Received April 27, 2018, accepted May 19, 2018, date of publication June 11, 2018, date of current version July 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2845876

Deep Learning for Link Prediction in Dynamic Networks Using Weak Estimators

CARTER CHIU AND JUSTIN ZHAN
Department of Computer Science, University of Nevada at Las Vegas, Las Vegas, NV 89154, USA

Corresponding author: Justin Zhan (justin.zhan@unlv.edu)

This work was supported in part by the United States Department of Defense under Grant W91 1 NF-13-1-0130 and Grant 72140-NS-RIP, in part by the National Science Foundation under Grant 1625677, Grant 1560625, and Grant 1710716, and in part by the United Healthcare Foundation under Grant 1592.

ABSTRACT Link prediction is the task of evaluating the probability that an edge exists in a network, and it has useful applications in many domains. Traditional approaches rely on measuring the similarity between two nodes in a static context. Recent research has focused on extending link prediction to a dynamic setting, predicting the creation and destruction of links in networks that evolve over time. Though a difficult task, the employment of deep learning techniques has shown to make notable improvements to the accuracy of predictions. To this end, we propose the novel application of weak estimators in addition to the utilization of traditional similarity metrics to inexpensively build an effective feature vector for a deep neural network. Weak estimators have been used in a variety of machine learning algorithms to improve model accuracy, owing to their capacity to estimate the changing probabilities in dynamic systems. Experiments indicate that our approach results in increased prediction accuracy on several real-world dynamic networks.

INDEX TERMS Deep learning, link prediction, dynamic networks, weak estimators, similarity metrics.

I. INTRODUCTION

Graphs representing a network of nodes and edges can be utilized to model a wide array of real life phenomena and are one of the most valuable structures we have for understanding the world around us. The interactions and relationships between entities that govern this world contain meaningful patterns that we seek to uncover. Social networks map friendships between individuals and contain a treasure trove of information about communities and influences in our societies. Computer networks form the foundation of the Internet and deliver information across nations, containing patterns on how the digital world works as well as the signatures of insidious threats to cyber security. And in bioinformatics, protein interaction networks (PINs) contain vital information about biomolecular behavior, containing secrets about disease and potential cures. So much knowledge can lie in the nodes and edges of these networks, but the analysis of these graphs is not a simple task. Networks modeling large-scale phenomena can contain millions to billions of nodes and many more edges. Therefore efficiency is just as key of a concern as effectiveness when designing algorithms for network analysis.

Associated with these motivations is a particularly valuable challenge in network analysis called the link prediction task. If given the current state of a network, link prediction focuses on evaluating the probability of the existence of an edge in a future state. This can be understood to represent the likelihood of a future friendship in a social network, a communication in a computer network, or a protein interaction in a PIN. It goes without saying that the ability to forecast interaction events before they occur is a significant but difficult endeavor. Traditionally, this task was performed in a static context, meaning a single snapshot of a network is used to forecast future links. But link prediction is intuitively understood to be a time-dependent problem, where a network evolves over time and we seek to predict the creation and destruction of edges in a changing temporal landscape. To this end, the concept of a dynamic network is introduced, consisting of several snapshots of a network structure over time. While link prediction in a dynamic context shares the same motivations as static link prediction, it is both a more valuable and challenging exercise. A history of network evolution provides more information for which to make a prediction in theory, but it also adds an entirely new dimension to the analysis. Thus, dynamic problems inherently involve more complexity along the temporal dimension. Properties such as concept drift, where the underlying statistical properties of the network evolve over time, need to be accounted for.

There are both unsupervised and supervised techniques for link prediction. Unsupervised approaches involve the development of heuristics to assign a score for the probability of each prospective link. Frequently used are similarity metrics that gauge the strength of the relationship between two nodes. These score functions are often based on topological properties of the nodes like common neighbors and graph distances, and operate on the principle that a stronger relationship implies a stronger likelihood of interaction. Alternatively, supervised approaches attempt to treat the link prediction task as a binary classification problem, with edges and non-edges being used to train a classifier. Modern link prediction research continues to explore both approaches, and there is ongoing investigation into basic topological features that are both easy to compute and contain a surprising depth of insight into link probability. Given the complexity of networks and particularly dynamic networks, a more recent development is the application of deep learning techniques. Deep learning is used to model complex relationships in the data, exhibiting extraordinary capacity to expose previously unseen patterns.

In this paper, we propose a novel method for building a computationally efficient, network-size-independent feature vector for neural network link predictions suitable for real time applications. We accomplish this by extending similarity metrics on static networks to the dynamic plane and also applying weak estimators, which have shown marked effectiveness in improving classification accuracy in dynamic systems. The result is a deep learning link prediction framework sensitive to dynamicity that is simultaneously cheap to train and accurate.

The remaining sections of our paper are as follows. In Section 2, we perform a survey on existing research in link prediction on dynamic networks and weak estimators. Section 3 contains our proposed approach, including the construction of the feature vector, training set, and deep learning model. Then in Section 4 we apply the proposed method to the real-world data, testing on three sizable networks and analyzing the results, and finally conclude the paper in Section 5.

II. RELATED WORK

A. HEURISTICS AND SIMILARITY METRICS

In Liben-Nowell and Kleinberg's seminal paper [1], they formalize the link prediction problem and evaluate the accuracy of several heuristics for link prediction. They present the notion that a future edge between two nodes can be predicted with accuracy using the basic topological features of the nodes that indicate the ''proximity'' or ''similarity'' of the nodes to each other such as common neighbors and distance. They find that several such measures have a significant correlation with the establishment of future links, notably Adamic/Adar [2] and heuristics based on the Katz centrality [3].

Further research focuses on the development of similar heuristics, often by extending neighbor-based metrics to second or higher degree adjacency. Yao *et al.* [4] propose a modification of common neighbors to include nodes separated by two hops. In addition, they propose the use of time-decay to give greater weight to more recent snapshots. Alternatively, Kaya et. al. [5] identify progressive, protective, and regressive events as the creation, maintenance, and destruction of edges in a dynamic network respectively, and utilize such events to score potential links in a time-weighted manner. Wang *et al.* [6] approach link prediction from the node level through the conception of a node evolution diversity metric. Community detection, which involves identifying clusters of high activity, has been employed for the purposes of link prediction by Deylami and Asadpour [7]. Common link prediction similarity metrics have also been utilized and repurposed for event detection in social networks [8] and [9] and contact prediction in cognitive radio networks [10].

B. MACHINE LEARNING AND DEEP LEARNING

In contrast to the metric-based approaches, machine learning techniques have been used to improve prediction accuracy. The primary challenge in such methods involves feature representation, since clearly an entire graph cannot be used as input in any reasonable manner. For instance, Hasan *et al.* [11] employ a variety of graph features for supervised learning using several algorithms including Naive Bayes, *k*-Nearest Neighbors, and Decision trees. Likewise, Bechettara *et al.* [12] use decision trees for topological metric-based prediction in bipartite graphs. Doppa *et al.* [13] introduce a supervised link prediction approach by building a feature vector based on chance constraints for use in a k-means classifier.

But a majority of recent techniques involve deep learning, and particularly deep neural networks, for their unparalleled learning capability. A deep neural network is a class of models in machine learning involving a network of several layers of connected, output-producing nodes whose parameters are iteratively tuned to minimize the error between the final output and the true value. Li et. al [14] describe a neural network architecture called a conditional temporal Restricted Boltzmann Machine (ctRBM) which extends the structure of a typical RBM to incorporate temporal elements of a dynamic network. Meanwhile, Zhang *et al.* [15] suggest a feature representation using what they term SPEAK, or social pattern and external attribute knowledge, as an alternative to traditional topology metrics, for input in a deep neural network. Many recent contributions have focused on semi-supervised or unsupervised representation learning [16]–[18] to further enhance accuracy. The disadvantage to such approaches is that they add a new and potentially time-consuming step to the prediction process.

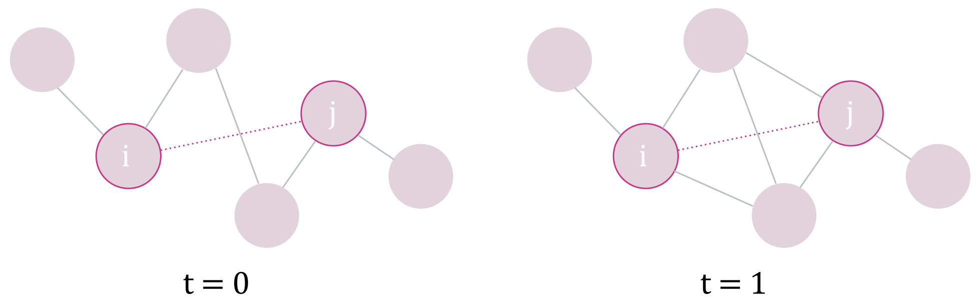

FIGURE 1. Two snapshots of a dynamic network with a link between nodes *i* and *j* considered.

C. DYNAMIC SYSTEMS AND WEAK ESTIMATORS

Dynamic systems pose unique challenges for classifiers because the underlying distributions for the target can change over time. Because this is a common characteristic of time-dependent data, a variety of techniques have been proposed to tackle this issue. One philosophy involves ignoring dynamicity at the algorithm level and employing classifiers that assume static distributions. The task then becomes recognizing when the distributions have changed, upon which the classifier must be retrained. This problem is referred to as concept drift detection, to which contributions have proposed tests [19], [20] to recognize drift with high statistical power. This approach comes with a few inherent flaws; the accuracy is necessarily "stair-step" due to intermittently-updating classifier over a continuously-transforming object of interest. Furthermore, there are challenges with identifying the data belonging to the approximate current state of the phenomenon after concept drift has been detected which must be addressed [21].

An arguably superior philosophy is to make use of algorithms that are dynamic by design. There are straightforward ways of accomplishing this, such as estimating data distributions by using a continually updating sliding window [22] or a weighted moving average [19]. These methods too have their imperfections as they rely on parameters that are inflexible to different instances of dynamic systems [23]. To overcome these deficiencies, Oommen and Rueda [24] proposed the use of weak estimators derived from the principles of stochastic learning, termed stochastic learning weak estimators (SLWEs), to detect and estimate changes in distributions. SLWEs have proven to be flexible, with work extending the application of SLWEs from binary to multiclass classification [25]. Subsequent work [26], [27] has verified that SLWEs can be applied to various types of classifiers to improve accuracy. The link prediction task for dynamic networks, which is very much so a dynamic system, stands to benefit from the application of stochastic learning weak estimators.

III. PROPOSED APPROACH

In this section, we first firmly establish the nature of the link prediction problem we wish to solve. Then, we explain the components of our deep learning approach for link prediction in dynamic networks. The first component is the format of the input feature vector. Then, we build a training set from the network using the specified feature representation, and finally construct a deep neural network framework to build a score function for link prediction on the network.

A. PROBLEM STATEMENT

We define a dynamic network as a series of snapshots in time, 1 to t, each containing a set of edges denoting the links present at that time. The link prediction problem is as follows: given snapshots $1 \ldots t$ of a dynamic network, return a score for the likelihood of edge e at time $t+1$. Figure 1 depicts a dynamic network with two snapshots. Given this information, we would seek to predict the likelihood of a link between nodes i and j at time $t = 2$.

Note that this problem formulation differs slightly from that of other papers that define link prediction as the evaluation of links that have not yet been added to a network. The alternate definition supposes that links can only be added and are persistent in later snapshots. We define the task in a more flexible manner, with both the addition and removal of edges permissible.

B. FEATURE REPRESENTATION

Our task is to build an input feature vector that is computationally inexpensive to compute and is length-independent of the complexity of the network. These are necessary requirements for an algorithm that can process large networks in real time.

1) SIMILARITY METRICS

A number of heuristics have been devised and utilized extensively for link prediction on static networks. We selected five such similarity measures which captured diverse aspects of the network topology and are efficiently computable. Each metric is a function of two nodes corresponding to the link, identified as i and j, while $N(n)$ indicates the nodes neighboring (adjacent to) n.

a: Common neighbors

The common neighbor metric is defined as $|N(i) \cap N(j)|$. Common neighbors reflect the belief that a link is more likely to be formed when the nodes share links with other nodes, analogous to the friend-of-a-friend (FOF) approach used in friend recommendation systems [28]. This metric has been applied extensively to existing link prediction research [4], [29] as it is straightforward and cheap to compute. In Figure 1, the value at time $t = 0$ is 0 since i and j share no neighbors, but at time $t = 1$ the value becomes 2.

b: Shortest path

The shortest path $d(i, j)$ represents the distance of the shortest path from i to j, reflecting the expectation that nodes with low degrees of separation are likely to interact. In a dynamic context, the shortest path can also encode the movement of two nodes toward or away from each other over time. The shortest path predictor has also found use in contemporary link prediction research [30] as a summary of network topology. The shortest path can potentially be demanding to calculate, but given the small world property of networks, we can limit the distance searched by the algorithm before stopping and presuming no path exists. The shortest path between i and j in Figure 1 decreases from 3 at time $t = 0$ to 2 at time $t = 1$, illustrating how the shortest path can detect closing distances in a dynamic network that are possibly indicative of a future interaction.

c: Adamic/adar index

The Adamic/Adar predictor is defined as follows:

$$\sum_{k \in N(i) \cap N(j)} \frac{1}{log(|N(k)|)}.$$

It captures the notion that neighbors with fewer links are more influential in facilitating future interaction. In other words, the Adamic/Adar index is a variant of the common neighbor metric that gives more weight to neighbors with lower degree. Liben-Nowell and Kleinberg [1] found Adamic/Adar to be among the most accurate predictors of the studied similarity metrics. For Figure 1, the Adamic/Adar index is 0 at time $t = 0$ due to an absence of common neighbors, but at $t = 1$ increases to $\frac{1}{log(3)} + \frac{1}{log(3)} \approx 1.62$.

d: Jaccard index

The common neighbors metric scores potential links highly if they share many neighbors, but does not account for the proportion of links shared. In this way, two nodes with many neighbors may score highly even if the relative strength of their relationship is week. The Jaccard Index, otherwise known as Intersection over Union, compensates for this by presenting common linkage as a probability. It is defined as

$$\frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}.$$

For Figure 1, no neighbors are shared at time $t = 0$, resulting in a value of 0. At $t = 1$ the Jaccard Index becomes $\frac{2}{4} = .5$.

e: Preferential attachment

Defined simply as $|N(i)| \cdot |N(j)|$, Preferential Attachment encodes the belief that nodes with many neighbors are more likely to form more in the future, and thus gives higher scores to potential links whose nodes have high degree. At $t = 0$ the score for Figure 1 is $2 \times 2 = 4$, and at $t = 1$ the score increases to $3 \times 3 = 9$.

These metrics are well-defined for static network link prediction. To extend their application to a dynamic network setting there are several options. However, in the context of preparing an input vector for use in a deep neural network, the most straightforward and information-preserving technique is to compute each metric for each snapshot. Previous research [5] has explored aggregate metrics by weighting snapshots, but the advantage of the deep neural network is to automatically construct an optimal weighting function for the snapshots. Altogether, given t snapshots, the length of the input vector corresponding to these five similarity metrics is $5t$.

2) WEAK ESTIMATORS

While the adaptation of static metrics to the dynamic domain is an adequate approach, the use of metrics that can natively capture the patterns in dynamic systems is desirable. To that end, we propose the use of stochastic learning weak estimators (SLWEs) to characterize the changing likelihood of the existence of a link over time. The goal of SLWEs is to estimate class probabilities in a dynamic system by applying stochastic learning principles to update probabilities as new instances are observed. Namely, the method by which we update should satisfy the Markov property, where the probability distribution update is dependent only on the current state and not on the entire history. In our application, the class probabilities refer to the probability that a link (1) is or (0) isn't present – a two-state process. Thus we can take a binomially-distributed random variable X as follows:

$$X = \begin{cases} 1 & \text{with probability } p_1 \\ 0 & \text{with probability } p_2 \end{cases},$$

where of course $p_1 + p_2 = 1$. To denote the changing probabilities p_1 and p_2 in a temporal context, we can write them as a function of time t with $p_1(t)$ and $p_1(t)$. According to work by Oommen and Rueda [24] and Zhan *et al.* [26], the change in the estimated likelihood of a link from time t to $t + 1$ can be expressed using the following update rule:

$$p_1(t+1) = \begin{cases} \lambda \cdot p_1(t) & \text{if } x(t) = 0 \\ 1 - \lambda(p_2(t)) & \text{if } x(t) = 1 \end{cases}$$

In our specific context, $x(t)$ is the existence of the link at time t and λ is a learning coefficient where $0 < \lambda < 1$. This learning coefficient can be understood to represent the sensitivity of the estimator to new information in that lesser values of λ results in greater changes in p_1 and p_2 from t to $t + 1$. This updating rule has been proven [25] to result in an expected value $E[p_n(\infty)] = p_n$ independent of λ.

For the purposes of providing an input feature for training a deep neural network, we are primarily interested in p_1, the estimated likelihood of the presence of a link, and not so for p_0. Knowing that $p_1 + p_2 = 1$, we can omit calculation of p_0 by rewriting p_1 as a function of a single probability, renamed p, as

$$p(t+1) = \begin{cases} \lambda \cdot p(t) & \text{if } x(t) = 0 \\ 1 - \lambda(1 - p(t)) & \text{if } x(t) = 1 \end{cases},$$

or alternatively as a single equation:

$$p(t+1) = \lambda \cdot p(t) + x(t)(1 - \lambda).$$

The application of stochastic learning weak estimators provides a uniquely valuable and efficiently computable probability measure that evaluates the likelihood of a link as underlying network dynamics shift over time. This is a single value regardless of the number of snapshots.

In total, the length of our input vector as a function of t snapshots is $5t + 1$. The length is completely independent of the size of the network, which is a highly desirable quality for use in time-sensitive applications.

C. TRAINING SET CONSTRUCTION FROM DYNAMIC NETWORK

With a input vector in place, it is necessary to construct a training set to train a deep learning network. This is not a trivial task as it often is in many deep learning applications, since there is no direct relationship between the dynamic network and the space of potential links; a network contains only edges that are present, so we must also generate links that are not present in the network. Algorithm 1 presents the steps for construction of the training set.

The training set consists of *teCt* true edges from the latest snapshot as well as *reCt* randomized edges from the entire edge space. Due to the sparsity of the vast majority of naturally-occurring networks, randomized edges have a low probability of being present, so the generation of randomized edges is approximately equivalent to yet computationally cheaper than generating false edges. That said, in order to obtain balanced classes in our training set, it is generally best practice to satisfy $teCt = reCt$. Once all edges have been generated in this manner, we build an input vector as specified in Section 3.2 using the last *ts* snapshots and then append the classification label (1 if present at time t, otherwise 0). The training set is finally returned once all edges have been processed and added.

D. DEEP LEARNING FRAMEWORK

The final step is to build and train a deep learning network for link prediction. While the training process is treated as a supervised classification problem, the goal is for the network to output a single value that represents the score for the edge. Toward this end, we propose a single node in the output layer with a sigmoid activation function to bound the score strictly between 0 and 1. This score function can be used analogously to the similarity metrics detailed in Section 3.2, but it has the key advantage of having been trained based on the properties of the network whose future states we wish to predict. The number of hidden layers and the nodes per layer is not defined, and is flexible based on the hardware capabilities and desires of the operator.

Algorithm 1 Training Set Construction

```
Require: dynamic network $DN$, timespan $ts$, true edge count
    $teCt$, randomized edge count $reCt$
 1: $te \leftarrow$ sample($teCt$ edges from $\{e \mid e \in DN(t)\}$)
 2: $re \leftarrow$ sample($reCt$ edges from $\{n \mid n \in N^2\}$)
 3: $TS \leftarrow \{\}$
 4: for $e$ in shuffle($te$, $re$) do
 5:     $iv \leftarrow \{\}$
 6:     for $ti$ from $t-1$ to $t-ts$ do
 7:         $iv \leftarrow$append CN($e$, $ti$)          ▷ Common Neighbors
 8:     end for
 9:     for $ti$ from $t-1$ to $t-ts$ do
10:         $iv \leftarrow$append SP($e$, $ti$)          ▷ Shortest Path
11:     end for
12:     for $ti$ from $t-1$ to $t-ts$ do
13:         $iv \leftarrow$append AA($e$, $ti$)          ▷ Adamic/Adar Index
14:     end for
15:     for $ti$ from $t-1$ to $t-ts$ do
16:         $iv \leftarrow$append JI($e$, $ti$)          ▷ Jaccard Index
17:     end for
18:     for $ti$ from $t-1$ to $t-ts$ do
19:         $iv \leftarrow$append PA($e$, $ti$)  ▷ Preferential Attachment
20:     end for
21:     $iv \leftarrow$append WE($e$, $ts$)              ▷ Weak Estimator
22:     if $e \in DN(t)$ then
23:         $iv \leftarrow$append 1
24:     else
25:         $iv \leftarrow$append 0
26:     end if
27:     $TS \leftarrow$append $iv$
28: end for
29: return $TS$
```

IV. EXPERIMENTS

Now that the proposed approach is laid out in full detail, we evaluate our contributions on real-world data. Our results show that our approach is effective and outperforms the baseline similarity metrics.

A. DATASETS

Three real temporal networks were chosen to evaluate the algorithm, sourced from [31], representing various graph sizes, densities, and time spans. Each dataset was provided with timestamps with precision in seconds. In order to obtain coarse-grain snapshots, we discretized the timestamps into five network snapshots using equal width binning. The basic characteristics of each network are listed in Table 1.

1) MATHOVERFLOW

The MathOverflow dataset represents interactions between users of the popular mathematics help forum

TABLE 1. Dataset characteristics.

Dataset	Edges	Nodes	Timespan (Days)
MathOverflow	506550	24818	2350
Eu-core	332334	986	803
CollegeMsg	59835	1899	193

mathoverflow.net, recorded over the span of more than six years. MathOverflow allows users to post math-related questions which others are encouraged to answer. In addition, users can comment on questions and answers to provide additional insights. The dynamic network captures three types of communications: answering a question, commenting on a question, and commenting on an answer. We opt to consider these communications uniformly and as undirected interactions.

2) EU-CORE

This dataset contains over 300,000 emails sent among faculty at a European university, with an edge representing an email sent from one to another institution member at a specific time. The dataset is notable for a high graph density and a strongly-defined community structure that poses both benefits and disadvantages for a link prediction task.

3) COLLEGEMSG

The CollegeMsg dataset describes the student interactions in a social network operating at the University of California Irvine over half a year. Here, an edge represents a private message sent from one student to another at a specific time.

B. EXPERIMENTAL SETUP

Our proposed approach was implemented in Python and run on a single Windows 10 machine with a Intel Xeon E5-1630 v4 @ 3.7 GHz processor and a NVIDIA GeForce GTX 1080 GPU.

We constructed training sets of 8000 links for MathOverflow, 4000 for Eu-core, and 2000 for CollegeMsg. A parameter of $\lambda = 0.8$ was used for the weak estimator learning coefficient. Column normalization was performed with feature scaling to standardize the ranges of each metric. We used a three-layer fully-connected deep neural network with hidden layers of 1024 ReLU neurons, minimizing cross-entropy using an Adam [32] optimizer. The configuration of our DNN is illustrated in Figure 2.

For evaluation purposes, we assembled testing sets by extracting 20% from each training set. Area Under the Curve (AUC) is the established metric to evaluate link prediction accuracy and is utilized here. Two fundamental statistical measures, the True Positive Rate (TPR) and the False Positive Rate (FPR) are used to calculate the AUC:

$$\mathrm{TPR} = \frac{n_{p,p}}{n_{p,p} + n_{p,a}}$$
$$\mathrm{FPR} = \frac{n_{a,p}}{n_{a,p} + n_{a,a}}.$$

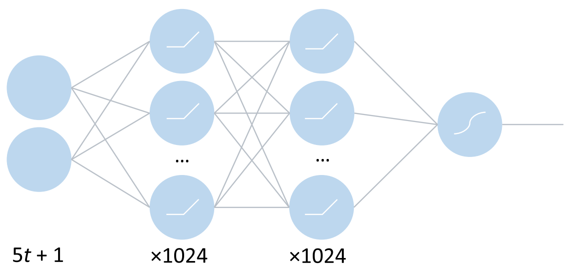

FIGURE 2. Configuration of the deep neural network used in our experiments.

Here, p denotes present, a denotes absent, and $n_{i,j}$ represents the number of $i \in \{p, a\}$ links classified as $j \in \{p, a\}$. In other words, the True Positive Rate represents the percentage of present links that were correctly identified as such, while the False Positive Rate represents the percentage of absent links incorrectly identified as present. From these measures we can build an Receiver Operating Characteristic (ROC) curve, generated by comparing the ratio of TPR and FPR when treating a heuristic as a binary classifier at different discriminating thresholds. The AUC is then calculated as the area under the ROC curve. As a supplement, we also compute the PRAUC, which is constructed similarly to the traditional AUC, but instead uses the Precision and Recall metrics respectively:

$$\mathrm{Precision} = \frac{n_{p,p}}{n_{p,p} + n_{a,p}}$$
$$\mathrm{Recall} = \mathrm{TPR} = \frac{n_{p,p}}{n_{p,p} + n_{p,a}}.$$

For comparison, we introduce ten baselines derived from the five similarity metrics presented in section 3.2. Since they are designed for static networks, we first use the metric for the fourth snapshot to predict link existence in the fifth snapshot, denoted as the ‘‘single’’ variant. Then, we calculate an arithmetic mean of the four snapshots, denoted as the ‘‘average’’ variant. We compare the AUC and PRAUC of our approach with that of these baselines.

C. RESULTS

The experimental results are displayed in tabular form in Table 2 (AUC) and Table 3 (PRAUC). A comparison of AUCs for each dataset are presented in graphical form by Figures 3, 4, and 5 for MathOverflow, Eu-core, and CollegeMsg respectively. In addition, Figures 6, 7, and 8 depict a comparison of the ROC curves of the single variants with our approach for each dataset. As can be clearly seen,

TABLE 2. Experimental results: AUC.

Dataset	Common Neighbors		Shortest Path		Adamic/Adar		Jaccard Index		Preferential Attachment		Our Approach
	Single	Average	Single	Average	Single	Average	Single	Average	Single	Average	
MathOverflow	.770	.777	.782	.773	.705	.743	.769	.776	.780	.778	**.806**
Eu-core	.895	.922	.956	.924	.896	.934	.889	.930	.771	.755	**.986**
CollegeMsg	.617	.574	.730	.650	.617	.584	.613	.577	.720	.590	**.855**

TABLE 3. Experimental results: PRAUC.

Dataset	Common Neighbors		Shortest Path		Adamic/Adar		Jaccard Index		Preferential Attachment		Our Approach
	Single	Average	Single	Average	Single	Average	Single	Average	Single	Average	
MathOverflow	.880	.869	.864	.837	.843	.847	.876	.867	.858	.845	**.879**
Eu-core	.903	.917	.965	.947	.908	.927	.894	.917	.786	.787	**.982**
CollegeMsg	.755	.542	.800	.660	.751	.572	.704	.551	.769	.529	**.864**

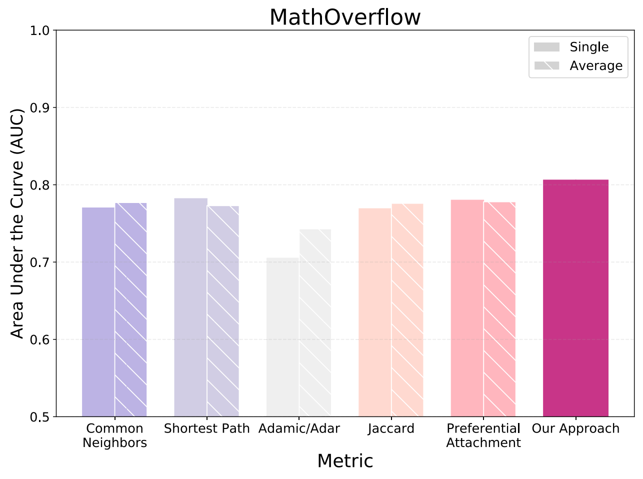

FIGURE 3. Comparison of the AUC measure for each link prediction approach for the MathOverflow dataset.

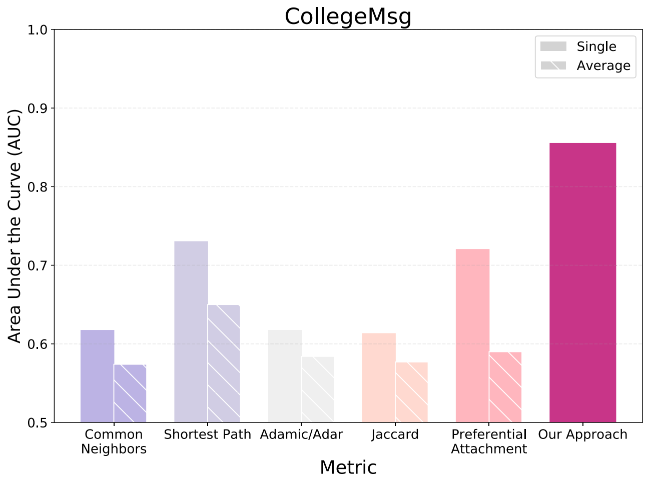

FIGURE 5. Comparison of the AUC measure for each link prediction approach for the CollegeMsg dataset.

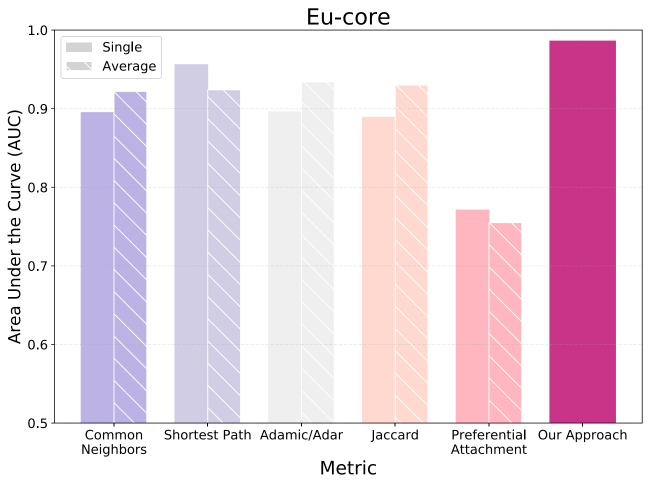

FIGURE 4. Comparison of the AUC measure for each link prediction approach for the Eu-core dataset.

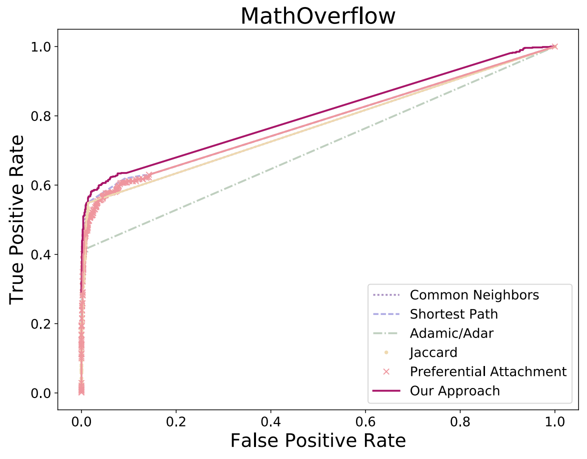

FIGURE 6. Comparison of the ROC curves for each link prediction approach for the MathOverflow dataset.

our approach produces superior results. This is particularly notable in CollegeMsg, where our approach bests the runner up AUC by over 12%. We theorize this may be due to the compressed timespan that exposed dynamic trends on a finer level. In addition, the AUC of .986% for Eu-core is impressive. In comparison to the tRBM and ctRBM approaches described by [14], our approach compares favorably when

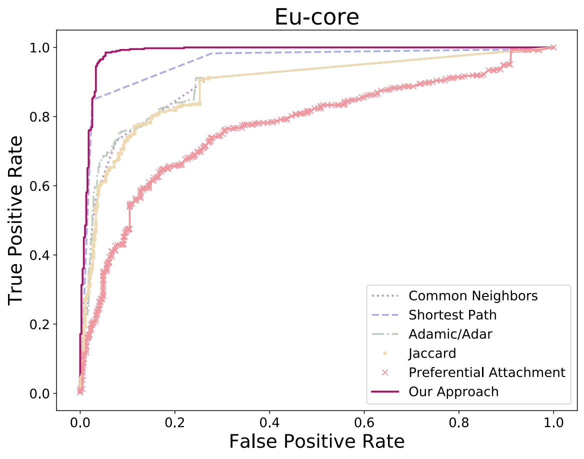

FIGURE 7. Comparison of the ROC curves for each link prediction approach for the Eu-core dataset.

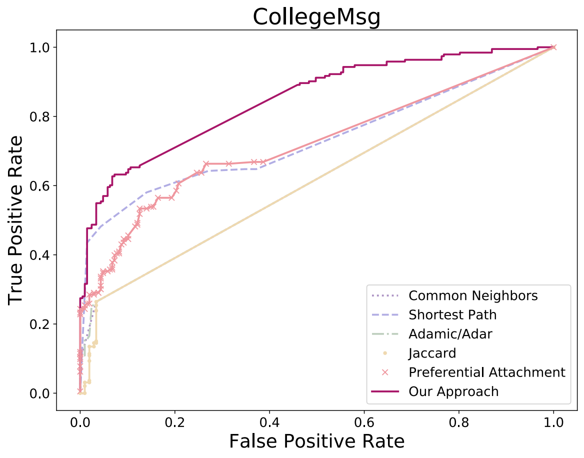

FIGURE 8. Comparison of the ROC curves for each link prediction approach for the CollegeMsg dataset.

assessing average AUC across the real datasets (.882 for our approach versus .840 for the tRBM and .866 for the ctRBM).

It is also worth mentioning that the training of the deep learning network was extremely fast; using the Adam optimizer, the model reached near-optimal accuracy in the first several iterations, taking mere seconds. The results affirm our key objectives in building a deep learning link prediction algorithm that is both accurate and efficient.

D. DISCUSSION

The improved accuracy of our approach over the baselines is promising, and we have directions for further improvement. A large selection of heuristics for link prediction have been proposed and researched, each possessing advantages and disadvantages in accuracy and computational cost. Therefore, there exists a wide array of potential variations to an input vector. Furthermore, there are extensions to be considered regarding the handling of time snapshots, discretization, and weighting. We too are curious about the potential of our algorithm to address the related challenge of link weight prediction, where edges have values and the learning objective becomes a regression task rather than a classification task.

V. CONCLUSION

In this paper, we explored the problem of link prediction in dynamic networks. With applications from sociology to cyber security and bioinformatics, link prediction is a significant research problem, but also a difficult one. By utilizing cheaply computable similarity metrics as well as stochastic learning weak estimators, we proposed an algorithm for constructing an input feature vector for use in a deep neural network. Compared to several baselines, the results on three large real-world dynamic networks demonstrate that our approach improves prediction accuracy while remaining remarkably fast to build and train.

ACKNOWLEDGMENTS

The authors gratefully acknowledge all the organizations for their support.

REFERENCES

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 58, no. 7, pp. 1019–1031, May 2007.
[2] L. A. Adamic and E. Adar, "Friends and neighbors on the Web," *Soc. Netw.*, vol. 25, no. 3, pp. 211–230, 2003.
[3] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, Mar. 1953.
[4] L. Yao, L. Wang, L. Pan, and K. Yao, "Link prediction based on common-neighbors for dynamic social network," *Procedia Comput. Sci.*, vol. 83, pp. 82–89, May 2016.
[5] M. Kaya, M. Jawed, E. Bütün, and R. Alhajj, "Unsupervised link prediction based on time frames in weighted–directed citation networks," in *Trends in Social Network Analysis*. Cham, Switzerland: Springer, 2017, pp. 189–205.
[6] H. Wang, W. Hu, Z. Qiu, and B. Du, "Nodes' evolution diversity and link prediction in social networks," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 10, pp. 2263–2274, Oct. 2017.
[7] H. A. Deylami and M. Asadpour, "Link prediction in social networks using hierarchical community detection," in *Proc. 7th Conf. Inf. Knowl. Technol. (IKT)*, May 2015, pp. 1–5.
[8] W. Hu, H. Wang, C. Peng, H. Liang, and B. Du, "An event detection method for social networks based on link prediction," *Inf. Syst.*, vol. 71, pp. 16–26, Nov. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306437917303976
[9] W. Hu, H. Wang, Z. Qiu, C. Nie, L. Yan, and B. Du, "An event detection method for social networks based on hybrid link prediction and quantum swarm intelligent," *World Wide Web*, vol. 20, no. 4, pp. 775–795, Jul. 2017, doi: 10.1007/s11280-016-0416-y.
[10] L. Zhang, F. Zhuo, C. Bai, and H. Xu, "Analytical model for predictable contact in intermittently connected cognitive radio ad hoc networks," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 7, p. 1550147716659426, 2016, doi: 10.1177/1550147716659426.
[11] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link prediction using supervised learning," in *Proc. Workshop Link Anal., Counterterrorism Secur.*, 2006, pp. 1–10.
[12] N. Benchettara, R. Kanawati, and C. Rouveirol, "Supervised machine learning applied to link prediction in bipartite social networks," in *Proc. Int. Conf. Adv. Soc. Netw. Anal. Mining*, Aug. 2010, pp. 326–330.
[13] J. R. Doppa, J. Yu, P. Tadepalli, and L. Getoor, "Learning algorithms for link prediction based on chance constraints," in *Machine Learning and Knowledge Discovery in Databases*. Berlin, Germany: Springer, 2010, pp. 344–360.
[14] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang, "A deep learning approach to link prediction in dynamic networks," in *Proc. SIAM Int. Conf. Data Mining*, 2014, pp. 289–297.

[15] C. Zhang, H. Zhang, D. Yuan, and M. Zhang, "Deep learning based link prediction with social pattern and external attribute knowledge in bibliographic networks," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Soc. Comput. (CPSCom), IEEE Smart Data (SmartData)*, Dec. 2016, pp. 815–821.

[16] M. Rahman and M. A. Hasan, "Link prediction in dynamic networks using graphlet," in *Machine Learning and Knowledge Discovery in Databases*. Cham, Switzerland: Springer, 2016, pp. 394–409.

[17] T. D. Bui, S. Ravi, and V. Ramavajjala, "Neural graph machines: Learning neural networks using graphs," *CoRR*, vol. abs/1703.04818, Mar. 2017.

[18] R. A. Rossi, R. Zhou, and N. K. Ahmed. (2017). "Deep feature learning for graphs." [Online]. Available: https://arxiv.org/abs/1704.08829

[19] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on Hoeffding's bounds," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 3, pp. 810–823, Mar. 2015.

[20] M. Bhaduri, J. Zhan, C. Chiu, and F. Zhan, "A novel online and non-parametric approach for drift detection in big data," *IEEE Access*, vol. 5, pp. 15883–15892, 2017.

[21] H. Wang and Z. Abraham, "Concept drift detection for streaming data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–9.

[22] H.-K. Song, "A channel estimation using sliding window approach and tuning algorithm for MLSE," *IEEE Commun. Lett.*, vol. 3, no. 7, pp. 211–213, Jul. 1999.

[23] M. M. Lazarescu, S. Venkatesh, and H. H. Bui, "Using multiple windows to track concept drift," *Intell. Data Anal.*, vol. 8, no. 1, pp. 29–59, Jan. 2004.

[24] B. J. Oommen and L. Rueda, "On Utilizing stochastic learning weak estimators for training and classification of patterns with non-stationary distributions," in *Advances in Artificial Intelligence*. Berlin, Germany: Springer, 2005, pp. 107–120.

[25] B. J. Oommen and L. Rueda, "Stochastic learning-based weak estimation of multinomial random variables and its applications to pattern recognition in non-stationary environments," *Pattern Recognit.*, vol. 39, no. 3, pp. 328–341, 2006.

[26] J. Zhan, B. J. Oommen, and J. Crisostomo, "Anomaly detection in dynamic systems using weak estimators," *ACM Trans. Internet Technol.*, vol. 11, no. 1, pp. 3:1–3:16, Jul. 2011.

[27] M. Bhaduri, J. Zhan, and C. Chiu, "A novel weak estimator for dynamic systems," *IEEE Access*, vol. 5, pp. 27354–27365, 2017.

[28] N. B. Silva, I.-R. Tsang, G. D. C. Cavalcanti, and I.-J. Tsang, "A graph-based friend recommendation system using genetic algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2010, pp. 1–7.

[29] P. Wang, B. Xu, Y. Wu, and X. Zhou, "Link prediction in social networks: The state-of-the-art," *CoRR*, vol. abs/1411.5118, pp. 1–38, Jan. 2015.

[30] A. Lebedev, J. Lee, V. Rivera, and M. Mazzara, "Link prediction using top-*k* shortest distances," *CoRR*, vol. abs/1705.02936, pp. 101–105, Jun. 2017.

[31] J. Leskovec and A. Krevl. (Jun. 2014). "SNAP datasets: Stanford large network dataset collection." [Online]. Available: http://snap.stanford.edu/data

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, May 2015.

[33] P. Ezatpoor, J. Zhan, J. M.-T. Wu, and C. Chiu, "Finding top-*k* dominance on incomplete big data using mapreduce framework," *IEEE Access*, vol. 6, pp. 7872–7887, 2018.

[34] P. Chopade and J. Zhan, "A framework for community detection in large networks using game-theoretic modeling," *IEEE Trans. Big Data*, vol. 3, no. 3, pp. 276–288, Sep. 2017.

[35] M. Bhaduri, J. Zhan, and C. Chiu, "A weak estimator for dynamic systems," *IEEE Access*, vol. 5, no. 1, pp. 27354–27365, 2017.

[36] C. Chiu, J. Zhan, and F. Zhan, "Uncovering suspicious activity from partially paired and incomplete multimodal data," *IEEE Access*, vol. 5, pp. 13689–13698, 2017.

[37] R. Ahn and J. Zhan, "Using proxies for node immunization identification on large graphs," *IEEE Access*, vol. 5, pp. 13046–13053, 2017.

[38] J. M.-T. Wu, J. Zhan, and J. C.-W. Lin, "Ant colony system sanitization approach to hiding sensitive itemsets," *IEEE Access*, vol. 14, pp. 10024–10039, 2017.

[39] J. Zhan and B. Dahal, "Using deep learning for short text understanding," *J. Big Data*, vol. 4, p. 34, Oct. 2017.

[40] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, J. Z. Zhan, and J. H. Zhang, "Exploiting highly qualified pattern with frequency and weight occupancy," *Knowl. Inf. Syst.*, to be published.

[41] J. C.-W. Lin, T.-P. Hong, P. Fournier-Viger, Q. Liu, J.-W. Wong, and J. Zhan, "Efficient hiding of confidential high-utility itemsets with minimal side effects," *J. Exp. Theor. Artif. Intell.*, vol. 29, no. 6, pp. 1225–1245, 2017.

[42] J. Zhan, S. Gurung, and S. P. K. Parsa, "Identification of top-K nodes in large networks using Katz centrality," *J. Big Data*, vol. 4, p. 16, May 2017.

[43] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, J. M.-T. Wu, and J. Zhan, "Extracting recent weighted-based patterns from uncertain temporal databases," *Eng. Appl. Artif. Intell.*, vol. 61, pp. 161–172, May 2017.

[44] J. Zhan, T. Rafalski, G. Stashkevich, and E. Verenich, "Vaccination allocation in large dynamic networks," *J. Big Data*, vol. 4, p. 2, Jan. 2017.

[45] W. Gan, J. C.-W. Lin, H.-C. Chao, and J. Zhan, "Data mining in distributed environment: A survey," *Wires Data Mining Knowl. Discovery*, vol. 7, no. 6, p. e1216, Nov./Dec. 2017.

[46] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, and J. Zhan, "Mining of frequent patterns with multiple minimum supports," *Eng. Appl. Artif. Intell.*, vol. 60, pp. 83–96, Apr. 2017.

[47] J. M.-T. Wu, J. Zhan, and J. C.-W. Lin, "An ACO-based approach to mine high-utility itemsets," *Knowl.-Based Syst.*, vol. 116, pp. 102–113, Jan. 2017.

[48] M. Pirouz, J. Zhan, and S. Tayeb, "An optimized approach for community detection and ranking," *J. Big Data*, vol. 3, p. 22, Nov. 2016.

[49] J. C.-W. Lin, W. Gan, P. Fournier-Viger, T.-P. Hong, and J. Zhan, "Efficient mining of high-utility itemsets using multiple minimum utility thresholds," *Knowl.-Based Syst.*, vol. 113, pp. 100–115, Dec. 2016.

[50] J. C.-W. Lin, T. Li, P. Fournier-Viger, T.-P. Hong, J. M.-T. Wu, and J. Zhan, "Efficient mining of multiple fuzzy frequent itemsets," *Int. J. Fuzzy Syst.*, vol. 19, no. 4, pp. 1032–1040, 2017.

CARTER CHIU is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Nevada at Las Vegas, Las Vegas, NV, USA. His research interests include deep learning and big data analytics. He is a member of the Big Data Hub with the University of Nevada.

JUSTIN ZHAN was a Faculty Member with North Carolina A&T State University, Carnegie Mellon University, and the National Center for the Protection of Financial Infrastructure, South Dakota, USA. He is a currently a Professor with the Department of Computer Science, College of Engineering, University of Nevada at Las Vegas, Las Vegas, NV, USA, where he is also the Director of the Big Data Hub. His research interests include big data, information assurance, social computing, and health science.