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.
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 includ-
ing matrix factorization (such as TMF [11] and TMNF [59]), modified random walk to incorporate
We construct a multilayer weighted graph \( G \) using a breadth-first and depth-first sampling method \([22]\). \( G \) is a sequence of graphs \( G_1, G_2, \ldots, G_T \) over time, and each layer \( G_t \) 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 layers \( G_t \) and \( G_{t-1} \) by directly connecting the corresponding nodes from \( G_t \) to \( G_{t-1} \), 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 \( \alpha \), such that for each step the probability of staying in the current layer is \( \alpha \) and the probability of going to the previous layer is \( 1 - \alpha \). A temporal backtracking walk on \( G \) is a sequence of vertices \( (v_1, v_2, \ldots, v_k) \) 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 \( G \). Assume we have got \( \langle v_1, v_2, \ldots, v_i \rangle \), and \( v_i \in M_t \) the transition probability at step \( i+1 \) is defined as:

\[
P(v_{i+1}|v_{i-1}, v_i) = \begin{cases} 
1 - \alpha & v_{i+1} \in M_{t-1} \\
\frac{\alpha}{d_{v_{i-1},v_{i+1}}} & d_{v_{i-1},v_{i+1}} = 0, v_{i+1} \in M_t \\
\frac{\alpha}{d_{v_{i-1},v_{i+1}}} & d_{v_{i-1},v_{i+1}} = 1, v_{i+1} \in M_t \\
\frac{\alpha}{d_{v_{i-1},v_{i+1}}} & d_{v_{i-1},v_{i+1}} = 2, v_{i+1} \in M_t \\
0 & \text{otherwise}
\end{cases}
\]

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, \( \alpha \): 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

1: Create the multilayer graph \( M \) from \( G \)
2: for \( t \in \{1, \ldots, T\} \) do
3: \( \pi[t] = \text{PrecomputeProbabilities}(M_t, p, q) \)
4: end for
5: docs = φ
6: for \( t \in \{1, \ldots, T\} \) do
7: for \( v \in V_t \) do
8: for \( i = 1 \) to \( n \) do
9: \( \text{walks} = \phi, s = v, \text{time} = t \)
10: while \( \text{len(walks)} \leq L \) \& \( \text{Neighbor(s)} \neq \phi \) do
11: \( \text{flag} = \text{Random}(0, 1) \)
12: if \( \text{flag} \leq \alpha \) then
13: \( s = \text{AliasSample}(s, \pi[\text{time}]) \)
14: \( \text{walks} = \text{walks} + s \)
15: else
16: \( \text{time} = \min(1, \text{time} - 1) \)
17: end if
18: end while
19: \( \text{docs} = \text{docs} + [\text{walks}, t] \)
20: end for
21: end for
22: end for
23: return \( X = \text{Doc2Vec}(\text{docs}) \)

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 \( \text{stay constant} \) 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 \textit{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 \textit{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:
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].

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes</th>
<th>Edges</th>
<th>Timestamps</th>
<th>Granularity</th>
<th>Temporal Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reddit (Game of Thrones)</td>
<td>156,732</td>
<td>834,753</td>
<td>62</td>
<td>Daily</td>
<td>4.67</td>
</tr>
<tr>
<td>Reddit (Formula1)</td>
<td>38,702</td>
<td>1,146,800</td>
<td>61</td>
<td>Daily</td>
<td>13.30</td>
</tr>
<tr>
<td>Facebook wall posts</td>
<td>87,062</td>
<td>857,815</td>
<td>30</td>
<td>Monthly</td>
<td>5.56</td>
</tr>
<tr>
<td>Enron</td>
<td>51,083</td>
<td>140,778</td>
<td>13</td>
<td>Monthly</td>
<td>2.88</td>
</tr>
</tbody>
</table>

- **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, 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 $\alpha$ 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

\[^2\text{Since these three methods are static ones, the “aligned” here means that each snapshot is trained separately, then have their embeddings rotated for alignment [19].}\]
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.

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.

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.

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 embeddings $X_t$ and $X_i$:

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

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.
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.

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

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 the consecutive timestamps $G_t$ and $G_{t+1}$ as the cosine distance between $X_t$ and $X_{t+1}$. 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


