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 implicit, forming cascade dynamics. Modeling cascade dynamics is the fundamental to understand information propagation and launch series of social applications, i.e., viral marketing, popularity prediction and rumor detection.

In order to model the cascades, many information diffusion models have been proposed in the literature, such as discrete-time independent cascade model, discrete-time linear threshold model and continuous-time independent cascade model. A diffusion model is often associated with a directed graph and parameterized by propagation probabilities defined on edges or interpersonal influence defined on nodes. Kempe et al. [Kempe *et al.*, 2003] implemented the linear threshold model by a degree-modulated edge weight. Goyal et al. [Goyal *et al.*, 2010] provided two static models in terms of Bernoulli distribution and Jaccard index, and learned temporal factors to maximize likelihood of cascades. Gomez-Rodriguez et al. [Gomez-Rodriguez *et al.*, 2013] estimated a transimission function over time associated with each edge, assuming that observed diffusion time is independently sampled according to the transimission function. Wang et.al [Wang *et al.*, 2015] captured users' influence and susceptibility in latent space and modeled the transition probabilities of users' activated status according to users' latent characteristics.

However, the information diffusion is too complicated to easily determine the most appropriate diffusion model in practice. The misspecified diffusion model would lead to large model bias. Besides, the diffusion network structure can be also hidden from us. We need to learn the underlying diffusion network structure with respect to parameter estimation in diffusion model at most cases. It often leads to under-determined high computational cost where specialized methods need to be designed.

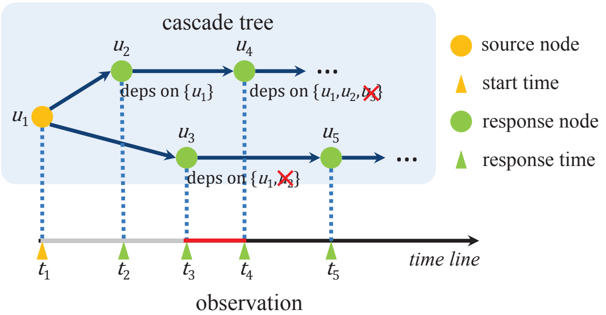


Figure 1: An example of crossing dependency problem in sequence modeling.

Instead of learning a particular diffusion model, sequence modeling can be used to discover complex dependencies and patterns between the following propagation and current histories, thus modeling the cascade dyanmics. For example, Manavoglue [Manavoglu *et al.*, 2003] proposed a user behavior generation method based on maxent and Markov mixture model. Recently, the efficient way of sequence modeling can be achieved by Recurrent Neural Network (RNN) [Goldberg and Levy, 2014; Mikolov *et al.*, 2010; Sundermeyer *et al.*, 2012]. Du et al. [Du *et al.*, 2016] proposed a RNN framework, called RMTPP, where the embedding of hidden units are used to parameterize the generation process of propagations. The benefits of sequence modeling are two-fold: 1) It avoids strong prior knowledge on diffusion model; 2) It has no limitation on hidden social network structure.

Despite of advantages in sequence modeling, the traditional methods may meet "crossing dependency" problem in cascade dynamics modeling, mainly caused by tree structure of propagation. Fig. 1 shows the crossing dependency dilemma in practical. As consideration of tree strucutre of propagations, we need to construct the dependence between the 1st and the 3rd propagations, causing redundant information usage passing from the 2nd one (the transitive dependence marked as grey in time line). If we abandon information inherited from the 2nd propagation, the generation of 4th propagation will lose useful information from the 2nd one as the corresponding user u_4 is directly connected to u_2 on the cascade tree (the transitive dependence marked as red in time line). In this paper, we propose a **C**ascade d**Y**namics modeling with **A**ttentio**N**-based RNN (CYAN-RNN) to better apply RNN in cascade dynamics modeling. We construct a dynamic pooling layer above hidden units of RNN, aggregating all embeddings of hidden units until current step. The generation of following propagation is relied on the aggregated embedding. We choose attention mechanism [Bahdanau *et al.*, 2014] to realize the pooling layer, where the parameters can be automatically learned by maximizing the likelihood of cascade. Moreover, we propose coverage in attention mechanism to ensure sufficient usage of each historical embedding involved in generation of following propagations. More specifically, our work makes the following contributions:

- We exploit crossing dependency problem existed in traditional sequence modeling when applied to model cascade dynamics and proposed attention-based RNN to solve the problem;
- The alignments in proposed attention mechanism are homologous to the underlying structure of propagation.
- The proposed coverage in attention mechansim further improve the performance of attention mechanism, including cascade prediction and propagation structure inference.

We conduct extensive experiments on synthetic and real-world datasets. Compared with several widely-used methods in cascade dynamics modeling, our proposed models consistently outperform them at cascade prediction.

2 Model

The input data is a collection of cascades $\mathcal{C} = \{S_m\}_{m=1}^M$. A cascade $S = \{(t_k, u_k) | u_k \in U, t_k \in R^+ \text{ and } k = 1, \dots, N\}$ is a sequence of propagations ascendingly ordered by time, where U refers to user set in cascade. The k-th propagation is recorded as a tuple (t_k, u_k) referring to pair of activated time and user. Let the history $\mathcal{H}_k$ be the list of activated time and user pairs up to the k-th propagation. The objective of sequence modeling in cascade dynamics is to formulate the conditional probability probability of next propagation $p((t_k, u_k)|\mathcal{H}_k)$.

2.1 Background

Firstly, we introduce RNN in cascade dynamics modeling. RNN is a feed-forward neural network, which can be used to generate a cascade by N steps sequentially. At each step k, the k-th propagation vectorized in x_k as input is fed into hidden units of RNN by nonlinear transformation f, jointly with the outputs from the last hidden units, updating the hidden state $h_k = \mathit{RNN}(x_k, h_{k-1})$. The representation of hidden state h_k can be considered as embedding of the k-th propagation, 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 cascade,

$$p(S) = \prod_{k=1}^{N} p((t_{k+1}, u_{k+1})|\mathcal{H}_k) = \prod_{k=1}^{N} p((t_{k+1}, u_{k+1})|h_k).$$

The conditional probability $p((t_{k+1}, u_{k+1})|h_k)$ can be formalized by a joint probability on both next activated time and user, such as,

$$\begin{aligned} p((t_{k+1}, u_{k+1})|h_k) =& p(t_{k+1}|h_k) \cdot p(u_{k+1}|h_k) \\ =& f(t; h_k) \cdot \text{softmax}(g(h_k)), \end{aligned}$$

where g is a non-linear function. The function $f(t; h_k)$ refers to a temporal point process parameterized by h_k,

$$\begin{aligned} & f(t; h_k) = \lambda(t) \exp\left(-\int_{t_k}^{t} \lambda(\tau) d\tau\right) \\ & s.t.\ \ \lambda(t) = \exp(wt + U^{(t)} h_k), \end{aligned}$$

where w is a scalar and $U^{(t)}$ is a parameter matrix. Based on sufficient observed cascades, RNN can find an optimal solution for the above equation in a huge functional space, avoiding the bias on diffusion model and limits on diffusion network. Thus, RNN can be a promising and flexible method to capture the complex propagation patterns in cascade dynamics modeling.

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

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

where the weight $\alpha_{k,i}$ refers to the proportion of dependence between next propagation and the i-th propagation 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 obtain the tree structure, even given the social relationships. Next we propose attention mechanism to implement the pooling layer.

2.2 Attention Mechanism

Attention mechanism is orginally used in neural machine translation (NMT). 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. For modeling cascade dynamics, the source sequence will be the observed cascade. The each element in target sequence is the following propagation corresponded to propagation in source sequence. There remains one problem when applying attention mechanism in cascade dynamics modeling: the target is the next propagation in source sequence, thus the alignments between source and target sequence need to be dynamically updated when the context $\{h_1, \dots, h_k\}$ is growing along with the proceeding propagations.

We propose a dynamic attention mechanism for CYAN-RNN. The proposed architecture is shown in Fig. 2. According to the architecture, we rewrite the conditional probability

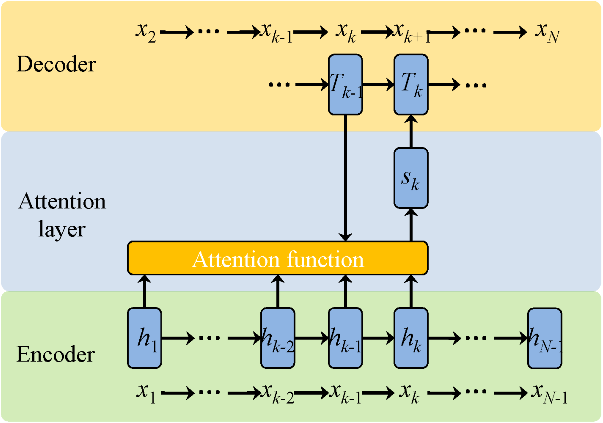
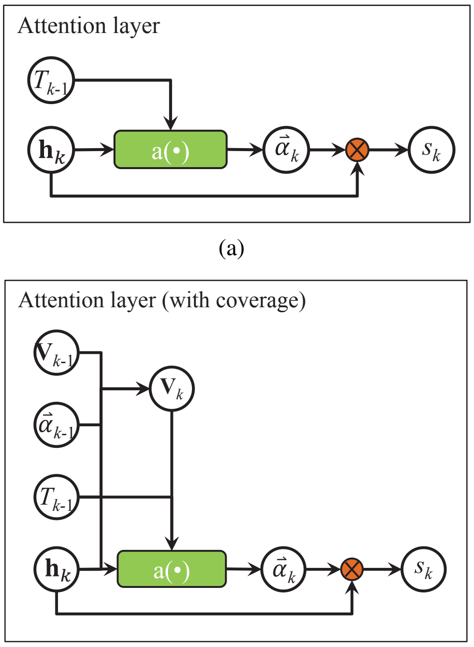

Decoder
$x_2 \rightarrow \cdots \rightarrow x_{k-1} \rightarrow x_k \rightarrow x_{k+1} \rightarrow \cdots \rightarrow x_N$
T_{k-1}
T_k
Attention layer
s_k
Attention function
Encoder
h_1
h_{k-2}
h_{k-1}
h_k
h_{N-1}
$x_1 \rightarrow \cdots \rightarrow x_{k-2} \rightarrow x_{k-1} \rightarrow x_k \rightarrow \cdots \rightarrow x_{N-1}$

Figure 2: The architecture of CYAN-RNN. The figure presents the case when modeling the generation of the $(k+1)$-th propagation. The sequence at bottom is the observed propagations and the sequence at top is the predictive propagations. The blue rectangles refer to representations of hidden units. The yellow rectangle is a general form of attention function $s_k = \mathit{AttentionFunc}(t_{k-1}, \{h_1, \ldots, h_k\})$.



(a)

(b)

Figure 3: Two kinds of implementation on attention layer. (a) The attention mechanism applied in CYAN-RNN; (b) The attention mechanism with coverage applied in CYAN-RNN (cov). Note that $\mathbf{h}_k = (h_1, \ldots, h_k)$ is matrix assembled by all historical propagation embeddings at step k, and $\mathbf{V}_k = (V_1, \ldots, V_k)$ is a coverage martix containing all coverage vectors at step k.

of next propagation

$$\begin{aligned} p((t_{k+1}, u_{k+1})|\mathcal{H}_k) &= p((t_{k+1}, u_{k+1})|x_k, T_k, s_k) \\ &= f(t; x_k, T_k, s_k) \cdot \text{softmax}(g(x_k, T_k, s_k)). \end{aligned}$$

The decoding state T_k for the k-th step in target sequence is computed by

$$T_k = f(x_k, T_{k-1}, s_k), \tag{2}$$

where f is a nonlinear activation function, which can be either a *tanh* or a *sigmoid* function. The context vector s_k is calculated by Eq. (1) where the weights $\alpha_{k,\cdot}$ is updated by the growth context $\{h_1, \ldots, h_k\}$ and T_{k-1}. The weight $\alpha_{k,i}$ is formalized as

$$\alpha_{k,i} = \frac{\exp(e_{k,i})}{\sum_{j=1}^{k} \exp(e_{k,j})}, \tag{3}$$

where

$$e_{k,i} = a(T_{k-1}, h_i) = v^T \tanh(W^{(a)} T_{k-1} + U^{(a)} h_i) \tag{4}$$

scores how well the dependence between the i-th propagation embedding and the output at the k-th step, and $W^{(a)}$ and $U^{(a)}$ are the parameter matrices. The implementation of attention mechanism in proposed model is briefly represented in Fig. 3(a).

With the attention mechanism, the alignments $\alpha_{k,\cdot}$ can be directly updated through the cost function, thus exploit an expected representation s_k over all historical propagation embeddings for each step k.

2.3 Coverage

Furthermore, we propose coverage to lead alignments focused more on recent propagation embeddings, following the similar perference of users' interests in real system. The example is shown in Fig. 1. If the user u_1 is an influential user and her propagation is key to the cascade, the propagation triggered by u_4 may depend more on the 1st propagation embedding triggered by u_1, achieving a larger alignment weight than the 2nd propagation embedding. But the u_4 is the immediate successor of u_2 in cascade tree instead of u_1 in practice. In this way, the 1st propagatoin embedding is *over-dependent* and the 2nd propagation embedding is *under-dependent* when modeling the generation of the 4th propagation.

The two problems are caused by memoryless of dynamic attention mechanism. Inspired by linguistic coverage model, we formulate the general form of coverage in cascade dynamics modeling, keeping historical alignments so as to release the over-dependent and under-dependent problems. The k-th step of coverage is defined as

$$V_{k,i} = f\left(V_{k-1,i}, \alpha_{k-1,i}, t_{k-1}, h_i\right). \tag{5}$$

Remarkably, as the increasing context and alignments, $V_{k,k}$ and $\alpha_{k,k}$ have no corresponding values in $V_{k-1,\cdot}$ and $\alpha_{k-1,\cdot}$. Instead we fill up with zero in our work. 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 propagation embeddings are in the past. The rewritten alignment calculation in Eq. (4) by coverage can be formalized [1] as

$$e_{k,i} = a(T_{k-1}, h_i, V_{k,i}) \\ = v^T \tanh(W^{(a)}T_{k-1} + U^{(a)}h_i + Z^{(a)}V_{k,i}), \quad (6)$$

where $W^{(a)}$, $U^{(a)}$ and $Z^{(a)}$ are parameter matrices. We expect that the alignment weights would be focus more on recent propagation embeddings. The expectation will be validated in section 4.

2.4 Length of Dependence

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, *length of dependence* l, limiting the size of alignments so that the predictive task can only depend on last l propagations. Empirically, we set $l = 200$ at most cases.

3 Optimization

In this section, we introduce the learning process of our proposed models. 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

$$\mathcal{L}(\mathcal{C}) = -\sum_{m=1}^{M}\sum_{k=1}^{N_m} \log p((t_{k+1}, u_{k+1})|x_k, T_k, s_k), \quad (7)$$

and we can learn parameters of the proposed model by minimizing the negative logarithmic likelihood $\arg\min_\theta \mathcal{L}(\mathcal{C})$, where θ is the parameter set in the model. We exploit backpropagation through time (BPTT) [Chauvin and Rumelhart, 1995] for training. In each training iteration, we vectorize the propagation as inputs, including user embedding and temporal features. The embedding matrix of users is learned along with the training process. The temporal features are formalized by 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 k-th inputs to h_k. We apply stochastic gradient descent (SGD) with mini-batch and the parameters are updated by Adam [Kingma and Adam, 2015]. For speeding up the convergence, we use orthogonal initialization method [Henaff *et al.*, 2016] in training process. We also employ early stopping method [Prechelt, 1998] and other techniques to prevent overfitting.

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 modeling methods on both synthetic and real data showing that CYAN-RNN can better model cascade dynamics and infer propagation structure.

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 modeling methods for either predicting next activated users or predicting next activated time for comparisons in two prediction tasks.

- **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.
- **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. The two models can be used to predict next activated users.
- **MC-1** Model: The markov chain model is a classic sequence modeling method, depicting the generation of activated users. Here we compare with markov chain with order one.
- **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

$$\lambda(t) = \lambda(0) + \alpha \sum_{t_i<t} \exp\left(-\frac{t - t_i}{\sigma}\right), \quad (8)$$

  where $\lambda(0)$ is a intrinsic rate when $t = 0$. We set $\sigma = 1$ in our experiments.

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 cascade 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 [Leskovec *et al.*, 2008] (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 in $[0.01, 10]$; 2) mixed rayleigh (Ray) distribution, controlled by scale parameters β 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

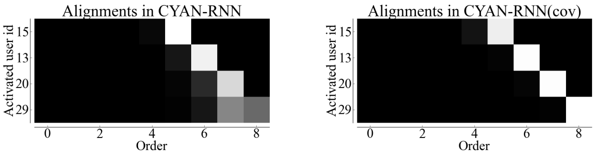
[1] If we use the last coverage $V_{k-1,\cdot}$ instead of $V_{k,\cdot}$ (like [Tu *et al.*, 2016]) to update $e_{k,\cdot}$ at each step k, we will lose certain coverage information and cause unbalanced calculation on k-th propagation embedding, proved by our preliminary experiments.

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, denoted by (CP, Exp), (CP, Ray), (Random, Exp) and (Random, Ray). 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]. The larger values in Acc@k and MRR indicate the better performance. On the prediction of next activated 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 are shown in Fig. 4(a)$\sim$ 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 propagation 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 propagation 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.

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 propagation 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 propagation embeddings, which is consistent to well-studied phenomonen 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.



(a)

(b)

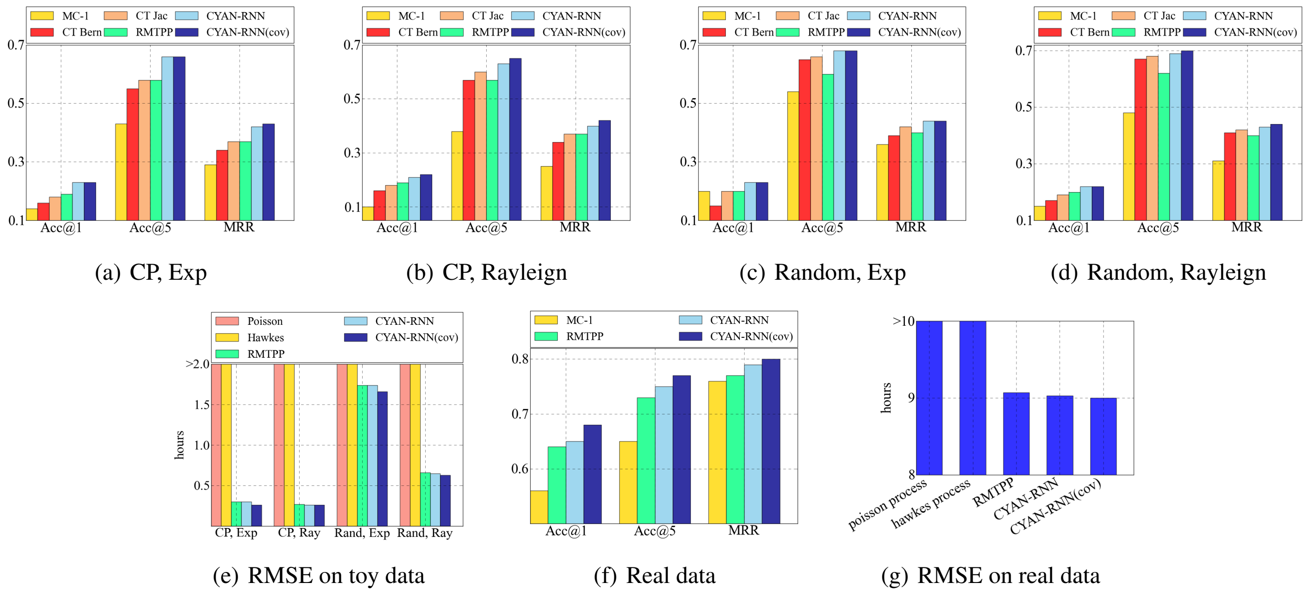
Figure 5: Sample alignments on a fragement of cascade. The y-axis is the users who will be activated next sequentiallly from top to bottom. The x-axis is the activated order related to next activated user in the cascade. Each pixel shows the weight $\alpha_{k,i}$ related to i-th propagation embedding 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).

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



(a) CP, Exp

(b) CP, Rayleign

(c) Random, Exp

(d) Random, Rayleign

(e) RMSE on toy data

(f) 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.

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.

[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.

[Kempe *et al.*, 2003] David Kempe, Jon Kleinberg, and Éva 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*, pages 137–146. ACM, 2003.

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

[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.

[Leskovec *et al.*, 2008] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 695–704, New York, NY, USA, 2008. 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.

[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.