Proceedings of Machine Learning Research 129:609–624, 2020 ACML 2020

Network Representation Learning Algorithm Based on Neighborhood Influence Sequence

Meng Liu 2191438@s.hlju.edu.cn
Ziwei Quan Quan_zwz@163.com
Yong Liu * 2010023@hlju.edu.cn
HeiLongJiang University, Harbin, China

Editors: Sinno Jialin Pan and Masashi Sugiyama

Abstract

Network representation learning (NRL) is playing an important role in network analysis, aiming to represent complex network more concisely by transforming nodes into low-dimensional vectors. However, most of the current work only uses network structure and node attribute to learn network representation, and often ignores the historical interactions between nodes that will affect the future interactions. Therefore, we propose a network representation learning algorithm based on neighborhood influence sequence (NIS), by investigating the influence of node historical interactions on future interactions. We propose three kinds of influence when two nodes interact, and integrate them into NIS by introducing the Hawkes process. In experiments, we compare our model with existing NRL models on four real-world datasets. Experimental results demonstrate that the embedding learned from the proposed NIS model achieve better performance than state-of-the-art methods in various tasks including node classification, link prediction, and network visualization.

Keywords: Temporal Network; Network Representation Learning; Hawkes Process; Neighborhood Influence Sequence

1. Introduction

Network representation learning (NRL), also known as network embedding (NE), aims to represent the large-scale networks by mapping nodes into a low-dimensional space. NRL provides an efficient way to represent the network structure and alleviates the computation and sparsity issues of conventional symbol-based representations Tu et al. (2017).

In recent years, researchers have become increasingly interested in network representation learning Cui et al. (2019). They use NRL algorithms in many real-world domains such as scientific citation and collaboration, communication analysis Cavallari et al. (2017), fraud and terrorist analysis Nguyen et al. (2018), etc. Most of the current works focus on network structure and node attributes at a certain time stamp, i.e., they do not take changes about nodes and edges into account during the learning process. One important but often-overlooked issue underlying these methods is that they seldom focus on temporal interactions between nodes.

In fact, node's interactions occur frequently in real-world networks. By utilizing temporal interaction between nodes in network representation learning, we can capture network

[*] Corresponding Author

dynamic and thus obtain better network embedding. Therefore, recent work has begun to focus on dynamic networks with temporal information.

Dynamic network embedding methods can generally be classified into two main categories: (1) generating static snapshots and (2) learning temporal sequences. The methods for generating static snapshots divide the network into multiple snapshots according to the timestamp, and learn network embedding at each timestamp which is used as initialization for network embedding at the next timestamp. The methods for learning temporal sequences attempt to capture network dynamic from interaction events and generate network embedding.

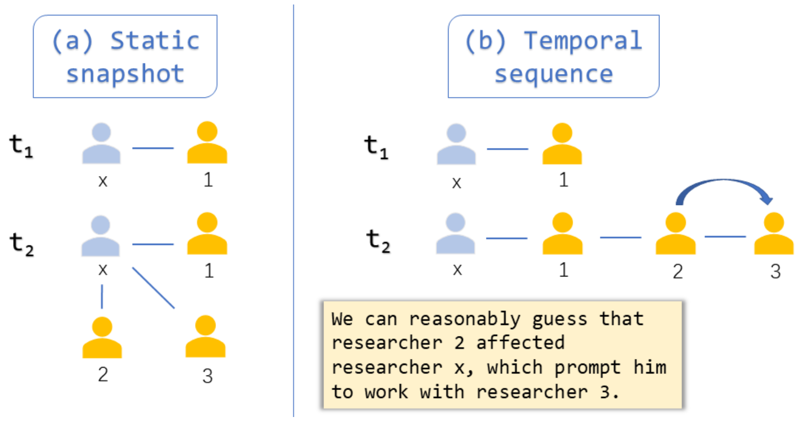
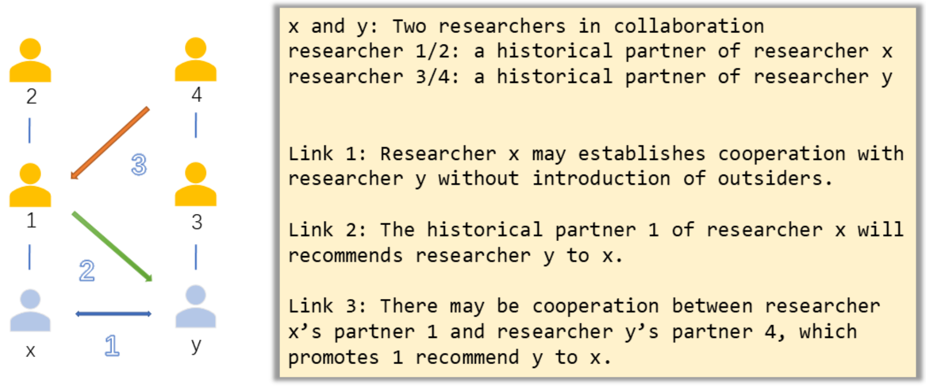

(a) Static snapshot
(b) Temporal sequence
t1
t2
x
1
2
3
We can reasonably guess that researcher 2 affected researcher x, which prompt him to work with researcher 3.

Figure 1: Comparison of static snapshot and temporal sequence

In Figure 1, we show the difference between two dynamic network embedding methods by exemplifying a researcher's co-author activity in timestamp t_1 and t_2. As shown in Figure 1, static snapshot can only show that a researcher x worked with researcher 2 and 3 between t_1 and t_2, but there is no cooperation order of two researchers. In contrast, after generating a temporal sequence, we find that the researcher x firstly worked with researcher 2, and then worked with researcher 3. According to the order, we can reasonably guess that researcher 2 affected researcher x, which prompt him to work with researcher 3. As a result, temporal sequence can contain more information for learning network embedding than static snapshot.

However, current work often ignores the influence of node's neighborhood when they focus on the temporal sequence. They usually consider the changes of a node's property at each time point, but the neighborhood of nodes may also change on time, thus causing different influence on future interaction. Specifically, in Figure 1, when researcher x worked with researcher 2, he might be influenced by researcher 1 he has previously worked with. When he worked with researcher 3, he might be influenced by the first two researchers.

Therefore, we need to focus not only on the change of a node's property over time, but also on the change of its neighborhood in the temporal network. How nodes interact with their neighbors sequentially can reveal the dynamic changes and should be exploited to better represent the network. This paper aims to learn the influence of a node's historical neighbor sequence on its future interactions. According to the Hawkes process theory Hawkes (1971), a node's historical interactions will affect its future interaction. The closer the interaction time is to the current time, the greater the influence. Therefore, inspired by the Hawkes process, we proposed a network representation learning algorithm based on neighborhood influence sequence, called NIS in this paper. In NIS, the interaction possibility between two nodes is not only related to their own attributes, but also to the neighbors that have interacted with the nodes in the past time, i.e., the historical neighbors of a node will influence its future interaction.



```
x and y: Two researchers in collaboration
researcher 1/2: a historical partner of researcher x
researcher 3/4: a historical partner of researcher y


Link 1: Researcher x may establishes cooperation with
researcher y without introduction of outsiders.

Link 2: The historical partner 1 of researcher x will
recommends researcher y to x.

Link 3: There may be cooperation between researcher
x's partner 1 and researcher y's partner 4, which
promotes 1 recommend y to x.
```

Figure 2: Example of the neighborhood influence

In Figure 2, we take the collaboration of researcher x and researcher y as an example to explain the possible influence of this process. First, we define two researchers 1 and 2 as historical partners of researcher x and two researchers 3 and 4 as historical partners of researcher y. In this process, there are three kinds of influence when we consider the possible interaction between researchers x and y.

The first influence is direct mutual influence between two nodes. As shown in link 1 in Figure 2, researcher x may establish cooperation with researcher y without the introduction of outsiders. **The second influence** is the influence of a node's neighborhood embedding on another node. As shown in link 2 in Figure 2, the historical partner 1 of researcher x will recommends researcher y to x. **The third influence** is mutual influence between the two node's neighborhood embedding. As shown in link 3 in Figure 2, there may be cooperation between researcher x's partner 1 and researcher y's partner 4, which promotes 1 recommend y to x. The above three influence constitute the main body of NIS, i.e., learning the influence of neighborhoods on nodes over time.

We conduct extensive experiments on four real-world datasets from different areas and compare NIS with the state-of-the-art algorithms. We show the performance of NIS via various tasks such as node classification Niepert et al. (2016), link prediction, network visualization, etc. The results demonstrate that modeling neighborhood influence sequence is critical for network analysis, in particular for dynamic network with complicated interactions between nodes. The contribution of our work is summarized as follows.

(1) We propose a novel algorithm for network embedding in temporal networks called NIS that take the influence of node neighborhood over time into consideration.

(2) We investigate three kinds of influence that will occur when two nodes intercat, and utilize them to obtain network embedding in NIS.

(3) We empirically evaluate NIS on several real-world datasets and show its superior performance.

The rest of the paper is organized as follows. We define the network in Section 2. We describe our network representation learning algorithm NIS in detail in Section 3. In Section 4 we introduce the experimental setup and results. We discuss related work in Section 5 and conclude the paper in Section 6. The source code and data for this paper can be downloaded from https://github.com/MGitHubL/NIS.

2. Network Definition

As discussed previously, the probability of future interaction between two nodes will be influenced by their neighbors which the two nodes have interacted with in the past. According to the interaction of each node, we can formally define temporal network as follows.

Definition 1. Temporal Network: Temporal network is a network with edges annotated by chronological interactive events between nodes. Suppose there is a temporal network $G = (V,\ E,\ O)$, where V is the set of nodes, $E \subseteq V \times V$ are the set of edges, and O denotes the set of events. Each event $(x,\ y,\ t) \in O$ represents an interaction between nodes x and y at time t. $O_{x,y}$ denotes the set of events between node x and node y, i.e., $O_{x,y} = \{(x, y, t_1), (x, y, t_2), \cdots\cdots, (x, y, t_n)\}$.

We believe that two nodes may interact at multiple moments. For example, user x and user y jointly published a paper in 2019, then user x and user y jointly published another paper in 2020. If we regard the nodes which a node has interacted with in the past as its neighbors, we can obtain its neighborhood influence sequence which is defined as follows.

Definition 2. Neighborhood Influence Sequence: Given a source node $x \in V$, we can obtain its neighborhood influence sequence N_x, i.e., $N_x = \{(y_1, t_1), (y_2, t_2), \cdots, (y_n, t_n)\}$. Each tuple in the sequence represents an event, i.e., the target node y_i interacts with the source node x at time t_i.

According to the neighborhood influence sequence of a node x, we can calculate the influence of neighbors on future interaction between x and other nodes, and learn an embedding for node x.

Definition 3. Temporal Network Embedding: Given a temporal network $G = (V,\ E,\ O)$, the neighbors and the corresponding chronological events of each node $x \in V$ can be induced into a neighborhood influence sequence N_x by tracking all events in which x interacts with its neighbors. Then, temporal network embedding aims to learning a D-dimensional vector to represent each node, which is indeed learning a mapping function $\phi : V \to D$, where $D \ll |V|$.

3. Method

3.1. Hawkes Process

Since our algorithm NIS is based on the Hawkes process, we briefly introduce the Hawkes process in this subsection. The Hawkes process is a point process that is widely used in various fields, such as the prediction of commodity flow, social network modeling, etc.

Hawkes process is a special linear self-excitation model, which can be regarded as a Poisson process satisfying a random process. According to the model, the probability of current events will be affected by historical events and will decrease with time. In other words, events in the past will affect the future in some way. Guoming et al. (2018) used the Hawkes process to recommend users' future concerns. Mei and Eisner (2017) used this model to process the historical event stream of a single user. All the above works use the most core idea of Hawkes process, i.e., a node's past behavior will affect the future, and the future event can be judged by calculating the influence.

Therefore, the probability of future events can be predicted by establishing the relationship between historical events and current events. The conditional intensity function of the Hawkes process is defined as follows.

$$\lambda_k(t) = \mu_k + \sum_{i:\ t_i<t} \alpha_{ik} v(t-t_i) \tag{1}$$

Where $\lambda_k(t)$ denotes the intensity of the current event k, $\mu_k \geq 0$ is the basic intensity of the process, $\alpha_{ik} \geq 0$ represents the degree to which a historical event i excites the current event k, $v(\cdot)$ is a kernel function representing time decay effect, t denotes the time when the current event k occurs and t_i represents the time when the historical event i occurs.

Hawkes process divides the conditional intensity into the basic part and the Hawkes increment. The Hawkes increment is the influence of historical events on the probability of the current event. After calculating each historical event separately, we can treat their sum as the overall Hawkes increment. Based on the Hawkes process, we propose our model NIS.

3.2. NIS Model

With the above definitions, we obtain the neighborhood influence sequence of nodes. Since the number of historical interaction events at different nodes is different, in order to facilitate the calculation for neighborhood influence sequence, we use the same length l for each neighborhood influence sequence. If the number of historical interaction events for a node is larger than l, we only choose the most recent l interactions for the node.

Neighborhood Embedding. Let z_i^x be the embedding of the ith node in the neighborhood influence sequence of node x. According to the source node x and its neighborhood influence sequence N_x, we can obtain its neighborhood embedding sequence $Z_x = (z_1^x,\ z_2^x, \cdots, z_l^x)$ and interactive time sequence $T_x = (t_1^x,\ t_2^x, \cdots, t_l^x)$. The final neighborhood embedding z_x^h of the source node x can be defined as follows.

$$z_x^h = \sum_{i=1}^{l} z_i^x \times w_i^x \tag{2}$$

$$w_i^x = \frac{\frac{1}{\left(1+\left|t-t_i^x\right|\right)}}{\sum_{k=1}^{l} \frac{1}{\left(1+\left|t-t_k^x\right|\right)}} \tag{3}$$

Where t denotes the current time, w_i^x is the weight of the interaction time of each neighbor in the neighborhood influence sequence. We use w_i^x to normalize the time and obtain the proportion of the time influence of each interaction. Thus, for source node x and target node y, we can calculate their final neighborhood embeddings z_x^h and z_y^h.

Conditional Intensity Function. According to the Hawkes process, the function is divided into two parts: basic intensity $\mu_{x,y}$ and Hawkes increment $h_{x,y}$, i.e.:

$$\lambda_{y|x}(t) = \mu_{x,y} + h_{x,y} \tag{4}$$

The basic intensity is the own influence of nodes x and y on future interactions:

$$\mu_{x,y} = -\left\| z_x - z_y \right\|^2 \tag{5}$$

The Hawkes increment can be divided into two parts: the first part calculates the influence of the node's neighborhood embedding on another node, the second part calculates the influence between the two node's neighborhood embedding. The example explanation has been introduced above in Figure 2.

We learn two parameters δ_1 and δ_2 to adjust the weight of the two parts respectively. The parameter δ_1 and δ_2 represent the influence weight of the first part and the second part respectively. In the process of calculating condition intensity, the weights of different types of influences are also different, and these weight parameters need to be constantly learning in training.

$$h_{x,y} = \delta_1 \left(-\left\| z_x^h - z_y \right\|^2 - \left\| z_y^h - z_x \right\|^2 \right) + \delta_2 \left(-\left\| z_x^h - z_y^h \right\|^2 \right) \tag{6}$$

Thus, the conditional probability of node interaction can be obtained, where y' means all possible nodes that may interact with x at time t.

$$p\left(y\left|x\right.\right) = \frac{\lambda_{y|x}(t)}{\sum_{y'} \lambda_{y'|x}\left(t\right)} \tag{7}$$

Loss Function. The log likelihood of neighborhood influence sequences for all nodes in the network can be defined as follows.

$$\log L = \sum_{x \in V} \sum_{y \in N_x} \log\left[p\left(y|x\right)\right] \tag{8}$$

Due to the huge cost computations of the loss function, we use negative sampling to optimize the loss function. Let K be the number of negative nodes sampled according to distribution $P\left(v\right)$ which is proportional to $3/4$ of degree for node v, and $\sigma\left(x\right) = 1/\left(1 + \exp\left(-x\right)\right)$ be the sigmoid function. The objective function of an interaction between a source node x and a target node y at time t can be computed as follows.

$$\log\left[p\left(y|x\right)\right] = \log\sigma\left(\lambda_{y|x}\left(t\right)\right) + \sum_{k=1}^{K} E_{z^k \sim P(v)}\left[-\log\sigma\left(\lambda_{z^k|x}\left(t\right)\right)\right] \tag{9}$$

After obtaining the loss function of the model, we use Adam method to optimize the objective function in Equation 9. The generated embeddings can well capture dynamic interaction process between nodes, and thus can be used in various downstream tasks. The pseudo code for NIS is shown in Algorithm 1.

Algorithm 1 An algorithm NIS for learning network embedding based on neighborhood influence sequence

```
Input: A temporal network $G = (V,\ E,\ O)$
Output: Dynamic node embeddings
 1: for each node $x \in V$ do
 2:   Initalize node embedding $z_x$ for $x$;
 3:   obtain $N_x = \{(y_1, t_1), (y_2, t_2), \cdots, (y_l, t_l)\}$ for $x$ from $G$;
 4: end for
 5: repeat
 6:   for each node $x \in V$ do
 7:     for each $(y, t) \in N_x$ do
 8:       Calculate $z_x^h$ and $z_y^h$ baed on Equation 2;
 9:       Calculate $\lambda_{y|x}(t)$ based on Equation 4;
10:       Calculate $\log[p(y|x)]$ based on Equation 9;
11:       Use Adam to get updated node embedding $z_x$ and $z_y$;
12:     end for
13:   end for
14: until Convergence
15: Output node embedding for all nodes;
```

3.3. Comparison of NIS and HTNE

The HTNE algorithm Zuo et al. (2018) is also based on Hawkes process, but there are obvious differences between NIS and HTNE.

Both NIS and HTNE draw on the idea of Hawkes process and regard the influence of historical interaction neighbors on the current node as part of the Hawkes increment. Both of them use negative squared Euclidean distance to denote the base intensity $\mu_{x,y}$ between source node x and target node y. But for Hawkes increment $h_{x,y}$, they are significantly different.

For Hawkes increment $h_{x,y}$, HTNE uses negative squared Euclidean distance to represent the degree to which a historical neighbor h of source node x excites target node y. In NIS, we made many modifications to Hawkes increment $h_{x,y}$. We first construct a neighborhood influence embedding z_x^h of source node x, and then use z_x^h to calculate the overall influence of x's historical neighbors on target node y. Further, we incorporated the idea of time decay in the process of generating z_x^h. The closer the interaction time between neighbor i and source node x, the greater the proportion of neighbor i in the neighbor influence embedding z_x^h of source node x, as shown in Equation 2 and 3.

In addition, HTNE only considers the influence of historical neighbors of source node x on target node y when calculating Hawkes increment. Howerver, HTNE ignores the influence of historical neighbors of target node y on source node x. HTNE also fails to consider mutual influence between the neighbors of x and the neighbors of y. In contrast, when calculating Hawkes increment, NIS takes into consideration the influence of historical neighbors of source node x on target node y, the influence of historical neighbors of target node y on source node x, and mutual influence between neighborhood embeddings respectively, as shown in Equation 6.

4. Experiments

To investigate the effectiveness of NIS on modeling relationships between nodes, we conduct experiments on four real-world datasets and compare with five state-of-the-art baselines.

4.1. Datasets

We first introduce the four real-world networks used in our experiments, with data statistics listed in Table 1.

Table 1: Data statistics

Datasets	DBLP	ML1M	Yelp	AMms
Nodes	28,085	9,745	110,649	74,526
Edges	236,894	1,000,206	537,136	89,689
Labels	10	5	5	5

DBLP Zuo et al. (2018): This is a co-authorship graph which focused on the Computer Science domain. Our dataset is extracted from it of ten research areas (Table 2). If more than half of the last ten papers of a researcher are published in corresponding conference, we assume he or she belongs to this particular area.

Table 2: Research areas in DBLP

Label	Conference	Research Area
0	ICDE, VLDB, SIGMOD	Database
1	KDD, ICDM, SDM, CIKM	Data Ming
2	SIGIR	Information Retrieval
3	IJCAI, AAAI, ICML, NIPS	Artificial Intelligence
4	CVPR, ICCV	Computer Vision
5	STOC, SODA, COLT	Theory
6	ACL, EMNLP, COLING	Computational Linguistics
7	SIGCOMM, INFOCOM	Computer Networks
8	SOSP, OSDI	Operating Systems
9	POPL	Programming Languages

ML1M Li et al. (2020): It is a widely used movie dataset for evaluating algorithms and we use the version (MovieLens-1M) that includes 1 million user ratings.

Yelp Zuo et al. (2018): We derive the dataset from the Yelp Challenge Dataset. Users and businesses are regarded as nodes, and commenting behaviors are taken as edges. We only retain the top five categories during the experiments.

AMms Ni et al. (2019): This dataset is taken from the magazine subscription part on the Amazon website (AMAZON-Magazine Subscriptions). Users rate the magazines and we select the most rated score for each magazine as its category.

4.2. Baselines

We compare NIS with five state-of-the-art algorithms.

DeepWalk Perozzi et al. (2014): This algorithm performs random walks over networks to generate node sequences and employ Skip-Gram Mikolov et al. (2013) model to learn vertex embeddings.

LINE Tang et al. (2015): This algorithm learns node embeddings in large-scale networks using first-order and second-order proximities.

Node2vec Grover and Leskovec (2016): This algorithm extends DeepWalk and proposes a biased random walk procedure to maintain a balance between the local and global properties of a network.

HTNE Zuo et al. (2018): This algorithm introduces the Hawkes process theory into network embedding to capture the influence of historical neighbors on the current neighbors.

tNodeEmbed Singer et al. (2019): This algorithm proposes a joint loss function that creates a temporal embedding of a node by learning to combine its historical temporal embeddings, and the joint loss function focus on optimizes per given task (e.g., node classification).

4.3. Tasks and Evaluation Measures

We evaluate our model with regard to various fundamental tasks: node classification, link prediction, network visualization, parameter sensitivity and ablation study.

Node Classification: We train a classifier to predict the node labels and use both Accuracy and Weighted-F1 as measures to evaluate the classification performance.

Link Prediction: We adopt the AUC score as the measure to determine whether there is an edge between two given nodes.

Network Visualization: We select some nodes in three fields to present their distribution in figure and evaluate models through the given network views.

Parameter Sensitivity: We use the Accuracy score to evaluate the node classification performance of NIS under different neighborhood influence sequence length l to verify its importance.

Ablation Study: We divide the conditional intensity function into three parts and evaluate different combinations. We use the Accuracy score to evaluate the effect of their embeddings in node classification to determine the necessity of each part in NIS model.

4.4. Parameter Settings

For all baselines, we set the node embedding dimension to 128, and use the default values for other parameters. For NIS, we set the length l of the historical sequence as 5, 5, 5, 2 for DBLP, ML1M, Yelp and AMms respectively. We set the mini-batch size, the learning rate and the number of negative sampling to be 128, 0.001, 5 respectively.

As described above, we evaluate the performance of node embeddings by feeding them into various tasks.

4.5. Node Classification

For node classification, we train a classifier and predict the node labels and use both Accuracy and Weighted-F1 as measures to evaluate the classification performance of various methods.

We use an 8:2 ratio to divide the training sets and the test sets. Since the datasets contain time information, we put a complete time period in the training set. Thus, the training set proportions of ML1M, Yelp and AMms are all around 80%, while DBLP is set to 70% ratio and its real training set instance accounts for 78%. The experimental results are shown in Table 3.

Table 3: Node classification results of all alogrithms on all datasets

Metric	Method	DBLP	ML1M	Yelp	AMms
Accuracy	DeepWalk	0.6210	0.6057	0.5092	0.5772
	LINE	0.6231	0.6096	0.5184	0.5763
	node2vec	0.6270	0.6189	0.5145	0.5774
	HTNE	0.6334	0.5910	0.5243	0.5698
	tNodeEmbed	0.6259	0.6023	0.5209	0.5755
	NIS	**0.6423**	**0.6197**	**0.5298**	**0.5777**
Weighted-F1	DeepWalk	0.6149	0.5837	0.3957	0.4246
	LINE	0.6189	0.5766	0.4034	0.4266
	node2vec	0.6194	0.5745	0.4120	0.4242
	HTNE	0.6272	0.5312	0.4087	0.4237
	tNodeEmbed	0.6203	0.5826	0.4039	0.4268
	NIS	**0.6336**	**0.5887**	**0.4181**	**0.4273**

In this case, NIS outperforms all other baselines over all datasets. This means that it is important to focus on the temporal information in network embedding. Compared with HTNE and tNodeEmbed which pay attention to the temporal information, NIS also focuses on the influence between neighborhoods which is one of the reasons for the better performance.

On the Yelp dataset, all algorithm results are relatively poor, we believe it's for the following reasons: business on Yelp usually contains multiple category labels and user's comments on businesses are often mapped to different categories. These comments are usually difficult to cause other reactions, so most of the interaction between user and business in the network is in an isolated state, which leads to poor performance of all models on this special network structure.

4.6. Link Prediction

For link prediction, we conducted experiments on DBLP and AMms datasets and took AUC Hanley and Mcneil (1982) to measure performance.

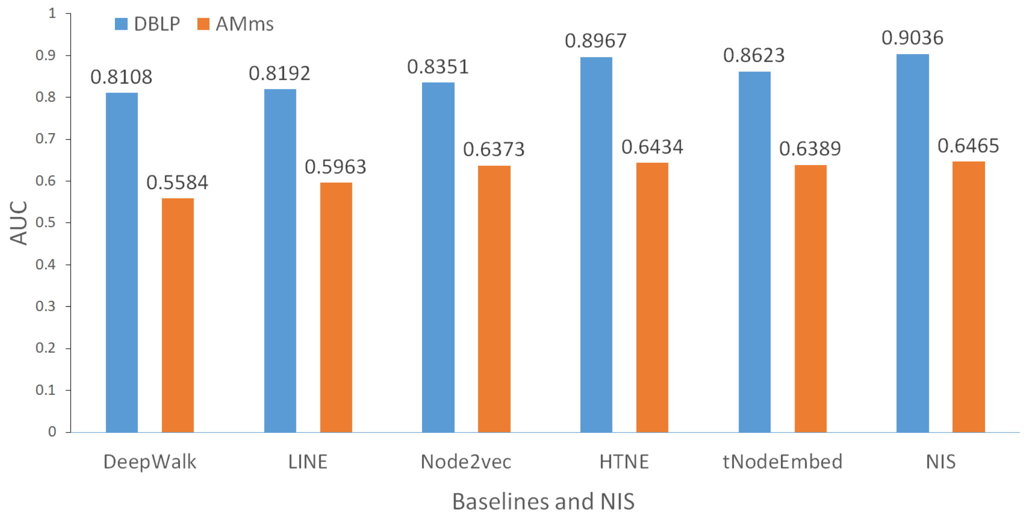


Figure 3: AUC comparison of NIS and baselines in link prediction task

The experimental results are shown in Figure 3. From the results, it can be seen that the overall performance of NIS is better, which proves the ability of the algorithm to capture the influence between nodes.

Note that, all algorithms have poor performance on AMms, we believe this problem is caused by the node distribution of datasets. As shown in Table 1, the number of nodes differs little from the number of edges in AMms, which means that many nodes are only connected to one or two nodes in the network, so the whole network structure is more like a chain. Therefore, all algorithms have poor results, but algorithms which focus on temporal information still work better than algorithms only focus on network structure.

4.7. Network Visualization

For network visualization, we employ the t-SNE method Der Maaten and Hinton (2008) to project embeddings of researchers to a 2-dimensional space on the DBLP dataset. In particular, we select three fields from the dataset: Computer Network, Data Mining and Computer Vision, and select 500 researchers as nodes in each field. Different fields are marked with different colors and presented with a scatter plot.

As shown in Figure 4, we use green for data mining, purple for computer vision, and blue for computer networks. It can be seen that both DeepWalk, LINE and Node2vec failed to separate the three areas apart clearly. HTNE and tNodeEmbed can only roughly distinguish the area boundaries. NIS clearly separates the three areas, one of which has a very obvious border. Above results can indicate that NIS has better performance on network visualization, i.e., the algorithm has the ability to handle tasks at the social application level such as community mining.

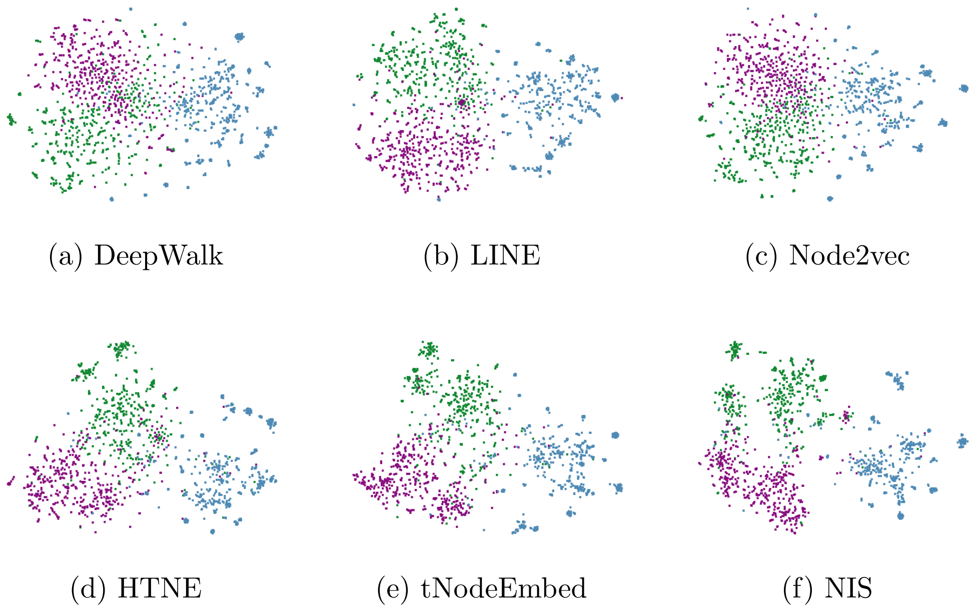


Figure 4: Network visualizations in three fields: green for data mining, purple for computer vision, blue for computer networks.

4.8. Parameter Sensitivity

In NIS model, we proposed an important parameter named neighborhood influence sequence length l which is designed to truncate the whole interaction sequence of a node at a specific time into a recent sequence with fixed length.

In parameter sensitivity experiment, we select 2, 3, 4, 5 and 10 for the length l, other parameters are the same as above. In particular, as illustrated in Figure 5, we report the Accuracy of NIS on DBLP and AMms.

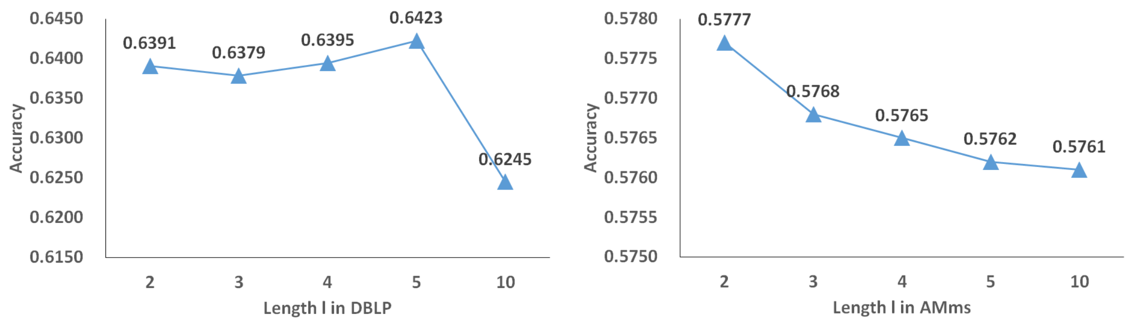


Figure 5: Parameter sensitivity of NIS on DBLP and AMms datasets

From the result, we can see that the accuracy score on DBLP first rises and then falls, finally reaching the highest score when l=5. However, the score keeps decreasing with the increase of l on AMms, and has reached the best performance when l=2.

The trend of the result curves on the two datasets is different. We think it is due to the following reason: as mentioned above in Table 1, DBLP is a co-author network which has 28,085 nodes and 236,894 edges. It means that on average ten interactions occur per node, i.e., many nodes may have 5 neighbors. For AMms, this dataset has 74,526 nodes and 89,689 edges. Each node in the AMms dataset may only have 1 or 2 neighbors. When l becomes larger, many useless nodes will be added to the neighbor sequence which affects the performance of embedding.

Therefore, compared with DBLP, AMms provides fewer historical neighbors, so the performance is better when l is smaller in AMms.

4.9. Ablation Study

NIS introduces the conditional function in the Hawkes process and divides it into two parts: base intensity and Hawkes increment. The Hawkes increment includes the influence of neighborhood on nodes and the influence between neighborhoods. Thus, we divide NIS model into three parts.

Let $NIS.a$ denotes the basic intensity part which is the own influence of nodes x and y on future interactions:, i.e., $NIS.a = \mu_{x,y} = -\|z_x - z_y\|^2$. $NIS.b$ denotes the influence of a node's neighborhood embedding on another node, i.e., $NIS.b = -\left\|z_x^h - z_y\right\|^2 - \left\|z_y^h - z_x\right\|^2$. $NIS.c$ denotes the influence between the two node's neighborhood embedding, i.e., $NIS.c = -\left\|z_x^h - z_y^h\right\|^2$. When we only declare NIS, it means combining the above three parts, i.e., $NIS.abc$.

In the experiment, we take $NIS.a$, $NIS.ab$, $NIS.ac$, NIS respectively to modify the method to find the true utility of each part. Specifically, we choose DBLP and ML1M as experimental datasets and all parameter settings are consistent with above experiments.

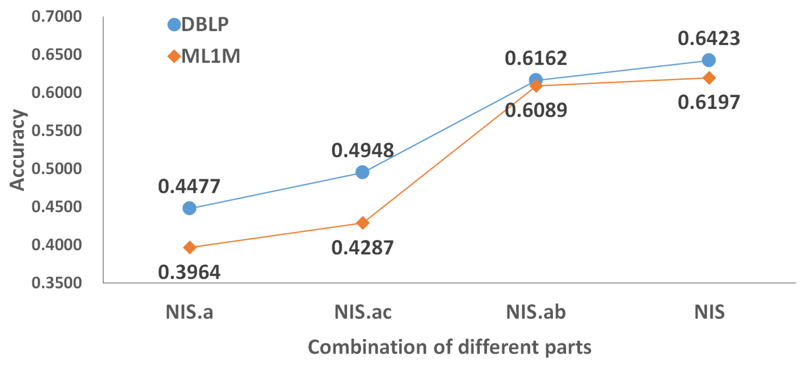


Figure 6: Ablation study of NIS on DBLP and ML1M datasets

From Figure 6, we find that the model can complete the node classification task by only retaining $NIS.a$, but the performance is not ideal. The model performance is slightly improved when $NIS.c$ is introduced, but it is much worse than $NIS.ab$.

Comparing the performance on the two datasets, we found that the impact of Part $NIS.b$ on ML1M is greater than that on DBLP. We believe this phenomenon is due to the following reasons: As shown in Table 1, compared with DBLP, ML1M has fewer nodes and more edges. This makes the nodes in the ML1M more sensitive to the influence of neighbors because each node has more interactions. Therefore, when the model considers $NIS.b$ (the influence of the neighborhood on nodes), it will significantly improve the embedding effect on ML1M.

In summary, NIS proves the importance of introducing neighbor influence sequences through ablation study, which will improve the performance of nodes embedded in real-world downstream tasks.

5. Related Work

Network representation learning (NRL), also known as network embedding (NE), becomes more and more important in network analysis tasks with the surge of network data in recent years. The goal of NRL is to transform nodes into low-dimensional vector representations which contains original attribute and structure information as much as possible. The low-dimensional representations for nodes is suitable for various network analysis tasks. According to whether the network structure evolves, NRL methods can be divided into two categories: static network-based methods and dynamic network-based methods Zhou et al. (2018).

Static Network: It means that there are no changes about nodes and edges into account during the learning process. In the development of network representation learning, algorithms were based on static networks at the earliest. For example, DeepWalk performs random walks over networks to generate node sequences and employ Skip-Gram model to learn vertex embeddings. LINE learns node embeddings in large-scale networks using first-order and second-order proximities. GraRep Cao et al. (2015) calculates the k-order similarity between nodes and constructs the loss function by matrix decomposition to obtain the global variable of nodes. Node2vec extends DeepWalk and proposes a biased random walk procedure to maintain a balance between the local and global properties of a network. SDNE Wang et al. (2016) first applies deep learning to network representation learning and embeds nodes with first-order and second-order similarity. CANE Tu et al. (2017) focuses on the text information attached to the node and learns the contextual embedding for the node.

Dynamic Network: With the development of representation learning, researchers began to pay attention to dynamic networks with nodes and edges changing. GrapSAGE Hamilton et al. (2017) learn to generate a vector-represented map for each node to sample the neighbors of the nodes in the graph and propose four different aggregation functions. CTDNE Nguyen et al. (2018) introduces temporal information to model the network and performs random walks in chronological order. HTNE introduces the Hawkes process theory into network embedding to capture the influence of historical neighbors on the current neighbors. tNodeEmbed proposes a joint loss function that creates a temporal embedding of a node by learning to combine its historical temporal embeddings, and the joint loss function focus on optimizes per given task(e.g., node classification).

However, although the dynamic network-based algorithms focus on the changes of time in the network, their models don't pay enough attention to the neighborhood information around nodes. To solve this problem, we propose a NIS algorithm to focus on both historical temporal sequence and neighborhood information in the network.

6. Conclusions

In this paper, we propose a NIS algorithm to focus on the influence of nodes' neighborhood in the interaction process. We propose three kinds of influence when two nodes interact, and integrate them into NIS by introducing the Hawkes process. The node embeddings which obtained by this algorithm can be well applied to the downstream tasks such as node classification and network visualization, etc. In future, we will consider the inductive method and the influence of the node's text information.

7. Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61972135), the Natural Science Foundation of Heilongjiang Province (No. LH2020F043), and the Innovation Talents Project of Science and Technology Bureau of Harbin (No. 2017RAQXJ094).

References

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Conference on information and knowledge management*, pages 891–900, 2015.

Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chenchuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *Conference on information and knowledge management*, pages 377–386, 2017.

Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):833–852, 2019.

Laurens Van Der Maaten and Geoffrey E Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Knowledge discovery and data mining*, pages 855–864, 2016.

Zhang Guoming, Wang Junshu, Jiang Nan, and Sheng Yehua. A point-of-interest recommendation method based on hawkes process. *Acta Geodaetica et Cartographica Sinica*, 47(9):1261, 2018.

William L Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Neural information processing systems*, pages 1024–1034, 2017.

James A Hanley and Barbara J Mcneil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

Alan G Hawkes. Point spectra of some mutually exciting point processes. *Journal of the royal statistical society series b-methodological*, 33(3):438–443, 1971.

Jiacheng Li, Yujie Wang, and Julian Mcauley. Time interval aware self-attention for sequential recommendation. In *Web search and data mining*, pages 322–330, 2020.

Hongyuan Mei and Jason Eisner. The neural hawkes process: a neurally self-modulating multivariate point process. In *neural information processing systems*, 2017.

Tomas Mikolov, Kai Chen, Greg S Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *international conference on learning representations*, 2013.

Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion of the The Web Conference*, pages 969–976, 2018.

Jianmo Ni, Jiacheng Li, and Julian Mcauley. Justifying recommendations using distantly-labeled reviews and fined-grained aspects. In *International joint conference on natural language processing*, pages 188–197, 2019.

Mathias Niepert, Mohamed H Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

Bryan Perozzi, Rami Alrfou, and Steven Skiena. Deepwalk: online learning of social representations. In *Knowledge discovery and data mining*, pages 701–710, 2014.

Uriel Singer, Ido Guy, and Kira Radinsky. Node embedding over temporal graphs. In *International joint conference on artificial intelligence*, pages 4605–4612, 2019.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *The web conference*, pages 1067–1077, 2015.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. Cane: Context-aware network embedding for relation modeling. In *Meeting of the association for computational linguistics*, volume 1, pages 1722–1731, 2017.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Knowledge discovery and data mining*, pages 1225–1234, 2016.

Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *National conference on artificial intelligence*, pages 571–578, 2018.

Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. Embedding temporal network via neighborhood formation. In *Knowledge discovery and data mining*, pages 2857–2866, 2018.