Journal of Ambient Intelligence and Humanized Computing (2021) 12:1981–1993
https://doi.org/10.1007/s12652-020-02289-0

ORIGINAL RESEARCH

Link prediction in dynamic networks using time-aware network embedding and time series forecasting

Anuraj Mohan[1] · **K. V. Pramod**[2]

Received: 11 February 2020 / Accepted: 27 June 2020 / Published online: 9 July 2020

Abstract
As most real-world networks evolve over time, link prediction over such dynamic networks has become a challenging issue. Recent researches focus towards network embedding to improve the performance of link prediction task. Most of the network embedding methods are only applicable to static networks and therefore cannot capture the temporal variations of dynamic networks. In this work, we propose a time-aware network embedding method which generates node embeddings by capturing the temporal dynamics of evolving networks. Unlike existing works which use deep architectures, we design an evolving skip-gram architecture to create dynamic node embeddings. We use the node embedding similarities between consecutive snapshots to construct a univariate time series of node similarities. Further, we use times series forecasting using auto regressive integrated moving average (ARIMA) model to predict the future links. We conduct experiments using dynamic network snapshot datasets from various domains and demonstrate the advantages of our system compared to other state-of-the-art methods. We show that, combining network embedding with time series forecasting methods can be an efficient solution to improve the quality of link prediction in dynamic networks.

Keywords Dynamic networks · Link prediction · Network embedding · Neural networks

1 Introduction

Network mining has gained a lot of attraction recently due to its importance in diverse areas which includes both scientific and business domains. Link prediction (Liben-Nowell and Kleinberg 2007; Lü and Zhou 2011) is one interesting problem in network mining which aims to predict future interaction between entities in a network that can model the evolution of networks. Friend recommendation in social networks, interaction prediction between proteins, disease outbreak prediction, etc. are some typical applications of link prediction. Many supervised and unsupervised learning methods (Al Hasan and Zaki 2011; Martínez et al. 2017) were applied to perform link prediction in networks. Recent studies show that embedding the network in a latent vector space, and using vector-based similarity measures for link prediction is more efficient compared to traditional methods. Various network embedding methods (Cai et al. 2018) are developed in recent years, and most of these methods deal with learning embeddings from static networks. These methods not only improve the prediction accuracy but also makes the task scalable for networks with millions of nodes and edges.

The main aim of network embedding is to map a complex network to a low dimensional latent space by preserving the structural properties of the network. Developments in the field of network embedding (Goyal and Ferrara 2017) can be classified into mainly three which include methods based on matrix factorization, work2vec architecture and deep neural networks. Initially, non-linear dimensionality reduction techniques were used for network embedding followed by specialized graph factorization based methods like locally linear embedding (LLE) (Roweis and Saul 2000), Laplacian Eigenmaps (Belkin and Niyogi 2002), Grarep (Cao et al. 2015), and Hope (Ou et al. 2016). Even if these methods can capture the structural properties of the network, they are computationally expensive. With the recent advancements in the field of neural networks and deep learning, researches

✉ Anuraj Mohan
anurajmohan@gmail.com

[1] Artificial Intelligence Lab, Department of Computer Applications, Cochin University of Science and Technology, Kerala 682022, India

[2] Department of Computer Applications, Cochin University of Science and Technology, Kerala 682022, India

moved towards using representation learning methods for network embedding. These methods are highly scalable and can handle the complex non-linear structure, which is an inherent property of networked data. Deepwalk (Perozzi et al. 2014), Node2vec (Grover and Leskovec 2016), SDNE (Wang et al. 2016), etc are some popular works which uses representation learning for generating network embeddings.

Most of the real-world networks like social networks and citation networks evolve with time. Social network evolves by adding new people and relationships, whereas new papers and citations lead to the growth of citation network. Such networks can be mathematically modelled either as a dynamic graph, which is represented as a series of snapshots, or as temporal network with the duration of interaction or interval timestamped on the edges. Even if some researches (Casteigts et al. 2012; Blonder et al. 2012) focused on learning the network dynamics, identifying the mechanism that leads to its evolution is a challenging issue. In this work, we investigate the way to incorporate the evolution mechanism into network embedding which can generate node embeddings that are evolved over time. Further, we create similarity based time series using the node embeddings generated from dynamic network snapshots and uses time series forecasting for predicting future links in the successive snapshots.

Generating embedding from a dynamic network is not straight forward. Embedding snapshots independently and aligning them across consecutive time steps is computationally expensive. Nodes and edges may vary across consecutive time steps which demand new techniques to be incorporated with static embedding methods. Successive snapshots may not differ much and the embeddings generated must be stable across various snapshots. The size of many real-world networks may grow exponentially over a period of time which demands the embedding method to be highly scalable. Some researchers have already attempted to generate node embedding from dynamic networks. All these works either used matrix factorization based models (Zhu et al. 2016) or deep neural architectures (Goyal et al. 2018) which are computationally complex.

In this paper, we present a word2vec based model for generating time aware embeddings from dynamic networks. Word2vec (Mikolov et al. 2013a, b) is a very successful model for generating word representations which showed tremendous improvements in solving many tasks related to natural language processing and text mining. Word2vec defines a shallow three layer neural network architecture whose objective is to maximize the probability of words that tend to co-occur to be close in a low dimensional vector space. Later, Word2vec provided inspirations for many network embedding researches (Perozzi et al. 2014; Grover and Leskovec 2016) which used neural network architectures to generate node embeddings. These works perform a random walk over the network to generate a sentence analogy in word2vec and use a skip-gram architecture to maximize the probability that the nodes that tend to co-occur on truncated walk lay closer in the embedding space. A general architecture of skip-gram based procedure for network embedding is shown in Fig. 1.

In all existing works, the embeddings generated after training the complete snapshots are used to predict the future links. But, non-linear fluctuations in connectivity patterns may occur within a series of consecutive snapshots which may reduce the efficiency of such predictions. To avoid such problems, we construct a time series based on node embedding similarities between consecutive snapshots and further uses time series forecasting using ARIMA model (Brockwell and Davis 2016) to perform link prediction.

The main contributions of our work are as follows.

i) We propose a representation learning architecture, which is modification over the skip-gram architecture, that can generate embeddings from dynamic networks. The proposed method can learn time-aware embeddings from network snapshots that are captured over a period of time.
ii) We utilize the time-aware embeddings to construct a univariate time series based on the node embedding similarities and performs time series forecasting using ARIMA model to predict future links.

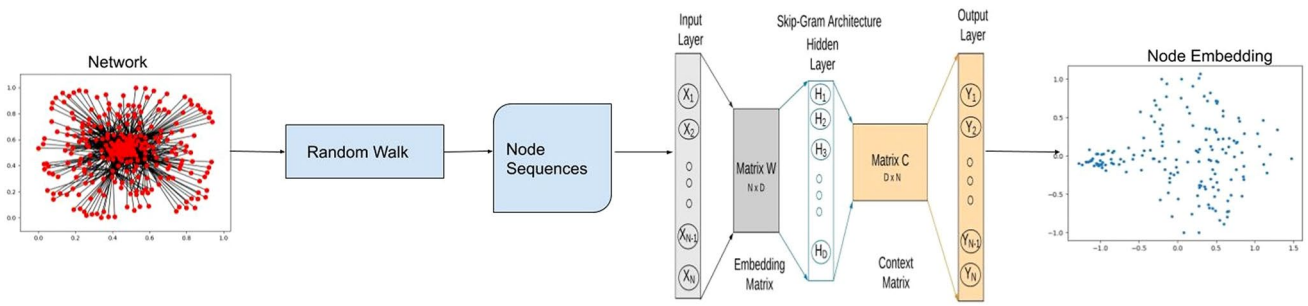

Fig. 1 Skip-gram based procedure for network embedding

iii) To the best of our knowledge, this is the first attempt which combines both dynamic network embedding and time series prediction based approaches for link prediction in dynamic networks.
iv) We conduct experiments on real-world dynamic networks and proves that the proposed method can provide better link prediction accuracy compared to state-of-the art methods.

2 Background

In this section, we define the terminologies used in this work and the problem definition.

Definition 1 A network is a graph $G = (V, E)$, where $V = \{v_1, v_2 ... v_n\}$, is the set of vertices and $e \in E$ is an edge between any two vertices. An adjacency matrix A defines the connectivity of G, $A_{ij} = 1$ if v_i and v_j are connected, else $A_{ij} = 0$.

Definition 2 A dynamic network can be modeled as contact sequences, snapshots, or interval graphs. In this work, we model a dynamic network as a series of snapshots $G = \{G_1, G_2, ...G_n\}$ where $G_i = (V_i, E_i)$ and n is the number of snapshots. New nodes and edges may be added and removed from the network over time, and we consider that the network snapshots at different time steps can capture this information.

Problem 1. Network embedding: Given a network $G = (V, E)$, the task is to learn a transformation function $f : V_i \rightarrow K_i \in R^d$, where $d << |V|$, such that f preserves the proximity information of the network. d defines the number of dimensions of the real-valued vector. Informally, if two nodes are connected or share similar neighborhoods in the network, their corresponding vector representations will occupy nearer positions in the vector space.

Problem 2. Time-aware network embedding (TANE): It is an extension to network embedding which can better represent dynamic networks in vector space. Given a as a series of network snapshots at n time steps, $G = \{G_1, G_2 ... G_n\}$, Time-Aware Network Embedding tries to learn a mapping function $f : V_i \rightarrow K_i \in R^d$, where $d << |V|$, from the network snapshots which captures both network proximities and temporal dynamics of the network. In the embedding space thus generated, two vectors, that correspond to two nodes are closer implies that the nodes preserve similar neighborhoods, and their interactions have constantly evolved.

Problem 3. Link prediction in dynamic network: Given a as a series of network snapshots at t discrete time steps, $G = \{G_1, G_2 ... G_n\}$, the task is to the predict the future graph at time $t + 1$, i.e G_{t+1}.

3 Related works

Apart from using graph algorithms, machine learning on graphs is one interesting direction towards network mining research. Network embedding is one such domain which gained attention in recent years. The basic purpose of network embedding is to encode the network structure to a low dimensional vector space by preserving the topological properties and thereby aims to perform the downstream mining tasks like node classification and link prediction with improved accuracy. The network embedding research started by applying non-linear dimensionality methods to generate node embeddings followed by specialized graph factorization based methods. As representation learning emerged as a successful method to solve many problems related to computer vision and text mining, the domain of network embedding is also influenced by these methods. Among these methods, word2vec (Mikolov et al. 2013a, b) is one popular architecture which uses a skip-gram model to generate word representations by training a neural network on a text corpus. Many network embedding methods like DeepWalk (Perozzi et al. 2014), Node2vec (Grover and Leskovec 2016) and Struc2vec (Ribeiro et al. 2017) are inspired from Word2vec. Another approach includes methods based on deep architectures like deep belief networks, Deep CNN and GAN. Nodes and edges of real-world networks may be attributed and heterogeneous. TADW (Yang et al. 2015), TriDNR (Pan et al. 2016) and DeepGL (Rossi et al. 2017) are some works on attributed network embedding and Metapath2vec (Dong et al. 2017) ,HNE (Chang et al. 2015) and Hin2vec (Fu et al. 2017) are some works on heterogeneous network embedding. All these methods are applicable only to static networks.

A dynamic network provide a more precise way of representing complex interactions compared to a static network. Analysis of dynamic networks has got wide applications in various domains. Embedding dynamic networks demand developing new techniques or adapting existing methods to incooperate dynamic network properties. Authors of (Zhu et al. 2016) attempt to incorporate a temporal regularizer into the matrix factorization framework so as to generate a temporal latent space from snapshot networks. Dyngem (Goyal et al. 2018) uses a deep belief network which generates node embeddings from a series of network snapshots. Dynamictriad (Zhou et al. 2018) considers triadic closure as a basic mechanism which leads to network evolution and uses it to generate node embeddings. DDNE (Li et al. 2018) is another work which uses a GRU encoder and a deep neural network decoder for node embedding generation. To capture the network dynamics effectively, the system needs to be trained using large number of network snapshots. Such a task will be computationally complex if we are using

matrix factorization and deep architecture based methods. In this paper, we propose a modified skip-gram architecture which is a shallow neural network that takes as input, node sequences generated using random walk from a sequence of network snapshots, and generates node embeddings that are constantly evolved over time.

Various approaches (Ma et al. 2017; Ahmed et al. 2018; Yasami and Safaei 2018; Wu et al. 2018) exist in literature to perform link prediction on dynamic networks. As a dynamic network is one which evolve over time, time series modeling and prediction is a good approach towards studying link prediction in dynamic networks. Time series forecasting has been successfully used in many applications where we aim to predict the future values of a variable using the past observations of the same variable. ARIMA (Brockwell and Davis 2016) is a successful model to capture the non-linearities in time series data. Recent works on time series forecasting use soft computing based methods which include fuzzy neural networks (Soto et al. 2018), fuzzy aggregation models with modular neural networks (Soto et al. 2019), and evolutionary computing approaches (Gupta et al. 2018). Many researchers approached the problems related to dynamic networks from perspective of time series forecasting. (Wu et al. 2018) studied the dynamics of a time-varying network as the problem of tracking a time series of finite dimensional vectors. The evolution of a temporal network can be well described using a time series of event sequences (Jo and Hiraoka 2019) which follow non-linear patterns. A detailed study (Zou et al. 2019) of complex network approaches to nonlinear time series analysis also exist in the literature. Recent studies (Güneş et al. 2016; Özcan and Öğüdücü 2016, 2017) show that time series forecasting based methods outperform other traditional methods for performing link prediction in dynamic networks. Most of these methods use node similarities and centrality measures for time series construction, and linear models for prediction which fails to capture the non-linear temporal variations of connectivity patterns in dynamic networks. The proposed system is to generate time aware embeddings from a dynamic network, performs time series construction using node embedding similarity scores and predicts future similarity scores using ARIMA model.

4 System design

4.1 Random walk

A random walk is a discrete-time stochastic process which is widely used in graph theory domain, particularly in applications like graph partitioning (Spielman and Teng 2004), community detection (Pons and Latapy 2005) and Pagerank (Chung and Zhao 2010). In this work, we consider random walk as a sampling technique to capture the local community structure of the network. A random walk on a graph G can be represented as a sequence of vertices $v_1, v_2, v_3, .., v_k$ such that the adjacent nodes in the sequence are connected in G. A random walk of length k starts from any node, randomly select a neighbor, visits the neighbor and the process continues until k edges are covered. The next vertex is selected with uniform probability from the set of neighbors. The same can be modeled as a markov chain, where the states of the chain are the vertices of the graph.

Given the adjacency matrix A_{ij} and $d(i) = \sum_j A_{ij}$, the degree of node i

The transition probability between two connected vertices i to j can be represented as

$T_{ij} = \frac{A_{ij}}{d(i)}$

This can be also represented as

$T = D^{-1}A$, where D is the diagonal degree matrix, $D_{ii} = d(i)$ and $D_{ij} = 0$ if $i \neq j$.

Let P^0_{ij} represents i 1-hot row vectors with the value of i^{th} entry is 1 and all others 0, which indicates that the random walk can start at any vertex, and let the stationary transition probability between two connected vertices to be T_{ij}. By assuming random walk without restart, the transition probability after a single walk can be computed as

$$P^1_{ij} = P^0_{ij} \times T_{ij}$$

The transition probability after k steps of random walk (walk of lenght k) can be computed as

$$P^k_{ij} = P^0_{ij} \times \prod_{i=1}^{k} T_{ij}$$

The i^{th} row of the matrix represent the transition probability from vertex i to all other vertices after a random walk of length k.

If the graph is strongly connected, it can be observed that the probability distribution reaches a steady state. For a random walk to capture the local structure, it should be long enough to gather the topological information and also should not be too long to avoid stationary distribution. So in this work, we are using a truncated random walk where the length of the walk is fixed. The k-step transition probability between i and j for a random walk with fixed length depends on the node degrees $d(i)$ and $d(j)$.

For vertices which are closer in the graph, the value of P^k_{ij} will be high. Fomally, we can say that two vertices i and j are similar if

$$\forall n \; P^k_{in} \approx P^k_{jn}$$

ie i and j have similar transition probability distribution w.r.t. all other nodes. The distance between i and j can be represented as

$$d_{ij} = \sqrt{\sum_{m=1}^{n} \frac{(P_{im}^k - P_{jm}^k)^2}{d(m)}}$$

The error value d_{ij} can be also interpreted as the distance between two probability distributions P_{im}^k and P_{jm}^k. When we have a truncated random walk of length k, ($k << n$), and when m is in the local neighborhood of i and j, the error value drops to minimum. This shows that the truncated random can capture the local community structure of the graph.

4.2 Skip-gram architecture

A skip-gram model is a shallow neural network architecture used by word2vec to generate word embeddings. The skip-gram for network embedding (Perozzi et al. 2014) deploys the same three-layer neural network architecture used by word2vec. Here the objective is to maximize the probability of neighboring nodes in the random walk, given the representation of a node. Skip-gram generates a d dimensional representation, $\phi(v_i) \in R^d$ for each node v_i, by maximizing the co-occurrence probability of its neighbors in the walk. The optimization function can be represented as,

$$\max \log P(v_{i-w}, ..., v_{i+w} | \phi(v_i)) \tag{1}$$

where $v_{i-w}, ..., v_{i+w}$ denotes the neighbors of v_i in the node sequence, and w is the context size. The output layer of skip-gram neural architecture is a softmax function which computes the co-occurence probability of output and input words, w_o and w_i as

$$p(w_o|w_i) = \frac{exp({v'_{wo}}^{\top} v_{wi})}{\sum_{w=1}^{N} exp({v'_w}^{\top} v_{wi})} \tag{2}$$

where v'_w and v_w are the vector representations of the context and input word respectively, and N is the size of the vocabulary. Estimating softmax function is computationally expensive, and word2vec approximates softmax using two strategies, hierarchical softmax (Morin and Bengio 2005), and negative sampling(Goldberg and Levy 2014). These strategies reduce the time complexity of the skip-gram model and speed up the training process.

4.3 Proposed system

The workflow of the proposed system is shown in Fig. 2. The input to the TANE algorithm is consecutive network snapshots, and the algorithm generates node embeddings from each snapshot by capturing the temporal dynamics of the evolving network. These embeddings are fed to a time series construction phase which generates a univariate time-series using node embedding similarity between each disconnected node pairs. The time series thus constructed is passed to the ARIMA model to predict the future similarity scores between disconnected vertices and thereby uses it to predict the future links.

4.3.1 TANE algorithm

In the case of dynamic networks, the network structure may vary at different snapshots and the basic skip-gram architecture cannot capture these changes. Even if we train the skip-gram model for one snapshot, for the next snapshot we need to retrain the entire model even if there is a small change in connectivity between two consecutive snapshots. This is a tedious task when we have a large number of snapshots. Moreover the training the model independently with different snapshots may also lead to sub-optimal results. This is because the embeddings thus generated for the same node with similar connectivity patterns across different snapshots may be different as they are not aligned to the same vector space. Our proposed architecture (TANE) is designed such that the neural network can be trained snapshot by snapshot,

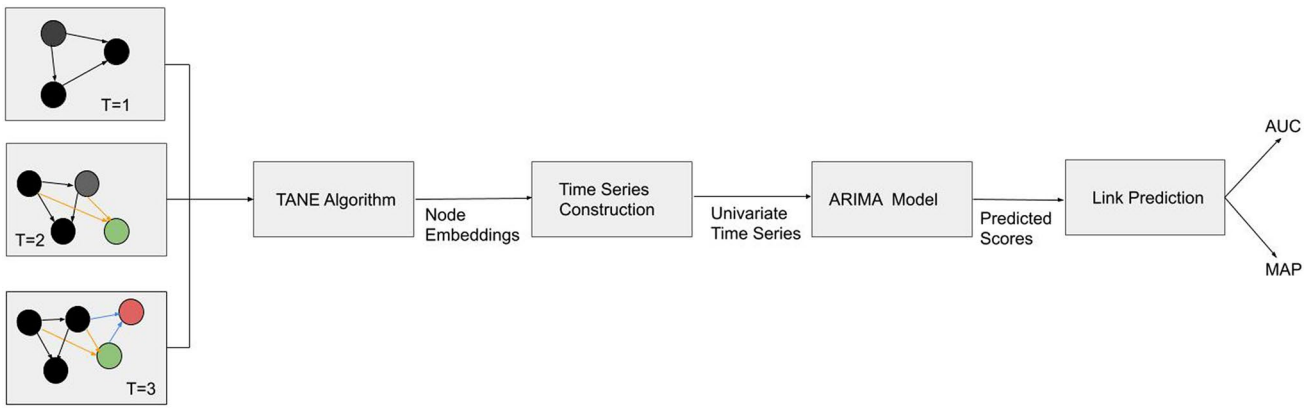

Fig. 2 Flow diagram of proposed system

by preserving the weight learned at each snapshot. At each time step, TANE aims to expand the neural architecture of skip-gram by preserving the input to hidden and hidden to output layer weights of the previous time step and thereby generating time aware network embeddings. The procedure is described in Algorithm 1.The algorithm takes as input n network snapshots $G = \{G_1, G_2...G_n\}$, the context window size w, the embedding dimension d, walk per vertex γ and walk length t. The algorithm generates an embedding matrix v x d as output, where v is the total number of vertices across all network snapshots. The procedure involves a two-step process. A random walk on the network snapshots to incrementally build the vertex vocabulary and a skip-gram objective optimization which incrementally learns the network embeddings from the snapshots. Vocabulary building starts from the first snapshot, and the embedding and context matrices are initialized as zero. The skip-gram objective is optimized by training a three-layer neural network which is optimized using stochastic gradient descent (SGD) (Bottou 2010). For successive snapshots, the skip-gram architecture is expanded by initializing the embedding and context vectors from the corresponding matrices of the previous snapshot and by initializing the vectors of newly observed vertices to zero. Further, the vocabulary is built incrementally by adding the newly observed vertices and by maintaining the occurrence count of each vertex. At each snapshot, the skip-gram objective is optimized to ensure that the embedding generated over successive snapshots have captured the temporal dynamics of the evolving network.

Algorithm 1 TANE Algorithm

Input: Network Snapshots G_1, G_2, G_3,..,G_n where $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$,...,$G_n = (V_n, E_n)$, window size w, embedding size d, walks per vertex γ and walk length t

Output: Embedding Matrix $E = R^{|V| \times d}$

for snapshots i from 1 to n **do**
  $walks(S_i) = randomwalk(S_i)$
  if i=1 **then**
    $Voc = BuildVocabulary(walks(S_i))$
    Initialize the embedding and context matrices, $E_v(S_i) = R^{|V| \times d}$, $C_v(S_i) = R^{d \times |V|}$
  else
    Load $Voc, E_v(S_{i-1}), C_v(S_{i-1})$
    for $u \in walks(S_i)$ **do**
      if $u \in Voc$ **then**
        $E_u(S_i) = E_u(S_{i-1})$
        $C_u(S_i) = C_u(S_{i-1})$
      else
        $E_u(S_i) = C_u(S_i) = 0$
    $Voc = BuildVocabulary(walks(S_i))$
  $SkipGram(E_v, walks, w)$
  Save $Voc, E_v(S_i), C_v(S_i)$

procedure BUILDVOCABULARY(WALKS)
  for $u \in walks$ **do**
    if $u \in Voc$ **then**
      count(u)++
    else
      addvocab(Voc,u)
      size(Voc)++

procedure SKIPGRAM(E_v,WALKS,W)
  for $v_k \in walks$ **do**
    for $v' \in walks(k - w : k + w)$ **do**
      $L(E) = -\log P(v'|E(v_k))$
      $stochasticgradientdescent(L(E))$

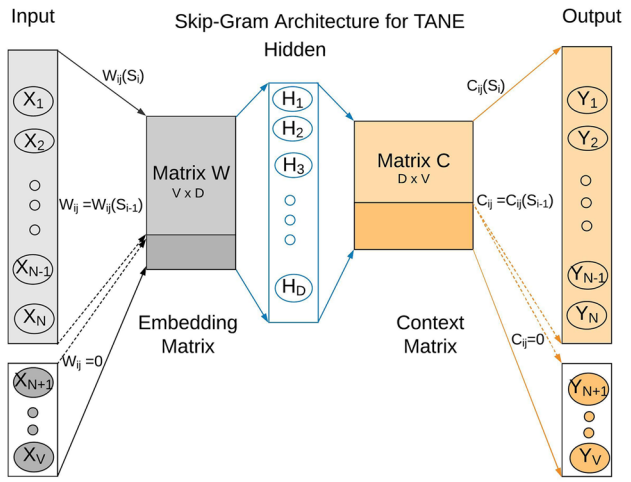

Fig. 3 Skip-gram architecture for TANE algorithm

Figure 3 shows the skip-gram architecture for TANE corresponding to two consecutive network snapshot S_{i-1} and S_i at time $i-1$ and i respectively. For those vertices in S_i which are already present in previous snapshot S_{i-1}, the weight vectors are loaded from the matrices of snapshot S_{i-1}, and the vectors of newly seen vertices are initialized to zero.The objective is optimized for each snapshot by preserving the embedding generated at each time step. This approximates the optimization of a global objective function which will learn the parameters of network embedding across all snapshots and thereby generates stable and time-aware network embedding.

4.3.2 Time series construction and forecasting

Once node embeddings are created for every snapshot, we pass the embeddings to a time series construction phase. Here, we first define a similarity matrix for every snapshot, where the similarity between two nodes x and y is computed as the cosine similarity between the node embeddings $v(x)$ and $v(y)$. We further align the similarity scores of every node pair across the series of snapshots to form a time series. Predicting the future association between two nodes can be now modeled as a time series forecasting problem. We pass the time series of node embedding similarities constructed in the previous step to a time series forecasting phase. We use the ARIMA model for time series forecasting. The $ARIMA(p, d, q)$ model is the combination of autoregressive and moving average processes, where p is the number of autoregressive terms, q is the number of moving average terms and d is the number of non-seasonal difference operations. The ARIMA model for forecasting the similarity score between two nodes x and y at snapshot t can represented as

$$Sim(x,y,t) = \mu + \phi_1 sim(x,y,t-1) + ... + \phi_p sim(x,y,t-p) + \Theta_1 e_{t-1} + ... + \Theta_q e_{t-q} + e_t \quad (3)$$

Where $\phi_1 sim(x,y,t-1) + ... + \phi_p sim(x,y,t-p)$ is the autoregressive terms which allows to incorporate the past similarity scores into the model and $\theta_1 e_{t-1} + ... + \theta_q e_{t-q}$ is the terms representing lagged forecast errors in prediction, and u is the average of past similarity scores. Model is tested with different values of p,d and q and the parameters which provided the minimum value of Akaike information criterion (AIC) is considered for generating similarity scores. For conducting link prediction experiments, we consider the missing links in the final snapshots with the highest predicted scores as the future links.

5 Experiments

We conduct experiments with four real-world networks to evaluate the quality of embeddings generated by the TANE algorithm. MAP and ROC are the measures which are used for evaluation. The results are compared with that of the baseline methods. All experiments were conducted using a machine with Ubuntu 16.04 operating system, 16 GB RAM and a hexa-core processor with 3.2 GHz speed. We modified the C implementation of google word2vec for developing TANE algorithm and used python packages including neworkx and scikit-learn for graph processing, time series prediction and link prediciton evaluation.

5.1 Datasets and evaluation measures

ENRON: This dataset (Klimt and Yang 2004) consists of emails between the employees in Enron Inc. from January 1999 to July 2002. Each node in the network represents a user, and a link represents email communication between them. The network consists of 150 users and 2609 edges.

HAGGLE: This network (Chaintreau et al. 2007) describes human contact informations where contacts between people are measured by some wireless devices. Nodes represent users and links between them indicates a contact. The dataset consists of 274 nodes and 28,244 edges.

HEP-PH: This a collaboration graph (Gehrke et al. 2003) of authors of scientific papers from Hep-Ph section of arXiv archive. It consists of 28,093 nodes and 4,596,803 edges. The data covers papers in the period from January 1993 to April 2003.

RADOSLAW: This network (Michalski et al. 2011) represents the email communication between employees in a mid-sized manufacturing company. Nodes in the network represent employees and edges between them are individual

Table 1 Statistics of various datasets used

Dataset	# of nodes	# of edges	# of times-tamps
Enron	150	2609	26
Haggle	274	28,244	8
Hep-ph	28,093	4,596,803	10
Radoslaw	167	82,927	10

emails. The dataset covers a period from January 2010 to October 2010 and consists of 167 vertices and 82,927 edges. A summary of various datasets used is shown in Table 1.

Area under curve (AUC): AUC is a widely used evaluation metric for link prediction. This metric can be interpreted as probability that a randomly chosen missing link is given a higher score than a randomly chosen non-existent link, provided the rank of all the non-observed links. The value of this metric is bounded between 0 and 1, and higher the value implies better the model. Among n independent comparisons, if there are n' times the missing link having a higher score and n'' times they have the same score, AUC score (Liu et al. 2011; Lü and Zhou 2011) is calculated as:

$$AUC = \frac{n' + 0.5n''}{n} \tag{4}$$

Mean average precision (MAP): This metric is an extension of average precision (AP) where the average of all APs is calculated to get MAP score. It estimates the precision of every node and computes the average over all nodes. It is calculated as:

$$AP(i) = \frac{\Sigma_j Precision@j(i).\Delta i(j)}{|\{\Delta i(j) = 1\}|} \quad MAP = \frac{\Sigma_{i \epsilon Q} AP(i)}{|Q|} \tag{5}$$

The performance of the proposed system is evaluated with various state-of-the-art works by considering all the links in the network at a future time as well as by taking only the new links that occur in the future time. A link (u, v) is said to be a new link if it is not present in the last snapshot.

6 Results and analysis

We conducted link prediction experiments with time-aware embeddings (TANE) independently and also by combining time-aware embeddings with time series forecasting (TANE-TS). The TANE algorithm takes as input, the network snapshots and generates embeddings that are evolved continuously over time. These embeddings can be used to perform link prediction. To test the link prediction performance in static network, the procedure is as follows. Hide 10–15% of the links to form the training set, generate node embeddings from the training set, use hammard product of the node embedding to form the edge embedding and build a classifier based on positive and negative edges. Hidden edges are used to test the accuracy of the classifier. To test link prediction performance with dynamic networks, we hide 10–15% of links of each network snapshot from time 1 to 't-1', generates embeddings and predict the links at time 't'. To test link prediction performance with TANE-TS, we build the time series using similarity of dynamic embeddings from snapshot 1 to t-1 and forecast the predicted scores for possible links in snapshot t. As TANE maps networks to a low dimensional space, the dimensionality of nodes is an important which affects the performance of link prediction task. We conducted experiments using different values of node dimensionality w.r.t each dataset and found 128 as the optimum value which is used for performance comparison. Now, we present the baseline methods along with the analysis and comparison of results obtained from conducting link prediction experiments on four real world networks.

6.1 Baseline methods

We selected baselines from different category of works. These include link prediction using static embedding and dynamic network embedding methods, and link prediction using node similarity based time series forecasting methods. A short introduction to specific baselines are listed below.

Node2vec (Grover and Leskovec 2016): Node2vec performs random walk on static network to generate node sequences and uses skip-gram with negative sampling to generate node representations. Unlike deepwalk, node-2vec performs a baised random walk which provides more

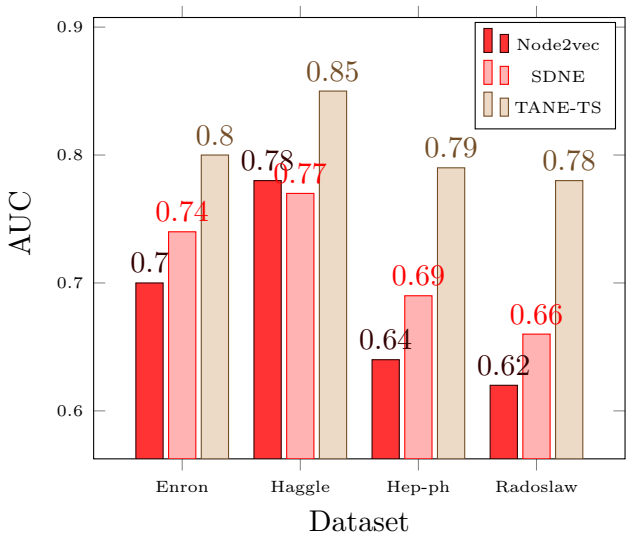

Fig. 4 AUC comparison of TANE-TS with static embedding methods

Table 2 MAP comparison of proposed system with static embedding methods

Method	Enron	Haggle	Hep-ph	Radoslaw
Node2vec	0.061	0.099	0.069	0.057
SDNE	0.066	0.149	0.054	0.051
TANE-TS	0.095	0.245	0.078	0.075

flexibility in exploring node neighborhoods. The learned representations can be used for link prediction using vector based similarity measures.

SDNE (Wang et al. 2016): is designed to preserve non-linear connectivity patterns in networks while generating embeddings. SDNE attempts to capture the first order and second order proximity between nodes and deploys a deep belief network for generating representations.

DynGem (Goyal et al. 2018): It is an extension of SDNE to dynamic networks. It dynamically expands the deep neural architecture and learn stable embeddings from a series of network snapshots by preserving the embeddings generated at each time step. The embeddings generated from complete training is used for link prediction.

Time series of node Similarities (TS-Sim) (Güneş et al. 2016): This work first computes node similarities using common neighbor, Adamic-Adar and Jaccard Coefficient. A time series is constructed using node similarities and the similarity of future links is predicted using ARIMA model.

6.2 Performance

The experiments conducted for evaluating the performance of the proposed system are threefold. We first present the performance improvement of TANE-TS over static network embedding methods followed by dynamic embedding methods. Further we compare the system with time series prediction based approaches. Links with top 20% highly predicted scores are considered as predicted links. We conduct experiments by both considering only the new links in the final snapshot and also for all links in the final snapshot. A link (u, v) is said to be a new link in the current snapshot if it is not present in the previous snapshot.

Figure 4 compares the AUC scores of TANE-TS against the static network embedding methods (Node2vec and SDNE) w.r.t. four datasets. Results show that the proposed method gives an AUC performance improvement of 8.1%, 10.3%, 14.4% and 18.1% over the baseline methods (SDNE) on Enron, Haggle, Hep-ph, and Radoslaw respectively. A similar improvement of can be also obtained on the MAP score for the given datasets, which is presented in Table 2. As the link prediction problem is directly related to evolution of the networks, we can observe that embedding the network by considering its evolution (proposed system) have good advantage over static network embedding methods.

Table 3 plots the AUC and MAP comparison of TANE-TS with dynamic network embedding method (DynGem). The results show that the TANE algorithm when combined with time series forecasting can achieve a performance improvement of 1.3%, 2.2%, 1.4% and 2.7% on AUC and 10.4%, 21.8%, 23.8% and 7.1% on MAP compared to dyngem w.r.t. Enron, Haggle, Hep-ph, Radoslaw respectively. In DynGem, the final embeddings generated after training

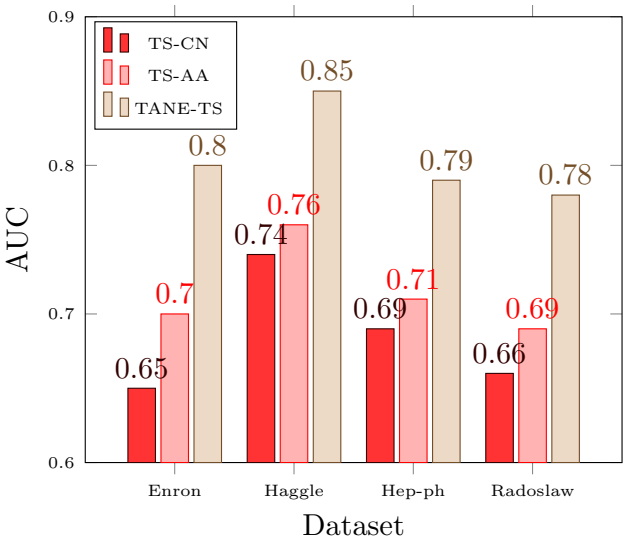

Fig. 5 AUC comparison of TANE-TS with time series prediction methods

Table 4 MAP comparison of proposed system with time series prediction methods

Method	Enron	Haggle	Hep-ph	Radoslaw
TS-CN	0.049	0.121	0.061	0.057
TS-AA	0.073	0.194	0.065	0.064
TANE-TS	0.095	0.245	0.078	0.075

Table 3 AUC and MAP comparison of proposed system with dynamic embedding method

Method	Enron		Haggle		Hep-ph		Radoslaw	
	AUC	MAP	AUC	MAP	AUC	MAP	AUC	MAP
DynGem	0.791	0.086	0.834	0.201	0.780	0.063	0.761	0.070
TANE-TS	0.802	0.095	0.853	0.245	0.791	0.078	0.782	0.075

Table 5 Comparison of proposed system with baseline methods by considering all links in the final snapshot

Method	Enron		Haggle		Hep-ph		Radoslaw	
	AUC	MAP	AUC	MAP	AUC	MAP	AUC	MAP
SDNE	0.735	0.055	0.773	0.149	0.757	0.044	0.743	0.081
TS-AA	0.791	0.077	0.842	0.182	0.801	0.088	0.794	0.183
DynGem	0.842	0.098	0.857	0.274	0.825	0.133	0.832	0.294
TANE-TS	0.881	0.126	0.932	0.471	0.851	0.156	0.869	0.382

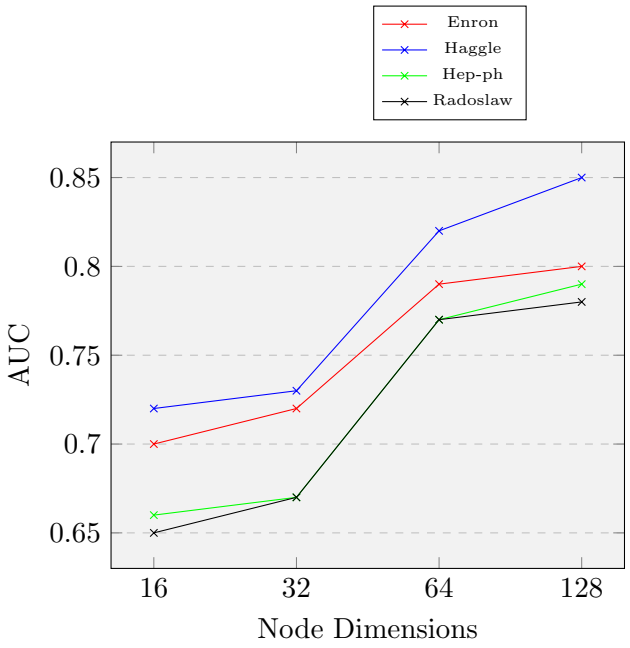

Fig. 6 Node dimensionality vs AUC

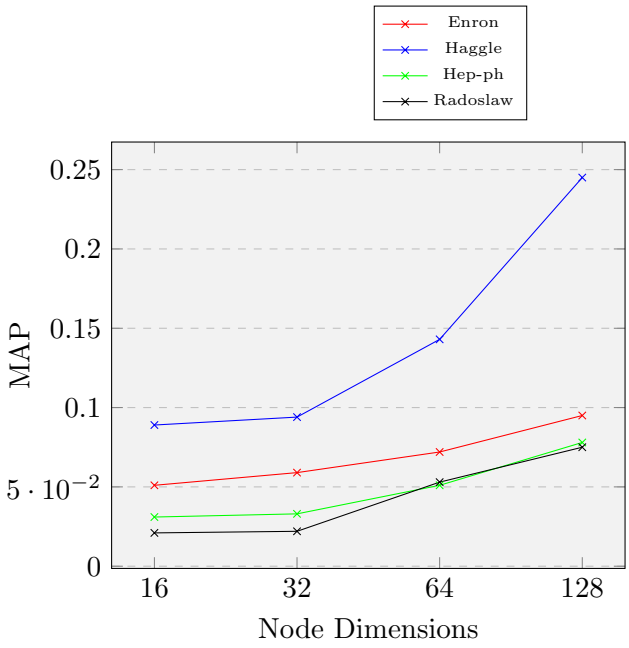

Fig. 7 Node dimensionality vs MAP

the deep model with all network snapshots are used in link prediction. In the proposed system, we are trying to learn the changes in the consecutive network snapshots and uses a time series forecasting model which can slightly improve the performance of link prediction task.

Figure 5 and Table 4 shows the AUC and MAP comparison of TANE-TS with time series forecasting based link prediction methods. Results show that TANE-TS outperforms TS-AA by 14.2%, 11.8%, 11.2% and 13.04% on AUC score and 30.1%, 26.2%, 20% and 17.1% on MAP score w.r.t. Enron, Haggle, Hep-ph, Radoslaw respectively. The network embedding method (proposed system) can more effectively capture the topological properties of the graph when compared to local similarity measures like common neighbors and adamic-adar which is used in TS-CN and TS-AA. Hence it can provide improved performance over these baselines.

The previously observed results on evaluating the performance of the proposed framework in terms of the AUC and MAP was computed by considering only the new links in the last snapshot. Now we present the AUC and MAP comparison of TANE-TS with baseline methods by considering

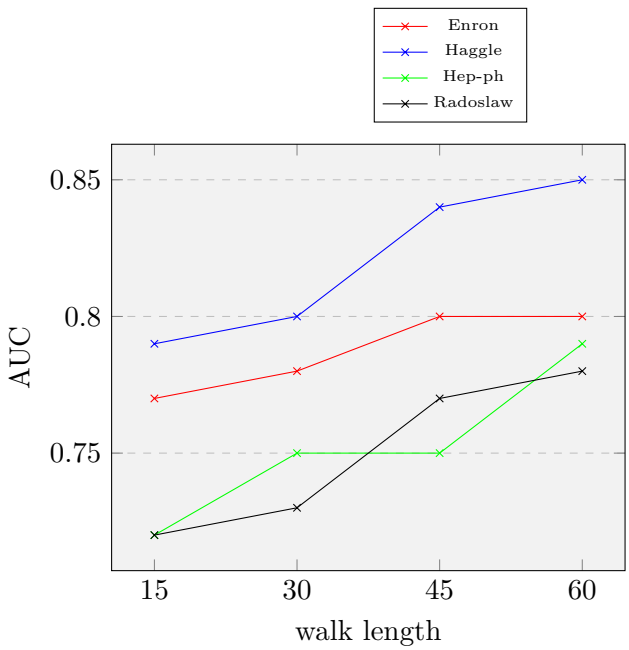

Fig. 8 Walk length vs AUC

all links in the final snapshot. Results from Table 5 reports the performance improvement of the proposed system over baseline methods.

6.3 Parameter sensitivity

In this section we investigate how the parameter tuning process affect the link prediction task. The various parameters include the dimensionality of the embeddings,the length of the random walk, the number of snapshots, the granularity of snapshots and the parameters of the ARIMA model.

We first discuss the effect of embedding dimension on the prediction results for various datasets. The AUC and MAP scores obtained by considering all new links in the final snapshot for different dimensions for different datasets are shown in Figs. 6 and 7 respectively. Results show that each vertex when represented as a higher dimensional vector (either 64 or 128) provides better AUC and MAP scores compared to low dimensional vector representations (16 and 32). AUC score for Enron, Hep-ph and Radoslaw datasets remains relatively close to 0.80 and 0.79 and 0.78 respectively when d = 64 and d = 128. For Haggle dataset,

Table 6 Effect of granularity of snapshots

Time-interval	AUC	MAP
6 months	0.621	0.051
3 months	0.671	0.066
2 months	0.782	0.081
1 month	0.802	0.095

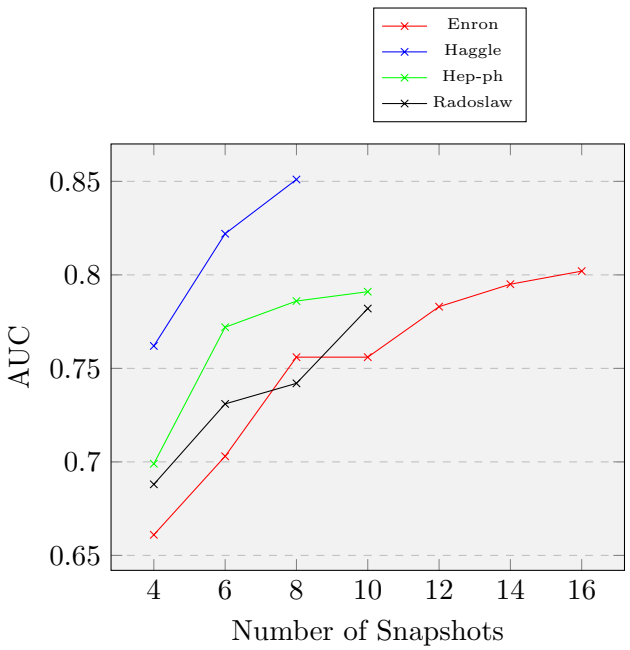

Fig. 9 No. of snapshots vs AUC

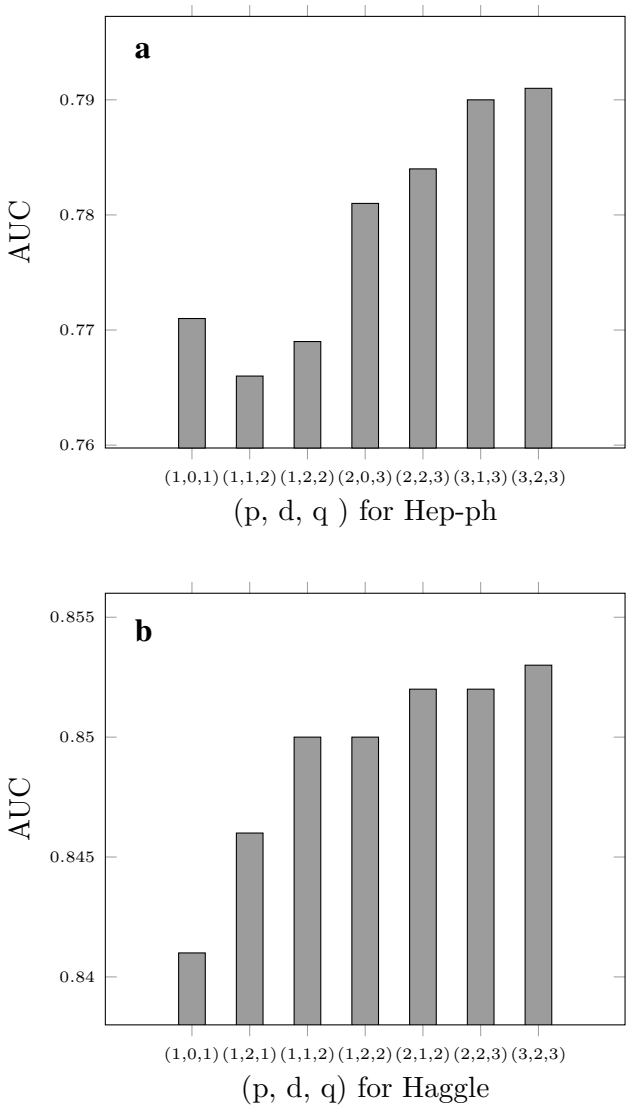

Fig. 10 **a** (p,d,q) values vs AUC. **b** (p,d,q) values vs AUC

AUC and MAP scores show some variations over various node dimensions and give an AUC and MAP score of .85 and .245 respectively when nodes are represented as 128 dimensional vectors.

The walk length of the random walk is one among the parameter which decides the extent to which the community structure of the network is explored. We have varied the walk length from 15 to 60 and the results are reported in Fig. 8. We can observe that for the smaller datasets (Enron and Haggle), a walk length of 45 can produce nearly optimal results. For bigger datasets, particulary for Hep-ph, a lengthy walk is needed to well capture the community structure and to generate optimum result for the proposed system.

We varied the number of snapshots from 5 to 10, conducted experiments on AUC and the results are presented

in Fig 9. It was found that for Enron and Hep-ph, the AUC increases up to 14 and 8 snapshots respectively and then reach an almost steady state. As the number of snapshots becomes high, the networks become dense and may lead to the saturation of AUC value. For haggle and radoslaw, an optimum AUC value of 0.85 and 0.78 respectively is obtained when the embeddings are generated by training 8 and 10 snapshots respectively.

Dynamics of real-world networks may differ in the range from small time-scales (fine-grained dynamics) to long time periods (coarse-grained dynamics). We conducted experiments on enron dataset by generating snapshots at various levels of granularity, and the results are shown in Table 6. The results show that the system gives better prediction accuracy while taking small time scales for generating network representations.

Finally we preset the influence of ARIMA model parameters on the performance of the proposed system. The parameter here we consider are the number of past time periods (p), the number of non-seasonal difference operations (d), and the number of lagged forecast errors (q). The AUC values against the different combinations of p,d,q values for Hep-ph and Haggle are shown in Fig. 10 a and b respectively. For hep-ph, 2 or 3 past values are required to build a robust model, where as for haggle dataset, the system can achieve good performance with 1 or 2 past values. We may conclude that more powerful time series model is required to get accurate predictions when we deal with more complex datasets like Hep-ph.

7 Conclusion and future works

Predicting the future links of a network by considering its evolving nature is an exciting direction of research in network mining. As network embedding emerged as an important technique to improve the performance of many network mining tasks, we investigate the effect of network embedding in link prediction on dynamic networks. We propose a method which combines time-aware network embedding and time series forecasting to perform link prediction on dynamic networks. The method uses a modified skip-gram architecture to generate time-aware node embeddings, constructs a time series of node embedding similarities and uses ARIMA model to predict the similarity scores between future links. We observe that the dynamic network embedding can well capture the temporal dynamics of the network, and the ARIMA model can predict future links with good accuracy. The benefits of the proposed are justified by conducting experiments with various real-world network datasets.

The current study can provide various directions for future work. Network topology will not change much between two consecutive snapshots and implementing better transfer learning techniques for neural network training can improve the scalability of the proposed system. The evolution of a network can be modelled as a temporal graph with time-stamped edges rather than a series of snapshots, and embedding such networks can further address the scalability issues. Combining node embedding similarity features with neighbourhood based similarity features to form a multivariate time series may further improve the prediction accuracy. System performance may be further improved by using advanced neural models like long short-term memory (LSTM) or gated recurrent units (GRU) for time series forecasting. In future, we also plan to perform link prediction in dynamic networks using graph neural network models like graph convolutional networks and graph attention networks.

Funding Not applicable.

Compliance with ethical standards

Conflict of interest The authors declare that there are no known conflicts of interest associated with this work and there has been no significant financial support or funding for this work that could have influenced its outcome.

References

Ahmed NM, Chen L, Wang Y, Li B, Li Y, Liu W (2018) DeepEye: link prediction in dynamic networks based on non-negative matrix factorization. Big Data Min Anal 1(1):19–33

Al Hasan M, Zaki MJ (2011) A survey of link prediction in social networks. In: Social network data analytics, Springer, pp 243–275

Belkin M, Niyogi P (2002) Laplacian eigenmaps and spectral techniques for embedding and clustering. Adv Neural Inf Process Syst 14:585–591

Blonder B, Wey TW, Dornhaus A, James R, Sih A (2012) Temporal dynamics and network analysis. Methods Ecol Evol 3(6):958–972

Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, Springer, pp 177–186

Brockwell Peter J, Davis RA (2016) Introduction to time series and forecasting. Springer, Berlin

Cai H, Zheng Vincent W, Chang K (2018) A comprehensive survey of graph embedding: problems, techniques and applications. IEEE Trans Knowl Data Eng 30(9):1616–1637

Cao S, Lu W, Xu Q (2015) Grarep: learning graph representations with global structural information. In: Proceedings of the 24th ACM international on conference on information and knowledge management, pp 891–900

Casteigts A, Flocchini P, Quattrociocchi W, Santoro N (2012) Time-varying graphs and dynamic networks. Int J Parallel Emerg Distrib Syst 27(5):387–408

Chaintreau A, Hui P, Crowcroft J, Diot C, Gass R, Scott J (2007) Impact of human mobility on opportunistic forwarding algorithms. IEEE Trans Mobile Comput 6:606–620

Chang S, Han W, Tang J, Qi G-J, Aggarwal CC, Huang TS (2015) Heterogeneous network embedding via deep architectures. In:

Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 119–128
Chung F, Zhao W (2010) PageRank and random walks on graphs. In: Fete of combinatorics and computer science, Springer, pp 43–62
Dong Y, Chawla NV, Swami A (2017) metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 135–144
Fu T-y, Lee W-C, Lei Z (2017) HIN2Vec: explore meta-paths in heterogeneous information networks for representation learning. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp 1797–1806
Gehrke J, Ginsparg P, Kleinberg J (2003) Overview of the 2003 KDD cup. Acm SIGKDD Explor Newslett 5(2):149–151
Goldberg Y, Levy O (2014) word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722
Goyal P, Ferrara E (2017) Graph embedding techniques, applications, and performance: a survey. Knowl-Based Syst 151:78–94
Goyal P, Kamra N, He X, Liu Y (2018) DynGEM: deep embedding method for dynamic graphs. arXiv preprint arXiv:1805.11273
Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp 855–864
Güneş İ, Gündüz-Öğüdücü Ş, Çataltepe Z (2016) Link prediction using time series of neighborhood-based node similarity scores. Data Min Knowl Discov 30(1):147–180
Gupta C, Jain A, Tayal DK, Castillo O (2018) ClusFuDE: forecasting low dimensional numerical data using an improved method based on automatic clustering, fuzzy relationships and differential evolution. Eng Appl Artif Intell 71:175–189
Jo H-H, Hiraoka T (2019) Bursty time series analysis for temporal networks. In: Temporal Network Theory, Springer, pp 161–179
Klimt B, Yang Y (2004) The enron corpus: a new dataset for email classification research. In: European conference on machine learning, Springer, pp 217–226
Li T, Jiawei Zhang SY, Philip YZ, Yan Y (2018) Deep dynamic network embedding for link prediction. IEEE Access 6:29219–29230
Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58(7):1019–1031
Liu Z, Zhang Q-M, Lü L, Zhou T (2011) Link prediction in complex networks: a local naïve Bayes model. EPL Europhys Lett 96(4):48007
Lü L, Zhou T (2011) Link prediction in complex networks: a survey. Phys A Stat Mech Appl 390(6):1150–1170
Ma X, Sun P, Qin G (2017) Nonnegative matrix factorization algorithms for link prediction in temporal networks using graph communicability. Pattern Recognit 71:361–374
Martínez V, Berzal F, Cubero J-C (2017) A survey of link prediction in complex networks. ACM Comput Surv 49(4):69
Michalski R, Palus S, Kazienko P (2011) Matching organizational structure and social network extracted from email communication. In: International conference on business information systems, Springer, pp 197–206
Mikolov T, Chen K, Corrado G, Dean J (2013a) Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781
Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013b) Distributed representations of words and phrases and their compositionality. Adv Neural Inf Process Syst 26:3111–3119
Morin F, Bengio Y (2005) Hierarchical probabilistic neural network language model. In: Aistats, vol 5, pp 246–252
Ou M, Cui P, Pei J, Zhang Z, Zhu W (2016) Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1105–1114
Özcan A, Öğüdücü ŞG (2016) Temporal link prediction using time series of quasi-local node similarity measures. In: 2016 15th IEEE international conference on machine learning and applications (ICMLA), pp 381–386
Özcan A, Öğüdücü ŞG (2017) Supervised temporal link prediction using time series of similarity measures. In: 2017 Ninth international conference on ubiquitous and future networks (ICUFN), pp 519–521
Pan S, Jia W, Zhu X, Zhang C, Wang Y (2016) Tri-party deep network representation. Network 11(9):12
Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 701–710
Pons P, Latapy M (2005) Computing communities in large networks using random walks. In: International symposium on computer and information sciences, Springer, pp 284–293
Ribeiro LFR, Saverese PHP, Figueiredo DR (2017) struc2vec: Learning node representations from structural identity. In: Proceedings of the 23rd ACM sigkdd international conference on knowledge discovery and data mining, pp 385–394
Rossi RA, Zhou R, Ahmed NK (2017) Deep feature learning for graphs. arXiv preprint arXiv:1704.08829
Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500):2323–2326
Soto J, Melin P, Castillo O (2018) A new approach for time series prediction using ensembles of IT2FNN models with optimization of fuzzy integrators. Int J Fuzzy Syst 20(3):701–728
Soto J, Castillo O, Melin P, Pedrycz W (2019) A new approach to multiple time series prediction using mimo fuzzy aggregation models with modular neural networks. Int J Fuzzy Syst 21(5):1629–1648
Spielman DA, Teng S-H (2004) Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In: Proceedings of the thirty-sixth annual ACM symposium on theory of computing, pp 81–90
Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1225–1234
Wu T, Cheng-Shang C, Wanjiun L (2018) Tracking network evolution and their applications in structural network analysis. IEEE Trans Netw Sci Eng 6(3):562–575
Yang C, Liu Z, Zhao D, Sun M, Chang EY (2015) Network representation learning with rich text information. In: IJCAI, pp 2111–2117
Yasami Y, Safaei F (2018) A novel multilayer model for missing link prediction and future link forecasting in dynamic complex networks. Phys A Stat Mech Appl 492:2166–2197
Zhou L, Yang Y, Ren X, Wu F, Zhuang Y (2018) Dynamic network embedding by modeling triadic closure process. In: Proceedings of the 32nd AAAI conference on artificial intelligence, pp 571–578
Zhu L, Guo D, Yin J, Steeg GV, Galstyan A (2016) Scalable temporal latent space inference for link prediction in dynamic social networks. IEEE Trans Knowl Data Eng 28(10):2765–2777
Zou Y, Donner RV, Marwan N, Donges JF, Kurths J (2019) Complex network approaches to nonlinear time series analysis. Phys Rep 787:1–97