for an excellent review of this topic. Though social networks share common representations as a graph, edges can have very different meanings, both within the same network and across different networks. What does an edge on the Flickr graph represent? At face value, each node comprises a user and each edge indicates a friendship, which need not be mutual. However, there are numerous reasons for friendship on a photo sharing site. It may be that two users are friends in real life, or they may share interest in a common subject matter, or they may share interest in a common style of photography. Recognizing the disparate meanings of graph edges leads to new interpretations of traditional link prediction methods.

Our approach to the IJCNN Social Network Challenge follows a classical paradigm in supervised learning, starting with feature extraction, then preprocessing, and lastly repeated classification using the posterior probabilities from Random Forests [5]. Instead of presenting a single novel methodology for link prediction, the foremost contribution of this paper is in the breadth and variety of techniques incorporated into the feature extraction step. Sec. II describes this process in detail, starting with subgraph extraction, descriptions of the features, and finally, meta approaches to make valuable, new predictors from these features. Aspects of the work which are novel, such as the application of Bayesian Sets and the development of the three-problem approach, are discussed in more detail at the end of the section.

II. Feature Extraction

We now introduce notation used throughout the paper. A graph G is a set of N vertices and directed edges (V, E), with associated adjacency matrix A. When considering whether a specific edge A_{ij} is real or fake, we label the outbound (i) node v_a and the inbound (j) node v_b. $\Gamma(v)$ denotes all neighbors of node v, while $\Gamma_{in}(v)$ and $\Gamma_{out}(v)$ denote just the inbound and outbound set of neighbors, respectively. We abbreviate the ith column of A as $A(:, i)$, and the jth row as $A(j, :)$. The undirected form of the adjacency matrix, A_u, is $A \vee A^T$. The area under the receiver-operator characteristic (ROC) curve is abbreviated AUC.

Link prediction methods are formed on the hypothesis that similar nodes are more likely to form connections. To this end, a link prediction method can be any general function $score(u, v)$ which produces a similarity ranking between all pairs of nodes in the graph. It is important to distinguish between the prediction task of the IJCNN challenge (namely, to separate real from fake edges in a given test set) and that of real-world link prediction, where one instead asks one of two questions:

1) Given a user v_a, with whom is that user most likely to connect?
2) Given a graph G, which user v_a is most likely to make a connection, and with whom?

These tasks are considerably more difficult. If the network is large, one must check N candidate nodes to see which has the highest probability of forming a new connection,

William Cukierski is with the Biomedical Engineering Department at Rutgers University and the University of Medicine and Dentistry of New Jersey.

Benjamin Hamner is a Whitaker Fellow at the École Polytechnique Fédérale de Lausanne.

Bo Yang is a software developer in the Vancouver, Canada area.

978-1-4244-9637-2/11/$26.00 ©2011 IEEE

then compute a similarity score with the remaining set of unconnected nodes (naively, checking $N - |\Gamma(v_a)|$ nodes). Additionally, certain methods which might work on the IJCNN data–for example using anomaly detection to locate edges which are "out of place"–are not directly applicable to real link prediction.

Instead of ranking the unconnected nodes, this paper employs similarity scores as features for supervised classification. This requires that, whenever possible, the scores be scaled to a common range, to eliminate the variation caused by different sized subgraphs and nodes with different numbers of neighbors. Since the final contest solution incorporates over ninety features, there is no way to describe each in full detail. Table I gives a short synopsis, with an appropriate reference if the feature cannot be succinctly summarized. In the rest of the section, we discuss methods to extract a local subgraph and meta approaches to build variations on link predictors. Lastly, we describe a limited subset of the most important features, focusing on those which worked best or were used in a novel way.

A. Extracting the Subgraph

The Flickr graph contains N = 1,133,547 users joined by 7,237,983 edges. Even with sparse matrices, there are similarity methods which can not run on the whole graph due to memory or processing limits. Certain methods are memory-bound because they require the similarity between all nodes be stored, with a worst case of $O(N^2)$. Other methods are runtime-bound, sometimes using computations which are $O(N^3)$ or worse (matrix inversions being the most frequent offender). It should be noted that there are published methods to approximate or optimize many of the significant link prediction algorithms (Katz [13], [14], SimRank [15], [16], [17], PageRank [18], [19], etc.). While these advanced approaches may or may not make the entire Flickr graph tractable, they often come with the added costs of specialized data structures, additional preprocessing steps, or rely on assumptions that only hold for undirected graphs. In a competition setting with limited time to implement, test, and run features, it can be impractical to adopt these more complicated approaches.

The natural alternative to processing the whole graph at once is to consider smaller neighborhoods relating to nodes of interest. Let G_{ab} be the subgraph consisting of all the nodes and edges defined as "relevant" to nodes v_a and v_b. Five methods to construct a local subgraph were tested, including combinations of the methods. Table II summarizes each method. Of the five, two were ruled out because they produced large subgraphs which required too much memory (2nd-level neighbors, distance-based criteria). The Bayesian Set method frequently created subgraphs in which the communities for v_a and v_b were isolated, causing a drop in performance of any subsequent graph-based similarity algorithms. The simple paths algorithm of Wang et al [20] showed promise, but had to be abandoned because it was too slow to enumerate paths in the Flickr data set[1]. Left was the simplest and fastest method to extract the subgraph: include only the first-level neighbors and any edges between them,

$$G_{ab} = \{\Gamma(v_a) \cup \Gamma(v_b)\}. \tag{1}$$

This subgraph was then used for all features that require computing and storing an all-pairs similarity.

B. Meta Approaches

Liben-Nowell and Kleinberg reviewed several "meta approaches" to link prediction [2], which encompass any approach that can be used to generate new features from existing link prediction methods. Some meta approaches are extremely simple. For example, one can apply the same predictors to A, A^T, and the undirected graph $A_u = A \vee A^T$. Additionally, one might use a weighted form of the adjacency matrix where each entry is row-normalized,

$$A_{ij} = A_{ij} / \sum_{j=1}^{N} A_{ij}, \tag{2}$$

thus having the effect that users with more connections count less than those with just a few.

The "unseen bigrams" method is a more sophisticated meta approach. If $V_a^{\{c\}}$ denotes the c nodes most related to v_a under $score(v_a, v)$, then a new score can be calculated from

$$score(v_a, v_b) = \sum_{v \in \Gamma(v_b) \cap V_a^{\{c\}}} score(v_a, v). \tag{3}$$

Rather than relying on the score between v_a and v_b, this approach calculates an aggregate similarity of v_b to the nodes most similar to v_a. This idea has both a significant advantage and disadvantage. When v has a small number of connections, the network contains less latent "information" about the node. Broadening the similarity score to include nodes similar to v can therefore improve knowledge about that node. However, finding $V_a^{\{c\}}$ (solving $\arg\max [score(v, v_a)]$) can take too long in a large graph, unless heuristics are put in place to limit the number of node similarities calculated. The need to calculate $V_a^{\{c\}}$ in reasonable time limited our use of bigrams to features in the small subgraph, G_{ab}.

We proposed an approximation to the unseen bigrams method which did not rely on computing the most similar nodes first. The already-connected neighbors were used in place of finding the most similar set of nodes. A similarity score was then defined as three separate problems (Fig. 1):

1) How similar is node v_a to v_b?

$$score(v_a, v_b)$$

[1]Even with the Boost Graph C++ Libraries, there were too many breadth-first searches required to process the nodes in the competition dataset in reasonable time

TABLE I

DESCRIPTION OF THE FEATURES

Name	Formula	Description
Katz [6]	$\beta A(v_a, v_b) + \beta^2 A^2(v_a, v_b) + \ldots + \beta^4 A^4(v_a, v_b)$	Truncated from $\sum_{l=1}^{\infty} A^l(v_a, v_b)$, weighted sum of all paths with length < 5 from v_a to v_b in G_{ab}
Common Neighbors	$\lvert\Gamma(v_a) \cap \Gamma(v_b)\rvert$	Num. common neighbors
Adar [7]	$\sum_{v \in \{\Gamma(v_a) \cap \Gamma(v_b)\}} \frac{1}{log\lvert\Gamma(v)\rvert}$	Weights connections with rare nodes more heavily
Jaccard [8]	$\lvert\Gamma(v_a) \cap \Gamma(v_b)\rvert / \lvert\Gamma(v_a) \cup \Gamma(v_b)\rvert$	Common neighbors normalized by total neighbors
Cosine	$\lvert\Gamma(v_a) \cap \Gamma(v_b)\rvert / (\lvert\Gamma(v_a)\rvert\lvert\Gamma(v_b)\rvert)$	Common neighbors normalized by pref. attachment
Preferential Attachment [9]	$\lvert\Gamma(v_a)\rvert\lvert\Gamma(v_b)\rvert$	New connections determined by size of current neighborhoods
Bayesian Sets [10]	$\max_c \sum bset_c(v_a) \cap bset_c(v_b)$	See Sec. II-E
	$\underset{c}{\text{argmin}} \lvert bayes\_set_c(v_a) \cap bayes\_set_c(v_b)\rvert = 1$	Position of first common node in the ranked queries
SVD Features	$\tilde{A}_u(v_a, v_b)$	Rank 80 SVD approximation
	$\tilde{A}_u(v_a, :) \cdot \tilde{A}_u(v_b, :)$	Dot product of columns v_a and v_b in low-rank approximation
	$mean\left[\tilde{A}_u(v_a, v \in \Gamma_{in}(v_b))\right]$	Mean SVD value of v_a with node v_b's neighbors
	% of $\tilde{A}_u(v_a, v \notin \Gamma_{in}(v_b)) < \tilde{A}_u(v_a, v_b)$	Percentage of non-existing edges with SVD values less than $\tilde{A}_u(v_a, v_b)$
SimRank [11]	$\frac{\gamma \sum_{v \in \Gamma(v_a)} \sum_{u \in \Gamma(v_b)} SimRank(v,u))}{\lvert\Gamma(v_a)\rvert\lvert\Gamma(v_b)\rvert}$	Recursive similarity calculated in G_{ab}. $SimRank(u, v) = 1$ if $u = v$
EdgeRank [12]	See Sec. II-C	Rooted PageRank over a weighted undirected adjacency matrix
Commute Time	See [2]	Random walk measure, from v_a to v_b and back on undirected G_{ab}
Bounded Walk	See [2]	Random walk with bounded path length in G_{ab}
PageRank	See [2]	Random walk with resets in G_{ab}
Maximum Flow	$\max f_{ab}$	Maximum flow treating v_a as the source and v_b as the sink
Betweenness Centrality	$\sum_{v \neq u} \frac{\sigma_{vu}(e_{ab})}{\sigma_{vu}}$, where $\sigma_{vu}(e_{ab})$ is the number of paths from v to u that traverse e_{ab}	Assume e_{ab} exists, calculate the edge betweenness centrality in G_{ab}
Core Number	N/A	Largest c s.t. v has $degree > 0$ when all vertices of $degree < c$ are removed
Shortest Paths Histogram	N/A	All shortest paths between $\Gamma(v_a)$ and $\Gamma(v_b)$
Power Law Exponent	$\frac{\log\left(\sum_{u,v} A(u,v)\right)}{\log(\lvert\Gamma(v_a) \cup \Gamma(v_b)\rvert)}$	Log ratio of number of edges to vertices in G_{ab}

TABLE II
METHODS FOR EXTRACTING LOCAL SUBGRAPHS

$G_{ab} = \{\Gamma(v_a) \cup \Gamma(v_b)\}$	1st-level neighbors of both nodes
$G_{ab} = \{\Gamma(\Gamma(v_a)) \cup \Gamma(\Gamma(v_b))\}$	2nd-level neighbors of both nodes
$G_{ab} = \{v \in G : d(v_a, v) < c\} \cup \{v \in G^T : d(v_b, v) < c'\}$	Set of nodes at distance less than c from node v_a, along with the nodes at distance less than c' to node v_b. c, c' were varied between 2 and 4.
$G_{ab} = \{bayes\_sets_c(v_a) \cup bayes\_sets_c(v_b)\}$	Nodes containing the top c scores in a bayesian sets query on v_a and v_b
G_{ab} = nodes most frequently appearing in the set of all simple (non-looping) paths from v_a to v_b	See section 3.1.1 of [20]

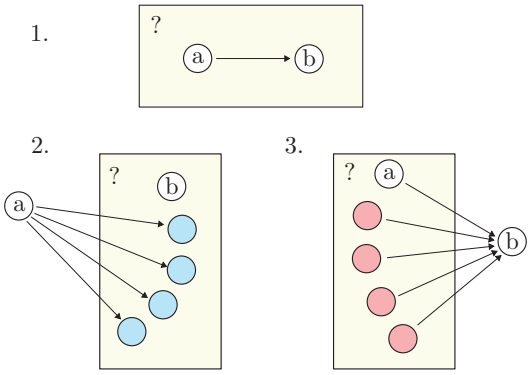

Fig. 1. Node similarity can be defined not just between v_a and v_b, but also between the appropriate neighbors of both nodes.

2) How similar are the outbound neighbors of v_a to v_b?

$$score(v_a, v_b) = \frac{1}{|\Gamma_{out}(v_a)|} \sum_{v \in \Gamma_{out}(v_a)} score(v_b, v)$$

3) How similar are the inbound neighbors of v_b to v_a?

$$score(v_a, v_b) = \frac{1}{|\Gamma_{in}(v_b)|} \sum_{v \in \Gamma_{in}(v_b)} score(v_a, v)$$

Once the code is written to compute $score(v_a, v_b)$, it takes only $|\Gamma_{out}(v_a)|$, $|\Gamma_{in}(v_b)|$ more comparisons to find $score(v_b, \Gamma_{out}(v_a))$ and $score(v_a, \Gamma_{in}(v_b))$.

The three problems above have a concrete meaning beyond their abstract mathematical role. Rewriting them in a more colloquial form, one can see they represent different categories of friendships which exist in the graph.

1) A and B are more likely to connect if A is like B. *(ex: A and B are friends in real life)*
2) A is more likely to connect to B if B is like A's friends. *(ex: A follows people who post photos of tennis, B is a tennis player)*
3) A is more likely to connect to B if A is like the people who like B. *(ex: B is famous photographer, A follows other famous photographers who have many fans)*

Using these three representations was central to our success in the contest. Number two, in particular, improved almost every link prediction method to which it was applied. Though less intensive than the unseen bigrams, we were still limited to using this approach on only the fastest features. Many users have hundreds or thousands of connections, so it can take hours per edge to calculate 2) and 3) if $score(v_a, v_b)$ is not extremely fast. This was too slow for the contest setting.

Singular value decomposition (SVD) was the final meta-approach used in our solution. SVD has been shown to work well for large sparse applications and gained popularity through its application to the Netflix Prize problem [21]. We used a rank 80 approximation of the Flickr graph, $\tilde{A}_u = (U\Sigma V^T)_{80}$. 80 was chosen as the largest rank the computer could reasonably handle. Since $\tilde{A}_u$ is not sparse and N is large, we store the matrices U, Σ, V^T and compute $svd(v_a, v_b)$ through the product $U_a \Sigma V_b$. The undirected graph was experimentally found to give higher AUC than the directed graph.

C. EdgeRank

The single highest-scoring feature found was termed EdgeRank and had an AUC of 0.926. This feature was a rooted PageRank (see [12]) over a weighted undirected adjacency matrix of the graph,

$$(1-d)\, x_0 + d\left(A + wA^T\right) x = x. \tag{4}$$

The vector x_0 is one at index v_a and zero elsewhere. The vector x represents the probability that a user starting at node v_a and randomly traversing the weighted graph will end up at each node. The user randomly restarts at node v_a with probability $(1-d)$. The solution to this equation was iteratively approximated as follows, starting with $n = 1$:

$$x_n = (1-d)\, x_0 + d\left(A + wA^T\right) x_{n-1}. \tag{5}$$

This feature has the advantage that, instead of simply computing the probability that an edge exists between nodes v_a and v_b, it ranks the likelihood of all nodes to which v_a does not point. An alternative optimization scheme was employed over one of the validation sets to solve the following problem:

$$\max_{d,w} \text{AUC}\left(V, \text{EdgeRank}\left(GV, V, d, w\right)\right). \tag{6}$$

For the IJCNN Flickr dataset, the optimal weights were found to be $d = 0.5$ and $w = 1$. These were relatively robust to variation, and the precise optimum varied based on the dataset used.

D. kNN

Initially, a genuine kNN method was implemented, however, it morphed into a group of related methods that can be considered kNN only in a loose sense. Nonetheless, we continued to refer to them as kNN throughout the competition and do so in this paper. They are more aptly described as general neighborhood and similarity-based methods.

In order to determine the similarity of nodes, an edge weight value was calculated between nodes. Edge weight decreases as the neighbor count goes up. Intuitively, consider one million people following a celebrity on a social network. Chances are most of them never met each other or the celebrity. On the other hand, if a user has 30 contacts in his/her social network, the chances are higher that many of them know each other.

For each node, edge weights were calculated separately for inbound and outbound edges,

$$w_i^{in} = \frac{1}{\sqrt{1 + |\Gamma_{in}(v_i)|}} \qquad (7)$$
$$w_i^{out} = \frac{1}{\sqrt{1 + |\Gamma_{out}(v_i)|}}.$$

The weight between v_a and v_b was then be constructed using different permutations of these weights,

$$\begin{aligned} w_{ab} &= 1 \qquad (8)\\ w_{ab} &= w_b^{in}\\ w_{ab} &= w_a^{out}\\ w_{ab} &= w_b^{in} + w_a^{out}\\ w_{ab} &= w_b^{in} w_a^{out}\\ &\dots \end{aligned}$$

Due to the sparseness of the graph, it was helpful to make use of 2nd-level neighbors. If v_c is a 2nd-level neighbor of v_a via v_b, then there are 4 possible types of paths from v_a to v_b to v_c: outbound-outbound (oo), outbound-inbound (oi), inbound-outbound (io), and inbound-inbound (ii). We will use w_{abc}^{type} to denote the link weight assigned to the path from v_a to v_b to v_c. The 2nd-level weight is calculated using a combination of the 1st-level neighbor weights. There are many ways to calculate this for each path type, for example:

$$\begin{aligned} w_{abc}^{oo} &= w_c^{in} \qquad (9)\\ w_{abc}^{oo} &= w_b^{out} + w_c^{in}\\ w_{abc}^{oo} &= w_a^{out} w_b^{out} w_c^{in}\\ &\dots \end{aligned}$$

The number of possible combinations is large (the general idea can also be extended to Nth levels). Due to the limited amount of time available, most time was spent on relatively simple forms of 1st and 2nd level functions. The final solution incorporated four variations of kNN, hand-picked for their usefulness when tested against the validation sets.

E. Bayesian Sets

Bayesian Sets [10] is a query method designed to retrieve items from a concept or cluster, based on example items from that cluster. The algorithm returns a ranked similarity score between the query and all other nodes in the graph. It was adapted to the task of link prediction by using the adjacency matrix as sparse association data, and then querying on v_a, v_b, $\Gamma(v_a)$, or $\Gamma(v_b)$. To our knowledge, this was the first application of Bayesian Sets to link prediction.

The best score using Bayesian sets was achieved by querying on v_a and v_b separately, then comparing the intersection of the two query results. The idea is that if v_a and v_b are connected, they will have similar sets of nodes returned by the query. Let $bset_c(v)$ be the top c scores resulting from a query on v. Then the similarity score was constructed using,

$$\max_c \sum bset_c(v_a) \cap bset_c(v_b), \qquad (10)$$

(where the intersect implies only the common members of each set are to be included in the sum). The "time" to the first common node in the query results was also included as a feature. This is the smallest number c such that the first c elements of each ranked query have at most one common node. Other types of queries, including those using the neighborhoods of each node, yielded worse AUC than using the single nodes.

III. Implementation

The feature extraction and classification was performed in parallel on an 8-core machine with 10GB of memory and an SSD hard drive, which was found to drastically improve performance when pageouts were necessary. The full graph was stored as a sparse matrix in shared memory, to allow each parallel worker access without the having a duplicate copy in memory. Two team members used Matlab for feature selection and the third used C++. The Boost Graph Library was used to perform efficient calculations of many graph-theoretic features, such as graph distances, betweenness centrality, and maximum flow. Feature extraction took approximately 10 seconds per edge. Edges whose nodes produced larger subgraphs ($N > 5000$) took several minutes.

The training data for the Random Forests was obtained by creating eight independent validation sets, each consisting of 4,000 data points (2,000 real edges and 2,000 fake edges). These validation sets were constructed to approximate the distributions of nodes and edges seen in the test set. Each validation set had 4,000 unique v_a nodes, while v_b nodes could be duplicates. Fake edge candidates were sampled from the full sets of nodes of each type. If the two nodes selected contained an edge, they were discarded and two new ones were sampled. No true edges were removed that would have left a node disconnected from any other nodes.

The predicted AUC from these validation sets did not match the reported leaderboard AUC for the competition;

the leaderboard AUC was typically around 0.03 to 0.04 lower, and methods which improved the predicted AUC ambiguously affected the leaderboard AUC. This brought our attention to an irregularity in the test dataset: there were far fewer v_a nodes with $|\Gamma_{out}(v_a)| = 1$ and far fewer v_b nodes with $|\Gamma_{in}(v_b)| \leq 1$ in the test dataset than there would have been, had the nodes been sampled in the above manner. This means that the test set did not contain many of the nodes that only had a single edge pointing to them, which almost always would have been false edges and thus easy to predict.

This explained the drop in AUC from the internal validation set to the leaderboard, and the methods to generate validation sets were modified to ignore nodes that were only connected to one other node for fake edges. This both improved the leaderboard score and brought the validation score in line with the leaderboard score. It also meant that if a node in the test set was only connected to one other node, it corresponded to a real edge.

IV. Results

There are several ways to measure performance in link prediction. The ROC area has the benefit of giving a global measure of a method's sensitivity and specificity. However, methods with a low ROC area might still be valuable if they are highly specific. The characteristic sparsity of social networks means the number of existing connections is small compared to the number which could possibly exist. This creates a bias towards specificity when suggesting new connections, since it is usually more desirable to produce accurate recommendations than thorough recommendations. There is also theoretical justification to include features with low ROC area. For example, the feature "$A(v_b, v_a) = 1$ and v_a forms mutual edges" has an AUC of only 0.541, but almost 100% specificity. This is valuable to a rule-based or decision tree classifier, even though it is not sensitive.

The best submission was obtained by learning 300 random forests with 5000 trees each for the binary link prediction problem. The number of variables considered at each split was chosen by cross validation to be $floor(\sqrt{num\_vars})$ (9 in the case of the 94 features used here). Random Forests performed the best out of all supervised machine learning methods evaluated, which included artificial neural networks, decision trees, boosted decision trees, support vector machines with a variety of kernels, relevance vector machines, and several Bayesian graphical models. Ensemble selection over different techniques was not found to significantly improve results. Random Forests performed well due to their ability to estimate variable importance and model noisy and complex interactions between features.

A total of 94 features were used in the final submission. Table III gives the individual AUCs of a subset of these features calculated on the test set. The AUCs are separated in to 5 columns according to which relationship they portray. Columns (1) through (3) are features applied to the three problems described in Sec. II. Columns (4) and (5) are features which pertain only to node v_a or v_b. Our method achieved an area under the ROC curve of 0.9695, the 2nd best overall score in the competition and the best score which did not employ de-anonymization on the graph. For comparison, the scores of the top five teams from the competition are listed in Table IV.

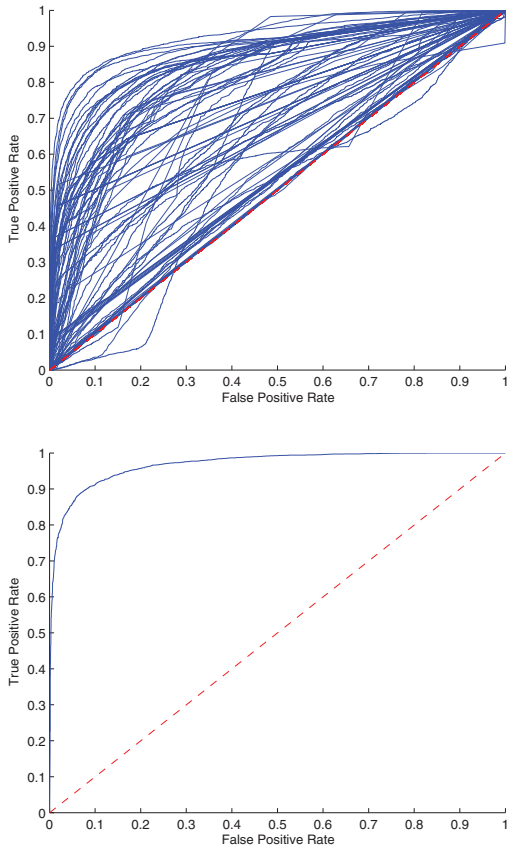

Fig. 2. ROC curves for all 94 features (top) and for the best final submission (bottom).

The results showed that the global features (run on the full G) tended to outperform local features (run on G_{ab}). This makes it difficult to directly compare features, since it is not known how much of the discrepancy is caused by the subgraph extraction step and how much is a limitation of the method. We suspect that some of the local features would improve significantly, given the computational power to run them on the whole graph.

V. Conclusions

Despite the large number of algorithms and ways to permute such algorithms, link prediction in large graphs is still a very challenging problem. Success in the IJCNN Social Network Challenge required a balance between feasibility, simplicity, and accuracy. There were many promising methods which were not tested due to hardware or time constraints. Our link prediction framework achieved high sensitivity and specificity when separating real from fake

TABLE III

AUCs OF INDIVIDUAL FEATURES (> 0.85 ARE IN BOLD)

Name	1)	2)	3)	v_a	v_b
Jaccard	0.721	**0.878**	**0.855**	-	-
Adar	0.721	0.837	0.687	-	-
Cosine	0.721	**0.861**	**0.867**	-	-
Common Neighbors	0.721	**0.864**	0.809	-	-
SimRank (G_{ab})	0.805	0.834	0.805	-	-
Katz (G_{ab})	0.772	0.542	0.760	-	-
EdgeRank	**0.926**	-	-	-	-
kNN 1	**0.880**	-	-	-	-
kNN 2	0.834	-	-	-	-
kNN 3	**0.914**	-	-	-	-
kNN 4	**0.907**	-	-	-	-
Bayesian Sets	**0.879**	-	-	-	-
Bayesian Sets time	0.839	-	-	-	-
SVD	0.823	-	-	-	-
SVD dot	0.804	-	-	-	-
SVD mean	0.845	-	-	-	-
SVD %	0.795	-	-	-	-
Commute Time (undirected G_{ab})	0.832		-	-	-
SimRank, Unseen Bigrams (G_{ab})	0.774	-	-	-	-
Katz, Unseen Bigrams (G_{ab})	0.663	-	-	-	-
Bounded Walks (G_{ab})	0.757	-	-	-	-
Maximum Flow (G_{ab})	0.751	-	-	-	-
Shortest Paths Histogram (G_{ab})	0.751	-	-	-	-
Global link distance	0.844	-	-	-	-
Power law exponent of G_{ab}	0.654	-	-	-	-
Betweenness Centrality (G_{ab})	0.628	-	-	-	-
$A(v_b, v_a) = 1$ and v_a forms mutual edges	0.541	-	-	-	-
# paths of length 2 from $v_a \rightarrow v_b$	0.680	-	-	-	-
$distance(v_a, v_b)$	0.731	-	-	-	-
$distance(v_b, v_a)$	0.536	-	-	-	-
Pref. Attach.	0.641	-	-	-	-
In Degree	-	-	-	0.501	0.718
Out Degree	-	-	-	0.504	0.535
Core Number In	-	-	-	0.564	0.730
Core Number Out	-	-	-	0.537	0.535
Pagerank (G_{ab})	-	-	-	0.693	0.535
Clustering Coeff. (G_{ab})	-	-	-	0.665	0.643

TABLE IV

TOP FIVE FINISHERS IN THE COMPETITION

Team Name	Final AUC
1. IND CCA	0.98115
2. **wcuk**	**0.96954**
3. vsh	0.95272
4. Jeremy Howard	0.94506
5. grec	0.92712

edges in a given test set, yet this is just one piece of the real-world link prediction problem.

It is common practice to abstract social networks as graphs and develop highly general methods for characterizing them. Unsurprisingly, few of these methods work best "out of the box." Networks possess different underlying dynamics of growth and attachment, while their edges can symbolize many forms of connections. The strength of our approach came not just from the breadth and depth of the individual methods, but also from empirically testing variations and permutations of those methods on the Flickr graph. Meta methods provided some of the best performing features in our approach, illustrating the importance of capturing a taxonomy of connections in otherwise identical graph edges.

REFERENCES

[1] M.E.J. Newman, "The structure of scientific collaboration networks," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 98, pp. 404, 2001.

[2] David Liben-Nowell and Jon Kleinberg, "The link prediction problem for social networks," in *Proceedings of the twelfth international conference on Information and knowledge management*, New York, NY, USA, 2003, CIKM '03, pp. 556–559, ACM.

[3] A. L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek, "Evolution of the social network of scientific collaborations," *Physica A: Statistical Mechanics and its Applications*, vol. 311, no. 3-4, pp. 590–614, 2002.

[4] M. E. J. Newman, "The structure and function of complex networks," *SIAM REVIEW*, vol. 45, pp. 167–256, 2003.

[5] Leo Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[6] Leo Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, March 1953.

[7] Lada A. Adamic and Eytan Adar, "Friends and neighbors on the web," *Social Networks*, vol. 25, no. 3, pp. 211–230, 2003.

[8] Paul Jaccard, "Étude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bulletin del la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

[9] M. E. J. Newman, "Clustering and preferential attachment in growing networks.," *Phys. Rev. E*, vol. 64, 2001.

[10] Zoubin Ghahramani and Katherine A. Heller, "Bayesian sets," in *Advances in Neural Information Processing Systems*, 2005.

[11] Glen Jeh and Jennifer Widom, "Simrank: a measure of structural-context similarity," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, 2002, pp. 538–543, ACM Press.

[12] Sergey Brin and Lawrence Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, pp. 107–117, April 1998.

[13] Pooya Esfandiar, Francesco Bonchi, David F. Gleich, Chen Greif, Laks V. S. Lakshmanan, and Byung-Won On, "Fast katz and commuters: Efficient estimation of social relatedness in large networks," in *WAW*, 2010, pp. 132–145.

[14] Kurt C. Foster, Stephen Q. Muth, John J. Potterat, and Richard B. Rothenberg, "A faster katz status score algorithm," *Comput. Math. Organ. Theory*, vol. 7, pp. 275–285, December 2001.

[15] Pei Li, Hongyan Liu, Jeffrey Xu Yu, Jun He, and Xiaoyong Du, "Fast single-pair simrank computation," in *SDM*, 2010, pp. 571–582.
[16] Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu, "Fast computation of simrank for static and dynamic information networks," in *Proceedings of the 13th International Conference on Extending Database Technology*, New York, NY, USA, 2010, EDBT 10, pp. 465–476, ACM.
[17] Dmitry Lizorkin, Pavel Velikhov, Maxim N. Grinev, and Denis Turdakov, "Accuracy estimate and optimization techniques for simrank computation," *VLDB J.*, vol. 19, no. 1, pp. 45–66, 2010.
[18] Gianna M. Del Corso, Antonio Gullí, and Francesco Romani, "Fast pagerank computation via a sparse linear system," *Internet Math*, vol. 2, pp. 118–130, 2004.
[19] Chris P. Lee, Gene H. Golub, and Stefanos A. Zenios, "A Fast Two-Stage Algorithm for Computing PageRank and Its Extensions," Tech. Rep., Stanford University, 2004.
[20] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy, "Local probabilistic models for link prediction," in *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, Washington, DC, USA, 2007, pp. 322–331, IEEE Computer Society.
[21] Chris Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, pp. 30–37, 2009.