International Journal of Data Science and Analytics (2024) 17:189–201
https://doi.org/10.1007/s41060-022-00376-3

REGULAR PAPER

Learning attentive attribute-aware node embeddings in dynamic environments

Nourhan Ahmed[1] · Ahmed Rashed[1] · Lars Schmidt-Thieme[1]

Received: 1 July 2022 / Accepted: 20 November 2022 / Published online: 3 December 2022

Abstract
Learning node embeddings is fundamental for numerous applications, such as link prediction and node classification. Node embeddings seek to learn a low-dimensional representation for each node in the graph. Many existing node representation learning methods for dynamic attributed graphs focus on preserving the temporal proximity of the nodes with relatively shallow models. However, real-life graphs are complex and usually exhibit evolutionary patterns of node attributes and graph structure. Therefore, the current state-of-the-art models fail to capture the information in the dynamic attributed graphs and settle for sub-optimal results. In this paper, we propose a novel model for embedding nodes in dynamic attributed graphs that captures the full extent of all relevant node information and the graph interrelations as well as graph evolutionary patterns. During model training, attribute-aware node embedding is learned using both graph and node properties in a dynamic context. Experiments demonstrate that our proposed method is superior to the state-of-the-art models in link prediction tasks. In addition, it introduces a novel way of learning richer representations by fully exploiting node attributes, graph structure, and evolutionary patterns in dynamic attributed graphs.

Keywords Attributed network · Dynamic network · Graph embedding · Unsupervised feature extraction · Self-attention

1 Introduction

Graph structures are the basis of many objects, concepts, and processes we use everyday. They appear spontaneously in a wide diversity of real-world environments since they are valuable in illustrating how objects are physically or logically connected. The Internet is one clear example. When we access the Internet, we implicitly enter a graph structure comprised of billions of nodes and edges. The graph topology has gained significant popularity in recent years because they help simplify complex systems of information into accessible representations.

✉ Nourhan Ahmed
ahmed@ismll.uni-hildesheim.de

Ahmed Rashed
ahmedrashed@ismll.uni-hildesheim.de

Lars Schmidt-Thieme
schmidt-thieme@ismll.uni-hildesheim.de

[1] Information Systems and Machine Learning Lab, University of Hildesheim, 31141 Hildesheim, Lower Saxony, Germany

Graph embedding is an efficient strategy that allows for the use of machine learning models with graph-structured data by preserving nodes, edges, and their properties into lower-dimensional embedding vectors [1–3]. Early literature on graph embedding has paid the most attention to factorization methods, such as Laplacian eigenmaps and node proximity matrix factorization. Different approaches were proposed to find the matrix approximation such as singular value decomposition (SVD) [4,5]. This technique is, however, computationally costly, especially when working with large networks [6]. More recently, scholars have employed random-walk-based methods to calibrate local knowledge. These methods—the first of which was Deepwalk [1]—learn node embedding by increasing the log-likelihood of a node given its neighboring nodes. Several research works, such as Node2vec [2] and LINE model [7], reported in the literature are variants of the Deepwalk approach.

Later, the research focus has turned to the question of how temporal information can be manifested into graph embedding. This strand of the literature was based on the idea that several tasks such as node classification and link prediction could be enhanced by incorporating graph dynamics [8,9]. In addition to evolutionary patterns, adding auxiliary informa-

tion such as node attributes in graph embedding has drawn further attention in recent years [10,11]. However, most models preserve many but not all essential characteristics in the dynamic attributed network, resulting in sub-optimal performance.

In this paper, we propose a novel node embedding model that connects two lines of research: node attributes and dynamic representation, yielding a model explicitly trained to preserve the full extent of node information in a rather dynamic fashion. We propose a novel architecture named deep attributed dynamic self-attention model *(DADSAT)* to learn latent node representation. First, in addition to self-attention layers that capture both structural and dynamic information in the graph, we introduce an unsupervised feature extraction module using the BERT model [12]. This module explicitly empowers the model to capture feature information for each node in the graph. This takes advantage of the fact that the training data contain more information than the structural information in the network; as result, we can expressly learn more informative node representations. Second, we use a recursive skip connection in conjunction with layer normalization to improve information delivery and integration between node latent representations at different levels. This relies on the assumption that the recursive skip connection, in conjunction with layer normalization, improves optimization and allows for more expressive information to be included. At different time steps, the recursive skip connection along with layer normalization is used to connect different self-attention layers—it is specifically used to support DADSAT in learning more informative latent representations. The key contributions of the current paper can be represented as follows:

- We propose a novel architecture for dynamic attributed graph representation learning to solve the challenging problem of embedding nodes in dynamic attributed graphs which convey more information.
- We develop a node representation learning algorithm that is not only capable of efficiently capturing both structural and dynamic information in a graph but is also supported by an unsupervised feature extraction module to fuse node features. The proposed model encodes the graph topological structure, temporal patterns, and node attributes into a reduced-dimensional representation.
- We conduct extensive experiments on real-world attributed dynamic networks. The findings indicate that our proposed approach consistently outperforms all baseline models in link prediction tasks. In addition, our approach generally exhibits superior stability over different time steps in comparison with baseline models.

2 Related work

This section provides an overview of the existing graph embedding models. We divide them into three main types: static, dynamic, and self-attention embedding models. In what follows, we discuss them consecutively.

2.1 Static network embedding models

Static graph representation learning dates back to factorization-based graph embedding models such as Laplacian [13], graph factorization [14], and nonnegative matrix factorization [11]. However, factorization-based models only focus on the transductive setting and cannot handle large-scale networks since they are computationally expensive.

Later, a variety of random walk-based methods have been proposed such as Deepwalk, Node2vec, LINE, DGI and BRNLE [1,2,7,15,16]. Deepwalk, for example, was inspired by the way language modeling algorithms learn node embedding from the local information obtained through a random walk [1]. The Node2vec model [2]—which can be considered an extension of Deepwalk—controls the random walk to provide richer node representation. In a similar fashion, the LINE model [7] is designed to address the problem of generating graph embeddings for large networks with millions of nodes in low-dimensional vector spaces. Moreover, SEAL [17], developed for link prediction tasks, is another approach that learns from sub-graph structures as well as latent and explicit node characteristics to encompass the whole scope of information. Additionally, HM-LDM [18] is an unsupervised embedding approach that combines the non-negativity constrained eigenmodel with the latent distance model. One main feature of HM-LDM is that it integrates soft and hard community detection with network embeddings to explore the whole scope of information in the embedding space.

On the other hand, a different, yet related, set of approaches combines GNNs with recurrent architectures. The most explored GNNs in this context are graph convolutional networks (GCNs) [19,20]. GCNs employ convolution over the graph to aggregate the features of all neighboring nodes for each node, followed by a linear transformation to construct node-specific representations more efficiently. Graph isomorphism network (GIN) [21], on the other hand, is a fully spatial-based GCN network with superior discriminatory power over existing GCNs for neighbor aggregation. Another work, GraphSAGE [5], can be viewed as a generalization of the aggregation function presented in the GCN architecture. LSPE [22] expands the capabilities of GNN architectures by incorporating positional encoding and injecting it into the input layer. Consequently, this method retrains the structural and positional information, leading to significantly improved performance as a result.

Although these models have significantly advanced the quality of node embeddings, they are not flawless. Specifically, the aforementioned models do not integrate evolutionary patterns and only operate on the graph structure in the static state. This shortcoming degrades the quality of the generated embeddings.

2.2 Dynamic network embedding models

Dynamic graphs can be classified into snapshot sequences or timestamped graphs depending on how they can be modeled with respect to time signals. A snapshot sequence refers to a series of static graphs captured at multiple discrete time steps. On the other hand, the timestamped graph refers to a single graph with a time stamp on each link [9]. The majority of dynamic graph embedding models adopt the snapshot-sequence approach [9,23].

There are several examples of the snapshot-sequence approach in the literature. For instance, DynamicTriad [24] retains both structural and evolution information by modeling how closed triads evolve in a given network. Another approach is Dyngraph2vec [25] which models the temporal patterns in the network using a recurrent deep architecture. Similarly, DySAT [9] learns node embeddings using joint self-attention to preserve structural and temporal properties in graphs.

Recently, research has focused on how to improve these models by adding auxiliary information such as node attributes and node embedding's uncertainty such as DNE [10], DANE [11], Online-Node2vec [26], weg2vec [27] and DynG2G [28]. For instance, weg2vec adopts an unsupervised embedding approach that concurrently considers past and future context for node (event) embedding learning. Another work is DynG2G [28] which is an autoencoding stochastic dynamic embedding method that adopts node triplet energy-based ranking loss to capture the node embedding's uncertainty.

A few papers have explored the problem of node embedding from a continuous-time perspective. For instance, CTDNE [29] tackles the problem of modeling graphs as a sequence of snapshots by introducing a novel architecture which extends the random walk notion by necessitating that walks maintain the temporal order for learning continuous-time preserving embeddings.

Another category of approaches generalizes the static models to maintain the temporal information. For example, Dynnode2vec [30] is a modified skip-gram model [31] in which the authors provide a more dynamic version by using evolving random walks to update the trained skip-gram model. GloDyNE [32] is another computationally efficient variant of the skip-gram model that selects representative nodes across a network to better maintain the overall global topology of the graph. Another work that generalizes Node2vec [2] is tNodeEmbed [33]. tNodeEmbed utilizes the temporal information by applying a joint loss function that embeds nodes based on their historical temporal embeddings, therefore preserving the network's structure and dynamics. FLDNE [34] is a generic method that can be applied to any static embedding technique. It integrates temporal information by aggregating embeddings across several time steps using a convex combination function. EvolveGCN [35] is an extension of the GCN architecture that utilizes an RNN module to update the GCN parameters in order to address the problem of lacking temporal characteristics. Despite the fact that these models have substantially improved the quality of node embeddings, they are incapable of capturing the whole range of associated node information and complex network interactions in dynamic contexts.

2.3 Self-attention embedding models

The graph attention network (GAT) is the first method that adopts self-attention layers to generate graph embeddings [36]. In particular, the GAT model employs stacked self-attention layers for node classification tasks in the static state. On the other hand, the SANNE architecture introduced an unsupervised embedding algorithm based primarily on the GAT architecture [37]. DySAT [9] model is yet another variant of the GAT model in the dynamic setting. Specifically, this work employs a self-attention-based model to retain structural and temporal dimensions. However, despite the promising performance, these models are still insufficient for constructing powerful embedding that takes into account all relevant auxiliary information in networks.

3 Problem setting

We consider the problem of inferring missing links. More formally, let's define a dynamic attributed graph as a sequence of observed static graphs $\mathbf{G} = \{\mathbf{G_1}, \ldots, \mathbf{G_T}\}$, where $\mathbf{T}$ denotes the total number of time steps. Each graph $\mathbf{G_t} = (\mathbf{V_t}, \mathbf{E_t})$ is a weighted undirected graph with a set of nodes $\mathbf{V_t}$ and a set of edges $\mathbf{E_t}$ at time step t. We presume that nodes are associated with D-dimensional attributes $\mathbf{M^t} \in \mathbb{R}^{|V| \times D}$, where m_i^t denotes the attributes for node i at time step t. Links can be added and removed over time. In our case, the goal is to predict the missing or formed links at time $t + 1$ given the link data at time t. In the literature, this is known as temporal link prediction [38].

The main goal is to learn latent representations $z_i^t \in \mathbb{R}^f$ for each node $i \in \mathbf{V}$ at time steps $t = 1, 2, ..., T$ that preserves the structure properties, node attributes, and temporal patterns, where f is a small number of latent dimensions. Thus, our target will be to learn and extract node feature information in an unsupervised fashion from the input graph. Next,

we fuse the features to self-attention layers to hierarchically capture all relevant information, which finally learns node embeddings to help boost the temporal link prediction task.

4 The proposed model: DADSAT

Here, we propose an architecture named *deep attributed dynamic self-attention network* (DADSAT), a novel model for learning node embeddings. Our algorithm learns node representations in dynamic graphs by employing self-attention layers. We argue that integrating network structure and semantic information in the proposed architecture by an early fusion on the input layer is the key to learning more informative node representations.

4.1 BERT for feature extraction

When operating with graphs, we usually have unstructured text defining the nodes' relevant information. Therefore, considering the graph structure without paying attention to textual features is insufficient to learn node embedding. To address this issue, we used Bidirectional Encoder Representations from Transformers (BERT) for feature extraction in an unsupervised manner to extract meaningful representations of node attributes as input for the DADSAT architecture.

The BERT model is built on the original implementation of Transformer developed by Vaswani et al. [12]. The process of training the BERT model consists of two main phases: the pre-training stage and fine-tuning stage. The first phase is the pre-training stage, which allows the model to leverage knowledge by making use of unlabeled data in multiple tasks. The second phase is the fine-tuning stage in which the already pre-trained parameters are fine-tuned using labeled data in different tasks. There are many variants of the BERT model such as RoBERTa [39], ALBERT [40] and DistilBERT [41]. Insofar as the text defining the nodes' information is clear, we avoid the degradation of performance caused by excessive parameters. To this end, we use DistilBERT as a lighter transformer model based on the BERT architecture. Accordingly, we use DistilBERT to extract feature embeddings where fixed features are extracted from the pre-trained model to be fed into our architecture without fine-tuning on a downstream task.

This module receives the text of the relevant information for each node as input. To obtain a single vector for each node, we averaged the hidden states of the last hidden layer of each token and compressed them into a single 768-length vector $\{x_i \in R^F, \forall i \in V\}$, where F is the dimension of the vector representation that captures the essential features related to each node. Therefore, our graph embedding model is built on top of this rich representation generated using the DistilBERT model.

4.2 Graph self-attention module

The graph self-attention module aims to fuse the node's information with its neighbors' information to handle structural and temporal dependencies in the graph, respectively. The graph self-attention module consists of two main layers to process structural and temporal information. The first layer consists of a stack of self-attention layers and a positional encoding layer. In particular, in the first layer, we use a setup similar to Bahdanau additive attention [42]. This layer takes as an input the vector representation of node attributes generated using DistilBERT model $\{x_i \in R^F, \forall i \in V\}$. This layer generates a different set of features $\{x'_i \in R^{F'}, \forall i \in V\}$. We perform a linear transformation using a LeakyReLU activation function to each node, which is parameterized by a weight matrix W, to convert the nodes' vector representation into higher-level latent space. Then, we compute the attention coefficient that reflects the contribution of node i to node j which are finally used to find the final representation for every node. The resulting representation preserves both node and structural characteristics of each snapshot with shared parameters, as presented in Fig. 1:

$$\alpha_{ij} = \frac{LeakyReLU\left(A_{ij}\left[W^s x_i \parallel W^s x_j\right]\right)}{\sum_{w \varepsilon N_i} LeakyReLU\left(A_{ij}\left[x_i \parallel x_j\right]\right)} \tag{1}$$

$$x'_i = \sigma\left(\sum_{j \varepsilon N_i} \alpha_{ij} W^s x_j\right), \ \forall (i,j) \in E \tag{2}$$

Here, $\parallel$ denotes the concatenation operation; $N_i = \{j \in V : (i,j) \in E\}$ represents the set of directly connected neighbors of node i in snapshot G; W^s is a shared weight transformation; $\sigma(.)$ represents the activation function; and the coefficient α_{ij} reflects the importance of node i to node j at each snapshot. In addition, A_{ij} represents the weight of the link (i, j) in the present snapshot. The number of interactions between a pair of nodes in each snapshot defines the weight of the connection.

We then add position-dependent signals to the final output representation of the first layer to incorporate the temporal position of snapshots [9]. These positional embeddings successfully represent the absolute temporal position of each snapshot $\{x'^1_i + p^1, x'^2_i + p^2, \ldots, x'^T_i + p^T\}, \forall i \in V$, where p^t represents each snapshot's temporal position. Positional encoding uses the sine and cosine functions to compute the position vectors, as it normalizes the position between -1 to 1. Let's consider that the position is denoted by pos, and the embedding size is d. The q-th dimension of the sinusoidal positional encoding is then considered as follows [43]:

$$p_{(pos,2q)} = \sin\left(\frac{pos}{10000^{\frac{2q}{d}}}\right)$$

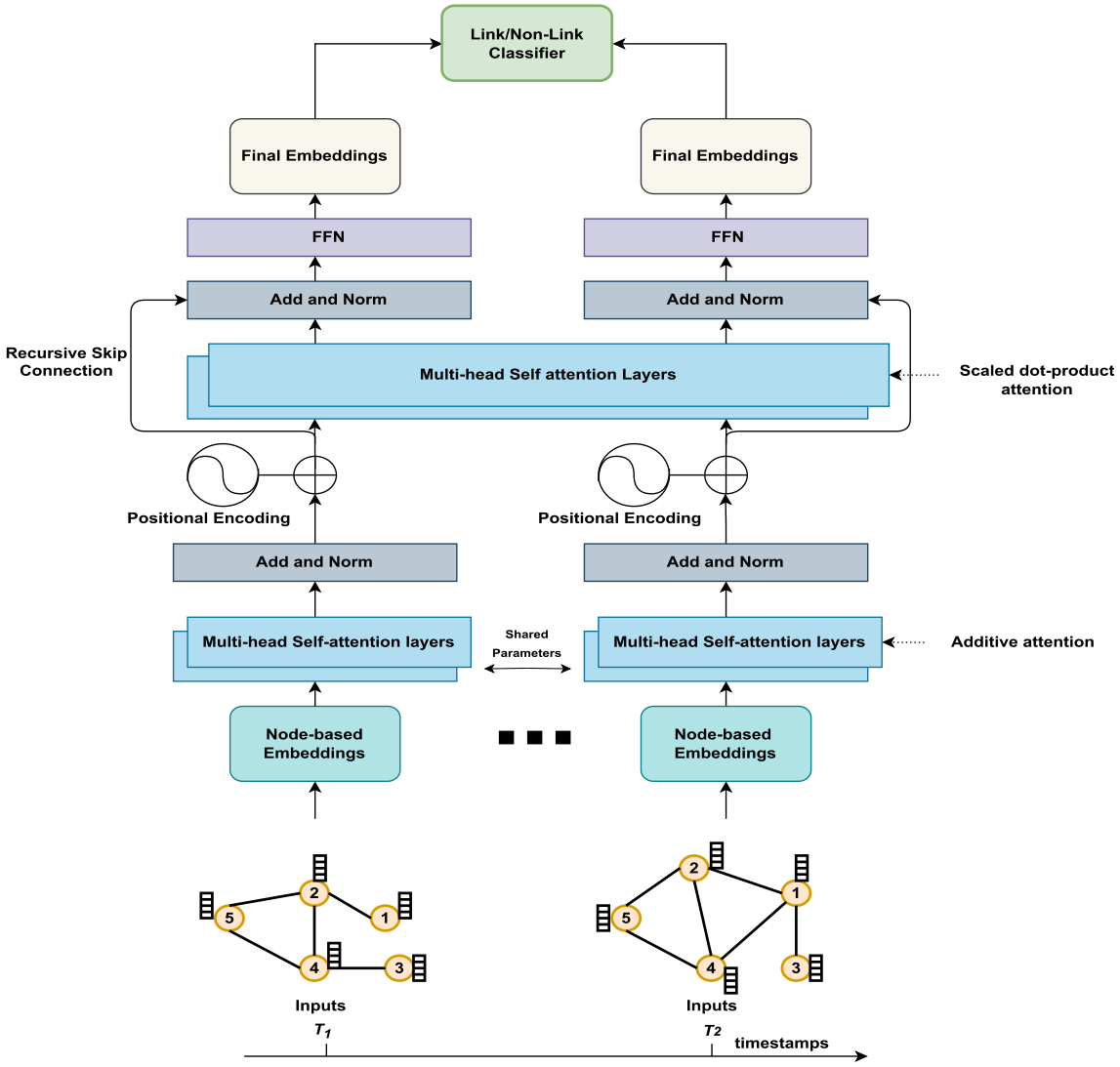

Fig. 1 Overall architecture of the DADSAT model. The input of the architecture is the node and attribute embeddings at time t

$$p_{(pos,2q+1)} = \cos\left(\frac{pos}{10000^{\frac{2q}{d}}}\right)$$

In the second layer, we adopt the scaled dot-product attention [9,43] to compute the output representation of node i at different time steps, $\{h_i^1, h_i^2, \ldots, h_i^T\}, \forall i \in V$. The multi-head attention mechanism is adopted in both layers in the architecture to capture graph evolution from different subspaces as in the transformer framework [43].

4.3 Recursive skip connections with layer normalization

We construct a recursive skip connection along with layer normalization between the output of the first layer and the output of the second layer in the graph self-attention module at each time step t which proved its capability to promote the performance of deep neural networks in different contexts compared to the plain version [44,45]. The main reason behind employing recursive skip connections is that these connections ensure better delivery and integration of information between the different components of the DADSAT model than abstract information. Accordingly, these recursive skip connections can be defined as follows:

$$\begin{aligned} s_i^{\lambda} &= LN\left(c_i + s_i^{\lambda-1}\right) \\ \text{s.t. } s_i^1 &= LN\left(c_i + h_i\right), \ \forall i \in V \end{aligned} \tag{3}$$

Here, LN denotes layer normalization, c_i denotes the output of the first layer in the graph self-attention module of a node i, h_i represents the output of the second layer in the graph self-attention module of a node i, and λ is a modulating scalar. According to the equation mentioned above, the

model is empowered to use recursive skip connection with layer normalization repeatedly to enhance optimization.

4.4 Position-wise feed-forward networks

The outputs of the graph attention module after employing recursive skip connection, $\{s_i^1, s_i^2, \ldots, s_i^T\}, \forall i \in V$, transfer through the position-wise feed-forward layer. Accordingly, it is applied to each position to get the final node representation, $\{z_i^1, z_i^2, ..., z_i^T\}, \forall i \in V$. This comes with two linear transformations using ReLU activation function [43].

4.5 Training DADSAT

To train the DADSAT model, we sample both positive and negative cases following the Deepwalk strategy [1]. Here, the DADSAT model is optimized using the following binary cross-entropy loss function:

$$L = \sum_{t=1}^{T} \sum_{j \in V} \left(\sum_{i \in walk_{(j)}^t} -log\left(\sigma\left(< z_i^t, z_j^t >\right)\right) - w_n \cdot \sum_{i' \in P_n^t} log\left(1 - \sigma\left(< z_{i'}^t, z_j^t >\right)\right)\right) \quad (4)$$

Here, σ denotes the sigmoid function, the inner product operation is represented by $< . >$, $walk^t$ indicates the selected positive instances in random walks of fixed length at snapshot G_t, P_n^t represents the sampled negative instances in snapshot G_t, and w_n is a constant fine-tuned hyper-parameter to determine the negative sampling ratio. This loss function enables the learned representations to present the local structure properties around a node in a dynamic configuration by promoting neighbor-similar representations [9].

5 Experiments

In this section, we present the performance of the proposed architecture on four real-world datasets. We also provide a comparison with different baselines along with a detailed analysis.

5.1 Datasets

We work with four datasets: DBLP [46], Epinions [47], Enron [48] and Yelp [49] for experimental evaluation as illustrated in Table 1. All of these datasets represent real-world dynamic attributed graphs. DBLP is our first dataset; it comprises bibliographic details to track the work of colleagues and to access their bibliographical information [46]. DBLP is a scientific collaboration network where actors are the authors of an article and edges are the co-authorship relationship [47]. The second dataset is Epinions which is a website where users can review different products such as items and services [50]. All trust relationships connect and form the trust web where the edges indicate a trust relationship. Enron is our third dataset. Enron is a corpus of employee interactions in which the actors are employees and the edges represent email interactions between them [48]. Finally, we make use of Yelp as a fourth dataset. Yelp is a website where businesses and users represent the actors, and the edges are the reviews [49].

Table 1 Summary statistics of the experimental datasets

	DBLP	Epinions	Enron	Yelp
#Nodes	328,080	163,543	143	6,569
#Edges	979,866	582,613	2,347	95,361
#Attributes	21,597	24,182	18,462	28,242
#Time steps	14	15	12	12

5.2 Datasets preprocessing

In this work, we take into account both the attributes of the nodes and the network properties. The node attributes are represented in the four datasets as follows: the articles' titles in the DPLB dataset; user reviews of different items of businesses in the Epinions and Yelp datasets; and finally the employees' email messages in the Enron dataset. In sum, we relied on the textual data associated with each node as our node attributes. The process of text representation using the feature-based approach of DistilBERT is done by feeding the text input into DistilBERT. The text input is tokenized before being fed into DistilBERT. Here, we use the token-level representation from last hidden states as extracted features from text. Then, we created snapshots for each dataset separately. For the DBLP data, we generated 4-year coauthorship record snapshots (i.e., our window size) amounting overall to 14 snapshots in the period 1959 to 2015. For Epinions, a temporal window of 8 months is selected to create 15 snapshots between 2001 and 2011. For Enron, a three-month temporal window was selected to make 12 snapshots between 1999 and 2002. For Yelp, a 7-month time period was chosen to create 12 snapshots between 2009 and 2016. Finally, we split the datasets using tenfold cross-validation and we reserved 10% of the data as our testing set following Sankar et al. and Li et al. [9,51].

5.3 Evaluation protocol

We assess the effectiveness of the suggested graph embedding model on the temporal link prediction task. Accordingly, we present the link prediction task in the dynamic settings in this section. Consider a series of observed graph snapshots $G_1, ..., G_t$, where each snapshot is represented by $G_t = (V_t, E_t)$. Here, V_t represents the nodes in the graph at time step t, and $E \subseteq (V \times V)$ represents the set of observed links across entities in the network. The goal of temporal link prediction is to predict the missing or formed links at time $t + 1$ given the link data at time t [9,38].

In other words, the link prediction problem can be formulated by classifying each pair of nodes into two groups: links and non-links. For this purpose, we used logistic regression classifier to predict links in G_{t+1}. After obtaining the latest embedding $z_i^t, \forall i \in V$, a binary classifier is used to predict the links at G_{t+1}. Accordingly, the effectiveness of the proposed model is evaluated at $t + 1$ after training the model at each snapshot t. For evaluation, micro- and macro-averaged AUC is used as evaluation metrics for this purpose. Micro-AUC is measured over all link instances which weighs all links equally. On the other hand, macro-AUC is measured by calculating the AUC score at each time step independently and then taking the average which weighs time steps equally [9]. Finally, micro- and macro-AUC scores are averaged across different snapshots.

5.4 Baselines

- **Deepwalk** [1]: A static embedding model that learns node embedding using local information obtained through random walk sampling.
- **Node2vec** [2]: A static embedding model that adopts biased random walks for providing richer node representation.
- **LINE** [7]: A model that produces node embeddings that retain first- and second-order proximity in the network.
- **GraphSAGE** [5]: An inductive embedding model that utilizes an embedding function that can be applied to unseen nodes in the training data. We compare the findings of various aggregators such as GCN, meanpool, maxpool, and LSTM and report the highest AUC scores following [9].
- **GCN-AE** [52]:A GCN autoencoder model that addresses the link prediction problem in graphs.
- **GAT-AE** [36]: A GAT autoencoder for link prediction.
- **SEAL** [17]: This embedding method was designed for link prediction tasks. It learns from subgraph structures as well as the implicit and explicit features of nodes to incorporate the whole spectrum of information.
- **GatedGCN-LSPE** [22]: This embedding approach is a vanilla GCN architecture that has been updated with node positional encoding and a soft-attention mechanism, as well as an edge gating mechanism, to enhance message aggregation in GCNs [53].
- **DynamicTriad** [24]: A dynamic embedding model that preserves both structural and temporal information by modeling how closed triads evolve in a given network.
- **DANE** [11]: A dynamic graph embedding model that makes use of the matrix perturbation theory to preserve the graph structure and node attributes.
- **DySAT** [9]: A dynamic graph embedding model that captures structural and temporal properties by employing joint self-attention.
- **GloDyNE** [32]: This work extends the static network embedding method-Skip-Gram [31] to cope with dynamic networks. GloDyNE selects representative nodes across a network to preserve the graph's global topology.
- **DynG2G** [28]: An autoencoding dynamic embedding method that learns probabilistic temporal graph representations through adopting node triplet energy-based ranking loss to capture node embedding uncertainty.

5.5 Experimental settings

We employed two self-attention sub-layers with the sizes 256 and 128 in both self-attention layers. As self-attention layers consist of multiple attention heads, we used 16 and 8 heads, respectively, for each layer. The proposed model is trained for 200 epochs with the option of early stopping with a default batch size of 256 nodes. In addition, we used a final embedding dimension of 128. For training, the objective function explained in Equation 4 uses node pairings that co-occur in random walks of fixed length as positive samples and those that do not co-occur as negative samples. We adopted the strategy described in Deepwalk [1] to sample unbiased random walks with the following parameters: 10 walks of length 40 for each node, and each walk has a context window size of 10. In addition, we utilize 10 negative samples for each positive pair [1,9]. For optimization, mini-batch gradient descent with Adam optimizer [54] is employed for training. For regularization, L_2 regularizer with a penalty of 5×10^{-4} and dropout rates of 0.1 and 0.5 are applied to self-attention layers, respectively. For recursive residual connections, we searched for the best value for λ in the range $\{1,2,3\}$. We configured the learning rate in the range $\{0.0001, 0.001, 0.01\}$ using the validation set following the DySAT model [9].

We evaluated the effectiveness of DADSAT with different state-of-the-art algorithms for static and dynamic graph embedding, with the goal of predicting temporal links. Regarding the static baseline models, we ran them on each snapshot. After that, in order to incorporate access to the temporal information in order to make an accurate com-

parison, we aggregated the graph up to the time t, and the weight of each link was defined as the cumulative weight up until time t following the DySAT model [9]. All of the codes for the aforementioned approaches can be found on the authors' websites. The parameters for these 13 comparative algorithms are either set to the default values provided by the authors, or they are fine-tuned via experimentation to get the optimal values. In addition, the embedding dimensionality is adjusted to either 128 or the default parameters recommended by the authors.[1]

5.6 Results and analysis

In this section, we compare the performance of DADSAT with other baseline models. We reckon that this comparative investigation will render our analysis and evaluation of the DADSAT performance more comprehensive.

As shown in Table 2, it is interesting to see that the DADSAT model consistently outperforms all baseline models. Moreover, we can notice from Table 2 that considering node attributes information in DANE and DADSAT yields a large boost in the performance when compared to models that only consider structural properties. The findings have demonstrated that the DADSAT model has contributed by around 3% improvement to all data-sets.

We finally compare the DADSAT with the baseline models over time. Figure 2 depicts the overtime change of the performance of all models. The X-axes represent the time steps, defined as the time-interval snapshots, while the Y-axes represent the average AUC scores. The figure depicts the differences between the DADSAT and baseline models discussed above. As explained earlier, DADSAT outperforms baselines models. Only DySAT's performance is comparable to DADSAT over some time steps. In addition, Fig. 2 indicates that DADSAT generally exhibits superior stability over different time steps in comparison with baseline models. As for the Epinions dataset, DADSAT is the most stable of all. The findings, therefore, demonstrate that DADSAT is not only achieving the highest performance but also exhibiting the highest overtime stability. Taken together, the findings demonstrate that the DADSAT model yields promising improvements to node representation learning.

6 Ablation study

DADSAT is distinguished by its (1) feature extraction module, (2) graph self-attention module, and (3) recursive residual connections. In order to assess the significance of each contribution, we provide the micro-AUC and macro-

Table 2 Performance comparison of micro- and macro-AUC scores (%) observed from different baselines when tested on our datasets

Model	Datasets							
	DBLP		Epinions		Enron		Yelp	
	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC
Deepwalk[1]	72.9±0.1	69.1±0.1	46.9±0.1	47.9±0.1	81.6±0.3	81.2±0.3	67.4±0.1	63.7±0.1
Node2vec[2]	73.3±0.2	69.6±0.1	47.7±0.1	49.2±0.1	83.7±0.2	83.1±0.2	67.4±0.1	65.1±0.1
LINE[7]	69.2±0.2	67.3±0.1	49.8±0.1	51.4±0.1	81.5±0.2	79.8±0.2	66.2±0.3	66.9±0.1
GraphSAGE[5]	74.8±0.3	73.9±0.2	71.7±0.1	71.2±0.2	78.5±0.2	78.3±0.2	67.4±0.1	66.8±0.1
GCN-AE[53]	72.7±0.2	72.4±0.2	69.8±0.1	71.5±0.2	73.5±0.2	74.1±0.2	65.2±0.1	64.4±0.1
GAT-AE[36]	73.5±0.3	73.4±0.3	68.7 ±0.1	69.5±0.2	75.5±0.2	75.7±0.2	67.8±0.1	68.2±0.1
SEAL[17]	75.8±0.3	76.8±0.4	72.3±0.1	72.7±0.2	74.4±0.2	74.7±0.2	70.2±0.3	70.9±0.3
GatedGCN-LSPE[22]	76.8±0.3	77.2±0.3	84.6±0.1	84.5±0.3	79.5±0.2	79.7±0.2	70.8±0.3	71.3±0.4
DynamicTriad[24]	76.7±0.2	76.2±0.2	74.5±0.1	75.7±0.2	81.5±0.2	81.8±0.2	69.2±0.1	69.4±0.1
DANE[11]	75.9±0.2	76.2±0.2	58.1±0.2	57.9±0.2	79.5±0.2	81.5±0.2	68.2±0.1	68.7±0.1
DySAT[9]	76.3±0.2	76.2±0.2	85.1±0.2	85.9±0.2	85.7±0.3	86.6±0.2	70.2±0.1	69.9±0.1
GloDyNE[32]	74.3±0.2	74.6±0.2	77.1±0.2	77.5±0.3	82.4±0.3	82.6±0.2	70.1±0.3	70.3±0.2
DynG2G[28]	76.9±0.2	77.2±0.3	84.6±0.2	84.8±0.2	86.1±0.3	86.4±0.2	70.5±0.3	70.8±0.3
DADSAT(ours)	**79.9±0.2**	**79.4±0.2**	**87.3±0.2**	**87.6±0.2**	**87.6±0.2**	**88.4±0.2**	**73.1±0.2**	**73.6±0.2**

The bold values indicate the best results

[1] In the appendix, we elaborate on experimental settings and baselines hyper-parameter tuning in detail.

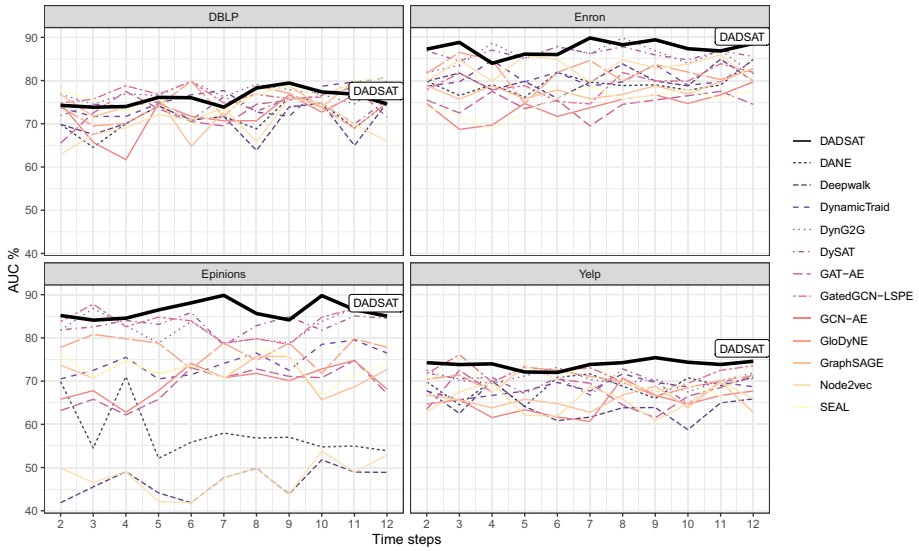

Fig. 2 Performance comparison of micro-AUC and macro-AUC scores (%) observed from different baselines on different time steps

AUC scores with the following changes to DADSAT: (1) without the feature extraction module, (2) without recursive residual connections, and (3) with varied embedding sizes.

In the first experiment, we investigated the impact of considering node attribute information on the performance of the DADSAT model. Here, we have tried to determine how big the difference is between retaining both the structural properties and the node attributes and just considering network structure. Generally, it can be concluded that in all cases, considering node attributes in the DADSAT model yields improvements of 1–3% over the initial model in the four datasets as shown in Table 3. It is clear that considering the high-quality representation of node attributes achieves higher performance, manifested by higher AUC scores, compared to just considering structural properties. Hence, we can infer that just considering structural properties does not convey node information properly.

In Table 4, we explore the influence of using residual connections on the DADSAT model. The findings indicate that employing residual connections can improve the performance of the DADSAT model by around 0.7–1.2%. This provides empirical evidence that these residual connections could be efficient in enhancing the DADSAT architecture. Moreover, this improvement is better shown in the DADSAT model when considering node attributes. A possible explanation for this improvement is that the DADSAT model holds two advantages: (1) It learns a meaningful representation of node attributes, and (2) it includes more detailed information about the network properties.

Finally, we investigate the effect of various embedding sizes on the DADSAT model. The findings show that a small embedding dimension does not incorporate enough information, resulting in poor DADSAT model performance; increasing the size improves performance, as shown in Table 5. However, increasing the embedding dimension beyond a certain point lowers the model's performance.

7 Computational complexity analysis

DADSAT consists of the BERT model as a module for feature extraction and a graph self-attention module. Thus, we separate the time complexity of DADSAT into the time complexity of the graph self-attention module and the time complexity of the BERT model. In the graph self-attention module, the nodes are encoded via a transformer backbone.

According to the analysis in [9], the graph self-attention module has a time complexity of $\mathcal{O}(|V|T^2d + |V|d^2T + \sum_{t=1}^{T} E_t)$ where V is the node set; d is the embedding dimension; and E_t is the link set per snapshot t over T snapshots. The computational complexity of the feature extraction module is $\mathcal{O}(|V|T^2F + |V|F^2T)$, where F is the latent dimension. In summary, the DADSAT's computational complexity is $\mathcal{O}(|V|T^2d + |V|d^2T + |V|T^2F + |V|F^2T + \sum_{t=1}^{T} E_t)$. Since the DADSAT architecture is fully parallelizable, it is scalable for larger datasets compared to CNN- and RNN-based methods.

8 Conclusion and future work

In this paper, we propose a node embedding model from a different perspective, namely DADSAT. To extract meaningful representations of node attributes, motivated by the intuition that node attribute information should be highly correlated to graph dynamics and structural information, we preserve the node attributes by employing an unsupervised feature extraction module that captures node feature infor-

Table 3 Performance comparison of micro- and macro-AUC scores (%) observed from different baselines when tested on our datasets

Model	Datasets							
	DBLP		Epinions		Enron		Yelp	
	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC
DADSAT$_{w/oDistilBERT}$	77.62±0.2	77.71±0.2	86.15±0.2	86.59±0.2	86.60 ±0.2	87.20±0.2	71.57±0.2	71.16±0.2
DADSAT$_{Word2Vec}$	78.24±0.2	78.2±0.2	86.45±0.1	86.81±0.1	86.93±0.2	87.49 ±0.2	72.19±0.2	71.91±0.2
DADSAT$_{DistilBERT}$	**79.93±0.1**	**79.35 ±0.2**	**87.25±0.1**	**87.59±0.1**	**87.60± 0.2**	**88.40±0.2**	**73.10±0.2**	**73.60±0.2**

The bold values indicate the best results

Table 4 The effect of recursive residual connections on DADSAT model in terms of micro-AUC and macro-AUC scores (%)

Model	Datasets							
	DBLP		Epinions		Enron		Yelp	
	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC
DADSAT$_{w/oResidual}$	78.79±0.2	78.93±0.2	86.75±0.2	86.89±0.2	86.9±0.2	87.4±0.2	72.7±0.2	72.9±0.2
DADSAT$_{Residual}$	**79.93±0.1**	**79.35 ±0.2**	**87.25±0.1**	**87.59 ±0.1**	**87.60±0.2**	**88.40±0.2**	**73.10±0.2**	**73.60±0.2**

The bold values indicate the best results

Table 5 The effect of embedding size (d) on DADSAT model in terms of micro-AUC and macro-AUC scores (%)

Embedding size (d)	Datasets							
	DBLP		Epinions		Enron		Yelp	
	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC	Micro-AUC	Macro-AUC
d = 64	76.29±0.2	76.66±0.2	85.55±0.2	85.84±0.2	85.87±0.2	86.23±0.2	70.87±0.2	71.11±0.2
d = 128	**79.93±0.1**	**79.35±0.2**	**87.25±0.1**	**87.59±0.1**	**87.60±0.2**	**88.40±0.2**	**73.10±0.2**	**73.60±0.2**
d = 256	78.64±0.2	78.72±0.2	85.85±0.1	86.21±0.1	86.47±0.2	86.89±0.2	71.59±0.2	71.96±0.3

The bold values indicate the best results

mation. After node feature extraction, we employ the graph self-attention module to hierarchically extract structural and temporal information in the graph. Our model captures graph structure, temporal patterns, and node attribute information and thus learns high-quality node representations in dynamic attributed graphs. The experimental results show that preserving attribute information while preserving network properties and temporal patterns effectively enhances the quality of node representations. Our approach proves promising for downstream applications.

Learning node embeddings on dynamic attributed graphs is still an open question, and there are several techniques and application-related aspects that might be investigated. For instance, learning about attribute-missing graphs might be considered for this problem. In addition, more sophisticated graph data, such as heterogeneous dynamic attributed graphs, might potentially be an intriguing topic to solve. These will be the subject of future research.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s41060-022-00376-3.

Author Contributions NA and AR conceived of the presented idea. NA developed the theory and the methodology. NA performed the experiments, and NA, AR and LST verified the proposed methodology. All authors discussed the findings and contributed to the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. The authors received no financial support for the research and the authorship of this manuscript.

Declarations

Conflict of interest The authors declare that there are no conflicts of interest to disclose.

References

1. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 701–710 (2014)

2. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 855–864 (2016)
3. Cai, H., Zheng, V.W., Chang, K.C.-C.: A comprehensive survey of graph embedding: problems, techniques and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018)
4. Lenßen, J.: Including attributes in graph embeddings (2018)
5. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in neural information processing systems, pp. 1024–1034 (2017)
6. Buford, J., Yu, H., Lua, E.K.: P2P Networking and applications, pp. 131–210 (2009)
7. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, pp. 1067–1077 (2015)
8. Zhang, Z., Yang, H., Bu, J., Zhou, S., Yu, P., Zhang, J., Ester, M., Wang, C.: Anrl: attributed network representation learning via deep neural networks. In: IJCAI, vol. 18, pp. 3155–3161 (2018)
9. Sankar, A., Wu, Y., Gou, L., Zhang, W., Yang, H.: Dysat: deep neural representation learning on dynamic graphs via self-attention networks. In: Proceedings of the 13th international conference on web search and data mining, pp. 519–527 (2020)
10. Du, L., Wang, Y., Song, G., Lu, Z., Wang, J.: Dynamic network embedding: an extended approach for skip-gram based network embedding. In: Proceedings of the 27th international joint conference on artificial intelligence, pp. 2086–2092 (2018)
11. Li, J.-H., Wang, C.-D., Huang, L., Huang, D., Lai, J.-H., Chen, P.: Attributed network embedding with micro-meso structure. In: International conference on database systems for advanced applications, pp. 20–36 (2018). Springer
12. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
13. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Nips, vol. 14, pp. 585–591 (2001)
14. Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. In: Proceedings of the 22nd international conference on world wide web, pp. 37–48 (2013)
15. Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. In: International conference on learning representations (2018)
16. Rashed, A., Grabocka, J., Schmidt-Thieme, L.: Multi-relational classification via bayesian ranked non-linear embeddings. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 1132–1140 (2019)
17. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. Advances in neural information processing systems, 31 (2018)
18. Nakis, N., Çelikkanat, A., Mørup, M.: Hm-ldm: A hybrid-membership latent distance model. arXiv preprint arXiv:2206.03463 (2022)
19. Welling, M., Kipf, T.N.: Semi-supervised classification with graph convolutional networks. In: J. International conference on learning representations (ICLR 2017) (2016)
20. Gao, H., Wang, Z., Ji, S.: Large-scale learnable graph convolutional networks. In: Proceedings of the 24th ACM SIGKDD International conference on knowledge discovery & data mining, pp. 1416–1424 (2018)
21. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International conference on learning representations (2018)
22. Dwivedi, V.P., Luu, A.T., Laurent, T., Bengio, Y., Bresson, X.: Graph neural networks with learnable structural and positional representations. In: International conference on learning representations (2021)
23. Barros, C.D.T., Mendonça, M.R.F., Vieira, A.B., Ziviani, A.: A survey on embedding dynamic graphs (2021)
24. Zhou, L., Yang, Y., Ren, X., Wu, F., Zhuang, Y.: Dynamic network embedding by modeling triadic closure process. In: Proceedings of the AAAI conference on artificial intelligence, vol. 32 (2018)
25. Goyal, P., Chhetri, S.R., Canedo, A.: dyngraph2vec: capturing network dynamics using dynamic graph representation learning. Knowl.-Based Syst. **187**, 104816 (2020)
26. Béres, F., Kelen, D.M., Pálovics, R., Benczúr, A.A.: Node embeddings in dynamic graphs. Appl. Netw. Sci. **4**(1), 1–25 (2019)
27. Torricelli, M., Karsai, M., Gauvin, L.: weg2vec: event embedding for temporal networks. Scientif. Rep. **10**(1), 7164–7164 (2020)
28. Xu, M., Singh, A.V., Karniadakis, G.E.: Dyng2g: an efficient stochastic graph embedding method for temporal graphs. IEEE Trans. Neur. Netw. Learn. Sys. (2022). https://doi.org/10.1109/TNNLS.2022.3178706
29. Nguyen, G.H., Lee, J.B., Rossi, R.A., Ahmed, N.K., Koh, E., Kim, S.: Continuous-time dynamic network embeddings. In: Companion proceedings of the the web conference 2018, pp. 969–976 (2018)
30. Mahdavi, S., Khoshraftar, S., An, A.: dynnode2vec: Scalable dynamic network embedding. In: 2018 IEEE International conference on big data (Big Data). IEEE, pp. 3762–3765 (2018)
31. Lazaridou, A., Baroni, M., *et al.*: Combining language and vision with a multimodal skip-gram model. In: Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies, pp. 153–163 (2015)
32. Hou, C., Zhang, H., He, S., Tang, K.: Glodyne: global topology preserving dynamic network embedding. IEEE Trans. Knowl. Data Eng. (2020). https://doi.org/10.1109/TKDE.2020.3046511
33. Singer, U., Guy, I., Radinsky, K.: Node embedding over temporal graphs. In: 28th international joint conference on artificial intelligence, IJCAI 2019, pp. 4605–4612 (2019)
34. Bielak, P., Tagowski, K., Falkiewicz, M., Kajdanowicz, T., Chawla, N.V.: Fildne: a framework for incremental learning of dynamic networks embeddings. Knowl.-Based Sys. **236**, 107453 (2022)
35. Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., Leiserson, C.: Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In: Proceedings of the AAAI conference on artificial intelligence, vol. 34, pp. 5363–5370 (2020)
36. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: 6th International conference on learning representations, ICLR 2018, Vancouver, Canada, Conference Track Proceedings (2018)
37. Nguyen, D.Q., Nguyen, T.D., Phung, D.: A self-attention network based node embedding model. In: Machine learning and knowledge discovery in databases - European Conference, ECML PKDD 2020, Ghent, Belgium, 2020, Proceedings, Part III, pp. 364–377 (2020)
38. Divakaran, A., Mohan, A.: Temporal link prediction: a survey. New generation computing, 1–46 (2019)
39. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: RoBERTa: a robustly optimized bert pretraining approach (2019)
40. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: ALBERT: a lite BERT for self-supervised learning of language representations (2020)
41. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (2020)
42. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: 3rd International con-

ference on learning representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference track proceedings (2015)
43. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems, pp. 5998–6008 (2017)
44. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on computer vision and pattern recognition, pp. 770–778 (2016)
45. Liu, F., Ren, X., Zhang, Z., Sun, X., Zou, Y.: Rethinking skip connection with layer normalization. In: Proceedings of the 28th international conference on computational linguistics, pp. 3586–3598 (2020)
46. Ley, M.: Dblp: some lessons learned. Proceed. VLDB Endowm. **2**(2), 1493–1500 (2009)
47. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. Knowl. Inf. Sys. **42**(1), 181–213 (2015)
48. Klimt, B., Yang, Y.: Introducing the enron corpus. In: CEAS (2004)
49. Luca, M.: Reviews, reputation, and revenue: The case of yelp. com. Com (March 15, 2016). Harvard business school nom unit working paper (12-016) (2016)
50. Meyffret, S., Guillot, E., Médini, L., Laforest, F.: Red: a rich epinions dataset for recommender systems (2012)
51. Li, J., Dani, H., Hu, X., Tang, J., Chang, Y., Liu, H.: Attributed network embedding for learning in a dynamic environment. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp. 387–396 (2017)
52. Zitnik, M., Agrawal, M., Leskovec, J.: Modeling polypharmacy side effects with graph convolutional networks. Bioinformatics **34**(13), 457–466 (2018)
53. Bresson, X., Laurent, T.: Residual gated graph convnets. arXiv preprint arXiv:1711.07553 (2017)
54. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 (2016)