Users and Pins and Boards, Oh My! Temporal Link Prediction over the Pinterest Network

Amani V. Peddada
Department of Computer Science
Stanford University
amanivp@stanford.edu

Lindsey Kostas
Department of Computer Science
Stanford University
lmkostas@stanford.edu

Abstract

In this project, we are interested in performing temporal link prediction over the uniquely structured social network underlying the on-line content sharing service Pinterest. Utilizing a newly released Pinterest dataset, we aim to predict "follows" and "pinned" relationships between user-board and pin-board node pairs, respectively. To accomplish this goal, we examine the efficacy of both proximity-based and supervised learning solutions, and propose a novel approach that takes advantage of the temporal structure of the Pinterest graph while integrating standard techniques used in link prediction. Our results are notable, with our trained models over temporal features obtaining substantial performance in predicting both types of relationships in the network. Our models additionally exhibit the ability to abstract transitive edges between unrelated nodes, thus indicating an understanding of higher-order relationships in the graph. We believe our findings are significant, and we look forward to continued research in this area.

1 Introduction

Link prediction is a remarkably rich and diverse domain in network analysis. It is uniquely challenging because it necessitates that computational models obtain a deep abstraction of local and long-range nodal interactions over time. There a tremendous number of applications associated with this domain, from enhancing recommendation systems, to modeling gene interactions, to discovering latent interactions between unrelated members of communities. Additionally, the ability to predict network structure at an arbitrary point in the future is highly pertinent in other networking challenges, including, but certainly not limited to, modeling the natural evolution of networks, graph completion, and forecasting the effects of a viral event or major network cascade.

We can now broadly frame the challenge of link prediction as follows: Given a snapshot of a graph $G =< V, E >$ at time t, can we predict what new links are formed between t and t', where $t' > t$? There are several key challenges associated with this problem. A sophisticated model, for instance, must be sensitive to local interactions between individual nodes while being cognizant of the overall structure of the graph. It must also be responsive to different timescales and the rate of edge formation – a Pinterest user, for instance, may not follow a certain board in the next day, but may perhaps do so in the next week. Such a model must additionally be robust to outliers and must elegantly handle the inherent problem of class imbalance, where we often have more pairs of nodes that are disconnected rather than connected.

In this project, we present, to the best of our knowledge, the first publicly accessible approach to investigate link prediction over the graph underlying the increasingly popular Pinterest social network. In particular, we define a novel, unified approach that integrates the temporal information of the graph with link prediction techniques specifically suited for the Pinterest data. Our challenge however differs from the standard link prediction problem in a number of ways. First, our network is not static; we are interested in how the network changes over time, and how the addition/removal of edges in the network affects the probability that two nodes will be linked. Second, we have many layers of interactions and nodes in our graph – we have separate classes of nodes for boards, pins and users, as well as separate edges for the relationships between these three entities. Finally, there is a unique structure to our graph, where, for instance, all edges have exactly one board as an endpoint, and there are never edges between nodes of the same type. Coming up with an algorithm that is sensitive to all of these defining features is thus key.

2 Previous Work

Though, as mentioned, there are no published examples of link prediction over the Pinterest social network, a significant amount of prior work has been published across the domain of link prediction as a whole. In general, there are two main classes of link

prediction techniques: *proximity-based methods* and *learning-based methods* (Li et al., 2015). A selection of pertinent works from these areas is detailed below.

2.1 Proximity-Based Methods

The main motivation behind proximity-based approaches is the idea that the "closer" two unlinked nodes are to each other, the more likely these two nodes will eventually be linked in the future. These methods generally rely on the topological structure or associated meta-data of the graph to assign scores to candidate pairs of nodes.

A key work in this area is Liben-Nowell and Kleinberg's seminal paper, which examines the efficacy of various proximity based measures over co-authorship networks extracted from `arXiv` (Liben-Nowell and Kleinberg, 2007). To approach their problem, they experiment with three types of scoring functions: neighbor-based metrics, such as number of common neighbors and the Adamic-Adar score; path-based metrics, such as the *Katz* function and the length of the shortest path; and matrix-approximation metrics, which normally make use of low-rank versions of the adjacency matrix as a form of noise reduction. Using these methods, they obtain results that provide significant improvement over a random baseline, though in absolute terms the overall performance is still coarse. Some limitations to this paper are that each scoring function is tested in isolation from the others, and the authors do not take advantage of prior knowledge of the particular structure of the co-authorship graph, which we imagine would be useful information.

Unlike the previous approach, Soares et al. propose a proximity-based method that relies on the *temporal information* of the network to calculate a similarity score between pairs of nodes (Soares and Prudêncio, 2013). They define a temporal event as follows: given two consecutive frames of a graph G_t and G_{t+1}, a temporal event is an event occurring between the two frames that affects the connectedness of any pair of nodes in the graph. They then compute proximity scores as a summation over all temporal occurrences seen during the transitions between frames in a given history. The results of applying this method to the `arXiv` dataset showed that the temporal-based model outperformed comparative static-analysis link prediction systems, including the common-neighbor, Adamic-Adar, and Jaccard Coefficient estimates. They also demonstrated the intuitive property that more recent events have a larger influence on node linkages than older ones. There are several logical extensions to this method, such as using topological and temporal proximity features together, as well as learning the weights for these scores automatically instead of choosing score thresholds manually.

2.2 Learning-Based Methods

Unlike proximity-based methods, which use handcrafted scoring functions to determine the relatedness of two nodes, learning-based approaches automatically learn the appropriate weights to assign to a set of input features.

In Link Prediction using Supervised Learning (2006), Hasan et al. formulate the link-prediction problem as that of binary classification, where, given a pair of nodes u, v, the goal is to determine whether this instance belong to a positive or negative class, where "positive" implies there is an edge between the vertices, and "negative" implies otherwise. They train a variety of classifiers, including SVMs, Multilayer Perceptrons, Decision Trees, K-Nearest Neighbors, and Naive Bayes models over the co-authorship network, using features such as keyword overlap, sum of papers published, clustering coefficient, and shortest path distance. Their results demonstrate that supervised approaches generally outperform proximity measures alone, with their fine-tuned SVM with a Gaussian kernel obtaining the highest performance. The authors also note that bagging (ensembling classifiers) does greatly enhance the predictive power of the models – the conclusion is that the feature selection is the bottleneck in accuracy rather than the classifiers themselves (Al Hasan et al., 2006).

Bliss et al. provide an alternative to the standard learning-based methods described above by proposing an *evolutionary algorithm* (Bliss et al., 2014). They extract a set of 16 features from the Twitter social network, and use the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) as an optimization strategy to learn a linear predictor. This method is essentially a genetic algorithm that initializes weights to random values, adds in mutations to get new candidate weights, and filters out candidates based on a fitness function. Applying this procedure iteratively, the authors find that an evolved predictor outperforms all other combined and individual features on the training data, as well as other machine-learning methods such as binary decision trees. Of course, one major limitation is the assumption of linearity and susceptibility to local optima – this assumption may not hold over the Pinterest dataset, and it might be necessary to use either neural nets or non-linear classifiers to obtain reasonable performance.

In the final paper that we examine, Davis et al. are concerned with performing link-prediction across

heterogeneous networks (Davis et al., 2011), where the input graph has different types of nodes and edges. Davis et al. approach this problem by propounding a multi-relational link approach (MRLP), which essentially counts occurrences of different types of triads, and probabilistically estimates how likely a new edge between candidate pair u, v will complete a triad of a certain type. These probabilities form a score that can be used in unsupervised contexts or as features in a supervised algorithm. They evaluate their method on three datasets: a YouTube network consisting of users with five possible types of relationships to other users; a Disease-Gene network, with two node types and four types of interactions between them; and a climate network with seven distinct edge types. Although they do demonstrate improvements with their method for both the supervised and unsupervised experiments, these improvements are only marginal. We nevertheless believe that this method could be used as an inspiration for our approach towards link prediction over Pinterest given the network's heterogeneous nature. In particular, this paper informs us that we must derive features and models that are especially suited to the structure of our network.

3 Technical Approach

In this section, we now describe the various methodologies that we apply to our task. In particular, we first highlight the baseline proximity measures used to perform basic link prediction, and then discuss our more advanced method that utilizes temporal information. Section 4 describes our experiments and evaluations, and Section 5 provides a more qualitative analysis of our results.

3.1 Dataset

For our dataset, we make use of a subset of the newly released Pinterest network kindly provided by the CS224W course staff. The dataset contains all food-related boards and pins created between February 2010 to December 2014 (thus spanning nearly five years). The schema of the dataset is as follows:

Pinner: UserID. Count: 18.0M

Food Boards: BoardID, BoardName, Description, UserID, CreatedTime. Count: 12.5M

Pins: CreatedTime, BoardID, PinID. Count: 736.0M

Follows: BoardID, UserID, CreatedTime. Count: 48.3M

As we can see, there are three distinct types of links that we can infer from this data. The first is the (User - Board) relationship, in which an edge extends from a user to a board if the user follows that board. The second is the (Board - User) link, which occurs when a board has been created by a user. Finally, there are (Pin - Board) edges, which describe if a pin has been attached to a particular board (note that a single pin can be attached to multiple boards). Since a board can only have one 'create' edge extending from it – which is formed upon the insertion of the board into the graph – we only concern ourselves with predicting 'follows' and 'pinned' links in the graph.

To understand the structure of the data, we have produced plots of the degree distribution of boards for the two relevant edge types:

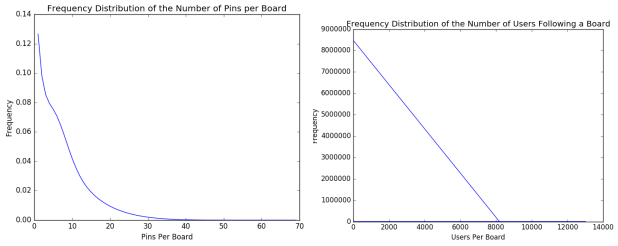


Figure 1: Distribution of pins per board (left) and users following a board (right).

As we can see, there are approximately 15 pins per board, and sometimes several thousands of users following a single board. To gain a better insight into the temporal structure of our data, we plot the change in the amounts of user activity and new board "follows" on a quarterly basis:

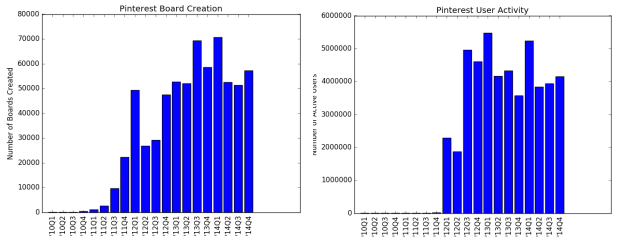


Figure 2: Number of new boards and new user activity over time.

As we can see, beginning in the first half of 2013, Pinterest begins to grow very active, which makes it a useful time period for analysis.

3.2 Constructing the Graph and Data

One of the key components of our project is finding a sensible and appropriate representation of the Pinterest network for our models. We ultimately decide

to build an undirected and unweighted graph, since this will allow us to run a variety of standard link prediction models and compare their results more effectively. In section 3.5, we explore an alternative representation that yields some interesting insights.

Given this undirected network representation G, we now would like to define a training and testing set over which to evaluate our models. Let us first define time intervals t_0, t_1, t_2, t_3, such that $t_0 < t_1 < t_2 < t_3$. Let us then define the graph $G_{\text{train}} =< V, E >$ as the set of nodes and edges created during the training interval $T_{\text{train}} = [t_0, t_1]$. Let us also define G_{test} over time period $T_{\text{test}} = [t_2, t_3]$ as a set of nodes $V' = V$ and a set of edges E' formed over T_{test}, such that $\forall e \in E'$, $e \notin E$. Our goal is to predict members of E' given G_{train}. An important note to make here is that we are ignoring the arrival/removal of any nodes between the training and testing graphs – as in (Liben-Nowell and Kleinberg, 2007), we thus only predict links that form between existing nodes.

Given G_{train} and G_{test}, we then sample pairs of nodes from G_{train} and G_{test} that serve as training and testing examples, respectively. To construct a training set $X_{\text{train}} = [(u_1, v_1), \ldots, (u_n, v_n)]$ of n examples, we sample $n/2$ positive and $n/2$ negative examples from G_{train}, where a pair is given a positive label if an edge exists between its constituent nodes, and a negative label otherwise. To construct a testing set X_{test} of m examples, we sample $m/2$ positive examples, where $(u, v) \in E'$ and $(u, v) \notin X_{\text{train}}$. We also sample $m/2$ negative examples, where each (u, v) that is chosen satisfies $(u, v) \notin E'$, $(u, v) \notin E$, and $(u, v) \notin X_{\text{train}}$.

3.3 Extracting Proximity Features

Once we have constructed a list of training and testing examples for our models, the next step is to extract a set of scores or features. That is, for every example node pair u, v, we compute a scoring function $f(u, v)$. We can use these feature functions individually as independent scoring functions, or we can learn weights over their values that will allow more nuanced predictions. In this section, we thus define several of these basic feature/scoring functions we use in our experiments.

- **Number of Common "Neighbors"** This feature as originally formulated is not well-defined over the bipartite-like structure of the Pinterest network. That is, users and boards, and pins and boards, can never share common neighbors. To rectify this, we can define an approximate neighborhood for node u as all vertices that are of length 2 away from u – then we can apply the metric as usual by computing the intersection of this set with direct neighbors of v.

- **Preferential Attachment** The product of the number of immediate neighbors of u and v. The intuition behind this feature is that the higher the degree of the two nodes, the more generally likely it is that they will be connected in the future.

- **Shortest Distance** The shortest possible distance between u and v on the graph.

- **Number of paths of length 3** Length 3 is a special value since it is the minimum number of links that a user or pin must traverse to reach an unconnected board. We thus expect that the more number of short paths that exist between a pair of candidate nodes, the more likely it is that they will eventually be connected.

- **Jaccard Coefficient** Measures the similarity of two nodes u and v by the proportion of the neighbors of u who have interacted with the same node v (and vice versa).

- **Adamic-Adar measure** Formalizes the notion that nodes which share a common neighbor of low degree are more similar than nodes that share a common neighbor of high degree. Calculated by summing the inverse of the degree of the common neighbors z of nodes u and v (where we use the definition of "common neighbor" earlier).

- **Mean of Neighbor Degrees** Finds the average degree of u's neighbors, the average degree of v's neighbors, and sums them.

- **Standard Deviation of Neighbor Degrees** Finds the standard deviation of degrees of u's neighbors, and the standard deviation of degrees of v's neighbors, and sums them.

- **Page Rank Sum** Measures the probability that two nodes u and v will be reached on a random walk through the network.

- **Eigenvector Centrality Sum** Characterizes the global prominence of a node by ranking a node by recursively computing the centralities of its neighbors (Bonacich, 2007). For a node pair u, v, we use the sum of their centralities as a score.

Note that for all features that we utilize in our models, we modify them to have unit mean and a standard deviation of one, as is standard in many machine learning problems.

3.4 Dynamic Temporal Feature Extraction

While we expect our basic proximity feature functions to perform fairly well as a baseline, they ignore the inherent temporal structure associated with the Pinterest network, which we imagine would be strongly correlated with process of link formation. One of our key contributions in this paper is thus defining a unique method of appropriately integrating this temporal information into our models.

Inspired by the work of (Soares and Prudêncio, 2013), we pursue a method that calculates the *evolution* of the topological structure of the graph over time. Intuitively, this process computes the rates of change of different characteristics of the Pinterest network. We expect, for instance, if the shortest distance between two nodes is rapidly decreasing, then eventually an edge is likely to form between them. We can thus use this knowledge to build more nuanced feature representations for candidate pairs of nodes.

We can now more formally summarize our approach as follows: We begin by taking our graph G_{train}, and partitioning it into n intervals of time: $[G_0, G_1], [G_1, G_2], \ldots, [G_{n-1}, G_n]$, where $G_n = G_{\text{train}}$. Each interval contains the same set of nodes, as well as all edges formed prior to and during this interval. Let us also assume that we have a set of r proximity functions $f_1, f_2, \ldots, f_r$ (such as those we defined earlier), where each f accepts a graph, a pair of nodes, and returns a corresponding score. For each interval $[G_i, G_{i+1}]$ and for each f_j, we compute the following for a candidate node pair u_k, v_k:

$$\delta_{ijk} = f_j(G_{i+1}, u_k, v_k) - f_j(G_i, u_k, v_k)$$

This value is then appended to the feature vector for u_k, v_k. We thus iterate this process for every pair of nodes, every feature function, and across the different time intervals. For training, we extract these features over the first $n-1$ intervals, and then train classifiers to predict the edges formed during the last interval $[G_{n-1}, G_n]$. In order to keep dimensionality of the feature vectors constant between training and testing, this means that during testing we must sample features from intervals $[G_1, G_2]$ to $[G_{n-1}, G_n]$ of the training set. Thus, if we utilize r scoring functions, the feature vector for each node pair will have dimension $r \cdot (n-1)$. A visual summary of our approach can be found in Figure 3.

We note that our approach is fundamentally different than that described in (Soares and Prudêncio, 2013). In particular, Soares et al. compute a single scalar that is the sum of edge formations and removals in a node's immediate neighborhood over time. Our approach, on the other hand, computes changes in higher-level quantities over time across the graph, and instead of simply summing over these values, we concatenate them into a feature vector so as not to lose any information in the process. Thus, we expect that our feature vectors will be more replete with valuable information for our classifiers.

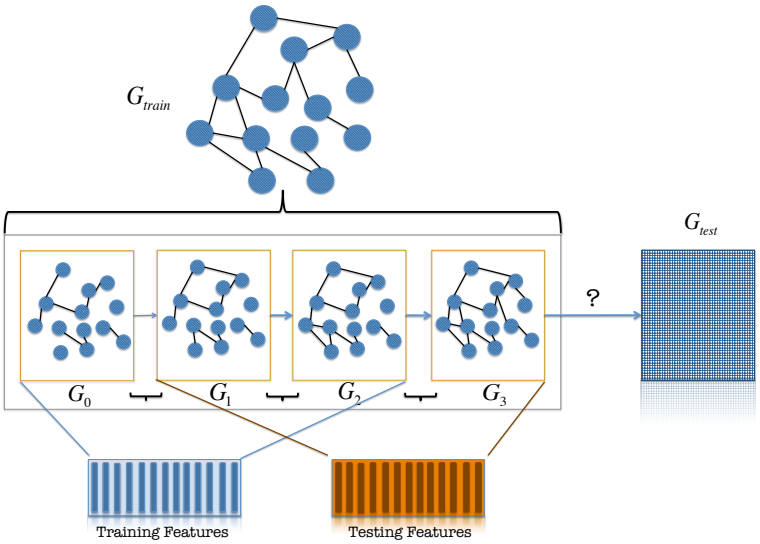


Figure 3: Dynamic Temporal Feature Extraction. Our training features are changes in topological features over the first $n-1$ time intervals of the training set, and the testing features are similarly extracted from the last $n-1$ intervals of the training set.

3.5 Models

In this section, we now describe the models that we use to actually predict link formation given our extracted features.

The most straightforward approach to integrate the information from our proximity functions is to simply apply each metric individually to every candidate pair of nodes in the test set, and produce a ranked list L_p by sorting the scores. This method takes inspiration from the work of (Liben-Nowell and Kleinberg, 2007), where after producing the ranked list, we select the first $|E'|$ candidates as our predictions. This method thus uses each feature in isolation, and, as Liben et al. mention, the absolute performance of these simple proximity based functions will be quite coarse – but they still serves as a reasonable baseline for our experiments.

A more logical approach is to concatenate all of the scoring functions from section 3.3 into a single feature vector for each example node pair, before feed-

ing these vectors into our models, which can then be trained to output positive and negative labels appropriately. Of course, we can simply input the temporal features from section 3.4 into our classifiers directly. The key models we utilize in this experiment are shown below.

- **Logistic Regression** The first supervised approach that we apply is a max-entropy classifier, where our goal is to fit a function of the form $h(x) = \frac{1}{1+e^{-\theta^T x}}$ to the data.
- **Random Forest** A random forest is a collection of decision trees where each tree is trained on a random subset of the data. As an ensemble approach, random forests are especially robust to overfitting, making them useful for our potentially noisy, real-world data (Liaw and Wiener, 2002). In this paper, we use a random forest comprised of 100 decision trees.
- **K-Nearest Neighbors** Our third model is the KNN algorithm, which examines the $k = 5$ training examples that are closest in Euclidean distance to a query point's feature vector, and outputs the majority label. KNN typically performs well with low-dimensional, hand-chosen features, so we expect that it might be well-suited to our task.
- **Artificial Neural Network (ANN)** One of our more advanced classifiers that we experiment with is a three hidden-layer artificial neural network, with 100, 50, and 50 nodes in each of the hidden layers, respectively. We use $tanh$ non-linearities between each layer, and a softmax classifier at the head of the network. Given the recent effectiveness in a large range of discriminative tasks of this model, we expect that it will likewise be useful for link prediction.
- **Covariance Matrix Adaptation Evolution Strategy (CMA-ES)** As described in section 2.2, Bliss et al. define a unique learning approach that uses a genetic algorithm to efficiently find weights over proximity features extracted from adjacency matrices. There are two useful properties of this algorithm – it is remarkably efficient in comparison to standard machine learning classifiers, and the magnitude of the weights that are learned have a direct correlation with the usefulness of the individual input features for prediction – therefore yielding important qualitative insights.

  The method works as follows: we begin by selecting a random initially weight vector w. We then repeat the following process. For N iterations:

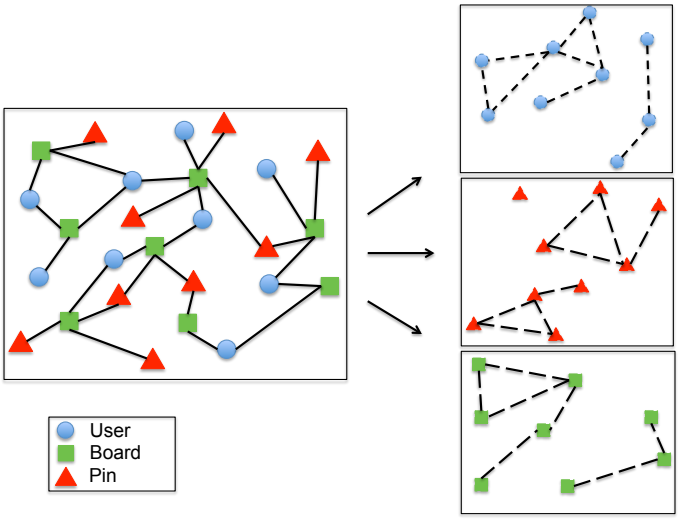


Figure 4: Coverting the Pinterest graph into a set of indirectly connected networks for transitive edge prediction.

  - Generate a set of C candidates by adding "mutations" to different elements of w. These mutations simply take the form of random, scaled Gaussian noise added to the individual elements of w.
  - For each candidate, produce a list of predictions over the training set and then compute a "fitness" score – in our case, we simply use the number of examples they classified correctly in the training set.
  - Choose the candidate with the highest fitness as our new weight vector w, and repeat.

3.6 Exploring Transitive Edge Prediction

As an interesting extension to our project, we investigate the effectiveness of our models in predicting higher-level, indirect relationships in the graph. In particular, we realize that the Pinterest graph G can in fact be broken up into three weighted "component" graphs User (U), Board (B), and Pin (P), where each graph consists of one type of node (See Figure 4).

In this formulation, we specify a user graph U, where two nodes are connected if and only if they follow at least one common board – the weight of the edge between them is the number of boards they follow in common. Likewise, two pins in graph P are connected if they are pinned to at least one common board. Lastly, two boards in B are connected if they share at least one common follower.

We see that this is an inherently interesting set of graphs, since their connectedness somehow reflects the "homogeneity" of the network – that is, the more well-connected the graph becomes, the more similar

Table 1: Proximity-based results on the testing set of the original graph

	Com. Neigh.	Pref. Attach.	Shortest Dist.	Len. 3 Paths	Jac. Coeff.	Adamic-Adar	Avg Neigh. Deg.	Std. Dev Neigh. Deg.	PRank Sum	Eigenvec. Centr. Sum
"Follows"	0.631	0.550	0.620	0.512	0.621	0.680	0.072	0.05	0.51	0.321
"Pinned"	0.712	0.691	0.721	0.522	0.670	0.651	0.153	0.101	0.612	0.511

Table 2: Supervised results for link prediction with standard features on the original graph.

	"Follows" Edge				**"Pinned" Edge**			
Metric	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**
KNN	0.7274	**0.8402**	0.5616	0.6732	0.8220	0.7997	0.8592	0.8283
Logistic Regression	0.7188	0.7052	0.7520	0.7278	0.8390	0.8478	0.8264	0.8369
Random Forest	**0.7674**	0.7669	0.7684	**0.7676**	0.4996	0.4997	**0.9992**	0.6663
3-Layer Neural Net	0.7412	0.7199	**0.7896**	0.7531	**0.8532**	0.8757	0.8232	**0.8487**
CMA-ES	0.6578	0.6802	0.5956	0.6351	0.8468	**0.9109**	0.7688	0.8338

Table 3: Supervised link prediction with **temporal features** over the testing set.

	"Follows" Edge				**"Pinned" Edge**			
Model + Temp. Feats	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**
KNN	**0.7938**	0.8044	0.7764	**0.7901**	0.7380	0.7418	0.7300	0.7358
Logistic Regression	0.7103	0.6790	0.7977	0.7336	0.8434	**0.9165**	0.7556	0.8283
Random Forest	0.5437	0.5259	**0.8863**	0.6601	**0.8564**	0.9163	0.7844	**0.8452**
3-Layer Neural Net	0.6664	0.6435	0.7460	0.6909	0.7738	0.7488	**0.8240**	0.7846
CMA-ES	0.7358	**0.9429**	0.5020	0.6551	0.7714	0.9051	0.6064	0.7262

Table 4: Proximity based methods over the indirect graphs.

	Com. Neigh	Pref. Attach	Shortest Dist	Len. 3 Paths	Jac. Coeff	Adamic-Adar	Avg Neigh. Deg.	Std. Dev Neigh. Deg	PRank Sum	Eigenvec. Centr. Sum
Users	0.571	0.534	0.421	0.534	0.422	0.591	0.311	0.432	0.522	0.201
Pins	0.542	0.597	0.399	0.522	0.523	0.622	0.412	0.417	0.533	0.521
Boards	0.512	0.571	0.310	0.412	0.481	0.562	0.373	0.390	0.452	0.461

Table 5: Link prediction over the indirect graphs with supervised methods.

	User-User				**Pin-Pin**				**Board-Board**			
Metric	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**	**F1**
KNN	0.9022	0.8380	0.9972	0.9107	0.8874	0.9287	0.8392	0.8816	**0.7678**	0.8950	0.6068	**0.7232**
Log. Regression	0.8958	0.8291	**0.9973**	0.9054	0.8868	0.9340	0.8324	0.8802	0.5120	0.5321	0.9899	0.6789
Random Forest	0.9024	0.8383	0.9972	0.9109	**0.8966**	0.9312	**0.8564**	**0.8922**	0.5301	0.5211	**0.9791**	0.6667
5-Layer Neural Net	**0.9044**	**0.8413**	0.9968	**0.9125**	0.8766	0.9250	0.8196	0.8691	0.5884	0.5519	0.9401	0.6955
CMA-ES	0.8960	0.8293	0.9972	0.9056	0.8772	**0.9686**	0.7796	0.8639	0.7524	**0.9055**	0.5636	0.6948

are the nodes involved. Predicting links at this level of abstraction thus provides a slightly more global view of network movements as a whole. Thus, as a separate experiment, we apply some of the features and models described above to each of the three graphs, which are likewise separated into training and testing graphs over the given time interval. The features we utilize in this case are the standard variations of the feature functions mentioned in section 3.3 (e.g. common neighbors refers to intersection of immediate neighbors, etc.).

4 Experiments and Results

For this project, we have implemented a complete end-to-end pipeline that reads in the raw data, constructs training and testing graphs over a given timescale, extracts features, and trains all of our classifiers over the data. We implemented Bliss's CMA-ES strategy, wrote optimized code for dynamic temporal feature extraction, and also built the three types of transitive networks P, B, U from the original graph. We implemented everything ourselves using SNAP and Python – our codebase can be found on GitHub[1].

One of the major challenges we encountered during experimentation was handling the sheer size of the dataset – parsing the four provided data files to create a single graph takes several hours, and the resulting graph is often so large that holding more than a year's worth of data causes out-of-memory errors even on Stanford clusters. Additionally, running a single experiment can take up to two days. Due to hardware limitations, we thus use the edges and nodes

[1] https://github.com/classicCoder16/CS224W_Project

formed in the first six months of 2013 as our data. We then define G_{train} as the first four months of this data, and the subsequent two months as G_{test}.

We sampled a total of 20,000 positive and negative examples from G_{train}, and tested on 10,000 examples drawn from G_{test}. For each edge type we aim to predict, we train only on instances of that edge type. For temporal feature extraction, we used $n - 1 = 5$ previous time intervals to build our feature vectors.

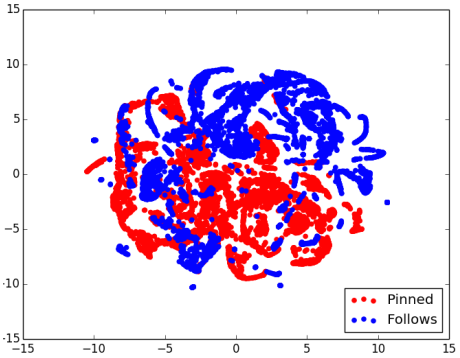
To evaluate the performance of our classifiers, we utilize accuracy, precision, recall, and F_1 metrics. To evaluate our proximity functions, we simply use F_1 score, since evaluating these methods, as noted in (Liben-Nowell and Kleinberg, 2007), requires that we know the number of positive examples in the testing set, and the other metrics are thus not informative. Our results are shown in tables 1 through 5.

5 Discussion

As we can see, all of our models perform reasonably well over the given data, with many methods surpassing the performance of a random baseline. We first observe that our basic proximity functions obtain fair performance in predicting individual link formation. Interestingly, the scoring functions that take advantage of the unique bipartite-like structure of the Pinterest graph seem to perform optimally, such as our modified Common Neighbors approach and new Adamic-Adar measure. In contrast, the more standard functions such as Average Neighbor Degree do not fare well over this network – thus, deriving feature functions that are sensitive to the anatomy of the Pinterest graph is important.

Lending further credence to our methods, our classification models are able to effectively combine these different feature functions to achieve considerable accuracies. In particular, the random forest and neural network seem to perform consistently well, with the neural network obtaining an impressive F_1 score of 0.8487 on "pinned" edges, and the random forest obtaining 0.7676 on "Follows" edges over standard features. However, Bliss's evolutionary algorithm, while comparable to the other models we experimented with, did not provide the increase in performance we expected. We do find, though, that the major benefits of this algorithm are more so its ease of interpretation and quick training times rather than sheer predictive power.

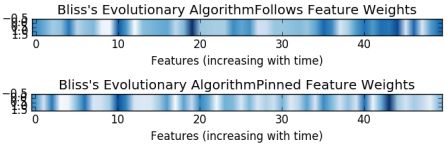
One thing we do notice from our results is that "follows" edges are seemingly much more difficult to predict than "pinned" edges, across almost all features and classifiers. This perhaps makes intuitive sense as users may be highly selective in choosing which boards they would like to follow, where as pins can rapidly be re-pinned onto different boards without as much inherent filtering. Perhaps incorporating meta-data and specific user information could improve results for these types of edges. One interesting thing we do observe though is that our features are actually seemingly expressive enough to discriminate between "follows" and "pinned" edges. To see this, we can run t-Sne (Maaten and Hinton, 2008) on the feature descriptions of pinned and follows edges in the testing set. The result is shown below, where we can see that there is somewhat of a distinction between the two types of edges:



This implies that the specific topological features of an edge varies with the specific type of the edge.

We now note from our results that, as expected, our temporal features appear to be very effective in our link prediction problem. In particular, our KNN algorithm over temporal features achieves the highest F_1 score for "follows" edges across all feature and classifiers, and the random forest over these features produces the second highest F_1 for "pinned" edges. Though the gain in performance is marginal, we believe our results are quite promising, and that our dynamic temporal feature extraction, with some parameter tuning, will be able to obtain superlative accuracies. One simple way to boost performance would be to concatenate the standard feature vectors with the temporal ones, while training on a larger number of training examples to balance the increase in dimensionality.

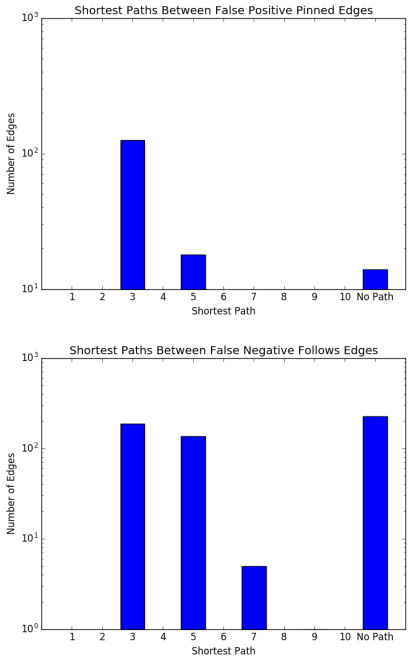
In order to examine the relationship between time and prediction accuracy, one interesting visualization we can produce is a heat-map of the weights learned by Bliss's classifier over the temporal features (darker values are higher magnitudes):



Intuitively, we see that the weights learned by Bliss's algorithm show that more recent features for

"follows" edges have a slightly higher weight than those in older time periods – thus, the recent activity of a user or board determines whether a "follows" relationship will be formed, and if we trained on a larger timescale, we would most likely notice an even sharper gradient of weight magnitudes. However, we also notice that the weight is more uniformly spread out for "pinned" edges, suggesting that these edges are not sensitive to historical changes in graph structure, and that re-pinnings are perhaps more random or spontaneous formations.

One qualitative fact that is useful to know is the type of misclassifications that our models frequently make. In particular, we can look at the false positive and negative classifications to gain an insight into our models' learning and predictions. We first examine the a few misclassifications made by the neural network on non-temporal features, shown below (false postives on top, and false negatives on bottom):



Note how false positives are almost exclusively made of short-path candidate pairs, where as false negatives have a significant fraction of pairs of nodes that have no path between them. These results follow intuition in that our models essentially only connect nodes that are close together. In order to address these mis-classifications, it might be useful to pursue a more careful sampling procedure for training examples. In particular, we should extract negative examples that are graphically close to each other, and positive examples that, when disconnected, are far from each other on the graph. This sampling would thus allow our models to generalize better.

Finally, we note that our algorithms achieve considerable performance on the indirect graphs consisting of nodes of the same type, obtaining F_1 scores as high as 0.9125. This fundamentally implies that we are able to both identify relationships between each of these homogeneous entities independent of other network activity while providing interesting insight into the true structure of Pinterest. Additionally, the ability to predict associations within these latent subgraphs has powerful implications for Pinterest in how it manages the content it displays and recommends to each user. Interestingly, predicting board-board relationships appears to be a more subtle problem than prediction over user-user or pin-pin relationships – perhaps because boards can often strongly overlap in content, but users generally would like to follow boards that are similar yet sufficiently different from each other; predicting whether a pair of boards will share a common user is thus fairly subtle.

6 Conclusion

In this paper, we have presented a novel approach towards link prediction over the social network underlying Pinterest. We have investigated the effectiveness of proximity-based and supervised algorithms, and have defined scoring functions desigined to handle the unique anatomy of the Pinterest graph. Our dynamic temporal feature extraction obtains comparable performance to our classifiers over standard features on "pinned" edges, and obtains improved results on the inherently more difficult "follows" edges. Additionally, our ability to predict indirect graph connections with respectable accuracy shows that our models are capable of understanding higher-order relationships in a graph.

7 Future Work

There are several directions that we could pursue for the continuation of this project. First, we believe that there is significant potential to be had with our dynamic temporal feature extraction. Perhaps one of the most logical extensions is to use a *sliding window* approach – intuitively, our current method trains on a single set of 5 previous graph transitions to predict edges formed in the next time frame. If we could "slide" this window across different points of the Pinterest network's history – extracting training examples over this window as we go – we might obtain more generalizable results. This approach would thus be akin to a bigram/trigram extraction used frequently in text classification. We might also consider changing the window size – our qualitative results demonstrate that more recent events are more important in determining link prediction, so perhaps

a smaller window size would increase both accuracy and efficiency.

Another key insight is that the entire Pinterest network, with regards to a single candidate node pair, can simply be represented as a vector time-series. That is, we can discretize the entire network's history as a series of graph transitions, where each transition is defined by a particular feature vector. A worthwhile extension would thus be to train a Recurrent Neural Network (such as a Gated Recurrent Unit or Long Short-Term Memory network) over this information, given that RNNs have proven to be inherently suited for forecasting events based on temporal sequences. At each time step, we could simply input the feature vector of the corresponding graph interval, and train the RNN to predict node-pair connectivity.

Lastly, another significant extension would be to utilize predictions at the indirect graph level as features for link prediction on the original graph. That is, if we believe that two users will ultimately follow a common board in the future, this perhaps makes it more likely that a given candidate (user, board) will be connected in the eventual graph. If we could then incorporate this knowledge into the link prediction over the original network, we might be able to produce a more informed model.

In any case, we believe that there is much potential for further (P)interesting developments in this field, and we look forward to performing continued research in this domain.

Acknowledgements

The authors would like to thank Jure Leskovec and the rest of the CS224W staff for providing access to the Pinterest dataset, as well as for their continued guidance and advice during the course of the project.

References

Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. 2006. Link prediction using supervised learning. In *SDM06: workshop on link analysis, counter-terrorism and security*.

Catherine A Bliss, Morgan R Frank, Christopher M Danforth, and Peter Sheridan Dodds. 2014. An evolutionary algorithm approach to link prediction in dynamic social networks. *Journal of Computational Science*, 5(5):750–764.

Phillip Bonacich. 2007. Some unique properties of eigenvector centrality. *Social networks*, 29(4):555–564.

Darcy Davis, Ryan Lichtenwalter, and Nitesh V Chawla. 2011. Multi-relational link prediction in heterogeneous information networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*, pages 281–288. IEEE.

Zhepeng Li, Xiao Fang, and Olivia R Liu Sheng. 2015. A survey of link recommendation for social networks: Methods, theoretical foundations, and future research directions. *Theoretical Foundations, and Future Research Directions (October 28, 2015)*.

Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomforest. *R news*, 2(3):18–22.

David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Paulo RS Soares and Ricardo BC Prudêncio. 2013. Proximity measures for link prediction based on temporal events. *Expert Systems with Applications*, 40(16):6652–6660.