Network Growth and Link Prediction Through an Empirical Lens

Qingyun Liu, Shiliang Tang, Xinyi Zhang, Xiaohan Zhao, Ben Y. Zhao and Haitao Zheng
Computer Science, UC Santa Barbara
{qingyun_liu, shiliang_tang, xyzhang, xiaohanzhao, ravenben, htzheng}@cs.ucsb.edu

ABSTRACT

Link prediction in dynamic networks is a well studied topic. Yet until recently, validation of algorithms has been hampered by limitations in the size and realism of empirical datasets. In this work, we seek to revisit and reassess the value and accuracy of prediction methods, by leveraging our access to several large, detailed traces of dynamics in online social networks (Facebook, Renren, YouTube). Our goals are to understand the absolute and comparative accuracy of existing prediction algorithms, and to develop techniques to improve them using insights from analysis of network dynamics.

We implement and evaluate 18 link prediction algorithms, labeled as either "metric-based" (those that predict potential links using a single similarity or proximity metric) or "classification-based" (those that use machine learning classifiers with multiple metrics as input features). Despite poor performance in absolute terms, SVM classifiers consistently perform the best across all our traces. Its accuracy is occasionally matched by metric-based algorithms, but never consistently across datasets. Finally, we use observations of network dynamics to build "filters" that dramatically reduce the search space for link candidates. Augmenting current algorithms with our filters dramatically improves prediction accuracy across all traces and algorithms.

1. INTRODUCTION

Link prediction is the problem of predicting formation of new edges on a given network. It is a fundamental problem that applies to networking in numerous contexts, including the Internet, the web, and online social networks. The sheer number of studies, including proposals for algorithms and models [2, 3, 9, 13, 14, 20, 23, 24, 26, 32, 33, 35, 39, 42], underscores the importance of the problem to a variety of applications, ranging from resource allocation in online services to offline efforts in counter-intelligence and counter-terrorism [19, 21].

As a technical problem, the efficacy of link prediction is generally not well understood. Today, link prediction algorithms are the basis for social recommendations in a wide range of social networks and applications, ranging from Facebook and Pinterest to personal streaming on Periscope and Q&A sites like Quora. The success of these sites and the sheer volume of prior literature lead many to believe the problem is well addressed. Only evidence to the contrary comes from anecdotes of failed recommendations that trigger potential privacy concerns [16].

Despite years of research in this space and hundreds of publications (only a small subset of which is cited here), there has been little opportunity to study these proposals from an empirical perspective. Until recently, public datasets of network dynamics have been limited to co-authorship studies and patent citation graphs, moderate sized networks which scale up to 20K nodes and 200K edges [4, 36]. In contrast, algorithms developed using and validated by these datasets are targeting dynamic networks that are two or more orders of magnitude larger, with millions or billions of nodes and billions of edges [44].

Thankfully, things are changing with the arrival of network traces from online social networks (OSNs). We are taking advantage of this opportunity (and availability to large traces of network dynamics) to step back and reassess the space of link prediction algorithms from an empirical perspective. We are motivated by questions such as:

- *How far have we come in understanding network growth and predicting the underlying processes that drive it? How far do we have to go?*
- *What lessons can we draw from the successes (and failures) of existing algorithms?*
- *Can we improve existing approaches by leveraging more data, e.g. detailed temporal network history?*

In this work, we perform an empirical study using large traces of network growth from three large OSNs, Facebook, Renren (Facebook equivalent in China), and YouTube. In each case, detailed timestamps capture the time when specific edges were created between nodes (users) in the network. To the best of our knowledge, these are the only

IMC 2016, November 14-16, 2016, Santa Monica, CA, USA
© 2016 ACM. ISBN 978-1-4503-4526-2/16/11…$15.00
DOI: http://dx.doi.org/10.1145/2987443.2987452

publicly available datasets suitable for this study, both sufficiently large and with sufficiently detailed timestamps to capture graph dynamics. These traces cover substantial subsets of users in each network, and in the case of Renren, the entire user population at a time when the network included 10 million users. We discretize these traces into numerous temporal snapshots, and use them to drive the evaluation of 18 representative link prediction algorithms. Finally, we use our lessons and observations from analyzing these network dynamics to build "filters" that help prune the set of candidate nodes for edge creation. By applying these filters before link prediction, we can reduce the search space and focus on regions of likely growth.

To better understand and compare results across prediction algorithms, we classify them into two groups. First, *Metric-based prediction algorithms* define specific metrics that can be computed for all potential links, where a potential link with a higher score on the metric indicates a higher probability of formation. For our analysis, we implemented 14 distinct metrics that had scalable algorithms in existing literature. In contrast, *classification-based prediction algorithms* utilize machine learning classifiers that take multiple metrics as input features, and produce a prediction of likelihood of formation for each potential link. Some methods produce a detailed probability while others produce a binary result. Experiments in our study cover support vector machines (SVM), logistic regression, naive Bayesian networks, and random forests, each using all 14 metrics as input features. Our experiments show that more complex techniques, *e.g.* larger ensemble methods do not produce noticeable improvements in accuracy.

As our first result, we find that link prediction performance remains poor in absolute terms. Correctly predicting link formation within some timeframe is difficult, and the problem only grows harder, as each new node brings $\sim N$ more potential links to a network of size N. Second, we find that for each of our traces, metric-based prediction algorithms vary significantly in accuracy. In each case, a small subset of metric-based predictors do as well as (and occasionally outperform) the most accurate machine learning based classifier (SVM in all cases). We note that while a few metrics perform consistently well, no single metric predictor consistently performs as well as SVM across all networks. Instead, there appears to be a strong correlation between network structure and the relative success of specific metric-based algorithms. Machine learning methods do well in part because they automatically adjust weights across different metrics, emphasizing those that match the targeted network without *a priori* knowledge of its structure. Without such knowledge, we can either achieve "good" accuracy by choosing a consistently strong metric, or achieve "near optimal" accuracy by using a ML classifier (at the cost of higher computational and training costs).

Finally, we revisit existing prediction algorithms with the goal of augmenting them by leveraging knowledge of past network dynamics. Our insight is to provide "temporal filters" that significantly reduce the set of potential new links, reducing the search space and computational cost, while focusing predictors on more probable link candidates. Our filters are focused around trends in node activity and potential link distances, both patterns observed in this and prior studies of network dynamics. Applying these filters produce very encouraging results, in many cases effectively doubling the predictive power of both metric-based and classifier-based algorithms. Not only do these filters outperform recent methods leveraging temporal information, but they can be combined with temporal methods to provide even better results.

Our key contributions can be summarized as follows:

- We carry out a comprehensive analysis of a wide range of link prediction algorithms, studying not only their performance but also possible causes of low prediction accuracy. We apply decision tree classifiers to identify the best metric-based algorithms for different networks.
- We compare the two categories of link prediction methods, *i.e.* metric-based and classification methods, study their cost versus accuracy tradeoffs, and identify strategies for choosing between them.
- We leverage insights from analysis of network growth to design filters that improve prediction accuracy by dramatically reducing the search space. In our tests, these filters significantly improve prediction power across all methods. Further, they outperform recent proposals that integrate temporal information, and can be combined with them to produce even better results.

2. BACKGROUND: LINK PREDICTION

Link prediction identifies new edges that will likely form in the near future, by analyzing the structure of the current network [23]. Given a graph $G_t = <V_t, E_t>$ observed at time t, it seeks to predict new edges to be created between nodes V_t at time t' ($t' > t$).[1] Note that we focus on predicting *future* links at some time t, which is different from the detection of *missing* links, where given a partially observed graph, it identifies link status for unobserved pair of nodes [17, 29].

Existing link prediction algorithms naturally fall into two categories, which we refer to as *metric-* and *classification-based*. We list and classify all of the known popular prediction algorithms in Table 1, which are algorithms we focus on in this work, and their details will be introduced later in Table 3. Metric-based algorithms estimate the likelihood of future connectivity between unconnected nodes, by generating a numeric score based on some graph-based heuristic [23] or models [9, 35]. All potential node pairs are sorted by score to determine the most likely future edges. In contrast, classification-based algorithms treat link prediction as a classification problem [3]. Using scores by metric-based algorithms and maybe other information as training features, these classifiers then "separates" the node pairs that will likely connect in the near future from those that will not. Some classifiers also produce a granular similarity score, which can be used to rank node pairs.

[1] This is the most common form of link prediction. It does not consider edges created by new nodes who join after t, nor edges that might disappear after their creation.

Metric-Based Prediction			Classification-Based Prediction
Heuristics	Learning Models		*e.g.*, SVM, Logistic Regression, Naive Bayes, Random Forest
e.g., CN, JC, AA, RA, LP, SP, PA, Katz, LRW, PPR	Probabilistic Models	Matrix/Tensor Based	
	e.g., BCN, BAA, BRA	*e.g.*, Rescal	

Table 1: Summary of link prediction algorithms, with details listed in Table 3.

Graph	Trace Start			Trace End			Time	Snapshot	# of
	Date	Nodes	Edges	Date	Nodes	Edges	Granularity	Delta (k)	Snapshots
Facebook (New Orleans) [41]	09/05/06	48,969	339,098	01/21/09	63,731	817,090	Seconds	15K	31
YouTube (Snowball Crawl) [30]	02/09/07	1,406,188	3,466,440	07/23/07	3,223,589	9,376,594	Days	250K	21
Renren (Non-sampled) [44]	01/01/07	1,413,731	13,616,792	12/31/07	10,572,832	199,564,006	Seconds	10M	17

Table 2: Statistics of the three OSN datasets.

Next, we describe these two categories of algorithms in detail and highlight their differences.

Metric-based Prediction. Metric-based link prediction algorithms quantify and rank node pairs by their likelihood of forming new edges, based on specific metrics that capture similarity or proximity between nodes [23]. For simplicity, we refer to the entire group as “similarity metrics,” and further divide them into *heuristics*, or more complicated *learning models*, as shown in Table 1.

Many popular metric-based algorithms are *heuristics* based on common intuitions of graph formation [23], *e.g.*, two currently unlinked nodes with the most commonly connected nodes are most likely to link in the future. Those hypotheses are driven by graph structural properties and do not require metadata. They generally focus either on node neighborhood information, where they capture properties of the common neighborhood between nodes of 2-hop distance, *e.g.* Common Neighbors [32] and Adamic Adar Index [2], or on path properties such as shortest path length [20].

Link prediction can also be performed by inferring the likelihood of two nodes forming an edge based on *learning models*. One way is to use probabilistic models calibrated by measurements on G_t. For example, [9, 13] assume a specific underlying structure of hierarchies or communities exists in the graph G_t, and model parameters are estimated using maximum likelihood. Another approach is to extend the field of relational learning to link prediction [39, 42]. However, these underlying models either do not scale to large graphs (due to complexity in parameter learning) or rely on special conditions that do not generalize to common networks such as social networks [42]). Only the local naive Bayes model [26] meets the needs of large, generalized graphs. Other metrics use matrix (tensor) techniques on matrix representations of graphs. They capture node similarity in a latent space, defined by different models [33, 35]. Among them, only *Rescal* [33] has been shown empirically to scale to large graphs of millions of nodes.

Classification-based Prediction. While metric-based algorithms are known for their simplicity [23], performance can vary significantly depending on the specific similarity metric used. Existing work has shown the best metric varies across datasets and there is no unified solution [23, 24].

The alternative is what we call classification-based methods. Instead of using a certain similarity metric, one can build automated classifiers to explore multiple similarity features [24]. Compared to single metrics, classifiers face the challenge of high computational complexity, (*e.g.* feature selection and training), especially for massive OSNs [14].

3. DATASETS AND METHODOLOGY

We now describe the datasets used for our study and our experimental methodology.

3.1 Datasets

Our study uses large traces of dynamic network growth from three different networks, Renren, Facebook, and YouTube. As far as we know, these are the only publicly available large-scale datasets suitable for this study, which have sufficiently detailed timestamps to capture graph dynamics, *i.e.*, the time when each edge (link) was created between nodes (users).

The Renren [44] data includes creation of every edge in the entire Renren network during a period of over 2 years (from its first edge, to 10 million users, 199 million edges when the trace ends). The Facebook trace [41] includes edges created in the New Orleans regional network over 2+ years. The YouTube trace [30] includes edges recorded from daily snowball crawls of a user community that grew from 1 million to 3 million users over a period of 5 months. To avoid disruptions from external events, *i.e.*, the network policy changes in Youtube, and a one-time network merge event for Renren (Renren merged with its largest competitor in December 2006), we use continuous subtraces that do not include the external events in question. Statistics on all three traces are summarized in Table 2.

We show each network’s daily growth in nodes and edges in Figure 1. While the three networks all continue on exponential growth trajectories (Facebook has a number of 49K users at the beginning while the other two has 1.4M users), we see Renren is on a much faster growing pace. This is because both the Facebook and YouTube datasets are sampled networks, *i.e.*, Facebook dataset is a regional network and YouTube dataset depicts the growth of a user community. Figures 2-4 provide a quick look at the change in basic net-

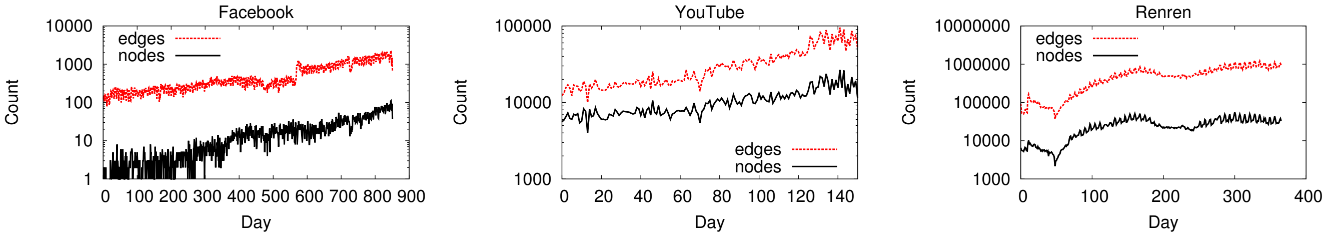


Figure 1: Daily new nodes and edges in the three networks.

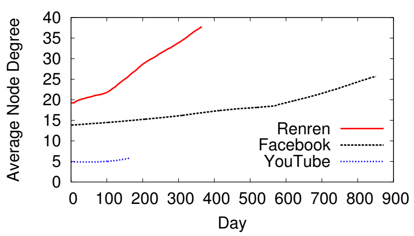


Figure 2: Average node degree

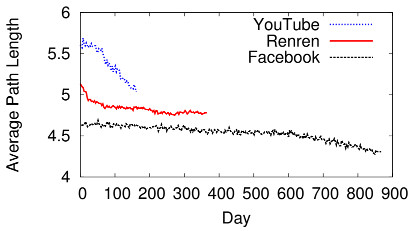


Figure 3: Average path length

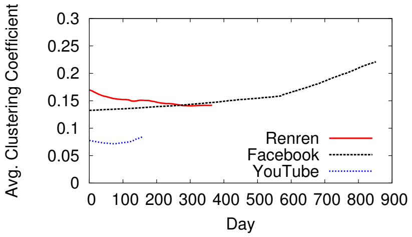


Figure 4: Average clustering coefficient

work properties over its evolution, including average node degree, path length, and clustering coefficient. Unsurprisingly, average node degree for all three networks grows over time. In comparison, Renren and Facebook are much denser than YouTube. Unsurprisingly, networks grow and densify over time, and their average path length shrinks. YouTube has the largest path length due to its sparsity.

3.2 Methodology

Existing link prediction studies focus on predicting edges between two static snapshots [23, 3, 11], and most do not capture the evolution of fast growing networks such as OSNs like Facebook, LinkedIn and Renren. In contrast, our work seeks to answer two key questions:

Q1: Can existing algorithms accurately predict the continuous edge (or link) growth of today's large, dynamic, online social networks?

Q2: Can we utilize temporal network data to improve prediction accuracy?

Evaluating Link Prediction on Graph Sequences. To answer these questions, we apply a *sequence-based* framework to evaluate existing link prediction algorithms as the network grows. We process each dataset to generate a sequence of graph snapshots $(G_1, G_2, ..., G_T)$ while keeping the number of new edges created in each snapshot constant. We refer to this number as the *snapshot delta*. We run each algorithm in every graph snapshot $G_{t-1}(1 < t \leq T)$ to predict new edges (among existing nodes) that will appear in the next snapshot G_t, and compare them to the ground truth, *i.e.* the actual new edges found in G_t. We choose the snapshot delta value to ensure sufficient number of snapshots for analysis (> 15) while ensuring duration between two successive snapshots is not too long (< 2 weeks). Specific values and the resulting number of snapshots for each dataset are listed in Table 2.

To address Q1, we evaluate 14 different metric-based algorithms that can scale to our large datasets, and 4 widely used classification-based algorithms. For each algorithm, we study its prediction accuracy as the network grows and identify potential causes for any loss of accuracy. We answer Q2 by analyzing the temporal properties in edge creation to identify and utilize trends as additional metrics to improve prediction accuracy.

Implementing Metric-based Algorithms. We first cover 10 most popular heuristics: Common Neighbors (CN), Jaccard's Coefficient (JC), Adamic/Adar Index (AA), Resource Allocation Index (RA), which focus on capturing properties of the common neighborhood between nodes of 2-hop distance; Preferential Attachment (PA), which is based on node degree; Local Path (LP), Local Random Walk (LRW), Shortest Path (SP), Personalized PageRank (PPR), and Katz, which are driven by path properties. We also include 1 tensor-based algorithm, *i.e.*, Rescal [33], which works by condensing the interaction among nodes into a latent space. And finally we implement 3 probabilistic algorithms [26], which account for different roles by different common neighboring nodes between node pairs, *i.e.*, Local Naive Bayes based Common Neighbors (BCN), Local Naive Bayes based Adamic Adar (BAA), and Local Naive Bayes based Resource Allocation (BRA). These cover the metric-based approaches (see Table 1), and we summarize their detailed implementation in Table 3.

We also fine-tune our implementation by identifying the best parameters and approximation methods (if any) based on results of our own experiments and from prior studies. Specifically, LP requires a weight parameter ϵ for 3-hop paths, and $\epsilon = 0.0001$ provides the highest accuracy. For PPR, we configure the restart probability $\alpha = 0.15$ as suggested by prior work [5]. For Katz, we set $\beta = 0.001$ as suggested by [1], and implement two approximation methods:

Metric-based Algorithms	Precise Formulation
CN (Common Neighbors) [32]	$\lvert\Gamma(u) \cap \Gamma(v)\rvert$
JC (Jaccard's Coefficient) [23]	$\frac{\lvert\Gamma(u)\cap\Gamma(v)\rvert}{\lvert\Gamma(u)\cup\Gamma(v)\rvert}$
AA (Adamic/Adar) [2]	$\sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{1}{\log(deg(w))}$
RA (Resource Allocation) [45]	$\sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{1}{deg(w)}$
BCN [26]	$\lvert\Gamma(u) \cap \Gamma(v)\rvert \log(s) + \sum_{w\in\Gamma(u)\cap\Gamma(v)} \log(R_w)$, where $s = \frac{\lvert V\rvert(\lvert V\rvert-1)}{2\lvert E\rvert} - 1$,$R_w = \frac{N_{\triangle w}+1}{N_{\wedge w}+1}$, $N_{\triangle u}$ ($N_{\wedge u}$): # of triangles (non-triangles) involving u.
BAA [26]	$\sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{1}{\log(deg(w))}(\log(s) + \log(R_w))$
BRA [26]	$\sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{1}{deg(w)}(\log(s) + \log(R_w))$
Katz [18]	$\sum_{l=1}^{\infty} \beta^l \cdot \lvert paths_{u,v}^{<l>}\rvert$ where $\beta > 0$, $paths_{u,v}^{<l>}$: all l-hop paths between u and$v$
LP (Local Path) [45]	$\lvert paths_{u,v}^{<2>}\rvert + \epsilon \cdot \lvert paths_{u,v}^{<3>}\rvert$
PPR (Personalized PageRank) [5]	$\pi_{u,v} + \pi_{v,u}$ where $\pi_{u,v}$: probability of a random walk from u to v with a restart probability $\alpha \in [0, 1]$
LRW (Local Random Walk) [25]	$\frac{deg(u)}{2\lvert E\rvert}\pi_{uv}(m) + \frac{deg(v)}{2\lvert E\rvert}\pi_{vu}(m)$, where $\pi_{uv}(m)$: probability from u to v after m steps
SP (Shortest Path)	# of hops on shortest path between u and v
PA (Preferential Attachment) [6]	$deg(u) \cdot deg(v)$
Rescal [33]	$XRX^T(u, v) + XRX^T(v, u)$ where adjacent matrix $A \approx XRX^T$, X: a $\lvert V\rvert \times r$ matrix, R: a $r \times r$ matrix

Table 3: The 14 metric-based algorithms used for our study. Notations: given graph $G =< V, E >$, u and v are two graph nodes, $\Gamma(u)$ denotes the neighbors of node u, $deg(u)$ represents the node degree of u.

low rank approximation (Katz$_{lr}$) [1] and scalable proximity estimation (Katz$_{sc}$) [38]. Our experiments in §4 show that while more accurate than Katz$_{sc}$, Katz$_{lr}$ does not scale to larger networks, since it computes Katz score for all candidate node pairs[2]. Thus for Renren and YouTube, we terminate the Katz$_{lr}$ experiments at snapshots of 65M edges and 5.5M edges, respectively.

In terms of computation cost, the local metrics (CN, JC, AA, RA, BAA, BCN, BRA) are easy to compute since we only need to compute each node's 2-hop neighbors. PA is also fast because one can optimize the implementation to only consider top-K node pairs. Even for our largest Renren graph, the computation for the above eight metrics finishes within a few minutes (we run the C++ implementation on 10 standard servers, each with 8 cores and 192GB RAM). The next three metrics (LRW, PPR and LP) take a few hours to compute because LRW and PPR require random walk computation while LP requires reaching 3-hop neighbors. Finally, the most complex metrics (Rescal, Katz and SP) take a few days to complete since they require node embedding. We also note that for the classifier-based methods, the computation complexity is dominated by feature calculation, *i.e.* computing the above similarity metrics.

4. METRIC-BASED PREDICTION

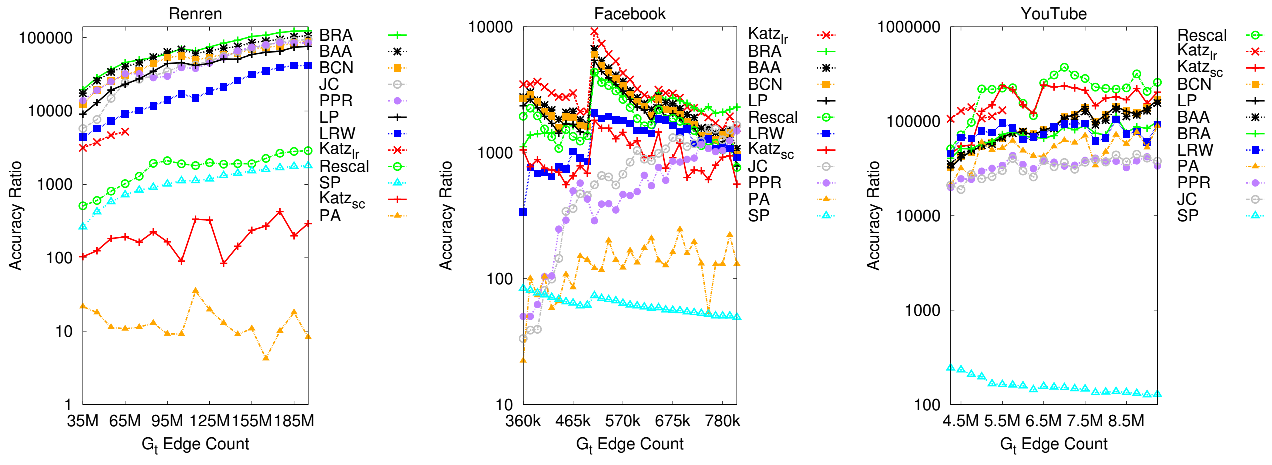
Our empirical evaluation begins with metric-based prediction algorithms. We seek to understand their prediction accuracy, and the key factors that lead to prediction errors.

4.1 Experimental Setup

Given a sequence of snapshots $\{G_1, G_2, ..., G_T\}$, we predict the new edges to appear in G_t based on observed G_{t-1}. For each of the 14 metric-based algorithms, we compute the similarity metric score for each unconnected node pair, and select the top k node pairs with the highest score. While the choice of k may affect prediction accuracy, we use the ground truth value, *i.e.* k equals the number of new edges among V_{t-1} nodes appeared in G_t but not in G_{t-1}. This allows us to focus on the effectiveness of similarity metrics. As a baseline for comparison, we also implement a *random prediction* algorithm, which uniform-randomly picks k unconnected node pairs from V_{t-1} as the predicted new edges in G_t.

Performance Metrics. We follow the established practice of evaluating each link prediction algorithm by comparing results to those from random prediction, *i.e.* in terms of the factor improvement over random [23]. Specifically, given a similarity metric M, let E_t^M represent the set of correctly predicted node pairs that become connected in G_t, *i.e.* the overlap between the predicted top k node pairs to connect and those that actually connect in G_t. Let E_t^R be the set of correctly predicted edges using random prediction, with an expected size of $\overline{|E_t^R|}$. Thus the performance metric is the improvement factor or *accuracy ratio* [23], $|E_t^M|/\overline{|E_t^R|}$. If the ratio is larger than 1, prediction using metric M is more accurate than random prediction (by predicting k edges). Note that we choose to use *accuracy ratio* rather than the area under the receiver operating characteristic curve (AUC) because AUC evaluates link prediction performance according to the entire list of the predicted node pairs [28], while our goal is to evaluate the accuracy of top k predicted node pairs. This allows us to focus on examining the effectiveness of similarity metrics.

[2] Even using 8 machines with 192GB memory each, calculating Katz$_{lr}$ for a Renren snapshot with 185M edges takes 27 days.



(a) (b) (c)

Figure 5: Link prediction performance (in terms of accuracy ratio) of all metric-based prediction algorithms. We omit the results of CN, AA and RA because they perform similarly (slightly worse) than their Local Naive Bayes versions, *i.e.* BCN, BAA and BRA. The results for Katz$_{lr}$ in Renren and YouTube are capped to 65M and 5.5M edges due to computation complexity.

Network	JC	BCN	BAA	BRA	LP	LRW	PPR	SP	$Katz_{lr}$	$Katz_{sc}$	Rescal	PA
Renren	1.72	2.40	3.22	3.52	1.75	1.06	2.44	0.053	0.82	0.018	0.091	0.0068
Facebook	1.21	6.17	6.82	4.43	5.53	2.11	1.06	0.10	9.41	1.85	4.45	0.21
YouTube	0.22	0.59	0.53	0.44	0.60	0.58	0.23	0.0021	0.98	1.44	1.75	0.38

Table 4: Best possible absolute accuracy (%) of all prediction methods on each dataset.

4.2 Metric-based Prediction Accuracy

Absolute Prediction Accuracy. We start by first looking at the raw prediction accuracy results in absolute terms, *i.e.* ratio of correctly predicted edges that match real new edges. For each consecutive pair of snapshots G_{t-1} and G_t, we apply each prediction algorithm on G_{t-1} generate the next k links likely to form, and compute the overlap in the result with the k links actually formed in G_t: $|E_t^M|/k$.

Prediction accuracy was quite low across the board, for all algorithms on all snapshots across all of our datasets. To highlight these accuracy results, we show in Table 4 the highest absolute accuracy results obtained by each algorithm over any snapshot pair across our datasets.

It is clear to see that in absolute terms, link prediction performs poorly in practice. While some methods consistently do better than others, the best they can do is accuracy in the single digits in percentages, *e.g.* 5–6%. The best results tend to come from the Facebook dataset, likely because it's significantly smaller (33 times fewer nodes) than the Youtube and Renren datasets. The single best result is Katz$_{lr}$, which reaches 9.41% on Facebook, but fails to reach even 1% on the larger datasets. Note that our definition of "accuracy" is loose, in that it only requires a predicted link to appear within some range of k new links (see Table 2), where k represents all links created in a time period ranging from one week (YouTube) to four weeks (Renren).

These numbers are likely to be significantly lower for real networks, which contain orders of magnitude more nodes (and therefore many orders of magnitude more potential new links) than our datasets, *e.g.* Facebook, WhatsApp, Pinterest etc. While our results are limited by reliance on only network structure (existing links), these results highlight the fact that link prediction is far from a solved problem. These results explain why link prediction literature typically uses the *accuracy ratio* [23], which compares results to a purely random algorithm. We will use the accuracy ratio metric for the rest of our analysis.

Accuracy Ratio Results. We present prediction results of our 14 metric-based algorithms in Figure 5, as the accuracy ratio over the sequence of snapshots for each OSN (marked by their total edge count). We omit the results of CN, AA and RA because they perform similarly (slightly worse) than their Local Naive Bayes versions, *i.e.* BCN, BAA and BRA. We include two implementations of Katz: Katz$_{lr}$ and Katz$_{sc}$, where Katz$_{lr}$ almost consistently outperforms Katz$_{sc}$, but is difficult to scale on Renren and Youtube. For the rest of the paper we only show analysis of Katz$_{lr}$ and refer to it as *Katz*.

We make two key observations from Figure 5. *First*, as expected, all metric-based algorithms outperform random prediction over each entire sequence of snapshots. The largest improvement on accuracy ratio is more than 100,000 times for Renren and YouTube, and 6000 for Facebook. A major contributor to this magnitude of differential is the large network sizes, where the accuracy of random prediction quickly decreases as network size grows, resulting in a much higher accuracy ratio.

Second, while the best algorithm varies across the three networks, there are algorithms, *i.e.*,SP and PA, which consistently perform poorly. SP gives all 2-hop node pairs the highest score, thus its prediction is actually random choice over all such pairs. PA tries to capture "the rich get richer" property, which is not dominant in friendship creation networks (*i.e.*, Renren and Facebook), where joint efforts from both users are required [44]. PA achieves marginally better accuracy ratio in YouTube, which is more of a subscription network where popular users attract more followers.

Impact of Network Structures. As mentioned before, Renren and Facebook are more similar in underlying structures since they are both traditional social networks. Our results from Figure 5 align with this observation that top algorithms are similar on Renren and Facebook, *i.e.*, both include common neighbor based algorithms BRA, BAA and BCN. Renren is slightly different from Facebook in that it is a non-sampled graph, and therefore captures higher connectivity between nodes compared to the subsampled regional network in Facebook. Thus Katz is hard to scale on Renren and JC and PPR perform much better. JC and PPR prefer pairs with both low degree nodes, which are usually inactive (more in §4.4) and are most common in the early phase of our Facebook trace, and decrease as the Facebook network grows over time. We can see their clearly increasing accuracy ratio in Figure 5(b).

In contrast, YouTube is more of a subscription network, where many super nodes with extremely high degrees remain super active in link creation. Thus YouTube has much higher node heterogeneity and lower network assortativity. We find that more than 40% new edges involve the top 0.1% nodes with highest degrees in YouTube, while only less than 3% for Facebook and Renren. Also, among edges created by super nodes, most are low degree nodes (80% with degree < 20). We confirm this by measuring the assortativity for each network. It stays consistently negative for YouTube, and generally positive for the other two.

The difference in network structures produces significantly different link prediction results in YouTube. Because they prefer node pairs with both high degrees, BRA, BAA and BCN do not rank highly amongst metrics (note that the y-axis in Figure 5 is in logscale). PPR and JC perform very poorly because most nodes have very low degree ($\sim$80% nodes with degree ≤ 3). The outperformer is Rescal, which achieves extremely good performance. Rescal works by condensing the interaction among nodes into a much smaller latent space, where it models the interaction between latent components instead, and assigns nodes corresponding weights for each latent component. Intuitively, super nodes are critical in many roles thus have much higher weights, leading to a higher final score. Our results also confirm that super nodes are weighted extremely highly while the rest share similar weights. In this way Rescal best captures the negative assortativity in YouTube.

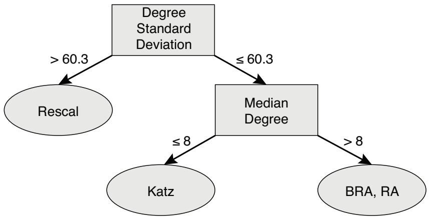


Figure 6: Visualization of classification results on choosing the best metric-based algorithm.

Correlation of Accuracy with 2-hop Edge Ratio. We observe that most algorithms increase in accuracy ratio with network growth, but only for Renren and YouTube, not Facebook. Our analysis shows that this could be explained by a dependence on link creation between 2-hop neighbors, i.e. λ_2, the percentage of 2-hop node pairs in G_{t-1} who form edges in G_t. A plot of λ_2 shows that it increases with network growth in Renren/YouTube, but decreases (after a matching spike) in Facebook. This is explained by the trend towards "densification" over time [22]. This is disrupted in the Facebook trace, because subsampling over the regional network breaks an increasing number of cross-regional edges as the network grows. We compute the average Pearson correlation of the top-performing 6 metrics for each graph to λ_2. The results are 0.95 for Renren, 0.83 for YouTube and 0.81 for Facebook.

Summary. Our results produce two key takeaways. *First*, the underlying network structure heavily impact prediction accuracy of metric-based algorithms (in terms of accuracy ratio). The more similar network structures in Renren and Facebook (links in which are both the abstraction of friendship between users, while YouTube is more of a subscription network) means their prediction results show consistent relative performance. *Second*, prediction accuracy of most metric-based algorithms strongly correlate with the ratio of 2-hop edges in network evolution, because their predictions are dominated by 2-hop edges.

4.3 Choosing Metric-based Algorithms

Since network structures heavily impact the performance of metric-based algorithms, a natural question is "given a network, can one *predict* the best link prediction algorithm?" And similarly, "given an algorithm, can we *characterize* the kind of networks on which it provides the most accurate link prediction?"

We answer the first question by training a multi-class classifier (decision tree), where the input features are the network properties and each class represents a (winning) link prediction algorithm (14 classes in total). We treat each graph snapshot as a data point, and create 69 data points across our three datasets. We consider the following features (computed from each snapshot): node count, edge count, node degree distribution (average, standard deviation, x-percentile), clustering coefficient, average path length, and network assortativity.

Figure 6 shows the resulting decision tree, where Rescal,

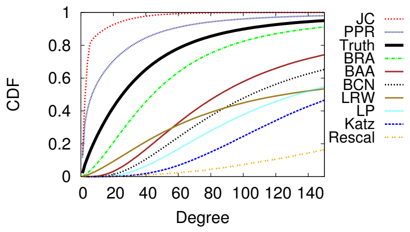


Figure 7: Degree distribution of nodes in predicted edges (Renren, $55M$ edges).

Metric	Predicted Edges	Real Edges
Rescal	99.5%	0.5%
LRW	66.7%	0.6%
Katz	39.7%	0.6%
LP	33.3%	0.5%
BCN	24.2%	0.5%
BAA	16.4%	0.5%
BRA	4.7%	0.8%

Table 5: Ratio of predicted and actual created edges that involve 0.1% most frequently predicted nodes (Renren snapshot with $55M$ edges).

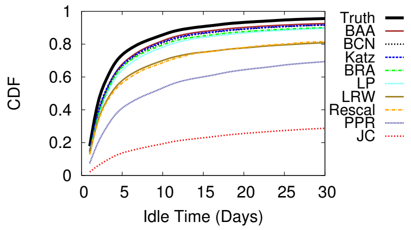


Figure 8: CDF of node idle time in predicted edges (Renren, $55M$ edges).

Katz and BRA (RA) are among the best performing algorithms (consistent with Figure 5). We see that the heterogeneity of node degrees in the network (captured as degree standard deviation) is the highest impact feature. It specifies that networks with high node degree heterogeneity should use Rescal, which aligns with our analysis in §4.2 that Rescal prefers node pairs with higher degree heterogeneity. The next factor is the median node degree where lower values ($\leq$8) marks Katz due to its limited scalability and higher values points to BRA (RA) which prefer high-degree node pairs.

Note that this result is not meant as a definitive guide to choosing link prediction approaches for different types of graphs. Our training set for the decision tree is relatively small, and only covers three distinct types of networks. A more "robust" result would require data from a wide range of networks with varying characteristics, with even more snapshots per network. We only use the results here to demonstrate general trends between key features, which are consistent with our detailed experimental results (Figure 5).

To answer the second question, we train a binary classifier (decision tree) for each algorithm where the inputs are the same set of network properties. We consider an algorithm to provide "good" prediction (*i.e.* positive) if its prediction accuracy ratio is within 90% of the optimal algorithm. The classification results are shown as below: (we omit algorithms for which there are few or no positive results):

- Rescal: standard deviation of node degree> 60.3
- Katz: # of edges$\leq 4.5M$
- BRA (RA): median node degree> 7

The results are consistent: Rescal is best for networks with high node degree heterogeneity, Katz is suitable for networks of limited scale and BRA (RA) is best for high-density networks.

Summary of Observations. We train classifiers to explore the correlation between the networks and metric-based link prediction algorithms. While we are limited to our three large network traces, we believe our results do provide some insights on today's metric-based link prediction algorithms:

- On sparse and small networks, Katz is a good choice.
- On dense and large networks, BRA (RA) performs well.
- On networks with high node heterogeneity, Rescal is likely the best solution.

4.4 Sources of Low Prediction Accuracy

While metric-based prediction largely outperforms random prediction, accuracy is still low in absolute terms. For example, the best similarity metric (BRA) on Renren boosts prediction accuracy over random prediction by more than 40,000 times (at 55M edges). Yet it only achieves 3% accuracy when predicting the next edge. To understand the key reasons behind such low accuracy, we investigate both structural and temporal aspects of each metric-based algorithm, with the exceptions of PA and SP, the worst performing metrics which we discussed in §4.2. We later take our findings into account when designing complementary prediction mechanisms in §6. Our analysis shows consistency over time and across networks. For brevity we focus our discussion on a sample of results (Renren, 55M edges).

Structural factors. We notice that all these similarity metrics are strongly biased by node degree. Figure 7 plots the degree distribution of nodes associated with the predicted edges (by each metric) and the ground truth distribution. We see that PPR and JC are heavily biased towards low-degree nodes, while the rest focus more on high-degree nodes. Such bias often comes from the construction of the similarity metric. Take for example BCN (and CN). In a small-world network, two nodes with high degree likely share more common neighbors, and are more likely to be chosen by the common neighbor algorithm.

We also observe that for metrics biased towards high-degree nodes, their results are dominated by a small number of nodes. To illustrate this, we find the 0.1% nodes most frequently predicted to create a new edge, and show their ratio of predicted and real edges in Table 5. We see that except for BRA, all other similarity metrics overpredict the involvement of a small group of nodes in edge creation. It makes sense that the worst offender, Rescal, is much better suited for a supernode-driven network like YouTube. There, its frequent link predictions around supernodes matches the network structure and produces much more accurate results.

Temporal factors. Our analysis also shows that these metrics tend to predict links between less active nodes. In particular, for each snapshot G_t, we measure the *idle time* for a node v in G_t as the time gap between t and the most recent time when v creates an edge. Figure 8 shows that the idle time of nodes in predicted edges by all metrics are larger than that of ground truth, meaning that they are all biased to nodes that are dormant recently, which are less likely to form new edges.

5. CLASSIFICATION-BASED PREDICTION

Classification-based algorithms apply supervised learning to predict links using multiple similarity metrics as features. The key challenge is how to scale to large OSNs, *i.e.* being able to predict edges among all possible node pairs. Prior works limit the prediction coverage to a very small subset of node pairs [36, 38]. Another challenge is that social networks are highly sparse, translating into highly "imbalanced" positive (connected) and negative (disconnected) subsets. Prior work cites data imbalance as a major cause of low prediction accuracy [15]. In our 55M-edge Renren snapshot for example, the ratio of positive to negative links is $1 : 179K$, and decreases further as the network grows.

In this section, we evaluate classification-based link prediction in practical scenarios, using our large OSN datasets with high data imbalance. To do so, we develop a scalable measurement mechanism for implementing and evaluating classification-based algorithms. We also study how they perform on imbalanced data and compare their results to metric-based prediction algorithms.

5.1 Evaluation Configuration

Classification-based algorithms first train models (classifiers) using labeled data and their corresponding features, then apply the trained classifiers to test data to predict their labels. Link prediction only requires binary classification ("+" for creating an edge and "-" for no edge). The key challenge in evaluating these algorithms is how to train and make prediction on all possible node pairs – this requires computing all the features for $O(|V^2| - |E|)$ node pairs and making a classification decision ($|V|$ and $|E|$ the graph node and edge count). Even for a "small" Renren snapshot (2.3M nodes, 25M edges), it takes 88 days to compute features!

Snowball Sampling. To address this challenge, we consider limiting our evaluation using snowball sampling [12], which has been shown to effectively reduce computation cost while preserving network structure and statistical representativity. Specifically, for a snapshot $G_{t-2} = \{V_{t-2}, E_{t-2}\}$ we first randomly select a node v as the seed, then run a breadth-first-search from node v until a fixed percentage p of nodes are visited. These visited nodes V^S_{t-2} are the sampled nodes in snapshot G_{t-2}. We repeat the process on the next snapshot $G_{t-1} = \{V_{t-1}, E_{t-1}\}$ using the same seed v, producing V^S_{t-1}. The choice of sampling percentage p must balance between computation cost and data representativity. We configure p based on the network size. Since our Facebook network is reasonably small, p=100%. For Renren and YouTube, p=2%.

Next, we apply common classification methods on these sampled node sets. During the training process, we measure the similarity features of all node pairs among V^S_{t-2} in G_{t-2}, labeling each node pair as either positive or negative depending on whether they are connected in G_{t-1}, and training a classifier using this labeled set. In the testing process, we collect features between node pairs among V^S_{t-1} in G_{t-1}, feed them into the trained classifier to compute prediction scores, and then choose the top k node pairs with the highest scores as the new edges for the next snapshot $G_t = \{V_t, E_t\}$. As in §4, we set k to the actual number of new edges created among node pairs in V^S_{t-1} for G_t, and use the accuracy ratio to evaluate prediction accuracy. To minimize the impact of seeds, we randomly select 5 nodes as seeds, repeat classification methods on them, and measure the average and standard deviation of prediction accuracy ratios.

Given the computation complexity, we limit our evaluation to two instances (listed in Table 6) of different sizes (small and large) for all three networks. Again, because these instances produce highly consistent results and space constraints, we only discuss the results for the large networks.

Features and Classifiers. We use scores from all 14 similarity metrics listed in Table 3 as features, and experiment with 4 well-known classifiers: Support Vector Machine (SVM), Logistic Regression, Naive Bayesian (NB), and Random Forests (RF)[3]. We also considered but ultimately rejected Decision Trees, because they can only produce binary recommendations, and are effectively subsumed by Random Forests.

We ran experiments on a wide range of network snapshots, and found the classifiers were consistent in their relative performance. RF and NB always performed poorly, and LR performed generally on par with SVM. In addition, we found that SVM outperformed LR with imbalanced training sets, which has been also shown in prior work [15]. Since our data is highly imbalanced, SVM's results are uniformly the best of the bunch, and we use SVM results for the remainder of our discussion. As an example of the relative accuracy results, we plot in Figure 9 prediction accuracy ratio for all 4 classifiers on a Facebook network snapshot of 345K edges.

5.2 Link Prediction Accuracy

To evaluate classification-based prediction, we must first understand the impact of data imbalance within training sets. Recall that link formations in social networks are extremely imbalanced, *i.e.* far fewer connected node pairs than disconnected. This imbalance has been shown to contribute to classification errors [15]. For this we apply the well-known undersampling technique to build training data, keeping all positive node pairs while varying the number of negative node pairs [15]. Here positive(negative) node pairs refer to those which will(not) connect in the prediction timeframe. Figure 10 plots prediction accuracy ratio while varying the under-sampling ratio, θ=(# of positive node pairs : # of neg-

[3] We use the implementation in an open source library [34] with default parameters for all classifiers in this paper.

Graph		G_{t-2}		G_{t-1}		Snowball
		Nodes	Edges	Nodes	Edges	Sampling p
Facebook	small	49K	345K	49K	360K	100%
	large	56K	600K	57K	615K	100%
YouTube	small	1.63M	4M	1.74M	4.25M	2%
	large	2.63M	7M	2.70M	7.25M	2%
Renren	small	2.3M	25M	2.7M	30M	2%
	large	6.2M	95M	6.7M	105M	2%

Table 6: Data instances for evaluating classification algorithms.

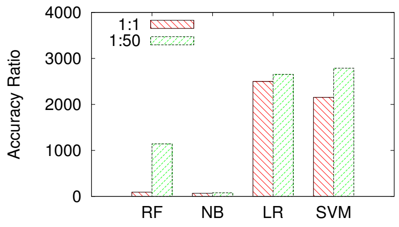


Figure 9: Accuracy ratio of four classifiers with under-sampling ratio θ 1:1 and 1:50 (Facebook, 345K edges).

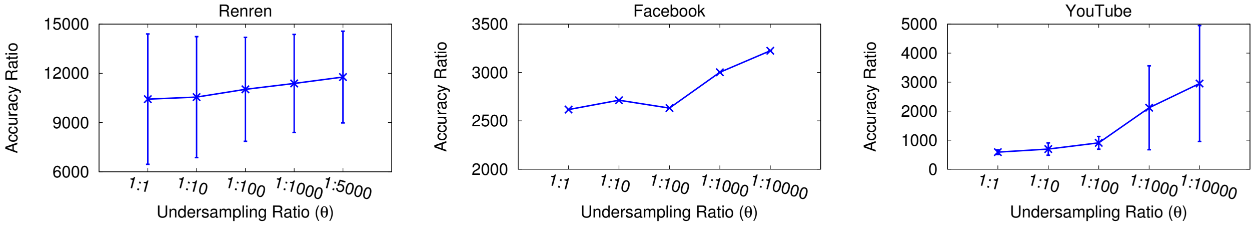


Figure 10: Performance of classification-based prediction as a function of the under-sampling ratio θ used during classifier training.

ative node pairs), from (1:1) to (1:10000)[4]. For our three OSNs, the true (unsampled) positive vs. negative ratio is around (1:100000). Note that existing classification-based prediction algorithms generally use balanced node pairs, or a ratio of (1:1).

These results show that classification-based prediction algorithms are significantly better than random prediction for all 5 sampling ratios. For Renren and Facebook, accuracy ratio improves as the sampling ratio θ approaches the actual positive vs. negative ratio (1:100000). Compared to conventional balanced sampling (1:1), a lower under-sampling ratio produces significantly more accurate results, and improvements in accuracy ratio, and also the accuracy, can be as high as a factor of 5.

The above results confirm the effectiveness of classification-based link prediction. More importantly, we show that the performance of these algorithms depends on the configuration of training data. Conventional methods of using balanced training data can lower prediction accuracy by as much as a factor of 5. To minimize such loss, we need to invest efforts on finding the right level of undersampling ratio (θ).

5.3 Comparing to Metric-based Algorithms

For a fair comparison, we run the metric-based methods again on the same sampled data (V_{t-1}^S). We plot the accuracy ratio for each algorithm (blue circle on the left) in Figure 11, and rank them in descending order from right to left. We see that the top (most accurate) similarity metrics are generally consistent on both the sampled data and the entire network (see §4) across different datasets. Also note that the test dataset V_{t-1}^S is smaller than V_{t-1} and better connected, the accuracy ratio of the metric-based algorithms is lower than results previously shown in §4 (accuracy ratio is lower because random prediction does better on this smaller dataset).

Comparing Accuracy. Figure 11 plots the accuracy ratio of SVM (red cross on the right) and metric-based methods (blue circle on the left). With a well-chosen θ, SVM consistently performs as well as, or outperforms the best metric-based algorithms. This outperformance stems from two factors: combining multiple similarity metrics to broaden coverage, and using under-sampling to address the issue of data imbalance. Overall, these results show that among existing algorithms, the SVM classifier provides consistently strong results. However, we also note that some similarity metrics, namely RA and BRA, provide consistently "good" results across all of our networks. In scenarios where the computational or training costs of SVMs were undesirable, RA and BRA provide reasonable alternatives with much lower computational complexity.

Similarity Metric Ranking vs. SVM Feature Weight. We seek to understand whether a good similarity metric in the metric-based method (identified from Figure 11) also becomes a dominant feature for the classification method. For this we use the feature coefficient provided by SVM, where a larger absolute value means the feature is more important. To make a fair comparison, we normalize the coefficients (using absolute values) within each classifier.

We take two steps to study the relationship between top similarity metrics and top SVM features. First, we directly compare the rankings of the two. For both Renren and Facebook, the rankings are very similar between the similarity

[4] We stop at (1:10000) for YouTube and Facebook and (1:5000) for Renren because this is the largest training size we can support on our memory-heavy servers (192GB RAM each).

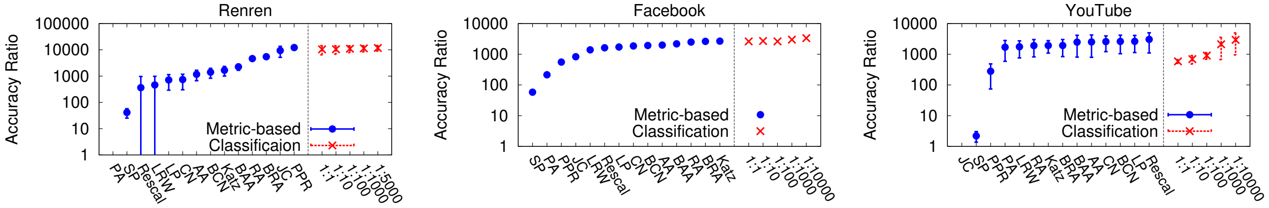


Figure 11: Comparing the prediction performance of metric- and classification-based prediction algorithms.

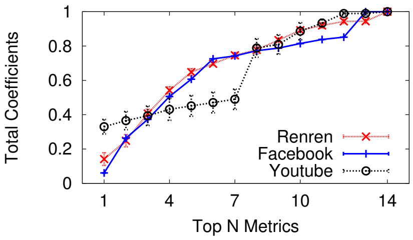


Figure 12: The relationship between top similarity metrics and top SVM features, shown as the total normalized SVM coefficient of top N similarity metrics, N=1,2,...,14.

metrics and SVM features, *i.e.* top similarity metrics are also top features in SVM. For YouTube, the orders are less consistent, except that Rescal always ranks first.

Next, we study how top similarity metrics contribute to SVM by comparing their feature coefficients. Specifically, for each graph we pick the top N similarity metrics and calculate their total normalized SVM coefficients, where $N = 1, 2, ..., 14$. Figure 12 presents the results for the large data instance listed in Table 6 with the largest θ. Results of small data instances and other values of θ are consistent and omitted for brevity.

We see that for Renren and Facebook the similarity metrics make similar contributions to the machine learning process. The top 6 similarity metrics have a slightly higher weight than the rest. For YouTube, the top first similarity metric (Rescal) and a lower ranked metric (Katz) are the key contributors while the rest make similar contributions.

Together, these results suggest that in general the metric-based and classifier-based methods share similar preferences on similarity metrics. But the classifier-based methods can combine prediction power of multiple metrics to achieve a higher accuracy and robustness across different datasets. Finally, the difference between Renren/Facebook and YouTube aligns with our earlier observation that as a subscription network YouTube's link prediction pattern differs from those of Renren and Facebook.

6. IMPROVING LINK PREDICTION

While our results show that today's prediction algorithms significantly outperform random prediction, they are still limited in their prediction accuracy. A fundamental contributing factor is that current prediction algorithms take a purely static approach to network analysis, and do not take into account temporal patterns exhibited by an evolving network. While recent studies seek to extend link prediction to support dynamic networks, they either do not scale [36], or are restricted to single model or metric [40] where performance vary significantly across datasets.

In this section, we improve existing link prediction algorithms by integrating them with dynamic network analysis. Specifically, we identify key patterns on network dynamics, and use them to build *temporal filters* that drastically reduce the search space for link prediction. Our proposed filters effectively augment existing link prediction algorithms, providing a significant boost in prediction accuracy. This is even true for algorithms that were already designed to capture network dynamics, *e.g.* [10].

6.1 Temporal Properties on Edge Creation

Using our dynamic OSN datasets, we investigate how different properties of network dynamics affect edge creation. These include node activeness, neighborhood structure evolution, neighborhood activeness, and arrival of common neighbor. We conclude that *node activity* and *arrival of common neighbor* are the key factors for all three networks, and thus we omit analysis for other explorations here. We have consistent observations across different snapshots and over different networks, and due to space limitation we only show figures for Renren snapshot at 55M edges in this subsection. We will briefly summarize our observations for other networks in the next subsection.

Node Activeness. Intuitively, a node that has recently actively created edges is more likely to create edges in the near future. We validate this by measuring node activity on both positive and negative node pairs (*i.e.* those with edges and those without). For each node pair, we mark the node with longer idle time (defined in §4.4) as the inactive node and the other as the active node. We measure activity by the idle time of the active node, the idle time of the inactive node, and the number of edges created by the active node in the past d days.

We found that for positive node pairs, *i.e.* those who will connect in the prediction timeframe, the idle times of both active and inactive nodes are significantly smaller. Figure 13 plots the CDF of the active node's idle time for the Renren snapshot at 55M edges. More than 90% of positive node pairs have <3 days idle time while only 40% of negative pairs do so. This 3-day threshold can effectively distinguish positive and negative node pairs. Similar patterns can be

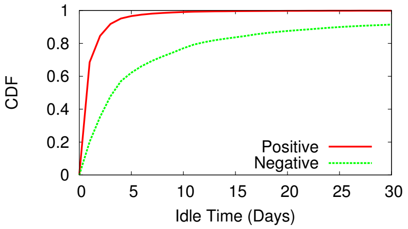


Figure 13: CDF of active node idle time in a Renren snapshot.

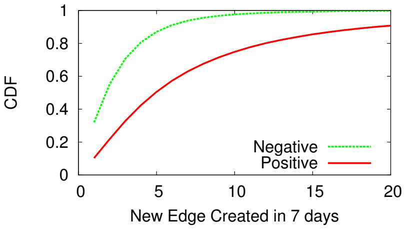


Figure 14: CDF of new edges created in the past 7 days by a node in a Renren snapshot.

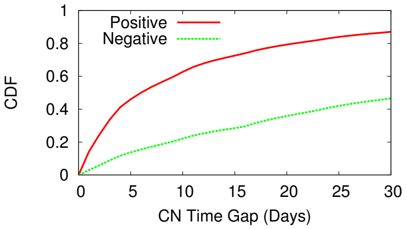


Figure 15: CDF of CN time gap of positive and negative node pairs in a Renren snapshot.

found when comparing the inactive nodes' idle times, with a 20-day idle threshold.

Furthermore, active nodes in positive node pairs tend to create more edges in a recent time. Using the same Renren snapshot, Figure 14 shows the CDF of new edges in the past week for both positive and negative sets. For more than 60% of positive node pairs, the active node creates more than 3 edges while only 20% of negative node pairs do so. This "3-edge in past 7 days" can also be used to help identify potential new links.

Arrival of Common Neighbor. We show in §4 that most similarity metrics focus on predicting edge formation between 2-hop neighbors. For these, the recent arrivals of common neighbors can often trigger the completion of a triad [46] and thus be critical in predicting edges. We test this hypothesis by measuring, for each node pair, the gap between the most recent time when they connect to a common neighbor and the current snapshot time, referred to as the *CN time gap*. Our results show that the CN time gap of positive set is much smaller than that of negative set. Figure 15 shows the result for the same Renren snapshot, where more than 60% of positive pairs create their last common neighbors in the last 10 days, while 20% of negative pairs do so.

6.2 Temporal Filtering

We propose to use these observations, which are consistent across networks, to develop "temporal filters" to drastically reduce the search space of new links by filtering out node pairs that are unlikely to create edges. Specifically, we remove any potential node pair from the candidate list if it fails to meet any of the following four criteria:

- Idle time of active nodes $< d_{act}$ days.
- Idle time of inactive nodes $< d_{inact}$ days.
- d-day new edges $\geq E_{new}$.
- CN time gap $< d_{CN}$. [5]

Our threshold values are listed in Table 7 , which hold across different snapshots for each corresponding network. While each parameter is network specific, the methodology to discover them is general.

Graph	Node Idle Time		d-day New Edges		d_{CN}
	d_{act}	d_{inact}	d	E_{new}	
Facebook	15	40	21	2	40
YouTube	3	30	7	3	20
Renren	3	20	7	3	10

Table 7: Parameters of the temporal filters.

Prediction Accuracy after Filtering. We now present the improvement in link prediction accuracy (in terms of accuracy ratio) after adding temporal filtering. We experiment with the same data instances used to evaluate classification-based algorithms (see Table 6) and present the result for the large instance. Results from the smaller instance show even more significant benefits and are omitted for brevity.

Table 8 lists the normalized improvement from applying the filter, *i.e.* the accuracy ratio of prediction with filtering divided by the accuracy ratio of prediction without filters. The improvement is quite significant for many cases, and somewhat incremental for others. For classification-based algorithms, our filtering raises the accuracy by 10%∼120%. For metric-based algorithms, the gain can be as much as a factor of 15.7.

We observe that filtering affects certain algorithms more than others. For metric-based algorithms, applying temporal filters changes the "best" prediction algorithm. For example in Facebook, JC was the weakest metric before the filters, but becomes the best metric after filtering. This is because temporal filters effectively identify and remove the unlikely-to-connect node pairs, *i.e.* inactive, low-degree nodes that JC is unable to identify.

6.3 Comparing to Other Temporal Methods

Recent works have exploited temporal information to improve prediction accuracy [8, 10, 40]. We compare our filtering design with the time series based prediction [10], a popular method that can also scale to our network datasets. For each potential node pair, this method computes its similarity metrics at multiple past time points, and aggregates these scores to produce a final score of the pair. We implement two aggregation approaches, Moving Average (MA) and Linear Regression (LR), shown by [10] as the two best approaches, and perform aggregation on equally spaced past time points (the space equals to the number of days between

[5] For node pairs beyond 2 hops, we do not apply this criterion.

Network	JC	BCN	BAA	BRA	LP	LRW	PPR	SP	Katz	Rescal	PA	1:1	1:10	1:100	1:1000	1:10000
Renren	2.2	5.8	4.1	2.3	9.7	3.2	1.7	**14.9**	1.5	2.4	-	1.9	1.9	1.8	1.8	1.8*
Facebook	**5.7**	1.2	1.2	1.4	1.3	1.3	5.3	4.4	1.3	1.2	2.1	1.3	1.4	1.5	1.3	1.2
YouTube	-	1.2	1.2	1.2	1.2	1.1	3.1	**15.7**	1.5	1.1	1.2	2.2	2.2	2	1.2	1.1

Table 8: Ratio of accuracy values after filtering vs. before filtering for all metric-based and classification methods. Bold value in each row is the maximum improvement for that network; "-" means the accuracy before filtering is "0". *Ratio in Renren is 1:5000.

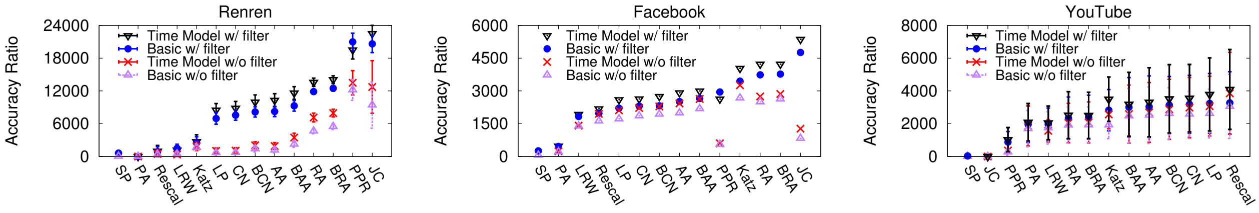


Figure 16: Our proposed temporal filtering method outperforms time-based models.

G_t and G_{t-1}). We observe that MA consistently outperforms LR, and thus omit the LR result for brevity.

Similar to Section 6.2, we also present the result for the large instance for each network in Table 6 and ignore the similar results from the smaller instance. Figure 16 shows prediction accuracy ratio for original similarity metrics (marked as Basic) and those enhanced with MA (marked as Time Model), both with and without our filtering method. For each metric, our filtering consistently improves the accuracy far more than the time series based prediction, especially for Renren and Facebook. Furthermore, even after applying the time series based prediction, our filter can still consistently improve prediction accuracy.

Summary. We show that by leveraging temporal information on network dynamics, we can effectively improve link prediction accuracy. Using these temporal filters, we can prune the set of candidate node pairs for edge creation, allowing link prediction algorithms to focus on regions of likely growth. By comparing to other temporal methods, we further confirm the effectiveness and generality of our filtering method.

7. RELATED WORK

We perform an in-depth study on two types of link prediction algorithms (metric- and classification-based). Prior works have evaluated metric-based algorithms using co-author networks [23], classification-based methods using balanced data [3, 11], and both methods using a small subset of Twitter (155K nodes) [7]. Our work differs by studying both methods using datasets of large, dynamic online social networks that recently became available. By discretizing these datasets into numerous temporal snapshots, we study the evolution of link prediction over fine time intervals, identify potential factors behind prediction errors, and propose filters that improve prediction power for all algorithms.

Recent studies have leveraged temporal information for link prediction. The key approach is to extend existing algorithms in the temporal domain, *e.g.* adding a temporal dimension [1], assigning more weights to new links [37, 40], integrating graph structure information over time [10]. Another approach applies past observations for prediction [8, 36], by identifying subgraphs that are similar to the target subgraph and use their time-evolving behaviors to help predict the target. Unfortunately, each of these methods suffers from at least one of the following limitations: i) the method is of high complexity and cannot scale to large networks; ii) the method is limited to a single model/metric, whose performance varies significantly across networks; iii) the method does not capture (and leverage) temporal patterns of the network. In contrast, our approach not only provides a general and scalable link prediction solution that supports a wide variety of similarity metrics, classifiers and network graphs, but also utilizes insights of network evolution to boost prediction accuracy and reduce complexity.

Finally, our work targets link prediction that only require graph topology information, *i.e.* nodes and edges. Additional information, such as edge weights [27], node connections on other social networks [31], and link direction [43], can improve prediction performance. We plan to consider these factors in future work.

8. CONCLUSIONS AND DISCUSSION

Using real traces of large dynamic networks, our work takes a concrete step towards objectively quantifying the predictive power of today's link prediction algorithms. By implementing a wide range of algorithms, we have already identified concrete challenges and issues with multiple algorithms, from high computational complexity that limits scalability, to binary classification results that lack granularity.

For metric-based approaches, we have shown the futility of some metrics (shortest path) and validated the scalability of others, *e.g.* scalable Katz heuristics [38]. At the same time, we have shown that it is indeed possible to scale some classifiers to large, multi-million node networks, and that classifiers such as SVMs can produce consistently strong results.

More surprisingly, we find that the best metric-based predictors (vary across different networks) perform on par with the most accurate classifier (SVM in all cases), and we derive potential guidelines for choosing metrics based on network structure. We also take a deeper look at current link prediction algorithms for the source of low accuracy, in terms of both structural and temporal aspects. Furthermore, we provide "temporal filters" that can greatly improve prediction accuracy (across different methods and networks) by leveraging knowledge of prior network dynamics, even for predictors that have already integrated temporal information.

Finally, our results underscore the fact that current prediction algorithms still perform poorly at the fine granularity of individual link predictions, even with our proposed temporal filters. While this confirms link prediction is still an unsolved problem, it is important to calibrate expectations depending on specific applications. For example, while current algorithms focus primarily on predicting nearby neighbors, a significant number of new links connect "distant" nodes. Overcoming these empirical limitations requires a better understanding of underlying network structures and dynamics.

This work only scratches the surface of a much larger problem space. Using datasets from just three networks, we already observe significant variance in accuracy for single metrics across different networks. Much more experimental and analytical work is necessary before we can identify specific properties of each network that make them more or less predictable by certain metrics. Our evaluation is limited by our reliance on network structural data, whereas deployed link prediction systems are likely to combine multiple information sources [14], *e.g.*, user profiles and behavioral data, which can boost prediction accuracy empirically.

Acknowledgments

The authors wish to thank the anonymous reviewers and our shepherd Krishna Gummadi for their helpful comments. This project was supported by NSF grants CNS-1527939 and IIS-1321083. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any funding agencies.

9. REFERENCES

[1] ACAR, E., DUNLAVY, D. M., AND KOLDA, T. G. Link prediction on evolving data using matrix and tensor factorizations. In *Proc. of ICDMW* (2009).

[2] ADAMIC, L. A., AND ADAR, E. Friends and neighbors on the web. *Social networks 25*, 3 (2003), 211–230.

[3] AL HASAN, M., CHAOJI, V., SALEM, S., AND ZAKI, M. Link prediction using supervised learning. In *Proc. of SDM Workshop on Link Analysis, Counter-terrorism and Security* (2006).

[4] BACKSTROM, L., AND LESKOVEC, J. Supervised random walks: predicting and recommending links in social networks. In *Proc. of WSDM* (2011).

[5] BAR-YOSSEF, Z., AND MASHIACH, L.-T. Local approximation of pagerank and reverse pagerank. In *Proc. of CIKM* (2008), pp. 279–288.

[6] BARABÁSI, A., AND ALBERT, R. Emergence of scaling in random networks. *Science 286*, 5439 (1999), 509.

[7] BLISS, C. A., FRANK, M. R., DANFORTH, C. M., AND DODDS, P. S. An evolutionary algorithm approach to link prediction in dynamic social networks. *Journal of Computational Science 5*, 5 (2014), 750–764.

[8] BRINGMANN, B., BERLINGERIO, M., BONCHI, F., AND GIONIS, A. Learning and predicting the evolution of social networks. *IEEE Intelligent Systems 25*, 4 (2010), 26–35.

[9] CLAUSET, A., MOORE, C., AND NEWMAN, M. Hierarchical structure and the prediction of missing links in networks. *Nature 453*, 7191 (2008), 98–101.

[10] DA SILVA SOARES, P. R., AND BASTOS CAVALCANTE PRUDENCIO, R. Time series based link prediction. In *Proc. of WCCI* (2012).

[11] FIRE, M., TENENBOIM, L., LESSER, O., ET AL. Link prediction in social networks using computationally efficient topological features. In *SocialCom* (2011).

[12] GOODMAN, L. A. Snowball sampling. *The annals of mathematical statistics* (1961), 148–170.

[13] GUIMERÀ, R., AND SALES-PARDO, M. Missing and spurious interactions and the reconstruction of complex networks. *Proc. of the National Academy of Sciences 106*, 52 (2009), 22073–22078.

[14] GUPTA, P., GOEL, A., LIN, J., ET AL. Wtf: The who to follow service at twitter. In *Proc. of WWW* (2013).

[15] HE, H., AND GARCIA, E. Learning from imbalanced data. *TKDE 21*, 9 (2009), 1263–1284.

[16] HILL, K. Facebook recommended that this psychiatrist's patients friend each other. Fusion.net, August 2016. http://fusion.net/story/339018/facebook-psychiatrist-privacy-problems/.

[17] KASHIMA, H., AND ABE, N. A parameterized probabilistic model of network evolution for supervised link prediction. In *ICDM* (2006).

[18] KATZ, L. A new status index derived from sociometric analysis. *Psychometrika 18*, 1 (1953), 39–43.

[19] KAUTZ, H., SELMAN, B., AND SHAH, M. Referral web: combining social networks and collaborative filtering. *Communications of the ACM 40*, 3 (1997), 63–65.

[20] KLEINBERG, J. M. Navigation in a small world. *Nature 406*, 6798 (2000), 845–845.

[21] KREBS, V. E. Mapping networks of terrorist cells. *Connections 24*, 3 (2002), 43–52.

[22] LESKOVEC, J., KLEINBERG, J., AND FALOUTSOS, C. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proc. of KDD* (2005).

[23] LIBEN-NOWELL, D., AND KLEINBERG, J. The link-prediction problem for social networks. *Journal of the American society for information science and technology 58*, 7 (2007), 1019–1031.

[24] LICHTENWALTER, R. N., LUSSIER, J. T., AND CHAWLA, N. V. New perspectives and methods in link prediction. In *Proc. of KDD* (2010).

[25] LIU, W., AND LÜ, L. Link prediction based on local random walk. *Europhysics Letters 89*, 5 (2010), 58007.

[26] LIU, Z., ZHANG, Q.-M., LÜ, L., AND ZHOU, T. Link prediction in complex networks: A local naïve bayes model. *Europhysics Letters 96*, 4 (2011), 48007.

[27] LÜ, L., AND ZHOU, T. Link prediction in weighted networks: The role of weak ties. *Europhysics Letters 89*, 1 (2010), 18001.

[28] LÜ, L., AND ZHOU, T. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications 390*, 6 (2011), 1150–1170.

[29] MENON, A. K., AND ELKAN, C. Link prediction via matrix factorization. In *Proc. of ECML PKDD* (2011).

[30] MISLOVE, A. *Online Social Networks: Measurement, Analysis, and Applications to Distributed Information Systems*. PhD thesis, Rice University, Department of Computer Science, May 2009.

[31] NARAYANAN, A., SHI, E., AND RUBINSTEIN, B. I. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *Proc. of IJCNN* (2011).

[32] NEWMAN, M. E. Clustering and preferential attachment in growing networks. *Physical Review E 64*, 2 (2001), 025102.

[33] NICKEL, M., TRESP, V., AND KRIEGEL, H.-P. A three-way model for collective learning on multi-relational data. In *Proc. of ICML* (2011).

[34] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., ET AL. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[35] RAYMOND, R., AND KASHIMA, H. Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs. In *Proc. of ECML PKDD* (2010).

[36] SARKAR, P., CHAKRABARTI, D., AND JORDAN, M. Nonparametric link prediction in dynamic networks. *Proc. of ICML* (2012).

[37] SHARAN, U., AND NEVILLE, J. Exploiting time-varying relationships in statistical relational models. In *Proc. of WebKDD Workshop on Web mining and social network analysis* (2007).

[38] SONG, H., CHO, T., DAVE, V., ZHANG, Y., AND QIU, L. Scalable proximity estimation and link prediction in online social networks. In *Proc. of IMC* (2009).

[39] TASKAR, B., WONG, M.-F., ABBEEL, P., AND KOLLER, D. Link prediction in relational data. In *Proc. of NIPS* (2003).

[40] TYLENDA, T., ANGELOVA, R., AND BEDATHUR, S. Towards time-aware link prediction in evolving social networks. In *Proc. of SNA-KDD* (2009).

[41] VISWANATH, B., MISLOVE, A., CHA, M., AND GUMMADI, K. On the evolution of user interaction in facebook. In *Proc. of WOSN* (2009).

[42] WANG, C., SATULURI, V., AND PARTHASARATHY, S. Local probabilistic models for link prediction. In *Proc. of ICDM* (2007).

[43] YIN, D., HONG, L., AND DAVISON, B. D. Structural link analysis and prediction in microblogs. In *Proc. of CIKM* (2011).

[44] ZHAO, X., SALA, A., WILSON, C., WANG, X., GAITO, S., ZHENG, H., AND ZHAO, B. Multi-scale dynamics in a massive online social network. In *Proc. of IMC* (2012).

[45] ZHOU, T., LÜ, L., AND ZHANG, Y.-C. Predicting missing links via local information. *European Physical Journal B 71*, 4 (2009), 623–630.

[46] ZIGNANI, M., GAITO, S., ROSSI, G. P., ZHAO, X., ZHENG, H., AND ZHAO, B. Y. Link and triadic closure delay: Temporal metrics for social network dynamics. In *Proc. of ICWSM* (2014).