Cascade modeling with multihead self-attention

1st Chaochao Liu
College of Intelligence and Computing
Tianjin University
Tianjin, 300350, China
chaochaoliu@tju.edu.cn

2th Wenjun Wang
College of Intelligence and Computing
Tianjin University
Tianjin, 300350, China
wjwang@tju.edu.cn

3nd Pengfei Jiao
Center of Biosafety Research and Strategy
Tianjin University
Tianjin, 300072, China
pjiao@tju.edu.cn

4rd Xue Chen
Law School
Tianjin University
Tianjin 300054, China
xuechen@tju.edu.cn

5th Yueheng Sun*
College of Intelligence and Computing
Tianjin University
Tianjin, 300350, China
* Corresponding Author
yhs@tju.edu.cn

***Abstract*—Modeling how information diffuses across social network platforms can be widely used. Recently, researchers have used deep learning methods to model information cascades and forecast their progression without dependence on the hypothesis of the underlying diffusion model. Most of these studies use sequential models (e.g., recurrent neural networks, RNNs) and model cascades of information spread without using the network structure information. However, the network structure information substantially affects information spread, cross-dependence should be considered in cascade modeling, and recurrent neural networks produce poor results on long sequence modeling. To solve these issues, in this paper, we present a new cascade modeling method with the multihead self-attention mechanism. We design an encoder that combines network structure information with multihead self-attention to learn the representations of cascades and consider diverse user dependencies on the network. Experiments are conducted on both synthetic and real-world datasets. The results show that the proposed method performs better on the long sequence cascade prediction than the state-of-the-art methods.**

***Index Terms*—cascade prediction, deep learning, network structure information, self-attention, multihead**

I. Introduction

A series of online social networks, such as Twitter and Facebook, make information sharing and dissemination convenient. If people want to share interesting information with others, they will re-edit or copy the information from one of their neighbors and then send it to their other neighbors [1]. Information cascades are generated when one person posts information on the social network, and the others re-post it.

Information cascades are important factors in most social media information dissemination phenomena, ranging from viral marketing, crowdsourcing, rumor spreading, cyber violence, and various types of persuasion campaigns [2]. There has been considerable research on information cascade modeling and prediction. The classical cascade models in this field are derived from epidemiological literature such as the independent cascading model (IC) [3] and the linear threshold model (LT) [4]. These models are used to model propagation dynamics; specifically, they estimate the interpersonal influence or parameters used to characterize the influence and susceptibility of individuals.

Recent research [5]–[7] has shown that deep learning methods can circumvent the cascade prediction problem without requiring an explicit underlying diffusion model. This characteristic makes deep learning-based models better suited for practical applications. These models are mostly based on the recurrent neural network, which is a sequential model. They focus on modeling how the historical sequence data can affect future sharing behavior. Moreover, this inherent sequence of features makes training parallelization difficult; however, processing long sequence data requires parallelization to increase computational efficiency and reduce computational space [8]. Additionally, the retrievable diffusion data are normally chronological, and the social network is unknown [9]. This feature makes it difficult to model the information spread of long sequence data produced from a social network structure.

Social network structure information is critical to understanding diffusion dynamics and cascading predictions [10]–[12]. Network structure information has not or has only partially been used in deep learning models for cascade resharing prediction tasks. Wang *et al.* [6] proposed an attention-based recurrent neural network model to capture the cross-dependence in cascades, but the network structure information is not used in the model. Wang *et al.* [9] developed a sequential neural information diffusion model with structure attention to incorporate the structural diffusion context into the cascade prediction of resharing, but they considered only nodes' neighborhood information. Liu *et al.* [13] proposed a cascade prediction model with enhanced community structure, but their model only focused on the community structure information and was hardly able to process the long cascade sequence data.

Long sequence data modeling is confronted with two challenges to be solved: how to process the long sequence data parallelized for handling the large datasets and how to capture the long-range dependencies of sequence data. The self-

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

attention mechanism proposed by Vaswani *et al* [8] can more easily solve the issue than can the recurrent neural network. Devlin *et al* [14] proposed a pretrained BERT model based on the self-attention mechanism for the long sequence data processing of the language representation task. Al-Rfou *et al* proposed a model based on the self-attention mechanism, and the results of experiments were shown to outperform recurrent neural network variants [15]. However, these works mostly focus on the natural language processing tasks, and they do not need to consider the task for combinations of the social network structure information with the dynamic information spread of cascade sequence data.

In our work, we propose a model based on multihead self-attention for cascade prediction with network structure attention called CPMHSA. The method utilizes a self-attention mechanism to capture the dependence of the spreading nodes, which performs better than the recurrent neural network model when processing the long sequence data. Since the network structure substantially affects the spread of information, we use the network information to restrict and regulate the node's dependence.

The main contributions of the proposed CPMHSA model are summarized as follows:

1) CPMHSA considers network structure information, which makes it more reasonable for handling the cascade prediction task.
2) CPMHSA is based on a self-attention mechanism that can learn long sequence dependence more effectively than can the RNN.
3) CPMHSA considers a multihead mechanism that can learn the complex dependence between nodes involved in the cascade.
4) CPMHSA achieves better performance than other compared algorithms.

II. Related work

We introduce the existing literature related to our work in this section. One related field is the empirical research on the effect of the network structure on spreading behavior. Considerable research on cascade modeling is underway. The work on self-attention is also related to this research.

Many studies have shown that the network structure strongly affects the spread of information. Lerman *et al* [16] analyzed the spread data of two social news sites and found that the structure of social networks affects information dissemination. Liu *et al* compared the epidemic propagation of the network that contains community structure to that of the random network. They found that the network with a community structure had a a smaller threshold for an epidemic outbreak and a greater prevalence to maintain the outbreak epidemic [17]. Pastor-Satorras *et al* conducted a theoretical study and found that the network topology affects the spread of epidemics [18]. Huang *et al* [19] compared two types of scale-free networks, with and without a community structure, and found that epidemic prevalence is reduced in networks with a strong community structure. However, minimal work has considered the network structure information in the cascade prediction task. Thus, in this paper, we include a network structure information constraint to model the information cascade.

Currently, many efforts have been carried out to solve the cascade behaivor predicton task by using deep learning frameworks. These models can capture complex relationships between activated nodes. Xiao *et al* [7] used two RNNs to model the time and the nodes' sequence. Zhang *et al* [5] proposed a model based on deep learning to capture the content of tweets, user interests and so on. Wang *et al* [6] used an attention mechanism to model the cross-dependence of the spreading nodes in the information cascades, but they did not actually use the network structure information. Most of these deep learning studies did not consider the social network structure information, and most of them were based on recurrent neural networks, which cannot capture the long dependence as well as self-attention. Thus, in this paper, we extend self-attention and consider the network structure information.

Self-attention has been widely used in many fields since Vaswani *el at* proposed the transformer [8]. Ma *et al* extended the transformer for language modeling [20] based on the ideas of tensor decomposition and parameter sharing, which largely compress the model parameters. Devlin *et al* [14] used self-attention and proposed a representation model to solve the language task. In this paper, we extend self-attention to the field of information cascade prediction.

III. Proposed method

A. Problem definition

In our paper, we define the information cascade prediction task as predicting further activated nodes given the historical sequence of activated nodes and the network structure information. We first give some definitions used in this paper in the following.

Definition 1. **Network.** A network is denoted as $G = (V, E)$, where V is a set of nodes and E is a set of edges between the nodes.

Definition 2. **Cascade.** A cascade $S = \{v_i | v_i \in V, 1 \leq i \leq N\}$ is a nodes' sequence set of information spreading starting from an original post and ordered by time, where N represents the frequency of information spreading behavior. The $i-th$ behavior refers to node v_i resharing the information at time t_i.

Given a network $G = (V, E)$ and a collection of F cascade sets $Q = \{S_f\}_{f=1}^{F}$, our task is to construct a model to fit the data. Once an observed cascade is received, the model can be used to predict the probability of the next activated node. We denote the observed cascade that contains a list of nodes up to the $k-th$ spreading behavior as $S_{\leq k}$. Thus, the probability of the next activated node v_{k+1} can be represented as $p(v_{k+1}|S_{\leq k})$.

B. Model framework

In our work, we solve the cascade prediction problem by proposing a multihead self-attention deep learning model

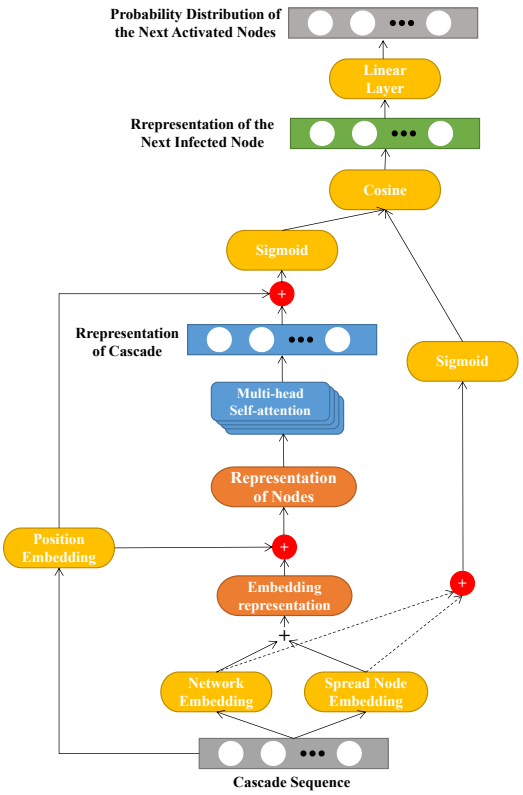

Fig. 1. **The structure of the proposed method.** Cascade sequence data are transformed to low-dimensional representations by using network embedding, spread node embedding, and position embedding. They are concatenated as the representation of nodes and converted as the representation of cascade by using multihead self-attention. The results are calculated for the cosine similarity with the cascade's embedding values. Thus, we can obtain the representation of the next infected node. Additionally, the probability distribution of the next activated nodes can be produced by using a linear layer.

that is network structure-aware, named CPMHSA. The model uses self-attention to capture the long sequence dependence between active nodes in a cascade. Since the dependencies are complex, we utilize the multihead mechanism to learn the complex relationship. Since the network structure has a substantial effect on the spread of information, we use network information to restrict and regulate the nodes' dependence. The structure of the method is shown in Figure 1, which can be trained end-to-end. For the inputs, the spreading nodes are transformed to the low-dimensional vectors by using a node embedding layer, which then adds with the spread node embedding. The nodes' embedding vectors can be obtained via SDNE [21]. Then, the positional encoding is concatenated. The cascade's representation can be produced by using multihead self-attention. The results concatenate the positional encoding and calculate the similarity with the summation of the network embedding and spread node embedding. After transformation by a linear layer, we can obtain the probability distribution of the next activated nodes.

1) Representation of Nodes: We model the input as the fusion of network structure embedding, spread node embedding, and positional encoding. Since the network embedding method SDNE [21] utilizes the first- and second-order proximity of nodes that can capture the local and global network structure information, we use SDNE to transform the sparse network matrix into low-dimension vectors. We represent the embedding matrix of nodes as $\mathbf{W}^M \in \mathbb{R}^{|V|\times d_m}$, where d_m represents the dimension of the embedding. $\mathbf{W}^P \in \mathbb{R}^{|V|\times d_p}$ represents the spread node embedding matrix, where d_p is the dimension of the embedding. In our model, we use positional encodings to capture the relative or absolute position of the tokens in the sequence following [8]. We represent the position embedding matrix of nodes as $\mathbf{W}^Q \in \mathbb{R}^{N\times d_q}$, where N represents the maximum length of the cascades and d_q represents the dimension of the embedding. Thus, the representation of nodes at step i is $r_i = (f(v_i)\mathbf{W}^M + f(v_i)\mathbf{W}^P) \oplus v_i\mathbf{W}^Q)$, where $f(\cdot)$ represents the mapping from the nodes at step i to the node labels and $\oplus$ represents the concatenate operation.

2) Multihead Network-Structure-Aware Self-Attention: We propose a multihead self-attention module with the network structure of spreading nodes considered. Since the relation between activated nodes in a cascade is complex, we employ h attention heads to fit this situation. The results of each head are concatenated, and we use a linear layer to transform the concatentated value to the output of the self-attention. The structure of the multihead self-attention module is shown in Figure 2.

All the attentions of the multihead operate on an input sequence embedding $\mathbf{r} = (r_1, ..., r_n)$ of n elements and output the new sequence $\mathbf{z} = (z_1, ..., z_n)$, where $z_i \in R^{d_z}$. Each output element, z_i, is computed as the weighted sum of the linearly transformed input elements:

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(r_j\mathbf{W}^V) \tag{1}$$

where α_{ij} is the attention coefficient and $\mathbf{W}^V$ represents the parameters of the model.

Each attention coefficient, α_{ij}, is computed using a softmax function with a network structure constraint:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{y=1}^{n}\exp(e_{iy})}a_{ij} \tag{2}$$

where a_{ij} represents the distance between node i and node j computed as the dot product between the network structure embeddings of nodes v_i and v_j:

$$a_{ij} = (f(v_i)\mathbf{W}^M) \cdot (f(v_j)\mathbf{W}^M)^T \tag{3}$$

e_{ij} is computed using a compatibility function that compares two input elements:

$$e_{ij} = \frac{(r_i\mathbf{W}^Q)(r_j\mathbf{W}^K)^T}{\sqrt{d_z}} \tag{4}$$

where $\mathbf{W}^Q$ and $\mathbf{W}^K$ are the parameter matrices, and d_z represents the dissemination size of the self-attention output

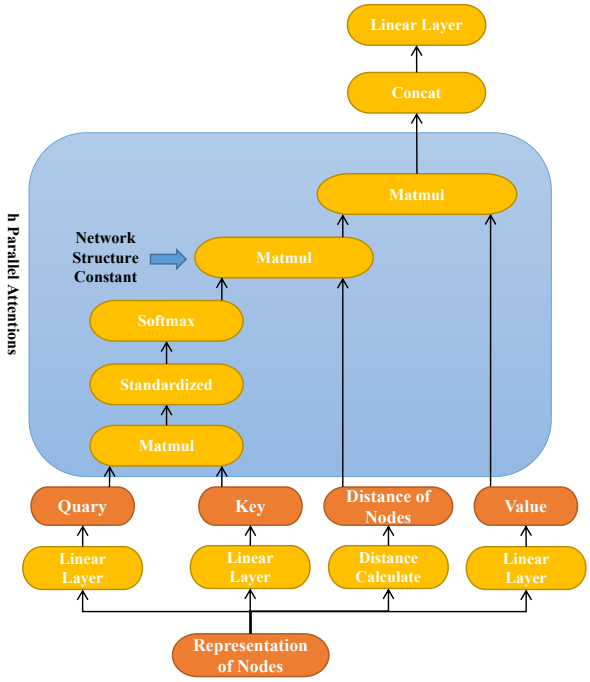

Fig. 2. **The structure of the proposed multihead self-attention module.** Representations of nodes are transformed as queries, keys, and values by using a linear layer. We use the network structure embeddings of nodes to calculate the distance between nodes and use the distance to constrain the attention coefficient.

z_i. The query, key, and value of the self-attention are, respectively, $r_i\mathbf{W}^Q, r_i\mathbf{W}^K$, and $r_i\mathbf{W}^V$. To make the multihead self-attention capture the complex relation between nodes, we make these parameters to be different in each attention head.

The scaled dot product, which enables efficient computation, was chosen as the compatibility function. Furthermore, the linear transformation layer of the inputs is also evolved to ensure sufficient expressive power.

Our model uses a multihead mechanism to model different relations among nodes. Thus, the output of sequence $r = (r_1, \cdots, r_n)$ is $z'_i = \text{Aggregate}(\text{Head}_i^{(1)}, \cdots, \text{Head}^{(c)_i}, \cdots, \text{Head}_i^{(C)})$, where $1 \leq c \leq C$, C is the number of single attentions and $\text{Head}_i^{(c)} = \sum_{j=1}^{n} \text{softmax}(\frac{(r_i\mathbf{W}_c^Q)(r_j\mathbf{W}_c^K)^T}{\sqrt{d_z}}) \cdot (f(v_i)\mathbf{W}^M) \cdot (f(v_j)\mathbf{W}^M)^T \cdot (r_j\mathbf{W}_c^V)$. We set $\text{Aggregate}(\cdot)$ to be an average operation following [8].

C. Output Layer

The model incorporates z'_i and position encoding $v_i\mathbf{W}^Q$ to obtain the representation of node i in the cascade as follows:

$$\mathbf{h}_i^{cas} = \text{sigmoid}(z'_i \oplus v_i\mathbf{W}^Q) \tag{5}$$

where $\text{sigmoid}(\cdot)$ represents the sigmoid function.

We calculate the cosine similarities between $\mathbf{h}_i^{cas}$ and incorporate the network structure embedding and spread nodes. Thus, we can obtain the hidden representation of the next activated node:

$$\mathbf{h}_i^{\text{next}} = cosine(\mathbf{h}_i^{\text{cas}}, \text{sigmoid}((\mathbf{W}^M \oplus \mathbf{W}^P)^T)) \tag{6}$$

Finally, we can predict the next activated node by using the probability distribution as follows:

$$\mathbf{p}_i^{\text{next}} = \text{sigmoid}(\mathbf{h}_i^{\text{next}}\mathbf{W}^E + \mathbf{b}) \tag{7}$$

where $\mathbf{W}^E$ and $\mathbf{b}$ are parameters.

IV. Optimization

To train the model on a set of cascades $Q = \{S_f\}_{f=1}^F$, we first treat all the data as independent and identically distributed. Then, the loss of the observed dataset can be represented by equation 8 in the following.

$$\textbf{Loss}(Q) = \sum_{f}^{F} \sum_{i=1}^{N_{f-1}} \log\, p(v_{k+1}|c_{v_k}, S_{\leq k}) \tag{8}$$

The equation is obtained by summing the logarithmic likelihood for all the individual cascade data. The parameters of the model can be trained by minimizing the negative loss of the model.

We use the back propagation through time (BPTT) method to train our deep learning model and use the stochastic optimization method Adam [22] to update the parameters of the model. The clip gradient norm is used in the training process to prevent the parameters from overfitting.

V. Experimental Setup

To illustrate the highlights of our proposed model, we conduct two types of datasets: synthetic datasets and a real-world dataset. Experiments are run on the models based on a deep learning framework and are presented in the section V-B. The evaluation metrics are shown in the section V-C.

A. Datasets

We use synthetic datasets to verify the good performance of our model when modeling and predicting the sequence cascade data produced on the network with community structure. Moreover, the real-world dataset is used to verify the good performance of our model in the real situation. The descriptions of all the datasets are shown in Table I.

The degree distributions of all the datasets' networks are shown in Figure 3. The x-axes are the degrees of the nodes, and the y-axes are the frequencies of the degrees. We use the logarithmic coordinates to show the datasets.

1) Synthetic Datasets: We perform dataset generation in two parts following the previous work, which contains the network generation part and the cascade generation part [6].

The Network Generation Part. In this part, we use the two popular types of network generation tools, which are the Kronecker graph model [23] and the LFR benchmark [24], to generate two types of networks. We generate a random network (RD) with parameters [0.5 0.5; 0.5 0.5] [6] using the Kronecker graph model. The size of the network is 1024,

TABLE I
THE DESCRIPTION OF ALL THE DATASETS.

	500,Exp	500,Ray	1000,Exp	1000,Ray	Rd,Exp	Rd,Ray	Digg
# Nodes	500	500	1000	1000	1024	1024	139,409
# Edges	5847	5847	12,285	12,285	2048	2048	1,731,658
# Cascades	10,000	10,000	20,000	20,000	20,480	20,480	3553
Avg. cascade length	173	166	257	252	62	53	350

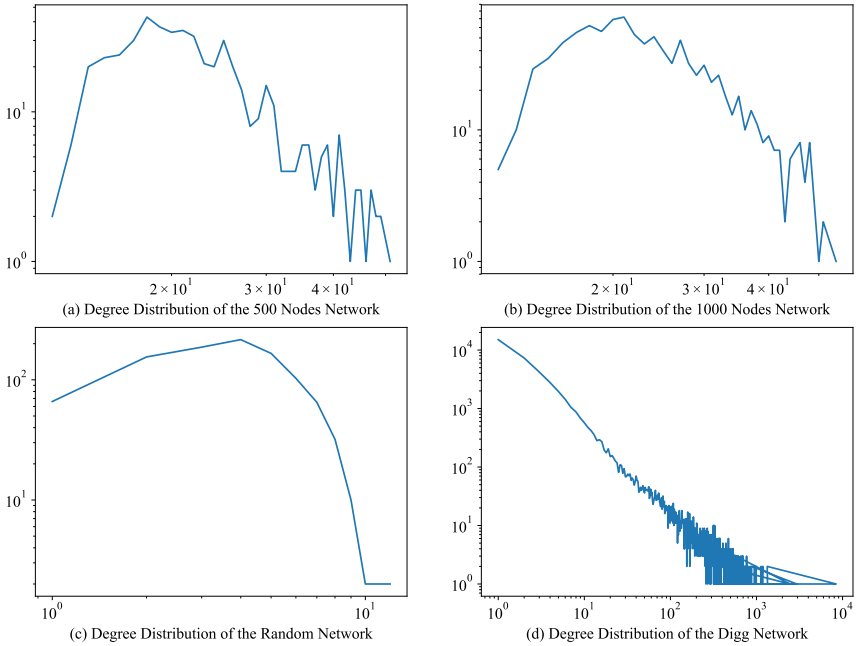

Fig. 3. **The degree distribution of all the datasets' networks.** The x-axes are the degrees of the nodes, and the y-axes are the frequencies of the degrees. We use the logarithmic coordinates to show the datasets.

and the average degree is 20. The LFR benchmark is used to generate the networks with heterogeneous community sizes and heterogeneous degree distributions. We set the parameters as the following: the average degree of nodes is 20, the maximum degree of nodes is 50, the power-law exponent for the degree distribution is 2, and the power-law exponent for the community size distribution is 1. We set the size of the network nodes to 500 and 1000 to generate two networks.

The Cascade Generation Part. We generate a cascades dataset by using the breadth-first search method to obtain the spreading nodes' sequences. Two types of time distributions for the node sampling method are used following Wang's setup [6]. One is the mixed exponential (Exp) distribution, whose parameters are scaled by [0.01, 10], and the other is the mixed Rayleigh (Ray) distribution, whose parameteres are scaled by [0.01,10]. The cascade sequence dataset generation progress will continue until the parameters of the time reach the maximum number ω or when no node is activated, following [6]. In this paper, ω is set as 100. For the each node, we generate the cascades 20 times.

Thus, we can obtain six synthetic datasets represented by: (500 Exp), (500, Ray), (1000, Exp), (1000, Ray), (Rd, Exp), and (Rd, Ray). Finally, for each dataset, we randomly select 80% as the training set, 10% as the validation set, and 10% as the testing set, following [6].

2) Real-world Dataset: We use the the Digg dataset, which is proposed by Nathan *et al* [25], to evaluate all the models. The dataset contains diffusions of stories as voted by users, along with the friendship networks of the users [26]. We drop the cascades with a size of less than 50 to verify the effectiveness of the long sequence modeling of our model. We randomly select 80% of the cascades for training, 10% for validation and 10% for testing.

B. Comparison Methods

Since our model is based on deep learning, we compare our proposed model with the current models based on deep learning models. All the compared models are shown in the following.

RNNPP [7]: This model extends the recurrent neural network(RNN) to model the point process. In this paper, we take the cascade sequence of the spreading nodes as the input, and the probability of the next activated node is obtained from the main-types event prediction layer.

Recurrent Marked Temporal Point Processes (RMTPP) [27]: This model is also based on the RNN model, and it models the intensity function of the temporal point process as a nonlinear function and can be used to predict the next activated nodes for an information cascade.

Sequential Neural Network with Structure Attention (SNNSA) [9]: The SNNSA is a recently proposed method for modeling information diffusion that can capture the structural dependency among users via an attention mechanism. The model is based on the RNN and only considers the local network structure information.

C. Performance Measures

We focus on the task to present the probability of the next activated node given the cascade sequence and the network structure. Since the result field is so large, we take the task as a users' ranking problem based on the users' transition probabilities [6]. Each model presents the activated probability distribution of all the nodes, and we take the node that has the highest probability as the next activated node [9]. In this situation, we use the popular metrics:

- **ACC@k:** The accuracy value of the top-k ranked prediction nodes. In this paper, we just use ACC@5 and ACC@10 metrics to evaluate the models.
- **MRR:** The mean reciprocal rank value of the ranked prediction nodes.

VI. Experimental Results

We use the PyTorch[1] library to train and evaluate our model and the compared baseline methods. We run all the models on a GPU server, which contains a V100 32 GB NVIDIA Tesla GPU, a Intel Xeon E5 CPU, and a 512 GB memory. The model is trained in the multibatch model and stopped when convergence is reached. To ensure reasonable results, we train each model 20 times on every dataset and report the average values of all training times as the final results. The training cost time (second) of the next activated node prediction task in each epoch on all the datasets is presented in Table II. We can see that our proposed model runs faster than the SNNSA model and that the SNNSA model runs out of memory on the Digg dataset. The results of the synthetic datasets and the real-world dataset are shown in the following.

A. Synthetic data results

We use the ACC@5, ACC@10, and MRR metrics to evaluate all the models on the prediction of the next activated node task, and the compared results are shown in Table III, Table IV, and Table V, respectively. The results show that the proposed CPMHSA method performs consistently and significantly better than the state-of-the-art models with respect to Acc@5, Acc@10, and MRR metrics on all of the datasets. Since our synthetic datasets are the cascade data generated from the random network or the network that contains the community structure, it illustrates that our proposed model can capture the network structure information from the dataset and achieve more accurate prediction results than the other models. The compared model performs better on the (Rd,Exp) and (Rd,Ray) datasets than the others, while our proposed model performs equally well. Importantly, the (Rd,Exp) and (Rd,Ray) datasets are known to have a shorter cascade length than the other datasets. This phenomenon shows that our proposed model is robust to the length of cascade.

[1]The PyTorch library is available at https://pytorch.org

B. Real data results

We show the prediction results of the next activated node in the Table VI for all the models on the Digg dataset. We can see that our proposed method achieves good performance on all the metrics, which verifies the effectiveness of our model. Moreover, the SNNSA model runs out of memory during the training process, which is interpretable. SNNSA uses the whole adjacency matrix of the network, whereas our model uses the result of the network embedding. The dimension of the network embedding is far less than the dimension of the network adjacency matrix; thus, the space complexity of SNNSA is much larger than that of our model. This feature makes the SNNSA model hardly capable of handling large-scale cascade datasets, while our model can solve the issue easily.

VII. Conclusion

In this paper, we propose a new model based on the multihead self-attention mechanism for the cascade modeling and prediction task. The proposed model can deal with the long cascade sequence easily because of the self-attention mechanism used. Since the network structure information considerably affects the spread of information, we considered it in the proposed model. We also use the multihead attention mechanism to capture diverse user dependencies on the network.

As evaluated by the experiments on both synthetic and real datasets, we find that our model can achieve better performance than the other models in the next activated node prediction task. Additionally, although the SNNSA model involves the network structure information, the model can hardly process the long cascade sequence and requires more time than our model.

Acknowledgment

This research is partly supported by the Chinese National Funding of Social Sciences 15BTQ056, the National Key Research and Development Program of China (No. 2018YFC0832101), and the Major Judicial Studies of the Supreme People's Court (ZGFYZDKT2019-01).

TABLE II
TRAINING COST TIME(SECOND) OF THE NEXT ACTIVATED NODE PREDICTION TASK IN EACH EPOCH ON ALL THE DATASETS.

Method	500,Exp	500,Ray	1000,Exp	1000,Ray	Rd,Exp	Rd,Ray	Digg
RNNPP	8	8	23	23	11	11	87
RMTPP	6	6	17	17	7	7	58
SNNSA	72	72	221	221	210	210	-
CPMHSA	31	31	130	130	53	53	159

TABLE III
PREDICTIVE PERFORMANCE (ACC@5) FOR THE NEXT ACTIVATED NODE OF THE BASELINES AND OUR PROPOSED MODEL CPMHSA.

Method	500,Exp	500,Ray	1000,Exp	1000,Ray	Rd,Exp	Rd,Ray
RNNPP	0.0793	0.1796	0.0756	0.1388	0.0800	0.1337
RMTPP	0.3486	0.3308	0.3443	0.3094	0.6807	0.7144
SNNSA	0.3466	0.2851	0.2835	0.2357	0.7135	0.6566
CPMHSA	**0.8377**	**0.8124**	**0.9121**	**0.9114**	**0.7568**	**0.8635**

TABLE IV
PREDICTIVE PERFORMANCE (ACC@10) FOR THE NEXT ACTIVATED NODE OF THE BASELINES AND OUR PROPOSED MODEL CPMHSA.

Method	500,Exp	500,Ray	1000,Exp	1000,Ray	Rd,Exp	Rd,Ray
RNNPP	0.1253	0.2142	0.0937	0.1604	0.0395	0.1389
RMTPP	0.4600	0.4508	0.4533	0.4117	0.7208	0.7925
SNNSA	0.4790	0.4217	0.4090	0.3515	0.8116	0.7792
CPMHSA	**0.8534**	**0.8379**	**0.9245**	**0.9301**	**0.8204**	**0.8986**

TABLE V
PREDICTIVE PERFORMANCE (MRR) FOR THE NEXT ACTIVATED NODE OF THE BASELINES AND OUR PROPOSED MODEL CPMHSA.

Method	500,Exp	500,Ray	1000,Exp	1000,Ray	Rd,Exp	Rd,Ray
RNNPP	0.0768	0.1566	0.0757	0.1215	0.0254	0.1312
RMTPP	0.2303	0.2208	0.2263	0.2063	0.4982	0.4736
SNNSA	0.2339	0.1960	0.1910	0.1579	0.4764	0.4309
CPMHSA	**0.8342**	**0.8014**	**0.9021**	**0.8942**	**0.7114**	**0.8521**

TABLE VI
PREDICTIVE PERFORMANCE OF THE NEXT ACTIVATED NODE ON THE DIGG DATASET.

Method	ACC@5	ACC@10	MRR
RNNPP	0.0054	0.0093	0.0064
RMTPP	0.0154	0.0236	0.0147
SNNSA	-	-	-
CPMHSA	**0.7846**	**0.7958**	**0.7544**

REFERENCES

[1] Z.-K. Zhang, C. Liu, X.-X. Zhan, X. Lu, C.-X. Zhang, and Y.-C. Zhang, "Dynamics of information diffusion and its applications on complex networks," *Physics Reports*, vol. 651, pp. 1–34, 2016.

[2] C. Li, J. Ma, X. Guo, and Q. Mei, "Deepcas: An end-to-end predictor of information cascades," in *Proceedings of the 26th international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 577–586.

[3] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf, "Uncovering the temporal dynamics of diffusion networks," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 2011, pp. 561–568.

[4] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 137–146.

[5] Q. Zhang, Y. Gong, J. Wu, H. Huang, and X. Huang, "Retweet prediction with attention-based deep neural network," in *Proceedings of the 25th ACM international on conference on information and knowledge management*. ACM, 2016, pp. 75–84.

[6] Y. Wang, H. Shen, S. Liu, J. Gao, and X. Cheng, "Cascade dynamics modeling with attention-based recurrent neural network." in *IJCAI*, 2017, pp. 2985–2991.

[7] S. Xiao, J. Yan, X. Yang, H. Zha, and S. M. Chu, "Modeling the intensity function of point process via recurrent neural networks," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[9] Z. Wang, C. Chen, and W. Li, "A sequential neural information diffusion model with structure attention," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018, pp. 1795–1798.

[10] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec, "Can cascades be predicted?" in *Proceedings of the 23rd international conference on World wide web*. ACM, 2014, pp. 925–936.

[11] M. G. Rodriguez, D. Balduzzi, and B. S. lkopf, "Uncovering the temporal dynamics of diffusion networks," *international conference on machine learning*, pp. 561–568, 2011.

[12] K. Saito, M. Kimura, K. Ohara, and H. Motoda, "Learning continuous-time information diffusion model for social behavioral data analysis," in *Asian Conference on Machine Learning*. Springer, 2009, pp. 322–337.

[13] C. Liu, W. Wang, and Y. Sun, "Community structure enhanced cascade prediction," *Neurocomputing*, vol. 359, pp. 276–284, 2019. [Online]. Available: https://academic.microsoft.com/paper/2951595213

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training

of deep bidirectional transformers for language understanding," in *NAACL-HLT 2019: Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 4171–4186. [Online]. Available: https://academic.microsoft.com/paper/2963341956

[15] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones, "Character-level language modeling with deeper self-attention," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3159–3166.

[16] K. Lerman and R. Ghosh, "Information contagion: An empirical study of the spread of news on digg and twitter social networks," in *Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

[17] Z. Liu and B. Hu, "Epidemic spreading in community networks," *EPL (Europhysics Letters)*, vol. 72, no. 2, p. 315, 2005.

[18] R. Pastor-Satorras and C. Castellano, "Distinct types of eigenvector localization in networks," *Scientific reports*, vol. 6, p. 18847, 2016.

[19] W. Huang and C. Li, "Epidemic spreading in scale-free networks with community structure," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2007, no. 01, p. P01014, 2007.

[20] X. Ma, P. Zhang, S. Zhang, N. Duan, Y. Hou, M. Zhou, and D. Song, "A tensorized transformer for language modeling," in *NeurIPS 2019 : Thirty-third Conference on Neural Information Processing Systems*, 2019. [Online]. Available: https://academic.microsoft.com/paper/2970213198

[21] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1225–1234.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 985–1042, 2010.

[24] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E*, vol. 78, no. 4, p. 046110, 2008.

[25] N. O. Hodas and K. Lerman, "The simple rules of social contagion," *Scientific reports*, vol. 4, p. 4343, 2014.

[26] T. Hogg and K. Lerman, "Social dynamics of digg," *EPJ Data Science*, vol. 1, no. 1, p. 5, 2012.

[27] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "Recurrent marked temporal point processes: Embedding event history to vector," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1555–1564.