Learning Dynamic Graph Embeddings Using Random Walk with Temporal Backtracking

Chenghan Huang[*]
Millennium Management, LLC
New York, NY 10022
njhuangchenghan@gmail.com

Lili Wang[*]
Dartmouth College
Hanover, NH 03755
lili.wang.gr@dartmouth.edu

Xinyuan Cao
Georgia Institute of Technology
Atlanta, GA 30332
xcao78@gatech.edu

Weicheng Ma
Dartmouth College
Hanover, NH 03755
weicheng.ma.gr@dartmouth.edu

Soroush Vosoughi
Dartmouth College
Hanover, NH 03755
soroush.vosoughi@dartmouth.edu

Abstract

Representation learning on graphs (also referred to as network embedding) can be done at different levels of granularity, from node to graph level. The majority of work on graph representation learning focuses on the former, and while there has been some work done on graph-level embedding, these typically deal with static networks. However, learning low-dimensional graph-level representations for dynamic (i.e., temporal) networks is important for such downstream graph retrieval tasks as temporal graph similarity ranking, temporal graph isomorphism, and anomaly detection. In this paper, we propose a novel temporal graph-level embedding method to fill this gap. Our method first builds a multilayer graph and then utilizes a novel modified random walk with temporal backtracking to generate temporal contexts for the nodes in the graph. Finally, a "document-level" language model is learned from these contexts to generate graph-level embeddings. We evaluate our model on five publicly available datasets for two commonly used tasks of graph similarity ranking and anomaly detection. Our results show that our method achieves state-of-the-art performance compared to all prior baselines.

1 Introduction

Graphs (i.e., networks) are a dominant type of data in many diverse domains, from social networks [24], to protein interactions [2], and scientific collaboration [20]. Through graph representations learning (also called graph embedding), we can represent graphs using general-purpose vector representations, removing the need for task-specific feature-engineering.

Networks (sometimes referred to as graphs)fall under two general categories, static and dynamic. As the name suggests, static networks are those whose structure does not change over time. Networks capturing natural phenomena usually fall under this category. For instance, the network representing

[*]The first two authors contributed equally to this work.

NeurIPS 2022 Temporal Graph Learning Workshop held in New Orleans, United States

the chemical interaction of different compounds is most likely a static network as the rules governing the interactions between compounds do not change over time. Conversely, dynamic networks, sometimes also referred to as temporal or evolving networks, are those whose structure does change over time. Networks capturing man-made systems and human interactions and behavior usually fall under this category. For instance, the Twitter social network is constantly changing with people frequently following/unfollowing each other [15]. Representation learning on static and dynamic networks are different from each other as the static embeddings need to only capture the structure of the networks while dynamic embeddings need to capture the structural and temporal aspects of the networks. Though static embedding methods can be applied to dynamic networks, the resulting embeddings are not ideal as they do not capture the evolving aspect of these networks.

Network embedding methods can be further categorized based on the granularity at which they operate, from node level to graph level. The most common type of network embedding is node embedding in which nodes in *one* network are represented as fixed-length vectors. While these vectors are supposed to preserve different scales of proximity between the nodes such as microscopic (such as `DeepWalk` [29] and `node2vec` [18]), macroscopic (such as `DP` [12] and `HARP` [7]), and structural role (such as `struc2vec` [30] and `GraphWave` [10]), they can not capture proximity between different networks as the node representations are learned within the context of the network they occupy. Considerable work has been done on node embedding for dynamic graphs (e.g., see [27, 41, 17, 23, 34]), which not only preserves the network structural information but also the temporal information for each node.

Whereas node embedding deals with a single graph, graph-level network embedding allows us to learn representations of whole graphs and directly compare different graphs. This enables us to investigate fundamental problems, such as whether two graphs are identical (also called the graph isomorphism problem). Babai [3] has shown that this problem can be solved in quasipolynomial time. In real-world applications, however, instead of determining whether two graphs are identical, we care about the degree of similarity between graphs. A typical application of this problem involves classifying graphs based on their similarity. Note that this is a generalization of the graph isomorphism problem as two graphs that are identical will be labeled the same. One approach to solving the graph classification problem is to learn a representation of the graph as a vector, called whole graph embedding, which is invariant under the graph isomorphism, and then adopt down-stream classifiers.

Several graph-level embedding methods have been explored, but they mainly deal with static networks [26, 8, 9, 35, 36, 13, 31]. However, in real-world applications, networks are typically dynamic and contain information besides their static structures.

In fact, to the best of our knowledge, there is only one prior method designed for dynamic graph-level embedding, called `tdGraphEmbed` [5]. However, a limitation of this method is that it treats dynamic graphs as a collection of independent static graph snapshots, without considering the connections and evolving information between them.

In this paper, we fill this gap by making the following contributions:

- We propose a novel method called temporal backtracking random walk, and combine it with *doc2vec* for dynamic graph-level embedding. Our method smoothly incorporates both graph structural and historical evolving information.
- We evaluate our method on five publicly available datasets for two tasks: graph similarity ranking and anomaly detection, and achieve state-of-the-art performance.
- We empirically show that our model scales linearly, which enables real-world application and scaling to large graphs.

2 Related Work

`tdGraphEmbed` is currently the only method for dynamic graph-level embedding, which we discussed in the introduction. Here, we go over two adjacent embedding categories: temporal node and static graph-level embedding. Different from static node embedding methods (such as `node2vec` [18], `SDNE` [37] and `GAE` [21]) which only consider structural information, temporal node embedding methods learn node representations from the historical information in evolving networks, to preserve both the structural and temporal information. Representative techniques including matrix factorization (such as `TMF` [11] and `TMNF` [39]), modified random walk to incorporate timestamps (such as dynnode2vec [25], CTDNE [27]), and deep-learning-based methods (such as DynGEM [17], dyngraph2vec [16], whose variations include DynAE, DynRNN and DynAERNN). DynamicTriad [40] relies on the triadic closure process, which is the development of closed triads from open triads.

For static graph-level embedding, a classic approach is to use graph kernels (such as Weisfeiler-Lehman kernel [33], random walk kernel [14], shortest path kernel [6] and deep graph kernel [38]). graph2vec [26] and GL2Vec [8] use graph kernels to extract features, which are then passed to a language model to extract embeddings. Sub2Vec [1] uses id-path and degree-path random walks to capture the neighborhood and structural property of a set of sub-graphs, and in this way get the representation of arbitrary sub-graphs. UGraphEmb [4] uses a multi-scale node attention to combine node-level embeddings into graph-level embeddings to preserve their graph-graph proximity. Other methods like SF [9], NetLSD [35], and FGSD [36] use the information from the Laplacian matrix and eigenvalues of graphs to generate embeddings.

3 Framework

In this section, we formally define the problem we address in this paper and introduce our framework. Our framework is simple but effective, which consists of two parts: (1) Building a multilayer graph and adopting temporal backtracking random walk on it (2) Learning a *doc2vec* language model on the output of the modified random walk to get graph-level embedding.

3.1 Background

Let $G = (V, E, T)$ be a discrete temporal graph where each temporal edge $(u, v)_t \in E$ is directed from a node u to a node v at time $t \in T$. A snapshot of G at time t is defined as $G_t = (V_t, E_t)$, as the graph of all edges occurring at time t. We consider the problem of representing each snapshot G_t as a low-dimensional vector which captures both the dynamic evolution information and graph topology as a n-dimensional Euclidean vector $X_t \in \mathbb{R}^n$, with $n << |V|$. We solve this problem in an unsupervised way and do not require any task-specific information.

3.2 Our Framework

We construct a multilayer weighted graph $M(V_M, E_M)$ that encodes the evolution between nodes. Each layer $M_t, t = 0, 1..., |T|$ is constructed by the nodes of G and the edges of snapshot G_t. Next, we build the inter-layer edges between each pair of M_t and M_{t-1} by directly connecting the corresponding nodes from t to $t-1$, note that the edges between the two layers are unidirectional. We model each snapshot G_t by using temporal backtracking random walk from each node as a sentence, and then all the sentences are concatenated to create a document to represent the whole snapshot.

For each step of the temporal backtracking walk, it can either walk inside the current layer to get the structural information or walk into the previous layer to get historical evolving information. We define the *stay* constant α, such that for each step the probability of staying in the current layer is α and the probability of going to the previous layer is $1 - \alpha$. A temporal backtracking walk on M is a sequence of vertices $\langle v_1, v_2, \cdots, v_k \rangle$ such that $\langle v_i, v_{i+1} \rangle \in E_M$ for $1 \leq i < k$, which can be derived by the transition probability on M. Assume we have got $\langle v_1, v_2, \cdots, v_i \rangle$, and $v_i \in M_t$the transition probability at step i+1 is defined as:

$$P\left(v_{i+1}|v_{i-1}, v_i\right) = \begin{cases} 1-\alpha & v_{i+1} \in M_{t-1} \\ \frac{\alpha}{pZ} & d_{v_{i-1},v_{i+1}} = 0, v_{i+1} \in M_t \\ \frac{\alpha}{Z} & d_{v_{i-1},v_{i+1}} = 1, v_{i+1} \in M_t \\ \frac{\alpha}{qZ} & d_{v_{i-1},v_{i+1}} = 2, v_{i+1} \in M_t \\ 0 & otherwise \end{cases} \tag{1}$$

Here, inspired by node2vec, the $d_{u,v}$ measures the length of the shortest path between node u and v. p is the return parameter and q is the in-out parameter and in this way smoothly interpolates breadth-first and depth-first sampling. Z is the normalizing constant. Each step of the temporal backtracking random walk can be done efficiently in $O(1)$ time complexity using a modified alias sampling method [22].

Algorithm 1 Dynamic Graph-Level Embedding

Input: $G = \{G_1, G_2, \ldots, G_{|T|}\}$: the snapshots of the dynamic graph, and $G_t = (V_t, E_t)$.
Parameter: p : return parameter, q : in-out parameter, α : staying constant, n : number of walks per node, L : length of walk
Output: $X_1, X_2, \ldots, X_{|T|} \in \mathbb{R}^n$, the embeddings for each snapshot

```
Create the multilayer graph M from G
for t ∈ {1, ..., T} do
    π[t] = PrecomputeProbabilities(M_t, p, q)
end for
docs = ϕ
for t ∈ {1, ..., T} do
    for v ∈ V_t do
        for i = 1 to n do
            walks = ϕ, s = v, time = t
            while len(walks) ≤ L & Neighbor(s) ≠ ϕ do
                flag = Random(0, 1)
                if flag ≤ α then
                    s = AliasSample(s, π[time])
                    walks = walks + s
                else
                    time = min(1, time − 1)
                end if
            end while
            docs = docs + [walks, t]
        end for
    end for
end for
return X = Doc2Vec(docs)
```

The idea behind the temporal backtracking random walk is that the contexts of nodes within one layer capture the neighborhood proximity; through backtracking to the former layer, it smoothly incorporates the structural information of previous timestamps. The *stay* constant is set to be larger than 0.5, so the influence of older timestamps will be smoothly decayed as the possibility of entering previous layers will be reduced exponentially.

The context from each node of a certain G_t can be seen as a sentence, which we combine into a document to represent a snapshot. Since these sentences do not have a specific order, we adopt a modified *doc2vec* language model to learn a representation of the snapshot "documents" where each sentence is tagged with the corresponding timestamp (t of G_t) as the paragraph id of *doc2vec*. After training, that final paragraph vector is the dynamic graph-level embedding of G_t. Algorithm 1 shows the pseudocode of our method.

4 Experiment

For a thorough evaluation of the performance of our proposed method, we conduct all the quantitative tasks for dynamic graph-level embedding introduced by Beladev et al. [5] (the only other dynamic graph-level embedding method). Specifically, these tasks are temporal similarity ranking and anomaly detection. We also the scalability evaluations to show our model's applicability to large networks commonly found in real-world applications. For a fair comparison, we use the exact same datasets, experiments, settings, and metrics as Beladev et al.

4.1 Datasets

We use all the five publicly available social graphs and corresponding ground truth for temporal similarity ranking and anomaly detection used by Beladev et al. [5]. Similar to Beladev et al., nodes with a degree less than five are removed. The descriptive statistics of these datasets are shown in Table 1. The datasets are described in detail below:

Dataset	Nodes	Edges	Timestamps	Granularity	Temporal Complexity
Reddit (Game of Thrones)	156,732	834,753	62	Daily	4.67
Reddit (Formula1)	38,702	254,731	61	Daily	13.30
Facebook wall posts	46,873	857,815	30	Monthly	5.56
Enron	87,062	$1,146,800$	182	Weekly	3.45
Slashdot	51,083	140,778	13	Monthly	2.88

Table 1: Descriptive statistics of the five datasets used in the temporal similarity ranking experiments. Graph temporal complexity denotes the average time interval between each snapshot to its most similar snapshot in previous timestamps. This table is taken from Beladev et al. [5].

- **Reddit Game of Thrones**: This dataset consists of the TV series 'Game of Thrones' subreddit. The nodes are Reddit users and the edges are replies from one user to the posts of another user. This dataset includes 62 daily granularity snapshots with 156,732 nodes and 834,753 edges.
- **Reddit Formula1**: This dataset consists of the 'Formula1' subreddit. The nodes are Reddit users and the edges are replies from one user to the posts of another user. This dataset includes 61 daily granularity snapshots with 38,702 nodes and 1,146,800 edges.
- **Facebook wall posts**: This dataset consists of a subset of posts to other users' walls on Facebook. The nodes are Facebook users and the edges are posts. This dataset includes 30 monthly granularity snapshots with 46,873 nodes and 857,815 edges.
- **Enron**: This dataset consists of emails sent between employees of Enron. The nodes are Enron employees and the edges are emails between them. This dataset includes 182 weekly granularity snapshots with 87,062 nodes and 1,146,800 edges.
- **Slashdot**: This dataset consists of the replies network on Slashdot website. The nodes are Slashdot users and the edges are replies from one to another. This dataset includes 13 monthly granularity snapshots with 51,083 nodes and 140,778 edges.

4.2 Experiment Settings

We compare our model with three types of baselines: static graph-level embedding (represented by `graph2vec`, `UGraphEmb`, and `Sub2vec`), temporal node-level embedding (represented by `node2vec aligned`, `SDNE aligned`, `GAE aligned`[2], `DynGEM`, `DynamicTriad`, `DynAE`, and `DynAERNN`), temporal graph-level embedding (the only existing SOTA method `tdGraphEmbed`). For all of these baselines, we use the same parameter settings introduced by Beladev et al. and report the best results between our experiments and the results reported by them. We do this to err on the side of caution and fairness.

For our model, we set the number of temporal backtracking random walks from each node to 40 with a length of 32. The return parameter p is set to 1, the in-out parameter q is set to 0.5, and the *stay* constant α is set to 0.8. For the *doc2vec* model training, the maximum distance between the current and predicted word within a sentence is set to 5, the initial learning rate is set to 0.025, and the size of the final embedding is set to 128.

4.3 Temporal Similarity Ranking

For a snapshot G_t of a dynamic graph G, the most similar snapshot to it may not be $G_t - 1$ or $G_t + 1$, but some other snapshot far away from it [5]. This task aims to test the ability of a model to capture the similarity among each snapshot. Temporal similarity ranking has many potential real-world applications. For example, this task can be used for detecting organized influence operations on social media. Many organized influence operations on social media rely on sharing and replying to each other to artificially boost support for their agenda. By analyzing the similarity of dynamic share/reply networks, we can detect new organized influence operations.

For this task, first we train our model to get the representations for all the snapshots. For each snapshot G_t, we rank all the other snapshots $G_i, (i \neq t)$ based on the cosine similarity between their

[2]Since these three methods are static ones, the "aligned" here means that each snapshot is trained separately, then have their embeddings rotated for alignment [19].

	Reddit - Game of Thrones				Reddit- Formula1			
	p@10	p@20	τ	ρ	p@10	p@20	τ	ρ
Static graph-level								
graph2vec	0.260	0.381	0.038	0.056	0.169	0.320	0.043	0.063
UGraphEmb	0.278	0.416	0.046	0.068	0.238	0.37	0.026	0.039
Sub2Vec	0.160	0.355	0.022	0.039	0.182	0.300	-0.030	-0.040
Temporal node-level								
node2vec aligned	0.336	0.431	0.069	0.103	0.214	0.361	0.047	0.083
SDNE aligned	0.352	0.457	0.120	0.197	0.262	0.388	0.044	0.078
GAE aligned	0.235	0.342	0.044	0.066	0.200	0.342	0.036	0.062
DynGEM	0.340	0.441	0.075	0.113	0.192	0.339	0.029	0.045
DynamicTriad	0.277	0.364	0.131	0.195	0.243	0.396	0.024	0.033
DynAE	0.192	0.357	0.019	0.030	0.229	0.397	0.009	0.012
DynAERNN	0.192	0.349	-0.002	-0.004	0.164	0.357	0.026	0.037
Temporal graph-level								
tdGraphEmbed	0.355	0.457	0.160	0.232	**0.274**	0.400	0.060	0.092
Our method	**0.435**	**0.481**	**0.177**	**0.272**	0.265	**0.410**	**0.076**	**0.106**

Table 2: The temporal similarity ranking results for the two Reddit datasets (Reddit Game of Thrones and Reddit Formula1). The baselines are divided into three categories: static graph-level embeddings, temporal node-level embeddings, and temporal graph-level embeddings. p@10 denotes precision at 10, p@20 denotes precision at 20, ρ denotes Spearman's rank correlation coefficient, and τ denotes Kendall's rank correlation coefficient.

	Enron				Facebook-wall posts				Slashdot			
	p@10	p@20	τ	ρ	p@10	p@20	τ	ρ	p@5	p@10	τ	ρ
Static graph-level												
graph2vec	.045	.059	-.033	-.046	.423	.713	.120	.176	.292	.800	.026	.045
UGraphEmb	.168	.269	.110	.150	.750	.871	.355	.452	.462	.900	.215	.271
Sub2Vec	.073	.137	.028	.044	.353	.685	.012	.021	.385	.808	.037	.074
Temporal node-level												
node2vec aligned	.379	.452	.107	.139	.680	.840	.303	.414	.538	.908	.229	.306
SDNE aligned	.316	.400	.087	.138	.400	.645	.095	.120	.415	.885	.095	.124
GAE aligned	.277	.360	.118	.156	.613	.820	.292	.397	.492	.885	.168	.227
DynGEM	.335	.377	.103	.143	.356	.733	.094	.115	.569	**.915**	.245	.314
DynamicTriad	.322	.425	.112	.153	.733	.818	.271	.395	.646	.869	.201	.276
DynAE	.069	.145	.009	.012	.389	.743	.122	.163	.473	.900	.002	.025
DynAERNN	.061	.110	.004	.006	.393	.755	.065	.076	.509	.900	.041	.088
Temporal graph-level												
tdGraphEmbed	.385	.489	.127	.188	.750	.892	.398	.522	**.785**	**.915**	.347	.463
Our method	**.479**	**.532**	**.172**	**.251**	**.806**	**.896**	**.447**	**.559**	.723	.885	**.400**	**.524**

Table 3: The temporal similarity ranking results for Enron, Facebook wall posts, and Slashdot datasets. The baselines are divided into three categories: static graph-level embeddings, temporal node-level embeddings, and temporal graph-level embeddings. p@5 denotes precision at 5, p@10 denotes precision at 10, p@20 denotes precision at 20, ρ denotes Spearman's rank correlation coefficient, and τ denotes Kendall's rank correlation coefficient.

embeddings X_t and X_i:

$$cos(X_t, X_i) = \frac{X_t \cdot X_i}{\|X_t\| \|X_i\|} \tag{2}$$

Using the predicted and ground truth ranking lists of G_t, we can get a correlation and precision score. For these metrics, we report the average precision at 10, precision at 20, Spearman's rank correlation coefficient (ρ), and Kendall's rank correlation coefficient (τ) for all the snapshots. For the Slashdot dataset we used precision at 5, and precision at 10 since there are only 13 time-steps. We show the results in Tables 2 and 3. As can be seen, our model outperforms all the baselines for all the datasets and metrics with three exceptions (out of 220), where `tdGraphEmbed` generates better results in the p@10 metric for the Reddit Formula1 dataset and p@5 and p@10 for the Slashdot dataset.

4.4 Anomaly Detection

This task aims to detect anomalies during the evolution of a dynamic graph. This task has several real-world applications. For instance, prior work has shown that there is a connection between the

	Reddit - Game of Thrones				Reddit- Formula1			
	p@5	p@10	r@5	r@10	p@5	p@10	r@5	r@10
Static graph-level embedding								
graph2vec	0.8	0.6	0.571	0.857	0.2	0.2	0.25	0.5
UGraphEmb	**1**	**0.7**	**0.714**	**1**	0	0	0	0
Sub2Vec	0.2	0.3	0.142	0.428	0.4	0.3	0.5	0.75
Temporal node-level embedding								
node2vec aligned	**1**	**0.7**	**1**	**0.714**	0	0.3	0	0.75
SDNE aligned	0	0	0	0	0	0	0	0
GAE aligned	0	0	0	0	0.2	0.3	0.25	0.75
DynGEM	0.4	0.4	0.285	0.571	0.2	0.1	0.25	0.25
DynamicTriad	0.4	0.4	0.285	0.571	0.4	0.3	0.5	0.75
DynAE	0	0.1	0	0.142	0	0	0	0
DynAERNN	0.2	0.1	0.142	0.142	0	0	0	0
Temporal graph-level embedding								
tdGraphEmbed	**1**	**0.7**	**0.714**	**1**	**0.8**	**0.4**	**1**	**1**
Our method	**1**	**0.7**	**0.714**	**1**	**0.8**	**0.4**	**1**	**1**

Table 4: Results for anomaly detection task. p@5 denotes the metric Precision at 5, p@10 denotes Precision at 10, r@5 denotes Recall at 5, and r@10 denotes Recall at 10.

stock market and social networks [32, 28]. Detecting anomalies in a social network could potentially be used to improve financial prediction and help with risk aversion.

For this task, after generating the representations for all the snapshots, we define the difference between all the consecutive timestamps G_t and $G_t + 1$ as the cosine distance between $\mathbb{X}_t$ and $\mathbb{X}_{t+1}$. The prediction of anomalies is made by sorting all the differences and selecting the top K as anomalies. We test this task on two datasets, Reddit Game of Thrones (the anomalies represent the air dates of the episodes of "Game of Thrones" season seven as during these times the volume of discussion is anomalous compared to other times) and Reddit Formula1 (dates of "Formula 1" races; same logic as before). As evaluation metrics, we report the Precision at 5, Precision at 10, Recall at 5, and Recall at 10 between our predicted results and the ground truth.

The results are shown in Table 4. Our method outperforms other baselines and achieves the same scores as the current SOTA method `tdGraphEmbed`. The reason may be because this task only takes the consecutive timestamps into account, and these two datasets only have 7 and 4 anomalies, respectively. Different from the temporal similarity ranking task, which considers all the pairs of similarities, this task is relatively simple allowing several methods to achieve high performance (as seen by the performance of `node2vec aligned` and `UGraphEmb` which are not temporal graph-level embedding methods).

4.5 Scalability

To investigate the scalability of our model, we learn temporal graph representations using our model with the default parameters on Erdos-Renyi graphs with increasing sizes from 100 to 1,000,000 edges with average degrees of 10 for each node. For each Erdos-Renyi graph, we uniformly split the edges into 10 different snapshots. We run these tests (and all other experiments in this paper) on a Lambda Deep Learning 2-GPU Workstation (RTX 2080) with 100GB of memory. Fig. 1 shows the log-log plot of the running time vs the number of nodes. The linear curve in log-log space indicates that our model is polynomial in time with respect to the size of the graph. In fact, the slopes of the curves are less than 1 in the log-log space, meaning that our model is performing in sub-linear time; due to its use of parallel processing. This suggests that our method is able to be scaled to large networks found in real-world scenarios.

5 Conclusion

In this paper, we proposed a novel dynamic graph-level embedding method based on temporal backtracking random walk. Our method smoothly incorporates both graph structural and historical evolving information. Through experimentation on five publicly available datasets for the tasks of graph similarity ranking and anomaly detection, we showed that our method achieves superior overall

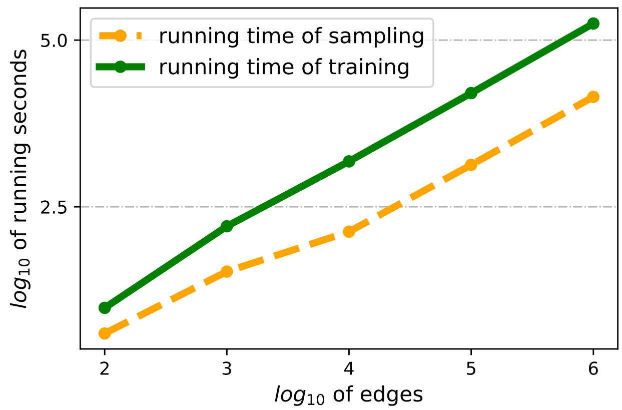

Figure 1: Scalability of our model on Erdos-Renyi graphs with an average degree of 10.

performance compared to other baselines and that our model is scalable to larger networks, making it applicable to real-world applications. An avenue for future work can be the extension of our proposed model to *heterogeneous* dynamic networks, which are dynamic networks that have multiple edge or node types.

References

[1] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 170–182. Springer, 2018.

[2] Nir Atias and Roded Sharan. Comparative analysis of protein networks: hard problems, practical solutions. *Communications of the ACM*, 55(5):88–97, 2012.

[3] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697, 2016.

[4] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. Unsupervised inductive graph-level representation learning via graph-graph proximity. *arXiv preprint arXiv:1904.01098*, 2019.

[5] Moran Beladev, Lior Rokach, Gilad Katz, Ido Guy, and Kira Radinsky. tdgraphembed: Temporal dynamic graph-level embedding. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 55–64, 2020.

[6] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[7] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[8] Hong Chen and Hisashi Koga. Gl2vec: Graph embedding enriched by line graphs with edge features. In *International Conference on Neural Information Processing*, pages 3–14. Springer, 2019.

[9] Nathan de Lara and Edouard Pineau. A simple baseline algorithm for graph classification. *arXiv preprint arXiv:1810.09155*, 2018.

[10] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1320–1329, 2018.

[11] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):1–27, 2011.

[12] Rui Feng, Yang Yang, Wenjie Hu, Fei Wu, and Yueting Zhang. Representation learning for scale-free networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[13] Feng Gao, Guy Wolf, and Matthew Hirn. Geometric scattering for graph data analysis. In *International Conference on Machine Learning*, pages 2122–2131. PMLR, 2019.

[14] Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pages 129–143. Springer, 2003.

[15] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Phani Gummadi. Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st International Conference on World Wide Web*, pages 61–70, 2012.

[16] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, 2020.

[17] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*, 2018.

[18] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016.

[19] William L Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096*, 2016.

[20] Norman P Hummon and Patrick Dereian. Connectivity in a citation network: The development of dna theory. *Social Networks*, 11(1):39–63, 1989.

[21] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[22] Aaron Q Li, Amr Ahmed, Sujith Ravi, and Alexander J Smola. Reducing the sampling complexity of topic models. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 891–900. ACM, 2014.

[23] Taisong Li, Jiawei Zhang, S Yu Philip, Yan Zhang, and Yonghong Yan. Deep dynamic network embedding for link prediction. *IEEE Access*, 6:29219–29230, 2018.

[24] Francois Lorrain and Harrison C White. Structural equivalence of individuals in social networks. *The Journal of Mathematical Sociology*, 1(1):49–80, 1971.

[25] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. dynnode2vec: Scalable dynamic network embedding. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3762–3765. IEEE, 2018.

[26] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[27] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976. International World Wide Web Conferences Steering Committee, 2018.

[28] Michael Nofer and Oliver Hinz. Using twitter to predict the stock market. *Business & Information Systems Engineering*, 57(4):229–242, 2015.

[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710. ACM, 2014.

[30] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 385–394, 2017.

[31] Benedek Rozemberczki and Rik Sarkar. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, page 1325–1334. ACM, 2020.

[32] Dehua Shen, Andrew Urquhart, and Pengfei Wang. Does twitter predict bitcoin? *Economics Letters*, 174:118–122, 2019.

[33] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[34] Uriel Singer, Ido Guy, and Kira Radinsky. Node embedding over temporal graphs. In *IJCAI*, 2018.

[35] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller. Netlsd: hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2347–2356, 2018.

[36] Saurabh Verma and Zhi-Li Zhang. Hunt for the unique, stable, sparse and fast feature learning on graphs. In *NIPS*, pages 88–98, 2017.

[37] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1225–1234. ACM, 2016.

[38] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.

[39] Wenchao Yu, Charu C Aggarwal, and Wei Wang. Temporally factorized network modeling for evolutionary network analysis. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 455–464, 2017.

[40] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[41] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. Embedding temporal network via neighborhood formation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2857–2866. ACM, 2018.