Continuous-Time Dynamic Network Embeddings

Giang Hoang Nguyen
Worcester Polytechnic Institute
ghnguyen@wpi.edu

John Boaz Lee
Worcester Polytechnic Institute
jtlee@wpi.edu

Ryan A. Rossi
Adobe Research
rrossi@adobe.com

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

Eunyee Koh
Adobe Research
eunyee@adobe.com

Sungchul Kim
Adobe Research
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.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Artificial intelligence**; **Logical and relational learning**; • **Mathematics of computing** → **Graph theory**; **Graph algorithms**; • **Theory of computation** → **Streaming, sublinear and near linear time algorithms**; • **Information systems** → **Data mining**;

KEYWORDS

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

1 INTRODUCTION

In recent years, network (graph/relational) data has grown at a tremendous rate; it is present in domains such as the Internet and the world-wide web [8, 11, 21], scientific citation and collaboration [43, 47], epidemiology [34, 42, 45, 51], communication analysis [60], metabolism [32, 66], ecosystems [13, 20], bioinformatics [31, 41], fraud and terrorist analysis [35, 46], and many others. The links in these network data may represent citations, friendships, associations, metabolic functions, communications, co-locations, shared mechanisms, or many other explicit or implicit relationships.

The majority of these real-world networks are naturally dynamic—evolving over time with the addition, deletion, and changing of nodes and links. The temporal information in networks is known to be important to accurately model, predict, and understand network data [47, 67]. Despite the importance of these dynamics, the majority of previous work has ignored the temporal information in network data [7, 14, 15, 26, 37, 38, 52, 56, 61, 65].

In this work, we address the problem of learning an appropriate network representation from *continuous-time dynamic networks* for improving the accuracy of predictive models. We describe a general framework for incorporating temporal dependencies into network embedding methods. The framework serves as a basis for incorporating temporal dependencies into existing node embedding and deep graph models based on random walks. The result is a more appropriate time-dependent network representation that captures the important temporal properties of the continuous-time dynamic network. 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 during the learning and therefore reduces noise by avoiding spurious or 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 representing the dynamic network as a sequence of static snapshot graphs where each static snapshot graph represents all edges that occur between a user-specified discrete-time interval (*e.g.*, day or week) [57, 59, 63, 64]. This can be seen as a very coarse and noisy approximation of the actual continuous-time dynamic network. Besides the 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) [2, 3, 5, 27] and therefore can be used for a variety of applications requring real-time performance [6, 12, 53].

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

WWW '18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3191526

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.*, [7, 15, 18, 26, 37, 38, 52, 56]) as it serves as a basis for incorporating important temporal dependencies into existing methods. Methods generalized by the framework are 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 proposed approach learns temporal embeddings by searching over the space $\mathbb{S}_T$ of temporal random walks that obey time. 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 of which the vast majority of them capture sequences of events that are invalid when considering time. In other words, many of the random walks used by these methods to derive embeddings for nodes are not actually possible when time is respected. 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. See Figure 1 for another example of information loss that occurs when time is not respected (in existing methods [7, 14, 15, 26, 37, 38, 52, 56, 61, 65]).

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 leverage random walks.
- **Continuous-Time Dynamic Networks**: Learns a time-dependent network representation for continuous-time dynamic networks. The approach avoids the issues and loss in information that arise from creating a sequence of discrete snapshot graphs from the continuous-time representation of the graph.

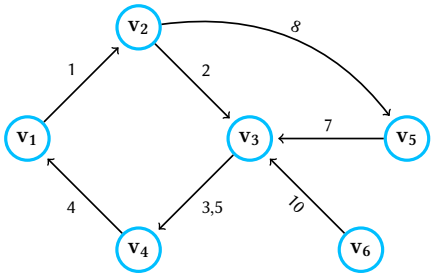

Figure 1: Dynamic network. Edges are labeled by time. Observe that v_4, v_1, v_2 is not a valid temporal walk since v_1, v_2 exists in the past with respect to v_4, v_1.

- **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.

2 FRAMEWORK

This section describes the general framework for learning time-dependent embeddings from continuous-time dynamic networks.

2.1 Temporal Model

In this work, instead of modeling the dynamic network as a sequence of discrete 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 as a *continuous-time dynamic network* (CTDN) defined formally as:

Definition 2.1 (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[1]), 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 denoted by 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 2.2 (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

[1]The terms temporal network and continuous-time dynamic network are used interchangeably.

in increasing order of edge times. As such, temporal walks are naturally asymmetric.

We now define the problem of learning time-respecting embeddings for continuous-time dynamic networks as follows: Given a temporal network $G = (V, E_T, \mathcal{T})$, the goal is to learn a function $f : V \rightarrow \mathbb{R}^D$ that maps nodes in G to D-dimensional *time-dependent feature representations* suitable for a down-stream machine learning task such as temporal link prediction. 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 now describe each component in Section 2.2 and 2.3.

2.2 Initial Temporal Edge Selection

Given a time-continuous dynamic network $G = (V, E_T, \mathcal{T})$, how can we select a node to begin a temporal random walk? Observe that most existing methods that ignore time simply perform a fixed number of random walks from each node in the graph. However, recall that 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 $1 \leq i < (k-1)$. Notice that in addition to a node v, a *temporal random walk* requires a starting time t. In time-continuous dynamic networks (Definition 2.1), every edge $e_i = (v, u) \in E_T$ is associated with a time $t = \mathcal{T}(e_i) = \mathcal{T}(v, u)$. Therefore, we can either sample an initial time t_* from a uniform or weighted distribution $\mathbb{F}_s$ and find the edge e_i closest to time t_* or select an initial edge $e_i = (v, w)$ along with its associated time $t_* = \mathcal{T}(e_i)$ by sampling from an arbitrary (uniform or weighted) distribution $\mathbb{F}_s$. The choice of where to begin the temporal random walk is used to our advantage as a way to *temporally bias* the temporal random walks and therefore improve predictive performance when the time-dependent embeddings are used on a downstream time-series regression or classification task. This is a important and fundamental difference between the proposed dynamic network embedding framework that uses temporal random walks and existing methods that use random walks on static graphs.

In general, each temporal walk starts from a temporal edge $e_i \in E_T$ at time $t = \mathcal{T}$ sampled 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 2.2.1 whereas strategies that temporally bias the selection of the initial edge are discussed in Section 2.2.2.

2.2.1 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:

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

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

2.2.2 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 sample the initial temporal edge by sampling a temporally weighted distribution based on exponential and linear functions. However, the continuous-time dynamic network embedding framework is flexible and 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:

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

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 $\eta : E_T \rightarrow \mathbb{Z}^+$ be a function that sorts (in ascending order by time) the edges in the dataset. 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:

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

2.3 Temporal Random Walk

After selecting the initial edge $e_i = (u, v, t)$ at time t to begin the temporal random walk (Section 2.2), 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 2.3 (Temporal Neighborhood). *The set of temporal neighbors of a node v at time t denoted as $\Gamma_t(v)$ are:*

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

Note that it is possible for the same neighbor w to appear in $\Gamma_t(v)$ multiple times since multiple temporal edges can exist between the same pair of nodes – for instance, two individuals may exchange multiple email messages over the course of time. 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.

2.3.1 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:

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

2.3.2 Biased. We describe two techniques to bias the temporal random walks by sampling the next node in a temporal walk via temporally weighted distribution 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:

$$\Pr(w) = \frac{\exp\big[\tau(w) - \tau(v)\big]}{\sum_{w' \in \Gamma_t(v)} \exp\big[\tau(w') - \tau(v)\big]} \tag{6}$$

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

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

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

2.3.3 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 sampled temporal 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 [44]). A maximum length L can be provided to accommodate longer walks. Hence, when generating a set of temporal walks, any temporal walk $\mathcal{S}_{t_i}$ with length $\omega \leq |\mathcal{S}_{t_i}| \leq L$ is considered valid. Given a set of temporal random walks $\{\mathcal{S}_{t_1}, \mathcal{S}_{t_2}, \cdots, \mathcal{S}_{t_k}\}$, we define a 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:

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

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

2.4 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:

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

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

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

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 very small and 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 continuous-time dynamic networks. However, the framework can easily be used for other deep graph models that leverage random walks (*e.g.*, [37]) as the temporal walks can serve as input vectors for neural networks [37].

2.5 Hyperparameters

While other methods have a lot of hyperparameters that require tuning such as node2vec [26], the proposed framework has only a single hyperparameter that requires tuning.

2.5.1 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. Regarding the sensitivity of ω and L, they do not affect the overall performance by a large margin for graphs used in our experiments. However, for much larger graphs, these values could be more data dependent and may be modified by the user.

Algorithm 1 Continuous-Time Dynamic Network Embeddings

Input:
a (un)weighted and (un)directed dynamic network $G = (V, E_T, \mathcal{T})$,
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 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 $S_t = [\, s, r \,]$
  Set $i = r$ ▷ current node
  for $p = 1$ **to** $min(L, C) - 1$ **do**
    $\Gamma_t(i) = \{(w, t') \mid e = (i, w, t') \in E_T \wedge \mathcal{T}(i) > t\}$
    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

2.5.2 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. In the case of initial temporal edge selection (Eq. 6), a large base b would cause the function to grow rapidly. Notice that if the observed temporal interactions (*e.g.* edges) in the dynamic network span a large time period, the probability of choosing one of the recent edges may be much larger than the probability to choose all other edges resulting in sampled walks that are skewed too much towards recent edges.

2.6 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 3. Recall that we use a distribution $\mathbb{F}_s$ to select the starting edge e_* of a temporal random walk (Section 2.2). Furthermore, we use another distribution $\mathbb{F}_\Gamma$ to bias the selection of each subsequent edge in a temporal random walk (Section 2.3). Thus, different distributions $\mathbb{F}_s$ and $\mathbb{F}_\Gamma$ can be used to bias the random walk sampling strategy. In particular, we investigated three different approaches to sample (1) the starting temporal edge e_* via $\mathbb{F}_s$, and (2) each subsequent edge in a temporal random walk via $\mathbb{F}_\Gamma$. This gives rise to nine different model variants by taking all possible combinations of unbiased and biased distributions discussed in Section 2.2 and Section 2.3.

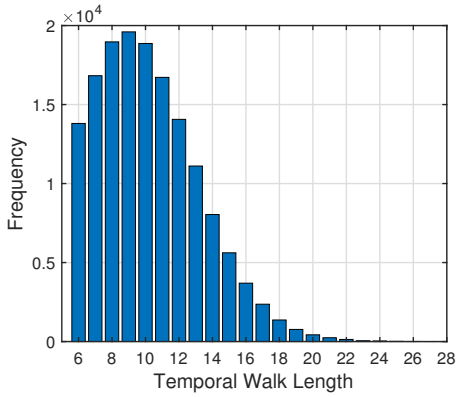

Figure 2: Frequency of *temporal random walks* by length (ia-contact).

3 EXPERIMENTS

The experiments are designed to investigate the effectiveness of the proposed *continuous-time dynamic network embedding* (CTDNE) framework using a wide range of temporal graphs with different structural and temporal characteristics from a variety of different application domains. A summary of the dynamic networks used for evaluation and their statistics are provided in Table 2. All networks 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 given as input. All data was obtained from NetworkRepository [58].

In particular, 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 perform link prediction on this labeled data $\mathcal{X}$ of positive and negative edges.

3.1 Experimental setup

We evaluate the framework presented in Section 2 for learning dynamic network representations against the following baseline methods: node2vec [26], DeepWalk [52], and LINE [65]. 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 [26]. We use the same hyperparameters for DeepWalk but with $p = q = 1$ as it is a special case of node2vec. As for our method, we

Table 1: AUC scores for Temporal Link Prediction.

Data	DeepWalk	Node2Vec	LINE	CTDNE	Gain
IA-CONTACT	0.845	0.874	0.736	**0.913**	10.369
IA-HYPERTEXT09	0.620	0.641	0.621	**0.671**	6.508
IA-ENRON-EMPLOYEES	0.719	0.759	0.550	**0.777**	12.999
IA-RADOSLAW-EMAIL	0.734	0.741	0.615	**0.811**	14.833
IA-EMAIL-EU	0.820	0.860	0.650	**0.890**	12.734
FB-FORUM	0.670	0.790	0.640	**0.826**	15.254
SOC-BITCOINA	0.840	0.870	0.670	**0.891**	10.962
SOC-WIKI-ELEC	0.820	0.840	0.620	**0.857**	11.319

use $\omega = 10$, $L = 80$, $D = 128$. For LINE, we also use $D = 128$ with 2nd-order-proximity and number of samples $T = 60$ million.

After the embeddings are learned for each node, we follow the methodology of [26] and compute the feature vector for an edge by combining the learned embedding vectors of the corresponding nodes using one of the following operations: $ops \in$ {weighted-L1, weighted-L2, average, hadamard}.

Recall that for each dataset, we generate a labeled dataset $\mathcal{X}$ for link prediction. We use logistic regression (LR) with hold-out validation of 25% on this dataset. Experiments are repeated for 10 random seed initializations and the average performance is reported. Unless otherwise mentioned, we use AUC to evaluate the models.

Table 2: Dynamic network data and statistics. Let $|E_T|$ = number of *temporal edges*; $\bar{d}$ = average temporal node degree; and $d_{\max}$ = max temporal node degree.

Dynamic Network	$\lvert V\rvert$	$\lvert E_T\rvert$	$\bar{d}$	$d_{\max}$	Timespan (days)
IA-CONTACT	274	28.2K	206.2	2092	3.97
IA-CONTACTS-HYPERTEXT09	113	20.8K	368.5	1483	2.46
IA-ENRON-EMPLOYEES	151	50.5K	669.8	5177	1137.55
IA-RADOSLAW-EMAIL	167	82.9K	993.1	9053	271.19
IA-EMAIL-EU	986	332.3K	674.1	10571	803.93
FB-FORUM	899	33.7K	75.0	1841	164.49
SOC-SIGN-BITCOINA	3.7K	24.1K	12.8	888	1901.00
SOC-WIKI-ELEC	7.1K	107K	30.1	1346	1378.34

3.2 Comparison

For fair comparison we set D to the same value for all compared methods. Furthermore, we ensure the same amount of information is used for learning by all baseline methods. In particular, the number of temporal context windows β used is

$$\beta = R \times N \times (L - \omega + 1) \tag{11}$$

where R denotes the number of walks for each node and L is the length of a random walk required by the baseline methods.

Table 1 shows the performance of all the compared methods on the temporal link prediction task. For this experiment, we use the simplest 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 2.2), the sampling 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 [26], 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 2. Obviously, we can expect to achieve even better predictive performance from the embeddings derived using the continuous-time node2vec generalization.

The last column of Table 1 provides the mean gain in 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 [15, 18, 37, 38, 56], as well as any future state-of-the-art embedding method. We also find that for many data sets, using a biased distribution (either linear or exponential) does indeed improve predictive performance in terms of AUC when compared to the uniform distribution. 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 short time span (more than 3 years at most). Table 3 provides results for a few other variants from the framework. In particular, Table 3 shows the difference of 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

Table 3: AUC scores for different variants that were tested. Note that $\mathbb{F}_s$ is the distribution used for initial edge sampling and $\mathbb{F}_\Gamma$ is the distribution for temporal neighbor sampling. Reference to formulation in paper in parentheses.

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

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.

4 RELATED WORK

The node embedding problem has received considerable attention from the research community in recent years. In this problem, a low-dimensional encoding is learned for each node in a graph. The goal is to learn encodings (*i.e.* embeddings) that capture key properties about each node such as their position in the graph, or their local neighborhood structure. Since nodes that share similar properties are grouped close to each other in the embedding space, one can easily use the learned embeddings for tasks such as ranking [50], community detection [48, 54], link prediction [39], and node classification [57].

Many of the techniques that were initially proposed for solving the node embedding problem were based on graph factorization [4, 9, 14]. More recently, the skip-gram model [44] was introduced in the natural language processing domain to learn continuous vector representations for words. Inspired by skip-gram's success in language modeling, various methods [26, 52, 65] have been proposed to learn node embeddings using skip-gram by treating a graph as a "document." Two of the more notable methods, DeepWalk [52] and node2vec [26], 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 [38], and heterogeneous networks [18]. However, the majority of the work in the area still fail to consider graphs that evolve over time (*i.e.* temporal graphs). Notably, the framework described in this work can be used to generalize these methods for learning more appropriate time-dependent embeddings since they are based on random walks.

A few work have begun to explore the problem of node embedding in temporal networks [28, 33]. However, our work differs from previous work on several key points and is, in particular, of a more general nature. While previous work have mostly been based on using discrete snapshots of temporal networks [28, 33], we propose a framework for incorporating temporal dependencies into node embeddings based on temporal random walks on a continuous representation of the temporal network. Furthermore, this work introduces a general framework that can serve as a basis for generalizing other random walk based deep learning (*e.g.*, [37]) and embedding methods (*e.g.*, [15, 18, 26, 38, 56]) for learning more appropriate time-dependent embeddings from temporal networks. In contrast, most other work has simply introduced new approaches for temporal networks [28, 33] and therefore are significantly different than the problem focused on in this work which is a general framework that can be leveraged by other non-temporal approaches to lift them for modeling time-dependent networks.

Random walks on graphs have been studied for decades [40]. The theory underlying random walks and their connection to eigenvalues and other fundamental properties of graphs are well-understood [17]. Our work is also related to uniform and non-uniform random walks on graphs [17, 40]. Random walks are at the heart of many important applications such as ranking [50], community detection [48, 54], recommendation [10], link prediction [39], influence modeling [30], search engines [36], image segmentation [25], routing in wireless sensor networks [62], and time-series forecasting [23]. These applications and techniques may also benefit from the notion of temporal random walks.

More recently, there has been significant research in developing network analysis and machine learning methods for modeling temporal networks. Temporal networks have been the focus of recent research including node classification in temporal networks [57], temporal link prediction [19], dynamic community detection [16], and dynamic mixed-membership role models [22, 59], anomaly detection in dynamic networks [55], influence modeling and online advertisement [24], finding important entities in dynamic networks [23, 49] temporal network centrality and measures [29]. For a survey on temporal network analysis, see [1, 29].

5 CONCLUSION

We have described a general framework for incorporating temporal information into network embedding methods. The framework provides a basis for generalizing existing 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. We demonstrated the effectiveness of the proposed framework and generalized dynamic network embedding method for temporal link prediction in several real-world networks. Overall, the proposed method achieves an average gain of 11.9% across all methods and graphs. Our results indicate that modeling and incorporating the temporal dependencies in graphs is important for learning appropriate and meaningful network representations. Future work will investigate using the continious-time dynamic network framework to generalize heterogeneous network embedding methods [18] and attributed network embedding methods [38], among other approaches [26, 56].

REFERENCES

[1] Charu Aggarwal and Karthik Subbian. 2014. Evolutionary network analysis: A survey. *CSUR* 47, 1 (2014), 10.

[2] Charu C Aggarwal, Yao Li, Philip S Yu, and Ruoming Jin. 2010. On dense pattern mining in graph streams. *VLDB* 3, 1-2 (2010), 975–984.

[3] Charu C Aggarwal, Yuchen Zhao, and S Yu Philip. 2011. Outlier detection in graph streams. In *ICDE*. IEEE, 399–409.

[4] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander S. Smola. 2013. Distributed large-scale natural graph factorization. In *WWW*. 37–48.

[5] Nesreen K. Ahmed, Nick Duffield, Theodore L. Willke, and Ryan A. Rossi. 2017. On Sampling from Massive Graph Streams. In *VLDB*. 1430–1441.

[6] Nesreen Kamel Ahmed and Ryan Anthony Rossi. 2015. Interactive Visual Graph Analytics on the Web.. In *ICWSM*. 566–569.

[7] Nesreen K. Ahmed, Ryan A. Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L. Willke, and Hoda Eldardiry. 2017. Inductive Representation Learning in Large Attributed Graphs. In *WiML NIPS*.

[8] R. Albert, H. Jeong, and A.L. Barabási. 1999. Internet: Diameter of the world-wide web. *Nature* 401, 6749 (1999), 130–131.

[9] Mikhail Belkin and Partha Niyogi. 2002. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation* 15 (2002), 1373–1396.

[10] Toine Bogers. 2010. Movie recommendation using random walks over the contextual graph. In *Context-Aware Recommender Systems*.

[11] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. 2000. Graph structure in the web. *Computer Networks* 33, 1-6 (2000), 309–320.

[12] Zhuhua Cai, Dionysios Logothetis, and Georgos Siganos. 2012. Facilitating real-time graph mining. In *Proceedings of the Fourth International Workshop on Cloud Data Management*. 8.

[13] J. Camacho, R. Guimerà, and L.A. Nunes Amaral. 2002. Robust patterns in food web structure. *Physical Review Letters* 88, 22 (2002), 228102: 1–4.

[14] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning graph representations with global structural information. In *CIKM*. ACM, 891–900.

[15] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. 2017. Learning community embedding with community detection and node embedding on graphs. In *CIKM*. 377–386.

[16] Rémy Cazabet and Frédéric Amblard. 2014. Dynamic community detection. In *ESNAM*. Springer, 404–414.

[17] Fan Chung. 2007. Random walks and local cuts in graphs. *Linear Algebra and its applications* 423, 1 (2007), 22–32.

[18] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *SIGKDD*. 135–144.

[19] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. 2011. Temporal link prediction using matrix and tensor factorizations. *TKDD* 5, 2 (2011), 10.

[20] J.A. Dunne, R.J. Williams, and N.D. Martinez. 2002. Food-web structure and network theory: The role of connectance and size. *Proceedings of the National Academy of Sciences of the United States of America* 99, 20 (2002), 12917.

[21] M. Faloutsos, P. Faloutsos, and C. Faloutsos. 1999. On power-law relationships of the internet topology. In *Proceedings of the ACM SIGCOMM International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. 251–262.

[22] Wenjie Fu, Le Song, and Eric P Xing. 2009. Dynamic mixed membership blockmodel for evolving networks. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 329–336.

[23] David F. Gleich and Ryan A. Rossi. 2014. A Dynamical System for PageRank with Time-Dependent Teleportation. *Internet Mathematics* (2014), 188–217.

[24] A. Goyal, F. Bonchi, and L.V.S. Lakshmanan. 2010. Learning influence probabilities in social networks. In *WSDM*. ACM, 241–250.

[25] Leo Grady. 2006. Random walks for image segmentation. *TPAMI* 28, 11 (2006), 1768–1783.

[26] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*. 855–864.

[27] Sudipto Guha and Andrew McGregor. 2012. Graph synopses, sketches, and streams: A survey. *VLDB* 5, 12 (2012), 2030–2031.

[28] Ryohei Hisano. 2016. Semi-supervised Graph Embedding Approach to Dynamic Link Prediction. *arXiv preprint arXiv:1610.04351* (2016).

[29] P. Holme and J. Saramäki. 2012. Temporal networks. *Physics Reports* (2012).

[30] Akshay Java, Pranam Kolari, Tim Finin, and Tim Oates. 2006. Modeling the spread of influence on the blogosphere. In *WWW*. 22–26.

[31] H. Jeong, S.P. Mason, A.L. Barabasi, and Z.N. Oltvai. 2001. Lethality and centrality in protein networks. *arXiv preprint cond-mat/0105306* (2001).

[32] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.L. Barabási. 2000. The large-scale organization of metabolic networks. *Nature* 407, 6804 (2000), 651–654.

[33] Nitin Kamra, Umang Gupta, and Yan Liu. 2017. Deep Generative Dual Memory Network for Continual Learning. *arXiv preprint, arXiv:1710.10368* (2017).

[34] A. Kleczkowski and B.T. Grenfell. 1999. Mean-field-type equations for spread of epidemics: The small world model. *Physica A: Statistical Mechanics and its Applications* 274, 1-2 (1999), 355–360.

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

[36] Jean-Louis Lassez, Ryan Rossi, and Kumar Jeev. 2008. Ranking Links on the Web: Search and Surf Engines. In *IEA/AIE*. 199–208.

[37] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2017. Deep Graph Attention Model. In *arXiv:1709.06075*.

[38] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2017. Attributed Social Network Embedding. *arXiv preprint arXiv:1705.04969* (2017).

[39] W. Liu and L. Lü. 2010. Link prediction based on local random walk. *Europhysics Letters* 89 (2010), 58007.

[40] László Lovász. 1993. Random walks on graphs. *Combinatorics* 2 (1993), 1–46.

[41] S. Maslov and K. Sneppen. 2002. Specificity and stability in topology of protein networks. *Science* 296, 5569 (2002), 910–913.

[42] R.M. May and A.L. Lloyd. 2001. Infection dynamics on scale-free networks. *Physical Review E* 64, 6 (2001), 66112.

[43] A. McGovern, L. Friedland, M. Hay, B. Gallagher, A. Fast, J. Neville, and D. Jensen. 2003. Exploiting Relational Structure to Understand Publication Patterns in High-Energy Physics. *SIGKDD Explorations* 5, 2 (2003), 165–172.

[44] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR Workshop*. 10.

[45] C. Moore and M.E.J. Newman. 2000. Epidemics and percolation in small-world networks. *Physical Review E* 61, 5 (2000), 5678–5682.

[46] J. Neville, O. Simsek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg. 2005. Using relational knowledge discovery to prevent securities fraud. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. 449–458.

[47] M.E.J. Newman. 2001. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences* 98, 2 (2001), 404–409.

[48] Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *NIPS*. 849–856.

[49] J. O'Madadhain, J. Hutchins, and P. Smyth. 2005. Prediction and ranking algorithms for event-based network data. *SIGKDD Explorations* 7, 2 (2005), 30.

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

[51] R. Pastor-Satorras and A. Vespignani. 2001. Epidemic spreading in scale-free networks. *Physical Review Letters* 86, 14 (2001), 3200–3203.

[52] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. 701–710.

[53] Robert Pienta, James Abello, Minsuk Kahng, and Duen Horng Chau. 2015. Scalable graph exploration and visualization: Sensemaking challenges and opportunities. In *BigComp*. 271–278.

[54] Pascal Pons and Matthieu Latapy. 2006. Computing communities in large networks using random walks. *J. Graph Alg. Appl.* 10, 2 (2006), 191–218.

[55] Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. 2015. Anomaly detection in dynamic networks: a survey. *Wiley Interdisc. Rev.: Comp. Stat.* 7, 3 (2015), 223–247.

[56] Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. 2017. Struc2Vec: Learning Node Representations from Structural Identity. In *SIGKDD*. 385–394.

[57] Ryan Rossi and Jennifer Neville. 2012. Time-evolving Relational Classification and Ensemble Methods. In *PAKDD*. 13.

[58] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. 4292–4293. http://networkrepository.com

[59] Ryan A. Rossi, Brian Gallagher, Jennifer Neville, and Keith Henderson. 2013. Modeling Dynamic Behavior in Large Evolving Graphs. In *WSDM*. ACM, 667–676.

[60] Ryan A. Rossi and Jennifer Neville. 2010. Modeling the Evolution of Discussion Topics and Communication to Improve Relational Classification. In *SIGKDD SOMA*. 89–97.

[61] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2017. Deep Feature Learning for Graphs. In *arXiv:1704.08829*.

[62] Sergio D Servetto and Guillermo Barrenechea. 2002. Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks. In *Wireless Sensor Networks & App.* 12–21.

[63] Sucheta Soundarajan, Acar Tamersoy, Elias B Khalil, Tina Eliassi-Rad, Duen Horng Chau, Brian Gallagher, and Kevin Roundy. 2016. Generating graph snapshots from streaming edge data. In *Proceedings of the 25th International Conference Companion on World Wide Web*. 109–110.

[64] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. 2007. GraphScope: parameter-free mining of large time-evolving graphs. In *SIGKDD*. 687–696.

[65] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*. 1067–1077.

[66] A. Wagner and D.A. Fell. 2001. The small world inside large metabolic networks. *Proceedings of the Royal Society of London. Series B: Biological Sciences* 268, 1478 (2001), 1803–1810.

[67] D.J. Watts and S.H. Strogatz. 1998. Collective dynamics of small-world networks. *Nature* 393, 6684 (1998), 440–442.