Cascade Dynamics Modeling with Attention-based Recurrent Neural Network

Abstract

1 Introduction

The emergence of Social Media platform has revolutionized dissemination of information via its great ease in inforamtion delivery, accessing and filtering. In Social Media, pieces of information, posted by users spontaneously, propagate along social relationships between users, explict or implict, forming cascade dynamics. Modeling cascade dynamics is the fundamental to understand information propagation and launch series of social applictions, i.e., viral marketing, popularity prediction and rumor detection.

The key to cascade dynamics modeling is to find a well-defined function in hypothesis space based on observed cascades. Existed methods for this problem fall into three main paradigms: pairwise, nodewise and eventwise modeling. The majority works in cascade dynamics modeling focus on pairwise modeling, defining the propagation probability of information between all pairs of users [Saito *et al.*, 2008; Goyal *et al.*, 2010; Gomez-Rodriguez *et al.*, 2013]. However, pairwise models suffer severe overfitting and overrepresentation problems especially in sparse social data, proved in [Wang *et al.*, 2015]. Nodewise modeling learn latent user-specific characteristics instead of pairwise manners, effectively combating overfitting and overrepresentation problems. Bourigault et.al. [Bourigault *et al.*, 2016] learn user-specific latent space in Independent Cascade (IC) model. Wang et.al [Wang *et al.*, 2015] capture users' influence and susceptibility in latent space and define propagation probability according to users latent characteristics. Kurashima et.al. [Kurashima *et al.*, 2014] embeded users into low-dimensional visualization space in Continous Time Independent Cascade (CTIC) model. But nodewise models require strong prior knowledge on generation processes of cascades in order to better fit the observations. Recently, eventwise models received great success in modeling sequence data.

Eventwise methods aims to learn history embedding in order to model the generation of next event, e.g., cascade. Manavoglu [Manavoglu *et al.*, 2003] propose users behavior generation method based on maxent and Markov mixture model. Recently, the efficient way of eventwise modeling can be achieved by Recurrent Neural Network (RNN) [Bengio *et al.*, 2003; Goldberg and Levy, 2014; Mikolov *et al.*, 2010; Sundermeyer *et al.*, 2012]. Du et.al. [Du *et al.*, 2016] proposed a Recurrent Marked Temporal Point Process (RMTPP) for event streams. The outputs of hidden layer in Recurrent Neural Network (RNN) represents the embedding of the event histories, then parameterizing the random process. The benefits of eventwise modeling are two folds: 1) avoiding strong prior knowledge on models and networks with respect to different observed cascades; 2) enlarging the functional space when searching the optimal cascade dynamics models, which may have great probability to better model cascade dynamics.

Despite of advantages in eventwise modeling, the traditional sequence models may meet "crossing dependency" problem in cascade dynamics modeling. The crossing dependencies problem is mainly caused by tree structure of propagation. Fig. 1 shows two typical crossing dependency cases in practical. For modeling dependence between 1st and 3rd event, we must use redundant information passing from 2nd event, called "redundant modeling". If we abandon useful information inherited by 3rd event when modeling the 4th event, the generation of 5th event would lose useful information from 3rd event, called "cut-off modeling". Crossing dependency problems limit the efficiency of sequence modeling.

In this paper, we propose a **C**ascade d**Y**namics modeling with **A**ttentio**N**-based RNN, named (CYAN-RNN). We construct a pooling layer above the output of hidden layer in RNN, aggregating event embedding in history. The weights in pooling layer pointing to each historical event embedding refers to connections between current event and history. We choose attention mechanism [Bahdanau *et al.*, 2014] to realize the pooling layer, automatically learned the connection weights. The benifits of our proposed model are three-fold: 1) We propose a eventwise method, using sequence modeling, for cascade dynamics modeling; 2) We point out crossing dependency problem in traditional sequence modeling when model cascade dynamics. Thus, we proposed CYAN-RNN to solve the problem; 3) We conduct experiments on synthetic and real-world datasets to show that our model consistently outperform than previous modeling methods in cascade dynamics modeling.

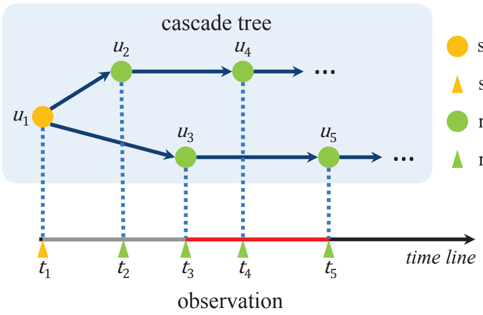


Figure 1: Tree structure of propagation and crossing dependency problems in sequence modeling. For modeling dependence between 1st and 3rd event, we must use redundant information passing from 2nd event, called "redundant modeling". If we abandon useful information inherited by 3rd event when modeling the 4th event, the generation of 5th event would lose useful information from 3rd event, called "cut-off modeling".

2 Model

2.1 Background

Firstly, we introduce RNN as an eventwise modeling method in cascade dynamics modeling. A cascade $S = \{x_i | x_i = (t_i, u_i), u_i \in U \text{ and } t_i \in R^+\}_{i=1}^N$ is a set of propagations asendingly ordered by time, where U refers to all possible users in cascades. The i-th propagation x_i is recorded as a tuple (t_i, u_i) referring to activated time and activated user respectively. At each step k, the k-th propagation are fed into hidden units by nonlinear transformation f, jointly with the outputs from the previous hidden units, updating the hidden state $h_k = f(x_i, h_{k-1})$. The representation of hidden state h_k can be considered as event embedding to the k-th propagation [Du *et al.*, 2016], and the output is trained to predict the next propagation x_{k+1} given h_k. In other words, we use RNN to maximize the likelihood probability of observed propagations,

$$p(\mathbf{x}) = \prod_{k=1}^{N} p(x_{k+1}|h_k) \tag{1}$$

Based on sufficient observed cascades, RNN can find an optimal solution for Eq. (1) in a huge functional space, avoiding the limits of prior knowledge. Thus, RNN can be a promising method to capture the complex propagation patterns in cascade dynamics modeling.

However, RNN suffers crossing dependency problem caused by tree structure propagations in cascade, shown in Fig. 1. One of the possible solutions is to construct a pooling layer above the hidden units in order to build the direct dependency between the generation of k-th propagation and all previous event embeddings, i.e., $p(x_{k+1}|\text{pooling}(h_1, \ldots, h_k))$. The simplest way of pooling can be formalized as

$$s_k = \sum_{i=1}^{k} \alpha_{k,i} h_i, \quad \text{s.t.} \sum_{i=1}^{k} \alpha_{k,i} = 1, \tag{2}$$

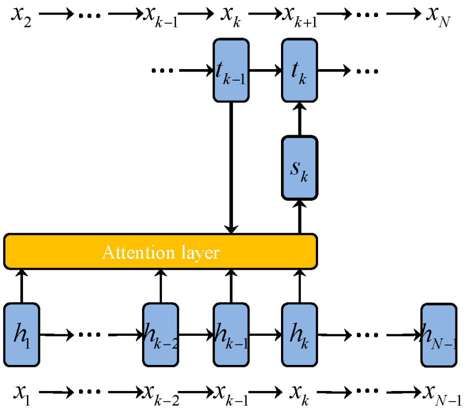

$x_2 \rightarrow \cdots \rightarrow x_{k-1} \rightarrow x_k \rightarrow x_{k+1} \rightarrow \cdots \rightarrow x_N$
t_{k-1}
t_k
s_k
Attention layer
h_1
h_{k-2}
h_{k-1}
h_k
h_{N-1}
$x_1 \rightarrow \cdots \rightarrow x_{k-2} \rightarrow x_{k-1} \rightarrow x_k \rightarrow \cdots \rightarrow x_{N-1}$

Figure 2: The architecture of CYAN-RNN. The figure presents the case when predicting the $(k+1)$-th propagation. The sequence in bottom is the observed cascade and the sequence in top is the cascade shifted one step according to the observation. The blue rectangles refer to representations from the hidden units in source sequence, attention layer, and hidden units in target sequence. The yellow rectangle is a component functioned in neural network.

where the weight $\alpha_{k,i}$ refers to the proportion of dependency between next propagation and the i-th event embedding. Mean pooling and max pooling are two popular choices for setting weights which takes the mean or element-wise max of all hidden states. However, these two methods still ignore the structure information in cascades. Intuitively, the weights can be settled according to the cascade tree, yet we can hardly observe the tree structure even given the social relationships. Next we propose attention mechanism to implement the pooling layer.

2.2 CYAN-RNN

Attention mechanism is orginally used in neural machine translation (NMT), firstly proposed by Bahdanau et.al [Bahdanau *et al.*, 2014]. In the senario of attention-based NMT, the target words are translated by the words in source sequence and attention mechanism can automatically learn the alignment between source words and target words []. However, there are two problems when applying attention mechanism in cascade dyanmics modeling: 1) only one single sequence can be observed in a cascade; 2) the size of alignments in attention mechanism need to be updated when the context $\{h_1, \ldots, h_k\}$ is growing along with the proceeding propagations.

Thus, we propose CYAN-RNN for cascade dynamics modeling, applying a dynamic attention mechanism to solve the crossing dependency problem. The architecture of CYAN-RNN is shown in Fig. 2. In CYAN-RNN, we conduct the observed cascade as the source sequence. The target sequence is the copy of source sequence, shifting one step of the copy backwards. The objective is to jointly predict the next propagation and learn the alignment of dependency between the next propagation and histories. According to the architecture, we define each conditional probability in Eq. (1) as:

$$p(x_{k+1}|x_1,\dots,x_k) = g(x_k, t_k, s_k), \tag{3}$$

where g is an objective function defined the joint probability of propagation on activated user and activated time. Here we use the objective function defined on RMTPP [Du *et al.*, 2016]. The t_k is a hidden state related to the k-th step of target sequence, computed by

$$t_k = f(x_k, t_{k-1}, s_k), \tag{4}$$

where f is a nonlinear activation function, which can be either a *tanh* or *sigmoid* function. The context vector s_k is calculated by Eq. (2) where the weights $\alpha_{k,\cdot}$ is updated by the growth context $\{h_1,\dots,h_k\}$ and t_{k-1}. We can get the weights

$$\alpha_{k,i} = \frac{\exp(e_{k,i})}{\sum_{j=1}^{k}\exp(e_{k,j})}, \tag{5}$$

where

$$e_{k,i} = a(t_{k-1}, h_i) = v^T \tanh(Wt_{k-1} + Uh_i) \tag{6}$$

scores how well the dependency between the i-th event embedding and the output at the k-th step. The implementation of attention mechanism in proposed model is briefly represented in Fig. 3(a).

Given a collection of cascades $\mathcal{C} = \{S_m\}_{m=1}^{M}$, we suppose that each cascade is independent on each other. As a result, the logarithmic likelihood of a set of cascades is the sum of logarithmic likelihood of individual cascade. In this way, the negative logarithmic likelihood of the set of cascades can be estimated as

$$\mathcal{L}(\mathcal{C}) = -\sum_{m=1}^{M}\sum_{k=1}^{N_m}\left[g(x_k^{(m)}, t_k^{(m)}, s_k^{(m)})\right], \tag{7}$$

and we can learn parameters of the proposed model by minimizing the negative logarithmic likelihood. With the attention mechanism, the alignment weigts $\alpha_{k,\cdot}$ can be directly updated through the cost function, thus exploit an expected representation s_k over all historic event embeddings for each step k.

2.3 CYAN-RNN with Coverage

Although CYAN-RNN is proposed to better model cascade dynamics in consideration of crossing dependency problem, the proposed model still suffer *over-dependent* and *under-dependent* problems when applying attention mechanism. As the exmaple shown in Fig. 1, if the user u_1 is an influential user and his propagation is key to the cascade, the propagation activated by u_4 may perfer to depend more on embedding of (t_1, u_1) instead of (t_2, u_2). Here embedding of (t_1, u_1) is over-dependent and embedding of (t_2, u_2) is under-dependent. In practice, it is a common phenomenon that users may have a higher probability activated by recent propagation than past ones [] and we also conduct experiments to illustrate it (see section 4).

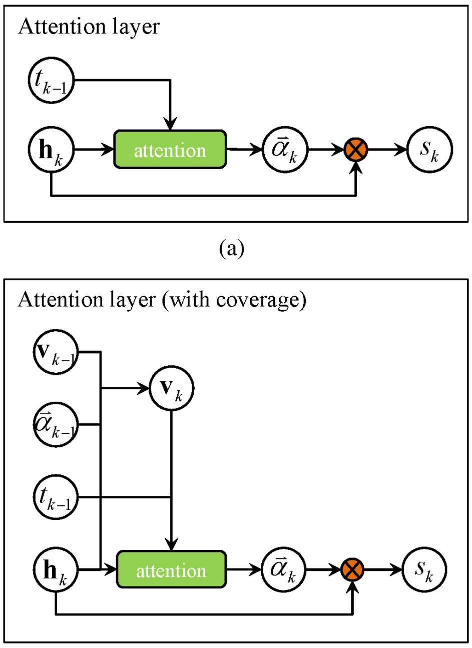
The two problems are caused by memoryless of dynamic attention mechanism. Inspired by linguistic coverage model [Tu *et al.*, 2016], we formulate the general form of coverage, keeping historical alignments so as to release the over-dependent and under-dependent problems. The k-th step of coverage is defined as

$$V_{k,i} = f\left(V_{k-1,i}, \alpha_{k-1,i}, t_{k-1}, h_i\right). \tag{8}$$

Remarkably, as the increasing propagations and alignments, $V_{k,k}$ and $\alpha_{k,k}$ have no corresponding values in $V_{k-1,\cdot}$ and $\alpha_{k-1,\cdot}$. Thus, we fill up with zeros in our works. Compared with $V_{k,i}$, $V_{k-1,i}$ At each step k, the k-th coverage serves an additional input to the attention mechanism, providing complementary information of that how about the dependencies of historical event embeddings are in the past. The rewritten alignment calculation in Eq. (6) by coverage can be formalized [1] as

$$\begin{aligned} e_{k,i} &= a(t_{k-1}, h_i, V_{k,i}) \\ &= v^T \tanh(Wt_{k-1} + Uh_i + ZV_{k,i}). \end{aligned} \tag{9}$$

We expect that the alignment weights would be focus more on recent event embeddings. The expectation will be validated in section 4.



(a)

(b)

Figure 3: Two kinds of implementation on attention. (a) Attention mechanism applied in CYAN-RNN; (b) Attention mechanism with coverage applied in CYAN-RNN (cov). Note that $\mathbf{h}_k = (h_1,\dots,h_k)$ is matrix assembled by all historic event embeddings at step k and $\mathbf{v}_k = (v_1,\dots,v_k)$ is a coverage martix containing all k-th coverage vectors.

[1] The formalization is determined by the incremental length of alignment weights. If we use the last coverage $V_{k-1,\cdot}$ instead of $V_{k,\cdot}$ (like [Tu *et al.*, 2016]) to update $e_{k,\cdot}$ at each k step, we will lose certain coverage information and cause unbalance calculation about k-th event embedding, proved by our preliminary experiments.

2.4 Window Size

Practically, a cascade may last long and the propagation length would be huge, causing an extreme computation cost when applying dynamic attention mechanism proposed in CYAN-RNN. According to perference of users interests on recent propagations, we consider a hyper-parameter, symboled by *window size* l, limiting the size of alignments so that the predicted task would only depend on last l propagations. Empirically, we set $l = 200$ in most cases.

3 Optimization

In this section, we introduce the learning process of CYAN-RNN. The k-th propagation are transformed into vectors as inputs including user embedding and temporal features. The user embedding matrix related to activated users is learned along with the training process. The temporal features related to activated time are formalized, inlcuding logarithm time interval $\log(t_k - t_{k-1})$ and discretization of numerical attributes on year, month, day, week, hour, mininute and second. We adopt GRU [Chung *et al.*, 2014] to encode the inputs of source sequence to event embeddings h. We apply back-propagation through time (BPTT) [Chauvin and Rumelhart, 1995] for parameter estimation. The parameters are iteratively updated by Adam [Kingma and Adam, 2015], an efficient stochastic optimization algorithm with mini-batch techniques. We also employ early stopping method [Prechelt, 1998] to prevent overfitting. The stopping criterion is achieved when the performance has no more improvement in validation set. For speeding up the convergence, we use orthogonal initialization method [Henaff *et al.*, 2016] in training process.

4 Experiments

In this section, we conduct empirical experiments to demonstrate the effectiveness of CYAN-RNN on cascade dynamics modeling. We compare the proposed model to the state-of-the-art cascade dynamics modeling methods on both synthetic and real data showing that CYAN-RNN can better model next propagations and infer the dependence structure in cascades.

4.1 Baselines

Rare methods can predict next propagations completely including both next activated users and time. To better illustrate the performance of our proposed model, we choose the state-of-the-art models for either predicting next activated users or predicting next activated time for comparisons in two prediction tasks.

(1) Prediction of next activated user

- **CT Bernoulli** and **CT Jaccard** models [Goyal *et al.*, 2010]: They are continuous time propagation models. The propagation probabilities between two users are defined by Bernoulli or Jaccard distribution and the probabilities are decayed over time.
- **MC-1** Model: The markov chain model is a classic sequence modeling method. Here we compare with markov chain with order one.

(2) Prediction of next activated time

- **Poisson process** model [Vere-Jones, 1988]: It is a stoachastic point process model, depicting the time consuming from one propagation to another. The intensity function is parameterized by a constant.
- **Hawkes process** model [Hawkes, 1971]: It is a stochastic point process model where the intensity function is parameterized by

$$\lambda(t) = \lambda(0) + \alpha \sum_{t_i < t} \exp\left(-\frac{t - t_i}{\sigma}\right), \tag{10}$$

where $\sigma = 1$ and $\lambda(0)$ is a intrinsic rate when $t = 0$.

We also compare with the model that has the ability to generate both mark and temporal sequences.

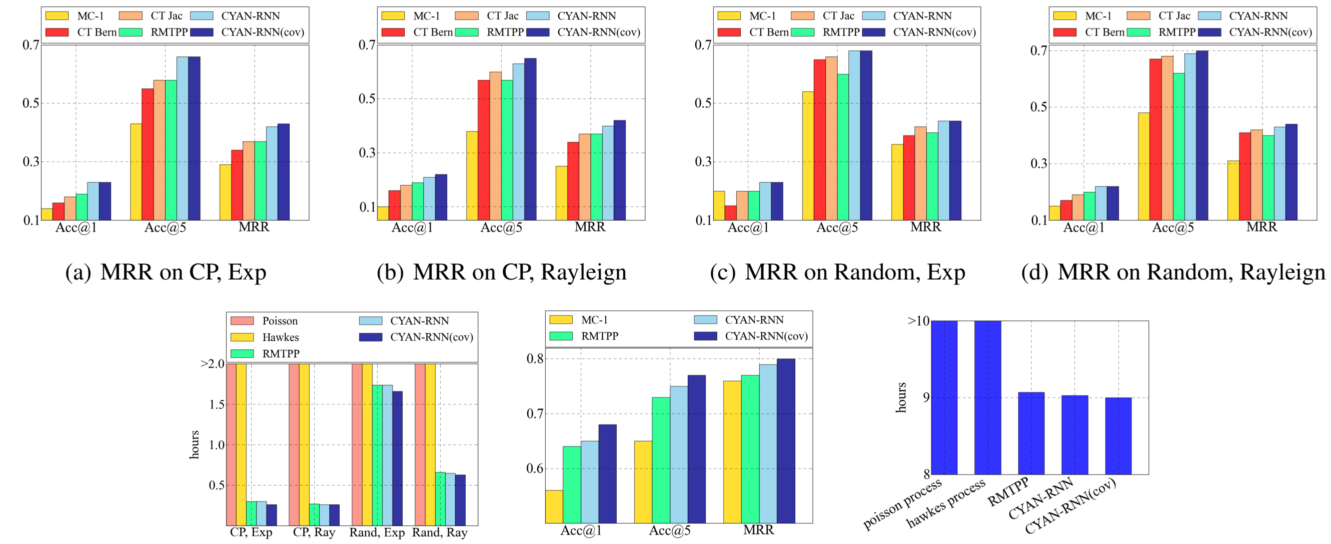
- **RMTPP** [Du *et al.*, 2016]: Recurrent marked temporal point process (RMTPP) is a method which can models both next activated user and time based on RNN.

4.2 Synthetic Data and Results Anlaysis

The goal of the experiments on synthetic data is to understand how the underlying network structure and propagation model affect our proposed model on both next propagation prediction and dependency structure inference.

Experimental setup. We use Kroneck generator [Leskovec and Faloutsos, 2007] to examine two types of networks with directed edges: 1) the Core-Periphery (CP) network (Kroneck parameter matrix $[0.962, 0.535; 0.535, 0.107]$), mimicing real-world social networks; 2) the Erdős-Rényi random (Random) network [Erdős and Rényi, 1960] ($[0.5, 0.5; 0.5, 0.5]$). The incubation time from activated user to anther on networks are sampled from two distributions: 1) mixed exponential (Exp) distributions, controlled by rate parameters α randomly choosing in $[0.01, 10]$; 2) rayleigh (Ray) distribution, controlled by scale parameters β randomly choosing in $[0.01, 10]$. To generate a cascade, we randomly choose a root user as the source of the cascade at first. For each neighbor of the activated user, its activated time is determined by incubation time. The propagation process will further conitune in breadth-first fashion until the overall time exceed the predefined observation time window or no new user being activated. In our experiments, we set the total number of users $|U| = 32$ and the maximal observation time $\max\{t\} = 100$. As a result, four kinds of datasets are obtained by the different unions between network and cascade generation distribution, i.e., (CP, Exp), (CP, Rayleign), (Random, Exp) and (Random, Rayleign). The number of simulated cascades is up to 20,000 in each dataset, where we randomly pick up 80% of completed sequences for training and the rest sequences are divided into two parts equally as validation and test set respectively.

Evaluation metrics. We regard the prediction task on next activated user as a ranking problem with respect to transition probabilities. The predictive performance is evaluated by *Accuracy on top* k (Acc@k) and *Mean Reciprocal Rank* (MRR) [Voorhees, 1999], functionized in top k and global perspectives. The larger values in Acc@k and MRR indicate the better performance. On the prediction of next activated



(a) MRR on CP, Exp

(b) MRR on CP, Rayleign

(c) MRR on Random, Exp

(d) MRR on Random, Rayleign

(e) RMSE on toy data

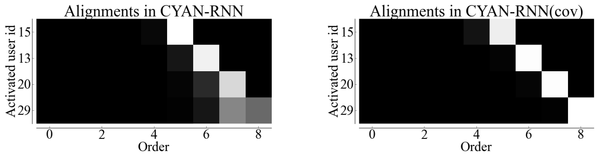
(f) MRR on real data

(g) RMSE on real data

Figure 4: Comparisons on baselines and our proposed models. (a)∼(e) The predictions of next activated user and time on toy data produced from different networks and propagation models; (f) and (g) The predictions of next activated user and time on real data.

time, we use Root Mean Square Deviation (RMSE) between estimated time and practical occuring time. The better performance means the small values in RMSE.

Prediction results. We conduct experiments on all four kinds of datasets. We firstly compare the predictive preformance on next activated users. The results on predictions of next activated users and time are shown in Fig. 4(a)∼ 4(d). As shown in the figures, CYAN-RNN and CYAN-RNN(cov) perform consistently and significantly better than other baselines in metrics of Acc@1, Acc@5 and MRR, included in all datasets. The results indicate that our proposed methods can better predict next activated users. It is interested that RMTPP has lower accuracy or MRR values than CT Bern and CT Jac in some cases shown in Fig. 4(b), 4(c) and 4(d), however, CYAN-RNN and CYAN-RNN(cov) can still performs better. It clearly demonstrates that the proposed attention mechanism has the ability to directly capture past event information which may be "forgetten" by sequential transitions in RNN, called crossing dependency problems in CDM. Fig. 4(e) compares the predictive performance on RMSE. We can observe that Poisson and Hawkes processes are the worst modeling methods, obtaining the errors larger than 2 hours in all datasets. Meanwhile, the RMSE values are similar between RMTPP and CYAN-RNN. But CYAN-RNN(cov) can perform slightly better than RMTPP and CYAN-RNN in all datasets when predicting next activated time. Additionally, we can observe that CYAN-RNN(cov) consistently perform better or even than CYAN-RNN in two prediction tasks. It indicates that the coverage can help to efficiently utilize the event embeddings in attention mechanism. Next we will exploit the answers how the coverage can help to boost predictive performance and lead to better inference on dependency structure.



(a)

(b)

Figure 5: Sample alignments on a fragement of cascade. The y-axis is the users who will be activated next sequentiallly from top to bottom. The x-axis is the activated order related to next activated user in the cascade. Each pixel shows the weight $\alpha_{k,i}$ related to i-th event embedding at each step k, in grayscale (0:black, 1:white). (a) the alignments inferred by CYAN-RNN; (b) the alignments inferred by CYAN-RNN(cov).

Alignment quality. Firstly, we expect to check if coverage can reduce the over-dependent and under-dependent problem mentioned in section 2.3. Fig. 5(a) and 5(b) show an example. Every grid in each plot represents the alignment weights associated with event embeddings. The brighter grid refers to the larger weights corresponding to next propagation. From this we see which positions in the past propagations were considered more important when predicting next propagation. Comparing to the alignments in CYAN-RNN, we can observe that the alignments in CYAN-RNN(cov) concentrate more on unused event embeddings, which is consistent to well-studied phenomenon in published works []. Moreover, we wonder if the inferred alignments is homologous to true propagation structures. Thus, we calculate alignments on each step of cascades in test data. We suggest that the propagation structure of next propagation can be inferred by the largest alignment at each step. We then get a matrix of statistic results based on all inferred structures. Every column in the matrix refers to the number of who mainly activate the propagation. As the high connection between propagation structure and network, the inferred structures would be the edges in network. Therefore, we normalize the matrix by rows, and check if the network edges can be classified by the matrix. As the inference results are same in other two datasets, we only show the AUC values computed by the models trained in dataset of (CP, Exp) and (Random, Exp), decribed in Table 1. The results from our proposed models are both worked in network inference. Besides, CYAN-RNN(cov) obtain better results of network inference than CYAN-RNN in two test data. It indicates that our proposed alignment mechanism can be natually used in inference of hidden propagation structure, which may have some potential applications in practice, e.g., advertisement and recommendation.

Table 1: AUC values of network inference.

Network	CYAN-RNN	CYAN-RNN(cov)
CP	0.83	0.84
Random	0.95	0.96

4.3 Real Data and Result Analysis

Experimental setup. The real data is extracted from Sina Weibo, a Chinese microblogging website, spanning from June 1st, 2016 to June 30th, 2016. We focus the records in June 1st and extract users whose posting numbers are ranged in $(100, 200]$. Then we sort records according to the root message IDs posted by the filtered users in 30 days and arrange them ascendingly by posting time. We drop the cascades that the number of propagations is larger than 1,000. The long length of propagations may dominate the training process, however, rarely occurred in practice. Finally, the processed data contains 2964 users and 596,088 cascades. We use 536,240 sequences for training, 29,758 for validation and 30,005 for testing. The task is to predict next propagation, including activated user and time.

Prediction results. The results on predictions of next activate users and time are shown in Fig. 4(f) and 4(g). The hyper-parameters of CYAN-RNN and CYAN-RNN(cov) are setted up as following: learning rate is 0.0001; hidden layer size of encoder is 20; hidden layer size of decoder is 10; window size is 200; coverage size is 10; and batch size is 128. Note that we have no social network in extracted real data, thus we cannot compare our proposed models with CT Bern and CT Jac. CYAN-RNN and CYAN-RNN(cov) outperform the other baselines with higher MRR values and lower RMSE values for predicting both next activated users and time. Comparing to RMTPP, CYAN-RNN(cov) recieves 6.25%, 5.48% and 3.90% relative increased performance on Acc@1, Acc@5 and MRR respectively, and reduces 0.78% relative errors on RMSE. Comparing to CYAN-RNN, CYAN-RNN(cov) recieves 4.62%, 2.67% and 1.27% relative increased performance on Acc@1, Acc@5 and MRR respectively, and reduces 0.33% relative errors on RMSE.

5 Conclusion

In this paper, we present the cascade dynamics modeling with attention-based RNN. As we know, it is a prior attempt on CDM based on RNN, which embed historical propagations into vectors and then determine the next propagation sequentially. Instead of traditional interpersonal influence modeling, RNN can be used to discover complex dependencies and patterns between the following propagation and current histories. However, RNN suffers crossing dependency problem when applying in CDM. To solve the problem, we propose attention mechanism above hidden units of RNN, named CYAN-RNN, aggregating all current historical embeddings. Thus, the generation of next propagation can directly depend on all current histories instead of transitive dependent way. Moreover, we propose CYAN-RNN(cov) to construct coverage on attention mechanism in order to solve over-dependent and under-dependent problem existed in CYAN-RNN. In experiments, we evaluate the effectiveness of our proposed model on both synthetic and real datasets. Experimental results demonstrate that our proposed models can consistently outperform compared modeling methods at both predicition tasks of next activated user and time. Addtionallly, CYAN-RNN(cov) performs better or even than CYAN-RNN on both synthetic and real datasets, proving that coverage can help to efficiently utilize historical embeddings in attention mechanism. Besides, we conduct experiments to exploit alignment quality. The results show that the alignments from our proposed models can reflect true propagation structures, which may be well applicable in practice.

References

[Bahdanau *et al.*, 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model, 2003.

[Bourigault *et al.*, 2016] Simon Bourigault, Sylvain Lamprier, and Patrick Gallinari. Representation learning for information diffusion through social networks: an embedded cascade model. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 573–582. ACM, 2016.

[Chauvin and Rumelhart, 1995] Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology Press, 1995.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[Du *et al.*, 2016] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le 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*, pages 1555–1564, 2016.

[Erdős and Rényi, 1960] P. Erdős and A Rényi. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[Goldberg and Levy, 2014] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[Gomez-Rodriguez *et al.*, 2013] Manuel Gomez-Rodriguez, Jure Leskovec, and Bernhard Schlkopf. Modeling information propagation with survival theory. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 666–674, 2013.

[Goyal *et al.*, 2010] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. Learning influence probabilities in social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 241–250. ACM, 2010.

[Hawkes, 1971] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[Henaff *et al.*, 2016] Mikael Henaff, Arthur Szlam, and Yann LeCun. Orthogonal rnns and long-memory tasks. *arXiv preprint arXiv:1602.06662*, 2016.

[Kingma and Adam, 2015] Diederik P Kingma and Jimmy Ba Adam. Adam: A method for stochastic optimization. In *International Conference on Learning Representation*, 2015.

[Kurashima *et al.*, 2014] Takeshi Kurashima, Tomoharu Iwata, Noriko Takaya, and Hiroshi Sawada. Probabilistic latent network visualization: Inferring and embedding diffusion networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1236–1245, New York, NY, USA, 2014. ACM.

[Leskovec and Faloutsos, 2007] Jure Leskovec and Christos Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 497–504, New York, NY, USA, 2007. ACM.

[Manavoglu *et al.*, 2003] Eren Manavoglu, Dmitry Pavlov, and C. Lee Giles. Probabilistic user behavior models. In *IEEE International Conference on Data Mining*, pages 203–210, 2003.

[Mikolov *et al.*, 2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3, 2010.

[Prechelt, 1998] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.

[Saito *et al.*, 2008] Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. Prediction of information diffusion probabilities for independent cascade model. In *Knowledge-Based Intelligent Information & Engineering Systems*, pages 67–75, 2008.

[Sundermeyer *et al.*, 2012] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *INTERSPEECH*, pages 194–197, 2012.

[Tu *et al.*, 2016] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. *ArXiv eprints, January*, 2016.

[Vere-Jones, 1988] D Vere-Jones. An introduction to the theory of point processes. *Springer Ser. Statist., Springer, New York*, 1988.

[Voorhees, 1999] Ellen M. Voorhees. The trec8 question answering track report. In *Text REtrieval Conference*, 1999.

[Wang *et al.*, 2015] Yongqing Wang, Huawei Shen, Shenghua Liu, and Xueqi Cheng. Learning user-specific latent influence and susceptibility from information cascades. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 477–483, 2015.