DYNAMIC GRAPH REPRESENTATION LEARNING VIA SELF-ATTENTION NETWORKS

Aravind Sankar*
Department of Computer Science
University of Illinois, Urbana-Champaign
asankar3@illinois.edu

Yanhong Wu, Liang Gou, Wei Zhang & Hao Yang
Visa Research
Palo Alto, CA
{yanwu,ligou,wzhan,haoyang}@visa.com

ABSTRACT

Learning latent node representations in graphs is an important task with widespread applications such as link prediction, node classification, and graph visualization. In this paper, we present Dynamic Self-Attention Network (DySAT), a novel neural architecture that operates on dynamic graphs and learns node representations by jointly employing self-attention layers along two dimensions: structural neighborhood and temporal dynamics. Compared with current methods modeling temporal graph evolution, dynamic self-attention is efficient, while achieving consistently superior performance. Our experiments on two graph classes: communication networks and bipartite rating networks indicate significant gains for DySAT has over several state-of-the-art graph embedding baselines, in both single and multi-step link prediction tasks.

1 INTRODUCTION

Learning latent representations of nodes in graphs is a fundamental problem due to its widespread applications in social media (Perozzi et al., 2014), biology (Grover & Leskovec, 2016), knowledge bases (Wang et al., 2014), etc. The goal is to learn low-dimensional vectors that encode the structural properties of a node and its neighborhood. Previous work mainly focuses on static graphs (Tang et al., 2015; Hamilton et al., 2017b) while many real-world graphs are intrinsically dynamic with constant evolution over time, and are usually represented as a sequence of snapshots at different time steps (Leskovec et al., 2007) such as co-authorship networks where authors may periodically switch their collaboration patterns. In such scenarios, the node representations should not only preserve structural proximity, but also jointly capture the temporal dependencies.

Recently, graph neural networks have significantly advanced representation learning in static graphs (Kipf & Welling, 2017; Velickovic et al., 2018; Hamilton et al., 2017b). Most existing work in dynamic graphs impose temporal smoothness of node representations from adjacent snapshots (Zhu et al., 2016; Zhou et al., 2018), which may fail when nodes exhibit distinct evolutionary trends.

Attention mechanisms have recently achieved great success in many sequential learning tasks (Bahdanau et al., 2015) by learning a function that aggregates a variable-sized input, while focusing on the most relevant parts to a certain context. *Self-attention* (Vaswani et al., 2017) uses a single sequence as both the inputs and context, and has achieved state-of-the-art performance in several tasks, while being significantly efficient in computation. Velickovic et al. (2018) extends self-attention to graphs by enabling each node to attend over its neighbors for representation learning in static graphs.

As dynamic graphs usually have periodical patterns such as recurrent links or communities, attention can focus on the most relevant historical snapshot(s), to facilitate future prediction. We present a novel Dynamic Self-Attention Network (DySAT) for dynamic graph representation learning that employs self-attention along two dimensions: structural neighborhoods and temporal dynamics. Structural attention extracts features from local node neighborhoods through self-attentional aggregation, while temporal attention captures graph structure evolution over multiple time steps.

We evaluate DySAT on single and multi-step link prediction on two communication networks (Klimt & Yang, 2004; Panzarasa et al., 2009) and two bipartite rating networks (Harper & Konstan, 2016).

*Work done while at Visa Research

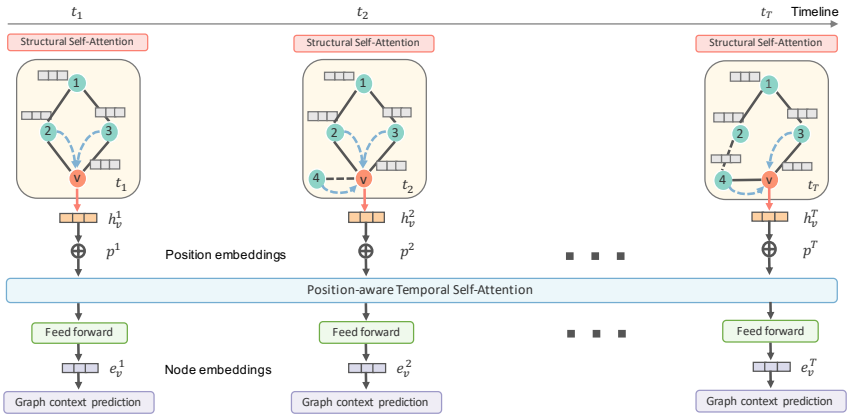

Figure 1: Neural architecture of DySAT: we employ structural attention layers followed by temporal attention layers. Dashed black lines indicate new links and dashed blue arrows refer to neighbor-based structural-attention.

DySAT achieves significant gains (4.8% macro-AUC on average) over several state-of-the-art baselines and maintains a consistently more stable performance over time.

2 Problem Dfinition

In this work, we address the problem of dynamic graph representation learning. A dynamic graph is a series of graph snapshots $\mathbb{G} = \{\mathcal{G}^1, \ldots, \mathcal{G}^T\}$ where T is the number of time steps. Each snapshot $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}^t)$ is a weighted undirected graph with a shared node set $\mathcal{V}$, link set $\mathcal{E}^t$, and weighted adjacency matrix $\boldsymbol{A}^t$. Dynamic graph representation learning aims to learn representations $\boldsymbol{e}_v^t \in \mathbb{R}^d$ for each node $v \in \mathcal{V}$ at time steps $t = \{1, 2, \ldots, T\}$, such that $\boldsymbol{e}_v^t$ preserves both the local graph structures centered at v and its temporal evolutionary behaviors up to time step t.

3 Dynamic Self-Attention Network

In this section, we first describe the high-level structure of our model. DySAT has three major building blocks as illustrated in Figure 1, (1) *structural* attention block, (2) *temporal* attention block, and (3) *graph context* prediction. We present each module in detail:

3.1 Structural Self-Attention

The input of this layer is a snapshot $\mathcal{G} \in \mathbb{G}$ and input representations $\{\boldsymbol{x}_v \in \mathbb{R}^D, \forall v \in \mathcal{V}\}$. The initial layer either takes one-hot vectors for each node or node attributes. The output is a new set of node representations $\{\boldsymbol{z}_v \in \mathbb{R}^F, \forall v \in \mathcal{V}\}$ that capture local structural properties in snapshot $\mathcal{G}$.

Specifically, *structural* self-attention attends over the immediate neighbors of a node v in $\mathcal{G}$, by computing attention weights as a function of their input representations. We set the structural attention layer as a variant of GAT (Velickovic et al., 2018) applied on a single snapshot $\mathcal{G}$, defined by:

$$\boldsymbol{z}_v = \sigma\Big(\sum_{u \in \mathcal{N}_v} \alpha_{uv} \boldsymbol{W}^s \boldsymbol{x}_u\Big),\ \alpha_{uv} = \frac{\exp(e_{uv})}{\sum_{w \in \mathcal{N}_v} \exp(e_{wv})},\ e_{uv} = \sigma\Big(A_{uv} \cdot \boldsymbol{a}^T [\boldsymbol{W}^s \boldsymbol{x}_u || \boldsymbol{W}^s \boldsymbol{x}_v]\Big)\ \forall (u, v) \in \mathcal{E}$$

where $\mathcal{N}_v$ is the set of immediate neighbors of v in $\mathcal{G}$; $\boldsymbol{W}^s \in \mathbb{R}^{D \times F}$ is a shared weight transform; $\boldsymbol{a} \in \mathbb{R}^{2D}$ parameterizes the attention function; $||$ denotes concatenation and $\sigma(\cdot)$ is a non-linear activation. The coefficients α_{uv}, obtained by softmax over neighbors, indicate the importance of u to v at the current snapshot. Thus, structural attention computes a self-attentional aggregation of neighboring embeddings, which constitutes a single round of message passing through the graph.

3.2 TEMPORAL SELF-ATTENTION

To capture temporal evolution in dynamic graphs, we design a temporal self-attention layer that takes as input, a sequence of representations that are assumed to capture local structure at each time step. Specifically, the input for node v is $\{\boldsymbol{x}_v^1, \boldsymbol{x}_v^2, \ldots, \boldsymbol{x}_v^T\}, \boldsymbol{x}_v^t \in \mathbb{R}^{D'}$. The output is a series of representations for v at each time step, $\boldsymbol{z}_v = \{\boldsymbol{z}_v^1, \boldsymbol{z}_v^2, \ldots, \boldsymbol{z}_v^T\}, \boldsymbol{z}_v^t \in \mathbb{R}^{F'}$. We denote the input and output representations of v, packed across time, by $\boldsymbol{X}_v \in \mathbb{R}^{T \times D'}$ and $\boldsymbol{Z}_v \in \mathbb{R}^{T \times F'}$ respectively.

We use $\boldsymbol{x}_v^t$ as the query to attend over its historical representations ($< t$), tracing the evolution of the local neighborhood around v to learn dependencies across different time steps. In contrast to structural attention which operates on neighbor representations, temporal attention depends entirely on the temporal history of each node, thus facilitating efficient parallelism. The decoupling of local neighborhood and temporal history of each node into independent layers, is one of the key factors contributing to the efficiency of our model.

We use scaled dot-product attention (Vaswani et al., 2017) where the queries, keys, and values (set to $\boldsymbol{X}_v$) are first transformed via linear projections $\boldsymbol{W}_q \in \mathbb{R}^{D' \times F'}, \boldsymbol{W}_k \in \mathbb{R}^{D' \times F'}$ and $\boldsymbol{W}_v \in \mathbb{R}^{D' \times F'}$ respectively. We preserve the auto-regressive property by allowing time-step t to attend over all steps up to and including t using a mask matrix $\boldsymbol{M} \in \mathbb{R}^{T \times T}$ with entries in $\{-\infty, 0\}$. $M_{ij} = -\infty$ switches off the attention from time-step i to j when $i > j$. Temporal self-attention is defined as:

$$\boldsymbol{Z}_v = \boldsymbol{\beta_v}(\boldsymbol{X}_v \boldsymbol{W}_v), \quad \beta_v^{ij} = \frac{\exp(e_v^{ij})}{\sum_{k=1}^{T} \exp(e_v^{ik})} \quad e_v^{ij} = \Big(\frac{((\boldsymbol{X}_v \boldsymbol{W}_q)(\boldsymbol{X}_v \boldsymbol{W}_k)^T)_{ij}}{\sqrt{F'}} + M_{ij} \Big)$$

where $\boldsymbol{\beta_v} \in \mathbb{R}^{T \times T}$ is the attention weight matrix obtained by the multiplicative attention function.

We employ multi-head attention (Vaswani et al., 2017) to increase model capacity and stability by enabling joint attention over different subspaces. The multi-head attention is defined by:

$$\text{Structural multi-head self-attention:} \qquad \boldsymbol{h}_v = \text{Concat}(\boldsymbol{z}_v^1, \boldsymbol{z}_v^2, \ldots, \boldsymbol{z}_v^{H_S}) \;\; \forall v \in \mathcal{V} \tag{1}$$

$$\text{Temporal multi-head self-attention:} \qquad \boldsymbol{H}_v = \text{Concat}(\boldsymbol{Z}_v^1, \boldsymbol{Z}_v^2, \ldots, \boldsymbol{Z}_v^{H_T}) \;\; \forall v \in \mathcal{V} \tag{2}$$

where H_S and H_T denote the number of structural and temporal attention heads, $\boldsymbol{h}_v \in \mathbb{R}^F$ and $\boldsymbol{H}_v \in \mathbb{R}^{T \times F'}$ are the outputs of structural and temporal multi-head attentions respectively.

3.3 DYSAT ARCHITECTURE

The input is a collection of T snapshots and the outputs are node representations at each time step. The *structural* block extracts features from higher-order neighborhoods via self-attentional aggregation and stacking, to compute intermediate node embeddings for each snapshot. These sequences feed into the *temporal* block, which captures temporal evolution of graph structure. The outputs comprise dynamic node representations optimized to preserve local graph context at each time step.

Structural attention block. This module comprises multiple stacked layers to extract features from nodes at different distances. We apply each layer independently at different snapshots with shared parameters, to capture local neighborhood structures at each time step. The output of the structural block is given by $\{\boldsymbol{h}_v^1, \boldsymbol{h}_v^2, \ldots, \boldsymbol{h}_v^T\}, \boldsymbol{h}_v^t \in \mathbb{R}^f$, which feed as input to the *temporal* block.

Temporal attention block. We use *position* embeddings (Gehring et al., 2017), $\{\boldsymbol{p}^1, \ldots, \boldsymbol{p}^T\}, \boldsymbol{p}^t \in \mathbb{R}^f$ to encode the absolute temporal position of each snapshot. The output of the structural block is added with the position embeddings to define the input representation sequence as: $\{\boldsymbol{h}_v^1 + \boldsymbol{p}^1, \boldsymbol{h}_v^2 + \boldsymbol{p}^2, \ldots, \boldsymbol{h}_v^T + \boldsymbol{p}^T\}$ for node v. The outputs after multiple stacked layers pass into a position-wise *feed-forward* layer to give the final node representations $\{\boldsymbol{e}_v^1, \boldsymbol{e}_v^2, \ldots, \boldsymbol{e}_v^T\}$ $\forall v \in \mathcal{V}$.

Graph context prediction. To model both structural and temporal information, our loss function preserves local structure across multiple time steps. We use a binary cross-entropy loss at each time step to encourage nodes co-occurring in fixed-length random walks, to have similar representations.

$$L = \sum_{t=1}^{T} \sum_{v \in \mathcal{V}} \Big(\sum_{u \in \mathcal{N}_{walk}^t(v)} -\log\left(\sigma(\langle \boldsymbol{e}_u^t, \boldsymbol{e}_v^t \rangle)\right) - w_n \cdot \sum_{u' \in P_n^t(v)} \log\left(1 - \sigma(\langle \boldsymbol{e}_{u'}^t, \boldsymbol{e}_v^t \rangle)\right) \Big) \tag{3}$$

where σ is the sigmoid function, $\langle . \rangle$ denotes inner product, $\mathcal{N}_{walk}^t(v)$ is the set of nodes that co-occur with v on fixed-length random walks in $\mathcal{G}_t$, P_n^t is a negative sampling distribution for $\mathcal{G}^t$ (usually a function of node degrees), and w_n is a hyper-parameter to balance the positive and negative samples.

Method	Enron		UCI		Yelp		ML-10M	
	Micro-AUC	**Macro-AUC**	**Micro-AUC**	**Macro-AUC**	**Micro-AUC**	**Macro-AUC**	**Micro-AUC**	**Macro-AUC**
node2vec	83.72 ± 0.7	83.05 ± 1.2	79.99 ± 0.4	80.49 ± 0.6	67.86 ± 0.2	65.34 ± 0.2	87.74 ± 0.2	87.52 ± 0.3
G-SAGE	$82.48^{*} \pm 0.6$	$81.88^{*} \pm 0.5$	$79.15^{*} \pm 0.4$	$82.89^{*} \pm 0.2$	$60.95^{\dagger} \pm 0.1$	$58.56^{\dagger} \pm 0.2$	$86.19^{\ddagger} \pm 0.3$	$89.92^{\ddagger} \pm 0.1$
G-SAGE + GAT	72.52 ± 0.4	73.34 ± 0.6	74.03 ± 0.4	79.83 ± 0.2	66.15 ± 0.1	65.09 ± 0.2	83.97 ± 0.3	84.93 ± 0.1
GCN-AE	81.55 ± 1.5	81.71 ± 1.5	80.53 ± 0.3	83.50 ± 0.5	66.71 ± 0.2	65.82 ± 0.2	85.49 ± 0.1	85.74 ± 0.1
GAT-AE	75.71 ± 1.1	75.97 ± 1.4	79.98 ± 0.2	81.86 ± 0.3	65.92 ± 0.1	65.37 ± 0.1	87.01 ± 0.2	86.75 ± 0.2
DynamicTriad	80.26 ± 0.8	78.98 ± 0.9	77.59 ± 0.6	80.28 ± 0.5	63.53 ± 0.3	62.69 ± 0.3	88.71 ± 0.2	88.43 ± 0.1
DynGEM	67.83 ± 0.6	69.72 ± 1.3	77.49 ± 0.3	79.82 ± 0.5	66.02 ± 0.2	65.94 ± 0.2	73.69 ± 1.2	85.96 ± 0.3
DynAERNN	72.02 ± 0.7	72.01 ± 0.7	79.95 ± 0.4	83.52 ± 0.4	69.54 ± 0.2	68.91 ± 0.2	87.73 ± 0.2	89.47 ± 0.1
DySAT	$\mathbf{85.71} \pm 0.3$	$\mathbf{86.60} \pm 0.2$	$\mathbf{81.03} \pm 0.2$	$\mathbf{85.81} \pm 0.1$	$\mathbf{70.15} \pm 0.1$	$\mathbf{69.87} \pm 0.1$	$\mathbf{90.82} \pm 0.3$	$\mathbf{93.68} \pm 0.1$

Table 1: Single-step link prediction (micro and macro AUC with std. dev). We show GraphSAGE (G-SAGE) with the best aggregators ($*$, †, and ‡ denote GCN, LSTM, and max-pool respectively).

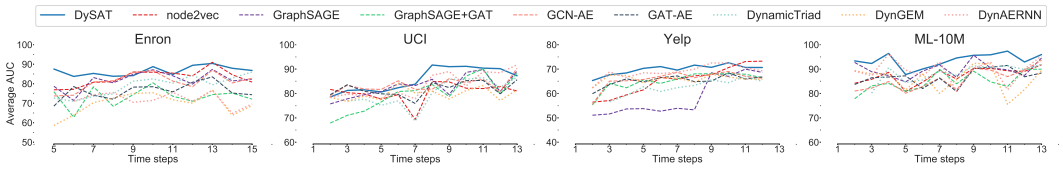

Figure 2: Performance comparison of DySAT on single-step link prediction: solid line denotes DySAT; dashed and dotted lines denote static and dynamic embedding baselines respectively.

4 Experiments

We compare DySAT against a variety of static and dynamic graph embedding baselines on four publicly available benchmarks and observe significant gains for DySAT over other methods.

4.1 Experimental Setup

We conduct experiments on single-step and multi-step link prediction (or forecasting). The single-step setting uses the latest embeddings at time step t to predict the links at $t+1$. We use a downstream logistic regression classifier by creating evaluation examples from the links in $\mathcal{G}^{t+1}$ and an equal number of randomly sampled pairs of non-links. In the multi-step scenario, the embeddings predict links at multiple future time steps $\{t+1, \ldots, t+\Delta\}$. In each dataset, we choose the latest $\Delta = 6$ snapshots for evaluation. We use AUC scores to evaluate link prediction.

Datasets. We use four datasets with graph sizes similar to Goyal et al. (2017; 2018) (Appendix **??**).

- **Communication networks.** We use two datasets: Enron (Klimt & Yang, 2004) and UCI (Panzarasa et al., 2009). In Enron, the links denote email interactions between core employees, while the links in UCI represent messages sent between peer users on an online social network platform.
- **Rating networks.** We examine two bipartite networks from Yelp and MovieLens. Yelp comprises links between users and businesses, derived from user ratings. ML-10M consists of a user-tag network where the links connect users with the tags they applied on movies.

Baselines. We compare DySAT with static embedding methods by aggregating the entire history of snapshots upto time t into a single graph, agnostic to time. We present comparisons with node2vec (Grover & Leskovec, 2016), GraphSAGE (Hamilton et al., 2017b), and graph autoencoders (Hamilton et al., 2017a), *i.e.*, GCN and GAT for link prediction (Zitnik et al., 2018), denoted by GCN-AE and GAT-AE respectively. We also include an attentional aggregator in the GraphSAGE framework, denoted by GraphSAGE + GAT. In the dynamic setup, we evaluate DynamicTriad (Zhou et al., 2018), DynGEM (Goyal et al., 2017), and DynAERNN (Goyal et al., 2018).

4.2 Experimental Results

Single-Step Link Prediction. We evaluate the methods at each time step t by training separate models up to snapshot t for each $t = 1, \ldots, T$. From Table 1, DySAT achieves consistent gains of 4–5% macro-AUC, in comparison to the best baseline across all datasets. DynAERNN typically comes

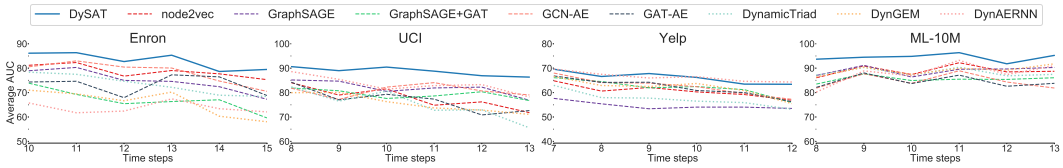

Figure 3: Performance comparison of DySAT on multi-step link prediction for the next 6 time steps.

second-best, validating the effectiveness of RNN-based methods. The performance comparison of different graph embedding methods yields several interesting insights.

First, GraphSAGE performs comparably with dynamic baselines despite being trained on static graphs. One possible reason is that GraphSAGE uses trainable neighbor-aggregation, while the dynamic methods either employ Skip-gram or adjacency reconstruction techniques to model structural proximity. This affirms the superior performance of DySAT which uses aggregations and multi-head attentions for joint structural and temporal modeling. Second, node2vec achieves consistent performance while being trained agnostic to temporal graph evolution. This points to further gains on applying second-order random walk sampling techniques to DySAT.

We also compare models at each time step (Figure 2) to obtain a deep understanding of their temporal behaviors. DySAT achieves a more stable performance especially in Enron and UCI, compared with static methods that encounter drastic performance drops at certain time steps.

Multi-Step Link Prediction. Now, we present results of different graph embedding methods on multi-step link prediction. Figure 3 depicts the performance variation over 6 future snapshots, indicating a slight decay over time for all models. DySAT achieves significant gains over the baselines, especially in later time steps. Static methods often exhibit large variations over time, while DySAT maintains a stable and consistently high performance. This demonstrates the capability of temporal attention in DySAT, to capture the most relevant historical context for forecasting.

5 Discussion

In DySAT, the temporal attention layers are stacked on top of structural attention layers. We chose this design since graph structures may not be stable over time, which makes the inverse option impractical. Another design choice we considered is applying self-attention along the two dimensions of neighbors and time together following a strategy similar to DiSAN (Shen et al., 2018). In practice, this would be computationally expensive due to variable number of neighbors per node across multiple snapshots. We leave exploring other architectural choices based on structural and temporal self-attentions as future work.

In the current setup, we store the sparse adjacency matrix of each snapshot in memory, which may pose memory challenges when scaling to large graphs. We plan to explore DySAT with memory-efficient mini-batch training strategies following GraphSAGE (Hamilton et al., 2017b).

DySAT can be directly extended to learn incremental embeddings in a streaming environment, enabling both computational and memory efficiency. We believe this opens the door to future exploration of efficient self-attentional architectures for incremental (or streaming) graph applications.

6 Conclusion

In this paper, we introduce a novel neural network architecture named DySAT that operates on dynamic graphs to learn node representations by jointly employing self-attention layers along two dimensions: structural neighborhoods and temporal dynamics. Our experiments on various real-world dynamic graphs indicate significant gains for DySAT over several state-of-the-art baselines. Though our experiments are conducted on graphs without node features, DySAT can be easily generalized to feature-rich graphs. Another interesting direction is exploring continuous-time generalizations of our framework to learn fine-grained temporal evolutionary patterns.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 1243–1252, 2017.

Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. In *IJCAI International Workshop on Representation Learning for Graphs (ReLiG)*, August 2017.

Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *arXiv preprint arXiv:1809.02657*, 2018.

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, San Francisco, CA, USA, August 13-17, 2016*, pp. 855–864, 2016.

William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017a.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1025–1035, 2017b.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, 5(4):19, 2016.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference for Learning Representations (ICLR)*, 2017.

Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *CEAS 2004 - First Conference on Email and Anti-Spam, July 30-31, 2004, Mountain View, California, USA*, 2004.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

Pietro Panzarasa, Tore Opsahl, and Kathleen M. Carley. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *JASIST*, 60(5):911–932, 2009.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pp. 701–710, 2014.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pp. 1067–1077, 2015. doi: 10.1145/2736277.2741093.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 6000–6010, 2017.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pp. 1112–1119, 2014.

Le-kui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7*, 2018.

Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Trans. Knowl. Data Eng.*, 28(10):2765–2777, 2016.

Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):457466, 2018.