Dynamic Network Embeddings: From Random Walks to *Temporal* Random Walks

Giang H. Nguyen, John Boaz Lee
Worcester Polytechnic Institute
{*ghnguyen,jtlee*}*@wpi.edu*

Ryan A. Rossi
Adobe Research
rrossi@adobe.com

Nesreen K. Ahmed
Intel Labs
nesreen.k.ahmed@intel.com

Eunyee Koh, Sungchul Kim
Adobe Research
{*eunyee,sukim*}*@adobe.com*

***Abstract*—Networks evolve continuously over time with the addition, deletion, and changing of links and nodes. Although many networks contain this type of temporal information, the majority of research in network representation learning has focused on static snapshots of the graph and has largely ignored the temporal dynamics of the network. In this work, we describe a general framework for incorporating temporal information into network embedding methods. The framework gives rise to methods for learning time-respecting embeddings from continuous-time dynamic networks. Overall, the experiments demonstrate the effectiveness of the proposed framework and dynamic network embedding approach as it achieves an average gain of 11.9% across all methods and graphs. The results indicate that modeling temporal dependencies in graphs is important for learning appropriate and meaningful network representations.**

***Keywords*-Dynamic network embeddings, temporal node embeddings, dynamic networks, network representation learning, temporal random walks, continuous-time dynamic networks, graph stream, feature learning, temporal networks**

I. Introduction

The majority of real-world networks are naturally dynamic—evolving over time with the addition, deletion, and changing of nodes and links. The temporal information in these networks is known to be important to accurately model, predict, and understand network data [1], [2]. Despite the importance of these dynamics, the majority of previous work on embedding methods have ignored the temporal information in network data [3], [4], [5], [6], [7], [8], [9], [10], [11], [12].

We address the problem of learning an appropriate network representation from *continuous-time dynamic networks* (Figure 1) for improving the accuracy of predictive models. We propose *continuous-time dynamic network embeddings* (CTDNE) and describe a general framework for learning such embeddings based on the notion of *temporal random walks* (walks that respect time). The framework is general with many interchangeable components and can be used in a straightforward fashion for incorporating temporal dependencies into existing node embedding and deep graph models that use random walks. Most importantly, the CTDNEs are learned from temporal random walks that represent actual *temporally valid sequences* of node interactions and thus avoids the issues and information loss that arises when time is ignored [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] or approximated as a sequence of discrete static snapshot graphs [13], [14], [15], [16], [17] (Figure 2) as done in previous work. The result is a more appropriate time-dependent network representation that captures the important temporal properties of the continuous-time dynamic network at the finest most natural temporal granularity without loss of information while using walks that are temporally valid (as opposed to walks that do not obey time and thus are invalid and noisy as they represent sequences that are impossible with respect to time). Hence, the framework allows existing embedding methods to be easily adapted for learning more appropriate network representations from continuous-time dynamic networks by ensuring time is respected and avoiding impossible sequences of events.

The proposed approach learns a more appropriate network representation from continuous-time dynamic networks that captures the important temporal dependencies of the network at the finest most natural granularity (*e.g.*, at a time scale of seconds or milliseconds). This is in contrast to approximating the dynamic network as a sequence of static snapshot graphs $G_1, \ldots, G_t$ where each static snapshot graph represents all edges that occur between a user-specified discrete-time interval (*e.g.*, day or week) [18], [19], [20]. Besides the obvious loss of information, there are many other issues such as selecting an appropriate aggregation granularity which is known to be an important and challenging problem in itself that can lead to poor predictive performance or misleading results. In addition, our approach naturally supports learning in *graph streams* where edges arrive continuously over time (*e.g.*, every second/millisecond) [21], [22], [23], [24] and therefore can be used for a variety of applications requring real-time performance [25], [26], [27].

We demonstrate the effectiveness of the proposed framework and generalized dynamic network embedding method for temporal link prediction in several real-world networks from a variety of application domains. Overall, the proposed method achieves an average gain of 11.9% across all methods and graphs. The results indicate that modeling temporal dependencies in graphs is important for learning appropriate and meaningful network representations. In addition, any existing embedding method or deep graph model that uses random walks can benefit from the proposed framework (*e.g.*, [3], [4], [8], [11], [9], [28], [10], [12]) as it serves as a basis for incorporating important temporal dependencies into existing methods. Methods generalized by the framework are

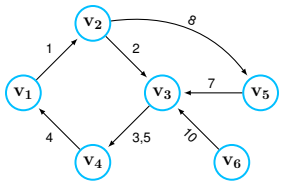

Figure 1. Dynamic network. Edges are labeled by time. Observe that existing methods that ignore time would consider $v_4 \longrightarrow v_1 \longrightarrow v_2$ a *valid* walk, however, $v_4 \longrightarrow v_1 \longrightarrow v_2$ is clearly *invalid with respect to time* since $v_1 \longrightarrow v_2$ exists in the past with respect to $v_4 \longrightarrow v_1$. In this work, we propose the notion of *temporal random walks* for embeddings that capture the *true temporally valid* behavior in networks.

able to learn more meaningful and accurate time-dependent network embeddings that capture important properties from continuous-time dynamic networks.

Previous embedding methods and deep graph models that use random walks search over the space of random walks $\mathbb{S}$ on G, whereas the class of models (continuous-time dynamic network embeddings) proposed in this work learn temporal embeddings by searching over the space $\mathbb{S}_T$ of temporal random walks that obey time and thus $\mathbb{S}_T$ includes only *temporally valid walks*. Informally, a *temporal walk* S_t from node v_{i_1} to node $v_{i_{L+1}}$ is defined as a sequence of edges $\{(v_{i_1}, v_{i_2}, t_{i_1}), (v_{i_2}, v_{i_3}, t_{i_2}), \ldots, (v_{i_L}, v_{i_{L+1}}, t_{i_L})\}$ such that $t_{i_1} \leq t_{i_2} \leq \ldots \leq t_{i_L}$. A temporal walk represents a *temporally valid* sequence of edges traversed in increasing order of edge times and therefore respect time. For instance, suppose each edge represents a contact (*e.g.*, email, phone call, proximity) between two entities, then a temporal random walk represents a feasible route for a piece of information through the dynamic network. It is straightforward to see that existing methods that ignore time learn embeddings from a set of random walks that are not actually possible when time is respected and thus represent invalid sequences of events.

The sequence that links (events) occur in a network carries important information, *e.g.*, if the event (link) represents an email communication sent from one user to another, then the state of the user who receives the email message changes in response to the email communication. For instance, suppose we have two emails $e_i = (v_1, v_2)$ from v_1 to v_2 and $e_j = (v_2, v_3)$ from v_2 to v_3; and let $\mathcal{T}(v_1, v_2)$ be the time of an email $e_i = (v_1, v_2)$. If $\mathcal{T}(v_1, v_2) < \mathcal{T}(v_2, v_3)$ then the message $e_j = (v_2, v_3)$ may reflect the information received from the email communication $e_i = (v_1, v_2)$. However, if $\mathcal{T}(v_1, v_2) > \mathcal{T}(v_2, v_3)$ then the message $e_j = (v_2, v_3)$ cannot contain any information communicated in the email $e_i = (v_1, v_2)$. This is just one simple example illustrating the importance of modeling the actual sequence of events (email communications). Embedding methods that ignore time are prone to many issues such as learning inappropriate node embeddings that do not accurately capture the dynamics in the network such as the real-world interactions or flow of information among nodes. An example of information loss that occurs when time is ignored or the actual dynamic network is approximated using a sequence of discrete static snapshot graphs is shown in Figure 1 and 2, respectively.

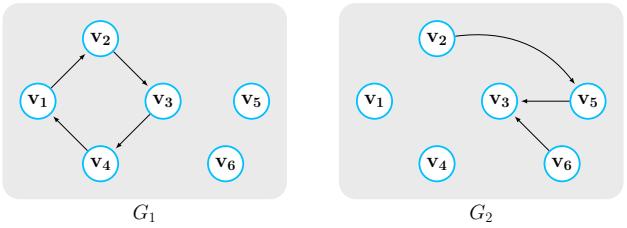

Figure 2. Noise and information loss occurs when the true dynamic network (Figure 1) is approximated as a sequence of discrete static snapshot graphs $G_1, \ldots, G_t$ using a user-defined aggregation time-scale s (temporal granularity). Suppose the dynamic network in Figure 1 is used and $s = 5$, then G_1 includes all edges in the time-interval $[1, 5]$ whereas G_2 includes all edges in $[6, 10]$ and so on. Notice that in the static snapshot graph G_1 the walk $v_4 \longrightarrow v_1 \longrightarrow v_2$ is still possible despite it being *invalid* while the perfectly valid temporal walk $v_1 \longrightarrow v_2 \longrightarrow v_5$ is impossible. Both cases are captured correctly without any loss using the proposed notion of *temporal walk* on the actual dynamic network.

The proposed approach has the following desired properties:

- **General & Unifying Framework**: We present a general framework for incorporating temporal dependencies in node embedding and deep graph models that use random walks.
- **Temporally Valid**: Embeddings are learned based on the proposed notion of *temporal random walks* that captures the *temporally valid interactions* (*e.g.*, flow of information, spread of diseases) in the dynamic network in a lossless fashion.
- **Continuous-Time Dynamic Network Embeddings**: Learns a time-dependent network representation for continuous-time dynamic networks. The approach avoids the issues and information loss that arise when time is ignored or the dynamic network is approximated as a sequence of discrete static snapshot graphs.
- **Effectiveness**: The proposed approach is shown to be effective for learning dynamic network representations. We achieve an average gain in AUC of 11.9% across all methods and graphs from various application domains.

II. Framework

This section introduces *continuous-time dynamic network embeddings* (CTDNE) and describes a general framework for deriving them based on the notion of *temporal random walks*.

A. Temporal Model

In this work, instead of approximating the dynamic network as a sequence of discrete static snapshot graphs defined as $G_1, \ldots, G_T$ where $G_i = (V, E_t)$ and E_t are the edges active between the timespan $[t_{i-1}, t_i]$, we model the *temporal interactions* in a lossless fashion as a *continuous-time dynamic network* (CTDN) defined formally as:

Definition 1 (Continuous-Time Dynamic Network) *Given a graph $G = (V, E_T, \mathcal{T})$, let V be a set of vertices, and*

$E_T \subseteq V \times V \times \mathbb{R}^+$ be the set of temporal edges between vertices in V, and $\mathcal{T} : E \rightarrow \mathbb{R}^+$ is a function that maps each edge to a corresponding timestamp. At the finest granularity, each edge $e_i = (u, v, t) \in E_T$ may be assigned a unique time $t \in \mathbb{R}^+$.

In continuous-time dynamic networks (*i.e.*, temporal networks), events denoted by edges occur over a time span $\mathcal{T} \subseteq \mathbb{T}$ where $\mathbb{T}$ is the temporal domain. For continuous-time systems $\mathbb{T} = \mathbb{R}^+$. In such networks, a *valid* walk is defined as a sequence of nodes connected by edges with non-decreasing timestamps. In other words, if each edge captures the time of contact between two entities, then a (valid temporal) walk may represent a feasible route for a piece of information. More formally,

DEFINITION 2 (TEMPORAL WALK) *A temporal walk from v_1 to v_k in G is a sequence of vertices $\langle v_1, v_2, \cdots, v_k \rangle$ such that $\langle v_i, v_{i+1} \rangle \in E_T$ for $1 \leq i < k$, and $\mathcal{T}(v_i, v_{i+1}) \leq \mathcal{T}(v_{i+1}, v_{i+2})$ for $1 \leq i < (k-1)$. For two arbitrary vertices u, $v \in V$, we say that u is temporally connected to v if there exists a temporal walk from u to v.*

The definition of temporal walk echoes the standard definition of a walk in static graphs but with an additional constraint that requires the walk to respect time, that is, edges must be traversed in increasing order of edge times. As such, temporal walks are naturally asymmetric. A *temporally invalid walk* is a walk that does not respect time. Any method that uses temporally invalid walks or approximates the dynamic network as a sequence of static snapshot graphs is said to have *temporal loss*.

We define the problem of learning continuous-time dynamic network embeddings (CTDNEs) for continuous-time dynamic networks as follows:

DEFINITION 3 (CONTINUOUS-TIME DYNAMIC NETWORK EMBEDDING) *Given a dynamic network $G = (V, E_T, \mathcal{T})$, the goal is to learn a function $f : V \rightarrow \mathbb{R}^D$ that maps nodes in the continuous-time dynamic network G to D-dimensional time-dependent embeddings using only data that is temporally valid (*e.g.*, temporal walks defined in Definition 2).*

Unlike previous work that ignores time or *approximates* the dynamic network as a sequence of discrete static snapshot graphs $G_1, \ldots, G_t$, CTDNEs proposed in this work are learned from temporal random walks that capture the true temporal interactions (*e.g.*, flow of information, spread of diseases, etc.) in the dynamic network in a lossless fashion.

The proposed continuous-time dynamic network embedding framework has two main interchangeable components that allow the user to *temporally bias* the learning of time-dependent network representations. We describe each component in Section II-B and II-C. In particular, the CTDNE framework generates *biased (and unbiased) temporal random walks* from CTDNs in Section II-B-II-C that are then used in Section II-D for deriving time-dependent embeddings that are learned from temporally valid node sequences that capture in a lossless fashion the actual flow of information or spread of disease in a network.

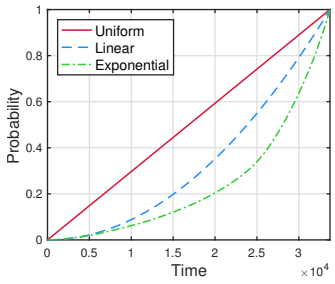

Figure 3. Example initial edge selection cumulative probability distributions (CPDs) for each of the variants investigated (uniform, linear, and exponential). Observe that exponential biases the selection of the initial edge towards those occurring more recently than in the past, whereas linear lies between exponential and uniform.

B. Initial Temporal Edge Selection

This section describes approaches to temporally bias the temporal random walks by selecting the initial temporal edge to begin the temporal random walk. In general, each temporal walk starts from a temporal edge $e_i \in E_T$ at time $t = \mathcal{T}$ selected from a distribution $\mathbb{F}_s$. The distribution used to select the initial temporal edge can either be uniform in which case there is no bias or the selection can be temporally biased using an arbitrary weighted (non-uniform) distribution for $\mathbb{F}_s$. For instance, to learn node embeddings for the temporal link prediction task, we may want to begin more temporal walks from edges closer to the current time point as the events/relationships in the distant past may be less predictive or indicative of the state of the system now. Selecting the initial temporal edge in an unbiased fashion is discussed in Section II-B1 whereas strategies that temporally bias the selection of the initial edge are discussed in Section II-B2.

1) Unbiased: In the case of initial edge selection, each edge $e_i = (v, u, t) \in E_T$ has the same probability of being selected:

$$\mathbb{P}(e) = 1/|E_T| \tag{1}$$

This corresponds to selecting the initial temporal edge using a uniform distribution.

2) Biased: We describe two techniques to temporally bias the selection of the initial edge that determines the start of the temporal random walk. In particular, we select the initial temporal edge using a temporally weighted distribution based on exponential and linear functions. However, the proposed continuous-time dynamic network embedding framework is flexible with many interchangeable components and therefore can easily support other temporally weighted distributions for selecting the initial temporal edge.

Exponential: We can also bias initial edge selection using an exponential distribution, in which case each edge $e \in E_T$ is assigned the probability:

$$\mathbb{P}(e) = \frac{\exp\left[\mathcal{T}(e) - t_{\min}\right]}{\sum_{e' \in E_T} \exp\left[\mathcal{T}(e') - t_{\min}\right]} \tag{2}$$

where $t_{\min}$ is the minimum time associated with an edge in the dynamic graph. This defines a distribution that heavily favors edges appearing later in time.

Linear: When the time difference between two time-wise consecutive edges is large, it can sometimes be helpful to map the edges to discrete time steps. Let $\eta : E_T \to \mathbb{Z}^+$ be a function that sorts (in ascending order by time) the edges in the graph. In other words η maps each edge to an index with $\eta(e) = 1$ for the earliest edge e. In this case, each edge $e \in \eta(E_T)$ will be assigned the probability:

$$\mathbb{P}(e) = \frac{\eta(e)}{\sum_{e' \in E_T} \eta(e')} \tag{3}$$

See Figure 3 for examples of the uniform, linear, and exponential variants.

C. Temporal Random Walks

After selecting the initial edge $e_i = (u, v, t)$ at time t to begin the temporal random walk (Section II-B) using $\mathbb{F}_s$, how can we perform a temporal random walk starting from that edge? We define the set of temporal neighbors of a node v at time t as follows:

DEFINITION 4 (TEMPORAL NEIGHBORHOOD) *The set of temporal neighbors of a node v at time t denoted as $\Gamma_t(v)$ are:*

$$\Gamma_t(v) = \left\{(w, t') \mid e = (v, w, t') \in E_T \wedge \mathcal{T}(e) > t\right\} \tag{4}$$

Observe that the same neighbor w can appear multiple times in $\Gamma_t(v)$ since multiple temporal edges can exist between the same pair of nodes. See Figure 4 for an example. The next node in a temporal random walk can then be chosen from the set $\Gamma_t(v)$. Here we use a second distribution $\mathbb{F}_\Gamma$ to *temporally bias* the neighbor selection. Again, this distribution can either be uniform, in which case no bias is applied, or more intuitively biased to consider time. For instance, we may want to bias the sampling strategy towards walks that exhibit smaller "in-between" time for consecutive edges. That is, for each consecutive pair of edges (u, v, t), and $(v, w, t + k)$ in the random walk, we want k to be small. For temporal link prediction on a dynamic social network, restricting the "in-between" time allows us to sample walks that do not group friends from different time periods together. As an example, if k is small we are likely to sample the random walk sequence $(v_1, v_2, t), (v_2, v_3, t+k)$ which makes sense as v_1 and v_3 are more likely to know each other since v_2 has interacted with them both recently. On the other hand, if k is large we are unlikely to sample the sequence. This helps to separate people that v_2 interacted with during very different time periods (*e.g.* high-school and graduate school) as they are less likely to know each other.

1) Unbiased: For unbiased temporal neighbor selection, given an arbitrary edge $e = (u, v, t)$, each temporal neighbor $w \in \Gamma_t(v)$ of node v at time t has the following probability of being selected:

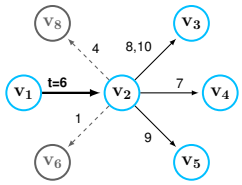

Figure 4. Temporal neighborhood of a node v_2 at time $t = 6$ denoted as $\Gamma_t(v_2)$. Notice that $\Gamma_t(v_2) = \{v_4, v_3, v_5, v_3\}$ is an ordered multiset where the temporal neighbors are sorted in ascending order by time with the nodes more recent appearing first. Moreover, the same node can appear multiple times (*e.g.*, a user sends another user multiple emails, or an association/event occurs multiple times between the same entities). This is in contrast to the definition of neighborhood used by previous work that is not parameterized by time, *e.g.*, $\Gamma(v_2) = \{v_3, v_4, v_5, v_6, v_8\}$ or $\Gamma(v_2) = \{v_3, v_3, v_4, v_5, v_6, v_8\}$ if multigraphs are supported.

$$\mathbb{P}(w) = 1/|\Gamma_t(v)| \tag{5}$$

2) Biased: We describe two techniques to bias the temporal random walks by sampling the next node in a temporal walk via temporally weighted distributions based on exponential and linear functions. However, the continuous-time dynamic network embedding framework is flexible and can easily be used with other application or domain-dependent *temporal bias functions*.

Exponential: When exponential decay is used, we formulate the probability as follows. Given an arbitrary edge $e = (u, v, t)$, each temporal neighbor $w \in \Gamma_t(v)$ has the following probability of being selected:

$$\mathbb{P}(w) = \frac{\exp\left[\tau(w) - \tau(v)\right]}{\sum_{w' \in \Gamma_t(v)} \exp\left[\tau(w') - \tau(v)\right]} \tag{6}$$

Note that we abuse the notation slightly here and use τ to mean the mapping to the corresponding time. This is similar to the exponentially decaying probability of consecutive contacts observed in the spread of computer viruses and worms [29].

Linear: Here, we define $\delta : V \times \mathbb{R}^+ \to \mathbb{Z}^+$ as a function which sorts temporal neighbors in descending order time-wise. The probability of each temporal neighbor $w \in \Gamma_t(v)$ of node v at time t is then defined as:

$$\mathbb{P}(w) = \frac{\delta(w)}{\sum_{w' \in \Gamma_t(v)} \delta(w')} \tag{7}$$

This distribution biases the selection towards edges that are closer in time to the current node.

3) Temporal Context Windows: Since temporal walks preserve time, it is possible for a walk to run out of *temporally valid* edges to traverse. Therefore, we do not impose a strict length on the temporal random walks. Instead, we simply require each temporal walk to have a minimum length ω (in this work, ω is equivalent to the context window size for skip-gram [30]). A maximum length L can be provided to accommodate longer walks. A temporal walk $\mathcal{S}_{t_i}$ with length

$|\mathcal{S}_{t_i}|$ is considered valid iff

$$\omega \leq |\mathcal{S}_{t_i}| \leq L$$

Given a set of temporal random walks $\{\mathcal{S}_{t_1}, \mathcal{S}_{t_2}, \cdots, \mathcal{S}_{t_k}\}$, we define the temporal context window count β as the total number of context windows of size ω that can be derived from the set of temporal random walks. Formally, this can be written as:

$$\beta = \sum_{i=1}^{k} \big(|\mathcal{S}_{t_i}| - \omega + 1\big) \tag{8}$$

When deriving a set of temporal walks, we typically set β to be a multiple of $N = |V|$.

D. Learning Time-preserving Embeddings

Given a temporal walk $\mathcal{S}_t$, we can now formulate the task of learning time-preserving node embeddings in a CTDN as the optimization problem:

$$\max_{f} \ \log \mathbb{P}\big(\, W_T = \{v_{i-\omega}, \cdots, v_{i+\omega}\} \setminus v_i \mid f(v_i)\big) \tag{9}$$

where $f : V \rightarrow \mathbb{R}^D$ is the node embedding function, ω is the context window size for optimization, and

$$W_T = \{v_{i-\omega}, \cdots, v_{i+\omega}\}$$

such that

$$\mathcal{T}(v_{i-\omega}, v_{i-\omega+1}) < \cdots < \mathcal{T}(v_{i+\omega-1}, v_{i+\omega})$$

is an arbitrary temporal context window $W_T \subseteq S_t$. For tractability, we assume conditional independence between the nodes of a temporal context window when observed with respect to the source node v_i. That is:

$$\mathbb{P}\big(\, W_T | f(v_i)\big) = \prod_{v_{i+k} \in W_T} \mathbb{P}\big(v_{i+k} | f(v_i)\big) \tag{10}$$

We can model the conditional likelihood of every source-nearby node pair (v_i, v_j) as a softmax unit parameterized by a dot product of their feature vectors:

$$\mathbb{P}\big(\, v_j | f(v_i)\big) = \frac{\exp\big[f(v_j) \cdot f(v_i)\big]}{\sum_{v_k \in V} \exp\big[f(v_k) \cdot f(v_i)\big]} \tag{11}$$

Using Eq. 10- 11, the optimization problem in Eq. 9 reduces to:

$$\max_{f} \ \sum_{v_i \in V} \left(-\log Z_i + \sum_{v_j \in W_T} f(v_j) \cdot f(v_i) \right) \tag{12}$$

where the term $Z_i = \sum_{v_j \in V} \exp\big[f(v_i) \cdot f(v_j)\big]$ can be approximated by negative sampling. Given a graph G, let $\mathbb{S}$ be the space of all possible random walks on G and let $\mathbb{S}_T$ be the space of all temporal random walks on G. It is straightforward to see that the space of temporal random walks $\mathbb{S}_T$ is contained within $\mathbb{S}$, and $\mathbb{S}_T$ represents only a tiny fraction of possible random walks in $\mathbb{S}$. Existing methods sample a set of random walks $\mathcal{S}$ from $\mathbb{S}$ whereas this work focuses on sampling a set of *temporal random walks* $\mathcal{S}_t$ from $\mathbb{S}_T \subseteq \mathbb{S}$. In general, the probability of an existing method sampling a temporal random walk from $\mathbb{S}$ by chance is extremely small and therefore the vast majority of random walks sampled by these methods represent sequences of events between nodes that are invalid (not possible) when time is respected. For instance, suppose each edge represents an interaction/event (*e.g.*, email, phone call, spatial proximity) between two people, then a temporal random walk may represent a feasible route for a piece of information through the dynamic network or a temporally valid pathway for the spread of an infectious disease.

We summarize the procedure to learn time-preserving embeddings for CTDNs in Algorithm 1. Our procedure in Algorithm 1 generalizes the Skip-Gram architecture to learn continuous-time dynamic network embeddings (CTDNEs). However, the framework can easily be used for other deep graph models that leverage random walks (*e.g.*, [12]) as the temporal walks can serve as input vectors for neural networks. There are many methods that can be adapted to learn CTDN embeddings using *temporal random walks* (*e.g.*,

Algorithm 1 Continuous-Time Dynamic Network Embeddings

Input:
a (un)weighted and (un)directed dynamic network $G = (V, \mathrm{E}_T, \mathcal{T})$,
temporal context window count β, context window size ω,
embedding dimensions D

Set maximum walk length $L = 80$
Initialize set of *temporal walks* $\mathcal{S}_T$ to $\emptyset$
Initialize number of temporal context windows $C = 0$
Precompute sampling distribution $\mathbb{F}_s$ using G
    $\mathbb{F}_s \in \{\text{Uniform, Exponential, Linear}\}$
$G' = (V, E_T, \mathcal{T}, \mathbb{F}_s)$
while $\beta - C > 0$ **do**
    Sample an edge $e_* = (v, u)$ via distribution $\mathbb{F}_s$
    $t = \mathcal{T}(e_*)$
    $S_t = \textsc{TemporalWalk}(G', e_* = (v, u), t, L, \omega + \beta - C - 1)$
    if $|S_t| > \omega$ **then**
        Add the *temporal walk* S_t to $\mathcal{S}_T$
        $C = C + (|S_t| - \omega + 1)$
end while
$\mathbf{Z} = \textsc{StochasticGradientDescent}(\omega, D, \mathcal{S}_T)$
return the *dynamic* node embedding matrix $\mathbf{Z}$

Algorithm 2 Temporal Random Walk

procedure $\textsc{TemporalWalk}(G', e = (s, r), t, L, C)$
    Initialize temporal walk $S_t = \big[\, s, r \,\big]$
    Set $i = r$ ▷ current node
    for $p = 1$ **to** $\min(L, C) - 1$ **do**
        $\Gamma_t(i) = \big\{(w, t') \mid e = (i, w, t') \in E_T \ \wedge \mathcal{T}(i) > t\big\}$
        if $|\Gamma_t(i)| > 0$ **then**
            Select node j from distribution $\mathbb{F}_\Gamma(\Gamma_t(i))$
            Append j to S_t
            Set $t = \mathcal{T}(i, j)$
            Set $i = j$
        else terminate temporal walk
    return temporal walk S_t of length $|S_t|$ rooted at node s

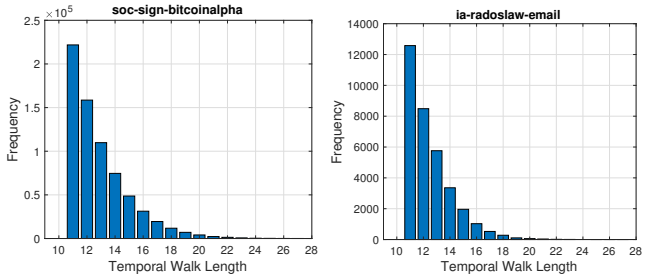

Figure 5. Frequency of *temporal random walks* by length

node2vec [4], struc2vec [8], role2vec [31]) and the proposed framework is not tied to any particular approach.

E. Hyperparameters

While other methods have a lot of hyperparameters that require tuning such as node2vec [4], the proposed framework has a single hyperparameter that requires tuning.

1) Arbitrary length walk: In our work, we allow temporal walks to have arbitrary lengths which we simply restrict to be between the range $[\omega, L]$. We argue that arbitrary-sized walks between ω and L allow more accurate representations of node behaviors. For instance, a walk starting at u can return to u after traversing L edges, showing a closed community. On the other hand, another walk starting from v can end immediately at minimum length ω without ever going back to v. These are two distant cases that would be misrepresented if a fixed random walk length is imposed. Observe that L can be set to any reasonably large value such as $L = 80$ as it serves as an upper bound on the temporal random walk length, which is of course bounded by the dynamic network (unlike random walks which can be infinite). However, we set $\omega = 10$ and found other reasonable values to not impact performance on most graphs.

2) Exponential base: Suppose the exponential function is used to bias the temporal random walk (Eq. 6) or bias the selection of the initial edge to begin the temporal walk (Eq. 2), then we allow the user to choose the base b of the exponential function for the exponential distribution.

F. Model variants

The proposed *continuous-time dynamic network embedding* (CTDNE) framework has two main interchangeable components that give rise to a variety of useful models. In this section, we discuss a few of the variants we investigate in Section IV. Recall that we use a distribution $\mathbb{F}_s$ to select the starting edge e_* of a temporal random walk (Section II-B) and another distribution $\mathbb{F}_\Gamma$ to bias the selection of each subsequent edge in a temporal random walk (Section II-C). Thus, different distributions $\mathbb{F}_s$ and $\mathbb{F}_\Gamma$ can be used to bias the temporal random walk sampling strategy. In this work, we investigated three different approaches for $\mathbb{F}_s$ and $\mathbb{F}_\Gamma$ giving rise to nine different CTDNE variants by taking all possible combinations of unbiased and biased distributions discussed in Section II-B and Section II-C.

III. Analysis

Let $N = |V|$ denote the number of nodes, $M = |E_T|$ be the number of edges, $D =$ dimensionality of the embedding, $R =$ the number of temporal walks per node, $L =$ the maximum length of a temporal random walk, and $\Delta =$ the maximum degree of a node. Recall that while R is not required, we use it here since the number of temporal random walks $|\mathcal{S}_T|$ is a multiple of the number of nodes $N = |V|$ and thus can be written as RN similar to previous work.

A. Time Complexity

Lemma 1 *The time complexity for learning CTDNEs using the generalized Skip-gram architecture in Section II-D is*

$$\mathcal{O}(M + N(R \log M + RL\Delta + D)) \tag{13}$$

and the time complexity for learning CTDNEs with unbiased *temporal random walks (uniform) is:*

$$\mathcal{O}(N(R \log M + RL \log \Delta + D)) \tag{14}$$

Proof. The time complexity of each of the three steps is provided below. We assume the exponential variant is used for both $\mathbb{F}_s$ and $\mathbb{F}_\Gamma$ since this CTDNE variant is the most computationally expensive among the nine CTDNE variants expressed from using uniform, linear, or exponential for $\mathbb{F}_s$ and $\mathbb{F}_\Gamma$. WLOG we assume E_T is sorted in increasing order of time such that $\mathcal{T}(e_1) \leq \mathcal{T}(e_2) \leq \cdots \leq \mathcal{T}(e_M)$. Similarly, the neighbors of each node are also ordered in increasing order of time.

Initial Temporal Edge Selection: To derive $\mathbb{F}_s$ for any of the variants used in this work (uniform, linear, exponential) it takes $\mathcal{O}(M)$ time since each variant can be computed with a single or at most two passes over the edges. Selecting an initial edge via $\mathbb{F}_s$ takes $\mathcal{O}(\log M)$ time. Now $\mathbb{F}_s$ is used to select the initial edge for each temporal random walk $S_t \in \mathcal{S}_T$ and thus an initial edge is selected $RN = |\mathcal{S}_T|$ times. This gives a total time complexity of $\mathcal{O}(M + RN \log M)$.[1]

Temporal Random Walks: After the initial edge is selected, the next step is to select the next temporally valid neighbor from the set of temporal neighbors $\Gamma_t(v)$ of a given node v at time t using a (weighted) distribution $\mathbb{F}_\Gamma$ (*e.g.*, uniform, linear, exponential). Note $\mathbb{F}_\Gamma$ must be computed and maintained for each node. Given a node v and a time t_* associated with the previous edge traversal in the temporal random walk, the first step in any variant (uniform, linear, exponential; Section II-C) is to obtain the ordered set of temporal neighbors $\Gamma_t(v) \subseteq \Gamma(v)$ of node v that occur after t_*. Since the set of all temporal neighbors is already stored and ordered by time, we only need to find the index of the neighbor $w \in \Gamma(v)$ with time $t > t_*$ as this gives us $\Gamma_t(v)$. Therefore, $\Gamma_t(v)$ is derived in $\log |\Gamma(v)|$ via a binary search over the ordered set $\Gamma(v)$. In the worst case, $\mathcal{O}(\log \Delta)$

[1]Note for uniform initial edge selection, the time complexity is linear in the number of temporal random walks $\mathcal{O}(RN)$.

where $\Delta = \max_{v \in V} |\Gamma(v)|$ is the maximum degree. After obtaining $\Gamma_t(v) \subseteq \Gamma(v)$, we derive $\mathbb{F}_\Gamma$ in $\mathcal{O}(\Delta)$ time when $d_v = \Delta$. Now, selecting the next temporally valid neighbor according to $\mathbb{F}_\Gamma$ takes $\mathcal{O}(\log \Delta)$ for exponential and linear and $o(1)$ for uniform. For the uniform variant, we select the next temporally valid neighbor in $o(1)$ constant time by $j \sim \text{UniformDiscrete}\{1, 2, \ldots, |\Gamma_t(v)|\}$ and then obtain the selected temporal neighbor by directly indexing into $\Gamma_t(v)$. Therefore, the time complexity to select the next node in a biased temporal random walk is $\mathcal{O}(\log \Delta + \Delta) = \mathcal{O}(\Delta)$ in the worst case and $\mathcal{O}(\log \Delta)$ for unbiased (uniform).

For a temporal random walk of length L, the time complexity is $\mathcal{O}(L\Delta)$ for a biased walk with linear/exponential and $\mathcal{O}(L \log \Delta)$ for an unbiased walk. Therefore, the time complexity for RN biased temporal random walks of length L is $\mathcal{O}(RNL\Delta)$ in the worst case and $\mathcal{O}(RNL \log \Delta)$ for unbiased.

Learning time-dependent embeddings: For the Skip-Gram-based generalization given in Section II-D, the time complexity per iteration of Stochastic Gradient Descent (SGD) is $\mathcal{O}(ND)$ where $D \ll N$. While the time complexity of a single iteration of SGD is less than a single iteration of Alternating Least Squares (ALS) [32], SGD requires more iterations to obtain a good enough model and is sensitive to the choice of learning rate [33], [34]. Moreover, SGD is more challenging to parallelize compared to ALS [32] or Cyclic Coordinate Descent (CCD) [35], [36]. Nevertheless, the choice of optimization scheme depends on the objective function of the embedding method generalized via the CTDNE framework.

B. Space Complexity

Storing the F_s distribution takes $\mathcal{O}(M)$ space. The temporal neighborhoods do not require any additional space (as we simply store an index). Storing $\mathbb{F}_\Gamma$ takes $\mathcal{O}(\Delta)$ (which can be reused for each node in the temporal random walk). The embedding matrix $\mathbf{Z}$ takes $\mathcal{O}(ND)$ space. Therefore, the space complexity of CTDNEs is $\mathcal{O}(M + ND + \Delta) = \mathcal{O}(M + ND)$.

IV. Experiments

The experiments are designed to investigate the effectiveness of the proposed *continuous-time dynamic network embeddings* (CTDNE) framework for prediction. To ensure the results and findings of this work are significant and meaningful, we investigate a wide range of temporal networks from a variety of application domains with fundamentally different structural and temporal characteristics. All networks investigated are continuous-time dynamic networks with $\mathbb{T} = \mathbb{R}^+$. For these dynamic networks, the time scale of the edges is at the level of seconds or milliseconds, *i.e.*, the edge timestamps record the time an edge occurred at the level of seconds or milliseconds (finest granularity given as input). Our approach uses the finest time scale available in the graph data as input. All data is from NetworkRepository [37] and is easily accessible for reproducibility.

A. Experimental setup

Since this work is the first to learn embeddings over a CTDN, there are no methods that are directly comparable. Nevertheless, we evaluate the framework presented in Section II for learning continuous-time dynamic network representations by first comparing CTDNE against a number of recent embedding methods including node2vec [4], DeepWalk [3], and LINE [5]. For node2vec, we use the same hyperparameters ($D = 128$, $R = 10$, $L = 80$, $\omega = 10$) and grid search over $p, q \in \{0.25, 0.50, 1, 2, 4\}$ as mentioned in [4]. We use the same hyperparameters for DeepWalk but with $p = q = 1$ as it is a special case of node2vec. As for the CTDNE methods, we use $\omega = 10$ and $D = 128$. For LINE, we also use $D = 128$ with 2nd-order-proximity and number of samples $T = 60$ million.

Table I
AUC scores for Temporal Link Prediction.

Data	DeepWalk	Node2Vec	LINE	CTDNE	(Gain)
ia-contact	0.845	0.874	0.736	**0.913**	(+10.37%)
ia-hypertext09	0.620	0.641	0.621	**0.671**	(+6.51%)
ia-enron-employees	0.719	0.759	0.550	**0.777**	(+13.00%)
ia-radoslaw-email	0.734	0.741	0.615	**0.811**	(+14.83%)
ia-email-eu	0.820	0.860	0.650	**0.890**	(+12.73%)
fb-forum	0.670	0.790	0.640	**0.826**	(+15.25%)
soc-bitcoinA	0.840	0.870	0.670	**0.891**	(+10.96%)
soc-wiki-elec	0.820	0.840	0.620	**0.857**	(+11.32%)

B. Comparison

We evaluate the performance of the proposed framework on the temporal link prediction task. To generate a set of labeled examples for link prediction, we first sort the edges in each graph by time (ascending) and use the first 75% for representation learning. The remaining 25% are considered as positive links and we sample an equal number of negative edges randomly. We take care to ensure edges in the training set do not appear in the test set and vice-versa. We perform link prediction on this labeled data $\mathcal{X}$ of positive and negative edges. After the embeddings are learned for each node, we derive edge feature vectors by combining the learned embedding vectors of the corresponding nodes. More formally, given embedding vectors $\mathbf{z}_i$ and $\mathbf{z}_j$ for node i and j, we derive an edge embedding vector $\mathbf{z}_{ij} \in \mathbb{R}^D$ as:

$$\mathbf{z}_{ij} = \Phi(\mathbf{z}_i, \mathbf{z}_j) \tag{15}$$

where

$$\Phi \in \left\{ (\mathbf{z}_i + \mathbf{z}_j)/2, \;\; \mathbf{z}_i \odot \mathbf{z}_j, \;\; |\mathbf{z}_i - \mathbf{z}_j|, \;\; (\mathbf{z}_i - \mathbf{z}_j)^{\circ 2} \right\} \tag{16}$$

and $\mathbf{z}_i \odot \mathbf{z}_j$ is the element-wise (Hadamard) product and $\mathbf{z}^{\circ 2}$ is the Hadamard power. We use logistic regression (LR) with hold-out validation of 25%. Experiments are repeated for 10

random seed initializations and the average performance is reported. Unless otherwise mentioned, we use ROC AUC (denoted as AUC for short) to evaluate the models and use the same number of dimensions D for all models.

For fair comparison, we ensure the same amount of information is used for learning by all baseline methods. In particular, the number of *temporal context windows* denoted by β is as follows:

$$\beta = R \times N \times (L - \omega + 1) \tag{17}$$

where R denotes the number of walks for each node and L is the length of a random walk required by the baseline methods. Recall that R and L are *not* required by CTDNE and are only used above to ensure that all methods use exactly the same amount of information for evaluation purposes. Note since CTDNE does not collect a fixed amount of random walks (of a fixed length) for each node as done by many other embedding methods [3], [4], instead the user simply specifies the # of temporal context windows (expected) per node and the total number of temporal context windows β is derived as a multiple of the number of nodes $N = |V|$. Hence, CTDNE is also easier to use as it requires a lot less hyperparameters that must be carefully tuned by the user.

Table I shows the performance of all the compared methods on the temporal link prediction task. For this experiment, we use the simplest CTDNE variant from the proposed framework and did not apply any *additional bias* to the selection strategy. In other words, both $\mathbb{F}_s$ and $\mathbb{F}_\Gamma$ are set to the uniform distribution. We note, however, that since our temporal walks are time-obeying (by Definition 2), the selection is already biased towards edges that appear later in time as the random walk traversal does not go back in time. Here we see that the proposed approach performs consistently better than DeepWalk, node2vec, and LINE. This is an indication that important information is lost when temporal information is ignored. Strikingly, our model does not leverage the bias introduced by node2vec [4], and yet it still outperforms this model by a significant margin. We could have generalized node2vec in a similar manner using the proposed framework from Section II. Obviously, we can expect to achieve even better predictive performance by using the CTDNE framework to derive a continuous-time node2vec generalization by replacing the notion of random walks in node2vec with the notion of *temporal random walks* biased by the (weighted) distributions $\mathbb{F}_s$ (Section II-B) and $\mathbb{F}_\Gamma$ (Section II-C).

The last column of Table I provides the mean gain in ROC-AUC averaged over all embedding methods for each dynamic network. In all cases, the proposed approach significantly outperforms the other embedding methods across all dynamic networks. Notably, we achieve an overall gain in AUC of 11.9% across all embedding methods and graphs. These results indicate that modeling and incorporating the temporal dependencies in graphs is important for learning appropriate and meaningful network representations. It is also worth noting that many other approaches that leverage random walks can also be generalized using the proposed framework [8], [11], [9], [28], [12], as well as any future state-of-the-art embedding method.

We also find that using a biased distribution (*e.g.*, linear or exponential) improves predictive performance in terms of AUC when compared to the uniform distribution on many graphs. For others however, there is no noticeable gain in performance. This can likely be attributed to the fact that most of the dynamic networks investigated have a relatively short time span (more than 3 years at most). Table II provides results for a few other variants from the framework. In particular, Table II shows the difference in AUC when applying a biased distribution to the initial edge selection strategy $\mathbb{F}_s$ as well as the temporal neighbor selection $\mathbb{F}_\Gamma$ for the temporal random walk. Interestingly, using a biased distribution for $\mathbb{F}_s$ seems to improve more on the tested datasets. However, for ia-enron-employees, the best result can be observed when both distributions are biased.

We investigate the difference between discrete-time models that learn embeddings from a sequence of discrete snapshot graphs, and the class of continuous-time models proposed in this work. A discrete-time dynamic network embedding (DTDNE) is defined as any embedding derived from a sequence of discrete static snapshot graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_t\}$. This includes any embedding learned from temporally smoothed static graphs or any representation derived from the initial sequence of discrete static snapshot graphs. All existing work to date for temporal networks have focused on DTDNE methods as opposed to the class of CTDNE methods proposed in this work. Notice that DTDNE methods use *approximations* of the actual dynamic network whereas CTDN embeddings do not and leverage the actual valid temporal information without any temporal loss. In this experiment, we create discrete snapshot graphs and learn embeddings for each one using the previous approaches. As an example, suppose we have a sequence of $T = 4$ snapshot

Table II
RESULTS FOR DIFFERENT CTDNE VARIANTS
$\mathbb{F}_s$ is the distribution for initial edge sampling and $\mathbb{F}_\Gamma$ is the distribution for temporal neighbor sampling.

Variant					
$\mathbb{F}_s$	$\mathbb{F}_\Gamma$	contact	hyper	enron	rado
Unif (Eq. 1)	**Unif** (Eq. 5)	0.913	0.671	0.777	0.811
Unif (Eq. 1)	**Lin** (Eq. 7)	0.903	0.665	0.769	0.797
Lin (Eq. 3)	**Unif** (Eq. 5)	0.915	0.675	0.773	0.818
Lin (Eq. 3)	**Lin** (Eq. 7)	0.903	0.667	0.782	0.806
Exp (Eq. 2)	**Exp** (Eq. 6)	**0.921**	0.681	**0.800**	0.820
Unif (Eq. 1)	**Exp** (Eq. 6)	0.913	0.670	0.759	0.803
Exp (Eq. 2)	**Unif** (Eq. 5)	0.920	**0.718**	0.786	**0.827**
Lin (Eq. 3)	**Exp** (Eq. 6)	0.916	0.681	0.782	0.823
Exp (Eq. 2)	**Lin** (Eq. 7)	0.914	0.675	0.747	0.817

Table III
RESULTS COMPARING DTNDES TO CTDNES (AUC)

DATA	**DTDNE**	**CTDNE**	(GAIN)
ia-contact	0.843	**0.913**	(+8.30%)
ia-hypertext09	0.612	**0.671**	(+9.64%)
ia-enron-employees	0.721	**0.777**	(+7.76%)
ia-radoslaw-email	0.785	**0.811**	(+3.31%)

graphs where each graph represents a day of activity and further suppose $D = 128$. For each snapshot graph, we learn a (D/T)-dimensional embedding and concatenate them all to obtain a D-dimensional embedding and then evaluate the embedding for link prediction as described previously.

One problem common with DTDNE methods is how to handle nodes that are not active in a given static snapshot graph G_i (*i.e.*, the node has no edges that occur in G_i). In such situations, we set the node embedding for that static snapshot graph to all zeros. However, we also investigated using the node embedding from the last active snapshot graph as well as setting the embedding of an inactive node to be the mean embedding of the active nodes in the given snapshot graph and observed similar results. More importantly, unlike DTDNE methods that have many issues and heuristics required to handle them (*e.g.*, the time-scale, how to handle inactive nodes, etc), CTDNEs do not. This is yet again another advantage of the CTDN embeddings introduced in this work. Results are provided in Table III. Since node2vec always performs the best among the baseline methods (Table I), we use it as a basis for the DTDN embeddings. For each graph, we carefully selected an appropriate time-scale for the DTDNE methods. Overall, the proposed CTDNEs perform better than the DTDNEs as shown in Table III. Note that CTDNE in Table III corresponds to using uniform for both $\mathbb{F}_s$ and $\mathbb{F}_\Gamma$. Obviously, better results can be achieved by learning $\mathbb{F}_s$ and $\mathbb{F}_\Gamma$ automatically as shown in Table II. The gain in AUC for each graph is shown in the rightmost column in Table III. The mean gain in AUC of CTDNE compared to DTDNE over all graphs is 7.25%.

V. RELATED WORK

The node embedding problem has received considerable attention from the research community in recent years. The goal is to learn encodings (embeddings, representations, features) that capture key properties about each node such as their role in the graph based on their structural characteristics (*i.e.*, roles capture distinct structural properties, *e.g.*, hub nodes, bridge nodes, near-cliques) [38] or community (*i.e.*, communities represent groups of nodes that are close together in the graph based on proximity, cohesive/tightly connected nodes) [39], [40]. Since nodes that share similar roles (based on structural properties) or communities (based on proximity, cohesiveness) are grouped close to each other in the embedding space, one can easily use the learned embeddings for tasks such as ranking [41], community detection [39], [40], role embeddings [38], [42], link prediction [43], and node classification [18].

Many of the techniques that were initially proposed for solving the node embedding problem were based on graph factorization [44], [45], [6]. More recently, the skip-gram model [30] was introduced in the natural language processing domain to learn vector representations for words. Inspired by skip-gram's success in language modeling, various methods [3], [4], [5] have been proposed to learn node embeddings using skip-gram by treating a graph as a "document." Two of the more notable methods, DeepWalk [3] and node2vec [4], use random walks to sample an ordered sequence of nodes from a graph. The skip-gram model can then be applied to these sequences to learn node embeddings.

Researchers have also tackled the problem of node embedding in more complex graphs, including attributed networks [9], heterogeneous networks [28] and dynamic networks [13], [46], [47]. However, the majority of the work in the area still fail to consider graphs that evolve over time (*i.e.*temporal graphs). A few work have begun to explore the problem of learning node embeddings from temporal networks [13], [14], [15], [48], [16], [17]. All of these approaches *approximate* the dynamic network as a sequence of discrete static snapshot graphs, which are fundamentally different from the class of continuous-time dynamic network embedding methods introduced in this work. Notably, this work is the first to propose *temporal random walks* for embeddings as well as *CTDN embeddings* that use temporal walks to capture the actual temporally valid sequences observed in the CTDN; and thus avoids the issues and information loss that arises when embedding methods simply ignore time or use discrete static snapshot graphs (See Figure 2 for one example). Furthermore, we introduce a unifying framework that can serve as a basis for generalizing other random walk based deep learning (*e.g.*, [12]) and embedding methods (*e.g.*, [8], [4], [11], [9], [28], [49]) for learning more appropriate time-dependent embeddings from temporal networks. In contrast, previous work has simply introduced new approaches for temporal networks [14] and therefore they focus on an entirely different problem than the one in this work which is a general framework that can be leveraged by other non-temporal approaches. Other work has focused on incremental algorithms (also called dynamic algorithms) for updating spectral clustering as new information arrives [50], which is different from the problem studied in this paper. Recently, Ahmed *et al.* [51] proposed the notion of *attributed random walks* that can be used to generalize existing methods for inductive learning and/or graph-based transfer learning tasks. In future work, we will investigate combining both attributed random walks and temporal random walks to derive even more powerful embeddings from networks.

VI. Conclusion

This paper introduced the notion of *temporal random walks* for embedding methods. Using the notion of temporal random walks, we proposed the class of continuous-time dynamic network embeddings (CTDNE) along with a general, expressive, and flexible framework for computing them. Our proposed framework provides a basis for generalizing existing (or future state-of-the-art) random walk-based embedding methods for learning dynamic (time-dependent) network embeddings from continuous-time dynamic networks. The result is a more appropriate time-dependent network representation that captures the important temporal properties of the continuous-time dynamic network. By learning time-dependent embeddings based on temporal random walks, we not only avoid the issues and information loss that arises when time is ignored or modeled as a sequence of discrete static snapshot graphs, but we also ensure the embeddings are learned using only *temporally valid walks*.

References

[1] D. Watts and S. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[2] M. Newman, "The structure of scientific collaboration networks," *PNAS*, vol. 98, no. 2, pp. 404–409, 2001.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.

[4] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.

[5] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale Information Network Embedding," in *WWW*, 2015.

[6] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *CIKM*. ACM, 2015, pp. 891–900.

[7] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Deep inductive network representation learning," in *WWW BigNet*, 2018.

[8] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "Struc2vec: Learning node representations from structural identity," in *SIGKDD*, 2017.

[9] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *arXiv:1705.04969*, 2017.

[10] N. K. Ahmed, R. A. Rossi, R. Zhou, J. B. Lee, X. Kong, T. L. Willke, and H. Eldardiry, "Inductive representation learning in large attributed graphs," in *WiML NIPS*, 2017.

[11] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *CIKM*, 2017, pp. 377–386.

[12] J. B. Lee, R. A. Rossi, and X. Kong, "Graph classification using structural attention," in *SIGKDD*, 2018, pp. 1–9.

[13] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling dynamic behavior in large evolving graphs," in *WSDM*, 2013, pp. 667–676.

[14] R. Hisano, "Semi-supervised graph embedding approach to dynamic link prediction," *arXiv:1610.04351*, 2016.

[15] N. Kamra, U. Gupta, and Y. Liu, "Deep generative dual memory network for continual learning," *arXiv preprint, arXiv:1710.10368*, 2017.

[16] T. K. Saha, T. Williams, M. A. Hasan, S. Joty, and N. K. Varberg, "Models for capturing temporal smoothness in evolving networks for learning latent representation of nodes," *arXiv:1804.05816*, 2018.

[17] M. Rahman, T. K. Saha, M. A. Hasan, K. S. Xu, and C. K. Reddy, "Dylink2vec: Effective feature representation for link prediction in dynamic networks," *arXiv:1804.05755*, 2018.

[18] R. Rossi and J. Neville, "Time-evolving relational classification and ensemble methods," in *PAKDD*, 2012, p. 13.

[19] S. Soundarajan, A. Tamersoy, E. B. Khalil, T. Eliassi-Rad, D. H. Chau, B. Gallagher, and K. Roundy, "Generating graph snapshots from streaming edge data," in *WWW Companion*, 2016, pp. 109–110.

[20] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu, "Graphscope: parameter-free mining of large time-evolving graphs," in *SIGKDD*, 2007.

[21] C. C. Aggarwal, Y. Zhao, and S. Y. Philip, "Outlier detection in graph streams," in *ICDE*. IEEE, 2011, pp. 399–409.

[22] N. K. Ahmed, N. Duffield, T. L. Willke, and R. A. Rossi, "On sampling from massive graph streams," in *VLDB*, 2017, pp. 1430–1441.

[23] C. C. Aggarwal, Y. Li, P. S. Yu, and R. Jin, "On dense pattern mining in graph streams," *VLDB*, vol. 3, no. 1-2, pp. 975–984, 2010.

[24] S. Guha and A. McGregor, "Graph synopses, sketches, and streams: A survey," *VLDB*, vol. 5, no. 12, pp. 2030–2031, 2012.

[25] R. Pienta, J. Abello, M. Kahng, and D. H. Chau, "Scalable graph exploration and visualization: Sensemaking challenges and opportunities," in *BigComp*, 2015, pp. 271–278.

[26] Z. Cai, D. Logothetis, and G. Siganos, "Facilitating real-time graph mining," in *International Workshop on Cloud Data Management*, 2012.

[27] N. K. Ahmed and R. A. Rossi, "Interactive visual graph analytics on the web." in *ICWSM*, 2015, pp. 566–569.

[28] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *SIGKDD*, 2017.

[29] P. Holme and J. Saramäki, "Temporal networks," *Phy. Rep.*, 2012.

[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR Workshop*, 2013, p. 10.

[31] N. K. Ahmed, R. A. Rossi, R. Zhou, J. B. Lee, X. Kong, T. L. Willke, and H. Eldardiry, "Learning role-based graph embeddings," in *arXiv:1802.02896*, 2018.

[32] I. Pilászy, D. Zibriczky, and D. Tikk, "Fast ALS-based matrix factorization for explicit and implicit feedback datasets," in *RecSys*, 2010.

[33] H. Yun, H.-F. Yu, C.-J. Hsieh, S. Vishwanathan, and I. Dhillon, "Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion," *VLDB*, vol. 7, no. 11, pp. 975–986, 2014.

[34] J. Oh, W.-S. Han, H. Yu, and X. Jiang, "Fast and robust parallel sgd matrix factorization," in *SIGKDD*. ACM, 2015, pp. 865–874.

[35] J. Kim, Y. He, and H. Park, "Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework," *J. of Glob. Opt.*, vol. 58, no. 2, pp. 285–319, 2014.

[36] R. A. Rossi and R. Zhou, "Scalable relational learning for large heterogeneous networks," in *DSAA*, 2015, pp. 1–10.

[37] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015, pp. 4292–4293. [Online]. Available: http://networkrepository.com

[38] R. Rossi and N. Ahmed, "Role discovery in networks," *TKDE*, vol. 27, no. 4, pp. 1112–1131, 2015.

[39] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *NIPS*, 2002, pp. 849–856.

[40] P. Pons and M. Latapy, "Computing communities in large networks using random walks." *J. Graph Alg. Appl.*, vol. 10, no. 2, p. 191, 2006.

[41] L. Page, S. Brin, R. Motwani, and T. Winograd, "PageRank citation ranking: Bringing order to the web," *Stanford Tech. Report*, 1998.

[42] N. K. Ahmed, R. A. Rossi, T. L. Willke, and R. Zhou, *Edge Role Discovery via Higher-Order Structures*, 2017, pp. 291–303.

[43] W. Liu and L. Lü, "Link prediction based on local random walk," *Europhysics Letters*, vol. 89, p. 58007, 2010.

[44] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. S. Smola, "Distributed large-scale natural graph factorization," in *WWW*, 2013, pp. 37–48.

[45] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, pp. 1373–1396, 2002.

[46] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," 2018.

[47] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *CIKM*, 2017, pp. 387–396.

[48] L. Zhu, D. Guo, J. Yin, G. Ver Steeg, and A. Galstyan, "Scalable temporal latent space inference for link prediction in dynamic social networks," *TKDE*, vol. 28, no. 10, pp. 2765–2777, 2016.

[49] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1025–1035.

[50] P.-Y. Chen, B. Zhang, M. A. Hasan, and A. O. Hero, "Incremental method for spectral clustering of increasing orders," *arXiv:1512.07349*, 2015.

[51] N. K. Ahmed, R. A. Rossi, R. Zhou, J. B. Lee, X. Kong, T. L. Willke, and H. Eldardiry, "Generalizing deep learning in graphs using attributed random walks," 2017, pp. 1–8.