Cognitive Computation (2020) 12:499–512
https://doi.org/10.1007/s12559-019-09690-8

Modeling Marked Temporal Point Process Using Multi-relation Structure RNN

Hongyun Cai[1] · Thanh Tung Nguyen[2] · Yan Li[3] · Vincent W. Zheng[4] · Binbin Chen[5] · Gao Cong[2] · Xiaoli Li[2]

Received: 30 March 2019 / Accepted: 10 October 2019 / Published online: 11 November 2019

Abstract

Event sequences with marker and timing information are available in a wide range of domains, from machine log in automatic train supervision systems to information cascades in social networks. Given the historical event sequences, predicting what event will happen next and when it will happen can benefit many useful applications, such as maintenance service schedule for mass rapid transit trains and product advertising in social networks. Temporal point process (TPP) is one effective solution to solve the next event prediction problem due to its capability of capturing the temporal dependence among events. The recent recurrent temporal point process (RTPP) methods exploited recurrent neural network (RNN) to get rid of the parametric form assumption in the density functions of TPP. However, most existing RTPP methods focus only on the temporal dependence among events. In this work, we design a novel multi-relation structure RNN model with a hierarchical attention mechanism to capture not only the conventional temporal dependencies but also the explicit multi-relation topology dependencies. We then propose an RTPP algorithm whose density function conditioned on the event sequence embedding learned from our RNN model for cognitively predict the next event marker and time. The experiments show that our proposed MRS-RMTPP outperforms the state-of-the-art baselines in terms of both event marker prediction and event time prediction on three real-world datasets. The capability of capturing both ontology relation structure and temporal structure in the event sequences is of great importance for the next event marker and time prediction.

Keywords Structure RNN · Recurrent temporal point process · Multi-layer attention

Introduction

Event sequence with marker and time information is becoming increasingly available in a broad range of domains, such as machine logs in automatic train supervision (ATS) systems [1], intelligent transport [2], financial time series [3], and information diffusion in social networks [4]. Each event sequence contains a list of information about the event marker (e.g., type, participator) ordered by event time (i.e., when the event occurs). Figure 1 shows an example event sequence, where (e_i, t_i) is an event marker–time pair. The event marker e_i has different meanings in different scenarios. For instance, in ATS system, e_i represents a train alarm/error (e.g., door signal not responding). But in social information cascades, e_i corresponds to a social user who diffuses a post. Given the event sequences in the past, one important problem is to predict *what event will happen next* and *when it will happen*. It can benefit a lot of applications in various domains. For example, predicting when a mass rapid transit (MRT) train will break down and what type of error causes the breakdown can facilitate the MRT maintenance service and help provide a more smooth travel experience to passengers. Also, predicting who will be the next user to diffuse a message and when the diffusion will happen can be useful for product advertising, news spreading, etc.

The next event marker and time prediction is not a trivial task. It is not easy to approximate the unknown model which governs the event occurrences in the historical event sequence. Temporal point process (TPP) [5] is one effective solution due to its capability of capturing the temporal dependencies among events. The early attempts usually made a hypothesis of the rules that govern the generative process of events. Based on their hypothesis, a conditional density function with a specific parametric form was designed to approximate the probability when the next event will happen. Representative studies include Poisson

✉ Vincent W. Zheng
vincentz@webank.com

Extended author information available on the last page of the article.

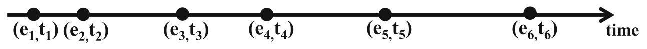


Fig. 1 An example of marked temporal event sequence

process [6], Hawkes process [7], self-correcting process [8], autoregressive conditional duration process [9], and so on.

Recently, some researchers observed that the performance of the above TPP methods highly depends on whether their hypothesis reflects the reality, which usually requires strong prior knowledge of the event sequences. To address this limitation, they start to explore recurrent neural network (RNN) [10] in their recurrent temporal point process (RTPP) models [11–13]. Specifically, they use an RNN model to embed the event sequence with both markers and time information, and then condition their density functions on the learned event sequence embedding. By utilizing RNN to learn the temporal dependence in event sequences, RTPP models get rid of specific assumptions about the parametric form. Moreover, they are non-Markovian processes assuming the probability of a future event not only depends on the present event, but also the prior events. Although effective, most RTPP methods focus on the sequential temporal dependence among events only. Specifically, the embedding of an event depends only on the event that happened just before it. As in Fig. 2a, h_i depends on its immediate predecessor h_{i-1} and the current status (e_i, t_i). There are two major limitations.

Firstly, they overlook the implicit cross-dependence in event sequences. The cross-dependence was first mentioned by CYAN-RNN [13]. Notice that an event may be triggered by its non-immediate predecessor in the event chain, CYAN-RNN proposes an attention mechanism to capture such cross-dependence in event sequences. The attention mechanisms are widely used in recent deep learning models to automatically capture elements' importance [14]. For example, in Fig. 2b, h_5 depends on the weighted aggregation of previous hidden states (h_1, h_2, h_3, and h_4). If e_5 is triggered mainly by e_3 (rather than e_4), the learned attention weight a_3 will be larger than a_4 to emphasize the impact from e_3. However, CYAN-RNN considers all the historical

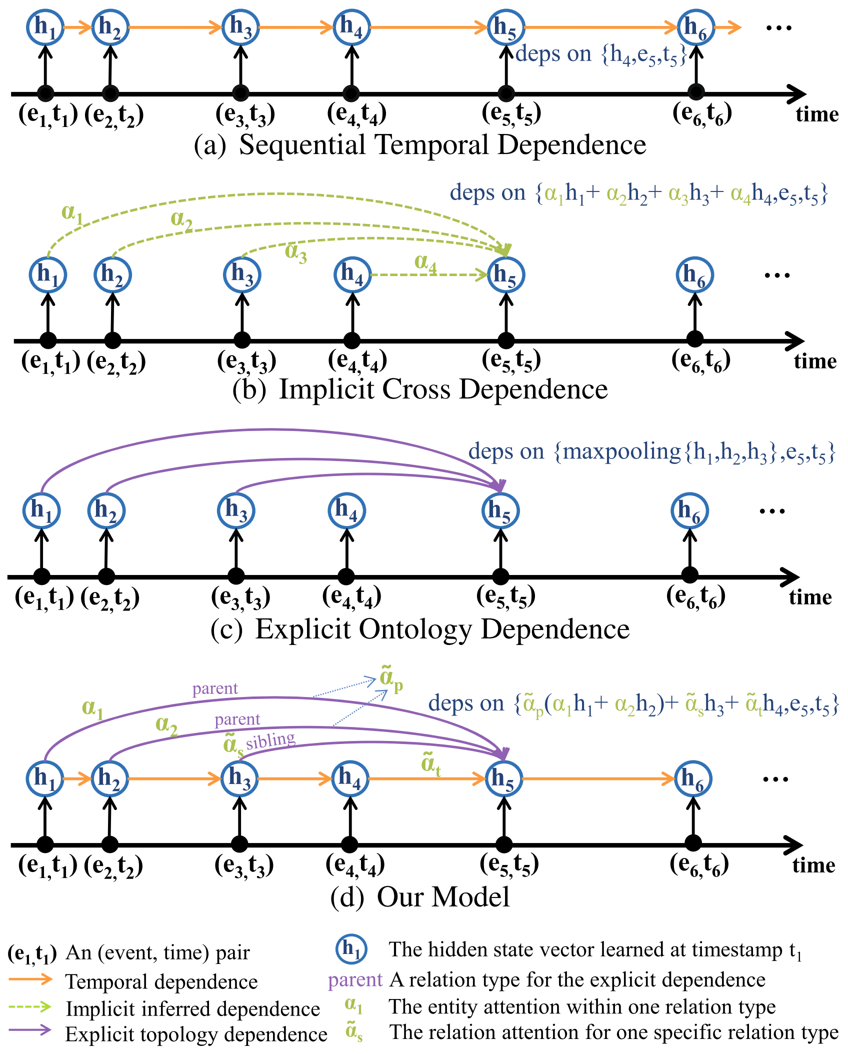

deps on {h4,e5,t5}
(a) Sequential Temporal Dependence
deps on {α1h1+ α2h2+ α3h3+ α4h4,e5,t5}
(b) Implicit Cross Dependence
deps on {maxpooling{h1,h2,h3},e5,t5}
(c) Explicit Ontology Dependence
parent
parent
sibling
deps on {α̃p(α1h1+ α2h2)+ α̃sh3+ α̃th4,e5,t5}
(d) Our Model
(e1,t1) An (event, time) pair
Temporal dependence
Implicit inferred dependence
Explicit topology dependence
The hidden state vector learned at timestamp t1
parent A relation type for the explicit dependence
The entity attention within one relation type
The relation attention for one specific relation type
time

Fig. 2 Illustrative example of modeling different dependencies in recurrent temporal point process

events when modeling each event. This may lead to high computation cost, especially for long sequences. Moreover, extensively modeling all the previous events may also introduce redundancy and noises. Ma et al. [15] proposed a two-step attention mechanism to capture both the target and sentence level attention for sentiment analysis; still, all words in the sentence are considered in their model.

Secondly, temporal dependence–based RTPP models neglect the possible explicit ontology dependencies in the event sequences. Sometimes, the explicit ontology dependencies may exist among the events. Take ATS log as an example, events (i.e., system alarms) naturally form a tree ontology. Figure 3 shows a toy example. These explicit dependencies not only provide additional domain knowledge about the event sequences, but also help remove the unnecessary implicit dependency modeling in an RTPP model like CYAN-RNN, e.g., "train broadcast error" should not be considered when modeling "point machine error" because their dependencies are very weak as indicated by the ontology dependency structure. In Fig. 2c, h_5 now depends on the aggregation of $\{h_1, h_2, h_3\}$ but not h_4 because e_4 and e_5 do not have direct explicit dependence. There exist some models (e.g., Topo-LSTM [4]) which utilize explicit dependence for event prediction. However, they do not employ a TPP framework and thus cannot predict the next event time. Moreover, they only model dependencies with one single relation type. But in reality, the relations can be of multiple types, e.g., the relations of event dependencies in ATS log can be parent (e.g., train broadcast in Fig. 3), or child (e.g., front sensor-train), sibling (front sensor-emergency brake). To the best of our knowledge, none of the existing RTPP models has studied the problem of modeling event sequence with explicit ontology dependencies, not to mention the settings with multi-relation dependencies.

In view of the limitations in existing studies, we design a novel multi-relation structure RNN with a hierarchical attention mechanism to embed the historical event sequence with explicit ontology dependencies. We then propose a multi-relation structure RNN–based recurrent marked temporal point process model (MRS-RMTPP) which conditions its density function on the learned event sequence embedding to predict the next event marker and time. As illustrated in Fig. 2d, to predict (e_7, t_7), we learn a hidden state at every time stamp. For each event, we leverage the embedding from all "relevant" events which happened before it. We consider both temporal dependence ($h_4 \rightarrow h_5$) and ontology dependencies with multiple types of relations (parent: $h_1 \rightarrow h_5, h_2 \rightarrow h_5$, sibling : $h_3 \rightarrow h_5$). Note that the impacts of parent events and sibling events may be different. And even for e_1 and e_2 that are both *parent* events, they may have different degrees of influence on the occurrence of e_5. Inspired by this, we design a hierarchical attention mechanism to automatically learn the impact of different relations (e.g., $\widetilde{\alpha}_p$, $\widetilde{\alpha}_s$, and $\widetilde{\alpha}_t$ for relation parent, sibling, and temporal predecessor in Fig. 2d), as well as the impacts of different events within one relation (e.g., α_1 and α_2 for relation parent in Fig. 2d). The advantages of our MRS-RMTPP are two-fold. Firstly, our model utilizes the explicit ontology dependencies to capture relations among historical events. It is more expressive compared with the temporal dependence–based RTPP and more efficient compared with the implicit dependence–based RTPP. Secondly, our model exploits different types of relations in the ontology dependencies. The proposed hierarchical attention mechanism can effectively capture impacts at different levels (i.e., from different relations and from different events within one relation).

We summarize our major contributions as follows.

- We design a novel multi-relation structure RNN, which embeds event sequences with an event marker, event time, and the explicit ontology dependencies among events.
- We propose a hierarchical attention mechanism to distinguish the impacts of the historical events within each relation and the impacts of different relations.
- We propose an MRS-RMTPP model with its density function conditioned on our designed multi-relation structure RNN for the next event marker and time prediction.
- Our evaluation results show that our model outperforms the contemporary baselines by 3.8 to 24.4% (event marker prediction) and 1.9 to 38.6% (event time prediction) on three real-world datasets.

Related Work

Before we introduce our method, we first review the representative studies in two relevant fields: structure RNN and temporal point process.

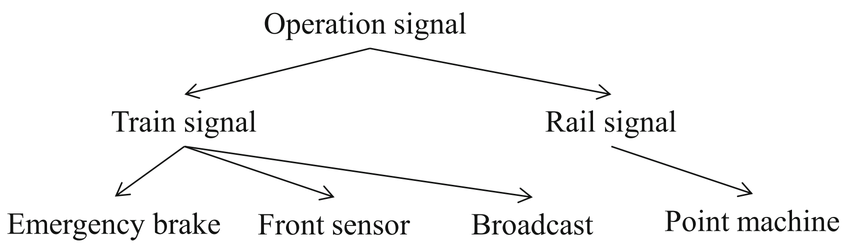


Fig. 3 An example of ontology dependency structure in ATS log

Structure RNN

Recurrent neural network (RNN) [10] processes variable-length sequences by having a recurrent hidden state whose activation at each time is dependent on that of the previous time [15]. Most RNN models (e.g., [16–19]) explore the linear chain dependence in the sequence, i.e., the hidden state at time t_i depends on that at t_{i-1} only. However, there may exist topology dependencies in the input sequence. For example, in a natural language sentence, words are naturally combined to phrases [20–22]. In an information diffusion cascade (a list of users who diffuse a specific content ordered by their diffusion time), the friendship relations among users form a network structure. Various structure RNN have been proposed to exploit such ontology dependencies in the sequences. The earliest attempt is Tree-LSTM proposed in [20]. Tree-LSTM takes a parse tree constructed from a natural language sentence as the input. Each LSTM unit conditions its components on the sum of child hidden states. The hidden state of the root node serves as the sentence embedding. Recent structure RNN models start to explore more complicated structures than a tree. For example, DAG-RNN [23] embeds a directed acyclic graph (DAG) structure for scene labeling. Topological RNN [4] embeds the social network structure in a cascade to predict future diffusion. ProxEmbed [24] learns the representations of the paths between two distant nodes in a heterogeneous graph to measure nodes' proximity. SPE [25] further improves ProxEmbed by using subgraphs to augment the paths between two nodes and then learns a subgraph-augmented path embedding for semantic user search.

Summary Most existing structure RNN models focus on one single relation type (e.g., spatial relationships in DAG-RNN, friendship in topological RNN). Although ProxEmbed and SPE work on heterogeneous graphs, their structure embedding is derived by a pipeline approach which first embeds each sequential path using RNN and then aggregates all paths' embedding as the structure embedding. In contrast to them, our RNN model directly embeds a multi-relation dependency structure. Moreover, none of the existing structure RNN models contains time information; hence, they cannot predict event time.

Temporal Point Process

Temporal point process (TPP) [26] is a principled framework for modeling the temporal event series data [27]. One important component of a TPP model is to design its conditional density function that captures the dynamics of event generation. There are usually the following two directions.

Conventional Temporal Point Process The conventional TPP models usually assume a particularly specified parametric form for their conditional intensity functions. For example, as the simplest point process, the Poisson process [6] assumes its intensity function to be independent of the history data. Both the homogeneous Poisson process ($\lambda(t) = \lambda_0 \geq 0$) and its time-varying generalization ($\lambda(t) = g(t) \geq 0$) share this assumption. Hawkes process [7], on the other hand, assumes that the occurrence of each historical event increases the intensity by a certain amount. Its intensity function is defined as $\lambda(t) = \gamma_0 + \alpha\sum_{t_i<t}\gamma(t, t_i)$, where $\gamma_0 \geq 0$ is a background intensity independent of the history and $\gamma(t, t_i)$ is a triggering kernel capturing temporal dependencies. In contrast to Hawkes process, the self-correcting process [8] assumes that the occurrence probability of new events decreases if an event has occurred recently. The intensity function is defined as $\lambda(t) = \exp(\mu t - \sum_{t_i<t}\alpha)$, where $\mu > 0$, $\alpha > 0$. That is, the intensity is decreased by multiplying a constant $e^{-\alpha} < 1$ every time a new event happens. Autoregressive conditional duration process [9] tries to capture the dependency between inter-event durations. Its intensity function is $\lambda(t) = \phi^{-1}_{N(t)}$, where $\phi_i = \gamma_0 + \sum_{j=0}^{m}\alpha_j d_{i-j}$ captures the influences from the most recent m duration, and $N(t)$ is the total number of events up to time t. One major limitation of the above methods is that a fixed parametric form has to be assumed in advance. However, such kind of hypothesis of the underlying event dynamics model is hard to specify or verify in practice. Thus, the expressive power of these models may be restricted.

Recurrent Temporal Point Process Recently, a few attempts have been made to design a more flexible model which can be automatically adapted to the data. The intuition is to use an RNN model to learn a more general and effective representation of the underlying dynamics from the event history automatically so as to get rid of the parametric assumption. The first RTPP model, the RMTPP [11], embeds both event time and marker using a standard RNN model. Although effective, RMTPP has two limitations. On one hand, RMTPP can only sample transient time series features when an event happens. Intensity RNN [12] then proposes to use a time series RNN to track the spontaneous background and an event sequence RNN to capture the long-range dependency with arbitrary time intervals. On the other hand, RMTPP models temporal dependencies only. With this observed, CYAN-RNN [13] proposes an attention-based RNN to capture the cross-dependence (implicit inferred dependencies) in the event sequence. Unlike the other RTPP models, Mei and Eisner [28] do not totally free the parametric assumption. They propose a novel continuous-time LSTM which updates its cell gate by a parametric form of a Hawkes process. In this way, they allow past events to influence the future in complex and realistic ways.

Summary Most existing RTPP work ([11, 12, 28]) models events sequentially. Although CYAN-RNN [13] uses an attention mechanism to capture the cross-dependence in event sequences, they still extensively consider all the previous events. Most of the existing RNN models with attention mechanism follow this setting. For example, Song et al. [29] use multi-head attention to capture the dependencies restricted within a neighborhood in the clinical time series. Ma et al. [30] introduces three attention mechanisms to measure the relationships of different visits for diagnosis prediction in healthcare. Considering all the previous nodes in the sequence may lead to a high computation cost for long sequences, as well as introduce some redundancies and noises in the event embedding. In this work, we consider the setting where an explicit ontology dependency structure is available. We design a novel structure RNN model which utilizes the explicit dependencies to remove unnecessary dependencies modeling. Moreover, our model distinguishes the impacts from different types of relations and within each relation, so as to capture the dynamics of event generation more accurately.

Methods

In this work, we study the problem of the next event marker and time prediction. We first introduce some key terminologies. Notations are listed in Table 1.

Table 1 Notations

Notation	Description
(e_i, t_i)	An event marker–time pair
S	A sequence of event marker–time pairs
$\mathcal{G}$	Ontology structure $\mathcal{G}$
$(\mathcal{E}, \mathcal{D}, \mathcal{R}, \tau)$	Events $\mathcal{E}$, dependencies $\mathcal{D}$, relations types $\mathcal{R}$, Dependency-relation type mapping function τ
$\mathcal{H}_{t_i}$	A sequence of event marker–time pairs up to time t_i
x_i	A dynamic feature extracted from event marker e_i
f_i	A time-dependent feature extracted from event time t_i
n, n'	The dimensions of x_i and f_i
h_i	The embedding of event sequence $\{(e_1, t_1), \cdots, (e_i, t_i)\}$
$\alpha^{(R)}$	The event attention for relation type R
$\widetilde{\alpha}$	The relation attention
d, d'	Embedding dimensions and attention dimensions

Problem Formulation

Definition 1 An event sequence is an ordered sequence of event marker–time pairs $\mathcal{S} = \{(e_1, t_1), \cdots, (e_N, t_N)\}$, where (e_i, t_i) denotes an event with marker e_i which happened at time t_i and $t_i \leq t_{i+1}$.

Definition 2 A multi-relation ontology dependency structure is a directed heterogeneous graph $\mathcal{G} = (\mathcal{E}, \mathcal{D}, \mathcal{R}, \tau)$, where $\mathcal{E}$ is the events (nodes) set, and $\mathcal{D}$ is the dependencies (edges) set. $\mathcal{R} = \{R_1, \cdots, R_K\}$ is a set of distinct relation types, and $\tau : \mathcal{D} \rightarrow \mathcal{R}$ is a relation type mapping function.

If $d_{i,j} \in \mathcal{D}$, there is a dependency from e_i to e_j. For example, in Fig. 2d, we have $\mathcal{D} = \{d_{1,5}, d_{2,5}, d_{3,5}\}$, $\mathcal{R} = \{\text{parent,sibling}\}$. For dependency $d_{1,5}$, $\tau(d_{1,5}) = \text{parent}$.

Problem 1 The input of our problem is a set of M training event sequences $\{\mathcal{S}_m\}_{m=1}^{M}$ and a multi-relation ontology dependencies structure $\mathcal{G}$. Our problem output is a trained model which is able to predict the next event marker-time pair (y_{i+1}, t_{i+1}) given a sequence of the past events $\{(e_1, t_1), \cdots, (e_i, t_i)\}$.

The event sequences and ontology dependency structure may have different meanings in different application scenarios. For example, in ATS, the event sequences correspond to the ATS log streams where each event is an error reported by the train system during its operation. The event marker represents the type of error. The components in the train system have different levels of granularity, and hence form a tree-structure ontology which explains their dependencies, e.g., the "train operation signal" component has sub-components such as "braking signal" and "engine signal." Thus, there is a parent dependency between event "train operation signal breakdown" and event "braking signal breakdown." In the social network information diffusion scenario, event sequences correspond to the cascades (i.e., sequences of re-sharing behaviors ascendingly ordered by time). In this case, each event is the action that a social user diffuses a piece of information and the ontology dependencies are the friendship relations among users. The event marker is the participator (i.e., social user) who performs the diffusion action.

Next, we will introduce how we design a model to predict the marker and time for the next event and how we conduct the end-to-end training.

Multi-relation Structure RNN

Denote the list of event marker–time pairs up to time t_i as $\mathcal{H}_{t_i} = \{(e_j, t_j) | t_j < t_i\}$. The standard TPP specifies a conditional density function $p(e_i, t_i | \mathcal{H}_{t_i})$ which describes the probability that the next event will happen at time t_i with marker e_i as follows:

$$\begin{aligned} p(e_i, t_i|\mathcal{H}_{t_i}) &= f(e_i|\mathcal{H}_{t_i}) f(t_i|\mathcal{H}_{t_i}) \\ &= f(e_i|\mathcal{H}_{t_i}) \lambda(t_i|\mathcal{H}_{t_i}) \exp\left(-\int_{t_{i-1}}^{t_i} \lambda(\tau|\mathcal{H}_{t_i}) d\tau\right) \end{aligned} \tag{1}$$

where $f(e_i|\mathcal{H}_{t_i})$ is a probability function for the next event marker, which follows a multinomial distribution. $f(t_i|\mathcal{H}_{t_i})$ is the conditional density function which denotes the likelihood that the next event will occur at time t_i given the timing sequence of the past events. $\lambda(t|\mathcal{H}_{t_i})$ is the conditional intensity function denoting the probability of the next event happening within a small window $[t_{i-1}, t)$. Note that event marker and event time are modeled independently to reduce the computation complexity.

One key component in TPP is to design the conditional density function $f(t|\mathcal{H}_t)$ (or the conditional intensity function $\lambda(t|\mathcal{H}_t)$). Different formulas have been designed as introduced in [27]. Motivated by previous recurrent temporal point process studies (e.g., [11, 12]), we use an RNN model to learn a general representation to approximate the unknown model that governs the event occurrences in the history sequences. The reason of adopting RNN is that RNN has a feedback mechanism which creates an internal state of the network to memorize the influence of each past event [11]. For example, in Fig. 2a, the hidden state at t_5 not only depends on current input (e_5, t_5), but also depends on the hidden state from the immediate temporal predecessor h_4. Similarly, h_4 depends on both (e_4, t_4) and h_3. Consequently, the hidden state at time t embeds the influence of the historical events up to time t. Recall that $\mathcal{H}_{t_i} = \{(e_j, t_j)|t_j < t_i\}$ denotes the list of event marker–time pairs up to the time t_i. It is easy to derive that the hidden state h_{i-1} in RNN corresponds to $\mathcal{H}_{t_i}$. Then, the conditional density function $f(t_i|\mathcal{H}_{t_i})$ in Eq. 1 can be naturally converted to

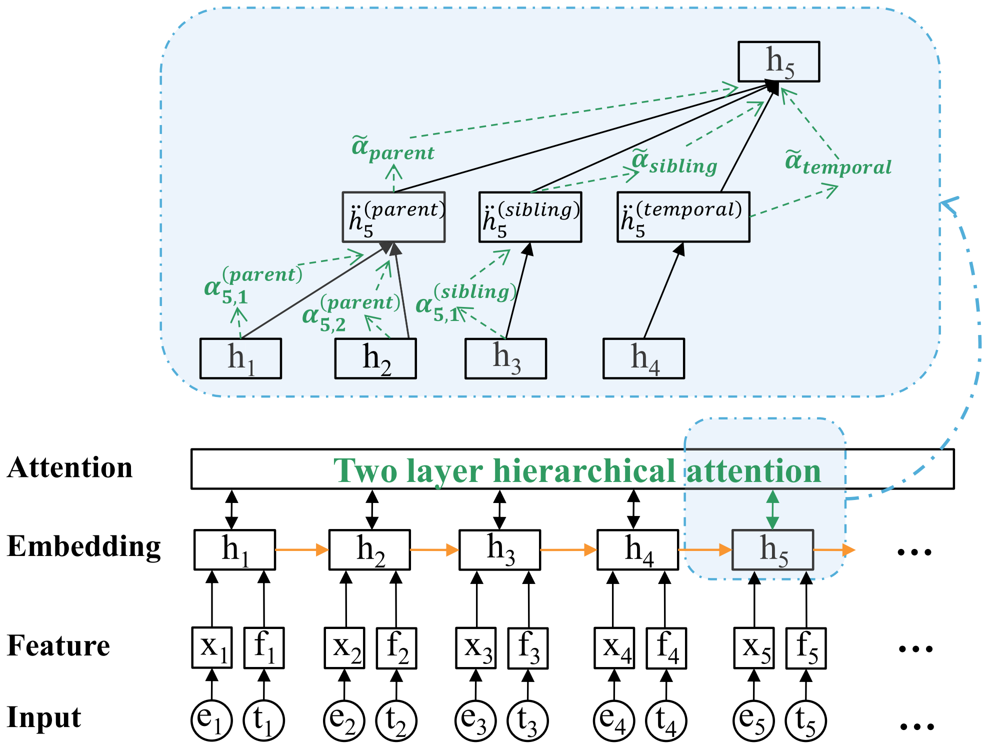
$$f(t_i|\mathcal{H}_{t_i}) = f(t_i|h_{i-1}) \tag{2}$$

By embedding the historical event marker and time information as a vector h_{i-1}, we now are able to design a more general conditional intensity function $\lambda(t)$ without having to specify a parametric form for the dependency structure over the history. Our objective is then to design an RNN model which is capable of modeling the event sequences with both the temporal sequential dependencies (e.g., $h_4 \rightarrow h_5$ in Fig. 2d) and the multi-relation ontology dependencies ($h_1 \rightarrow h_5, h_2 \rightarrow h_5$ and $h_3 \rightarrow h_5$ in Fig. 2d).

Our intuition is that the probability of an event happening increases if any of its related events have been observed recently. The event relations are reflected in an ontology dependency structure. Take ATS log and Fig. 3 as an example, after a "front sensor signal error" event happened, it is more likely to observe its sibling event "emergency brake signal error" rather than an irrelevant event such as "point machine signal error." Moreover, different relations may have different impacts on the next event. For instance, for an observed event, its parent event has a higher probability to be observed soon than its sibling event, e.g., a "sensor" error does not necessarily lead to a "broadcast" error. In an event for the same relation (e.g., sibling), different events may have different impacts: "emergency brake signal error" is more triggered by "front sensor signal error" than "broadcast error."

To properly model the multi-relation ontology in event sequences, we propose a structure GRU model with a two-layer attention mechanism. Our structure GRU is able to (1) explicitly model the ontology dependencies in event sequences for event sequence embedding; (2) distinguish the impacts from different types of relations in the dependencies; (3) differentiate the influences from different events within each type of relation. Specifically, we design a two-layer attention mechanism to leverage the inferences from different relations more accurately. The first attention layer is the event attention, which is constructed above the hidden units within one type of relation so as to differentiate the effects from different predecessors of the same relation type. The second attention layer is the relation attention, which is constructed above the aggregated hidden state of different relations (i.e., the outputs of the first layer attention) to differentiate the contributions of each relation type.

Figure 4 shows an illustrative example of how our proposed RNN model embeds the event sequence in Fig. 2d. Specifically, for each event marker-time pair (e_i, t_i), we learn a vector representation using a feature extraction layer as $x_i = W^{(e)} e_i + b^{(e)}$, where e_i is a one-hot vector representation of the event marker. In addition to this dynamic embedding learned from our model, we also extract a 134-d feature f_i based on the timestamp of the event (t_i). f_i is a concatenate vector that indicates the information of "month of the year," "day of the month," "day of the week," "hour of the day," and "minute of the hour." Both x_i and f_i are inputs into our multi-relation structure RNN. Suppose $R \in \mathcal{R}$ is a single relation type as indicated by the ontology structure $\mathcal{G}$, we use $R(i)$ to denote the set of events who have relation with e_i and happen before t_i. That is, $R(i) = \{e_j|\tau(d_{i,j}) = R, t_j < t_i\}$. For example, in Fig. 2d, we have parent(e_5)=$\{e_1, e_2\}$, sibling(e_5)=$\{e_3\}$. In reality, for ATS log, $\mathcal{R} = \{$parent, child, sibling, self$\}$. For social network cascades, $\mathcal{R} = \{$friend, non-friend$\}$. Note that in addition to the explicit ontology dependence indicated by the ontology structure $\mathcal{G}$, we also consider the temporal dependence. For example, in Fig. 2d, we have temporal(e_5) = e_4.



h_5
$\tilde{\alpha}_{parent}$
$\tilde{\alpha}_{sibling}$
$\tilde{\alpha}_{temporal}$
$\ddot{h}_5^{(parent)}$
$\ddot{h}_5^{(sibling)}$
$\ddot{h}_5^{(temporal)}$
$\alpha_{5,1}^{(parent)}$
$\alpha_{5,2}^{(parent)}$
$\alpha_{5,1}^{(sibling)}$
h_1 h_2 h_3 h_4
Attention
Two layer hierarchical attention
Embedding
h_1 h_2 h_3 h_4 h_5 …
Feature
x_1 f_1 x_2 f_2 x_3 f_3 x_4 f_4 x_5 f_5 …
Input
e_1 t_1 e_2 t_2 e_3 t_3 e_4 t_4 e_5 t_5 …

Fig. 4 Multi-relation structure GRU

Then for each single relation type $R \in \mathcal{R} \cup \{\text{temporal}\}$, we learn an update gate $z_i^{(R)}$, reset gate $r_i^{(R)}$ and candidate activation $\hat{h}_i^{(R)}$ as follows:

$$z_i^{(R)} = \sigma(W_z^{(x)} x_i + W_z^{(f)} f_i + U_z^{(R)} \ddot{h}_i^{(R)} + b_z) \quad (3)$$

$$r_i^{(R)} = \sigma(W_r^{(x)} x_i + W_r^{(f)} f_i + U_r^{(R)} \ddot{h}_i^{(R)} + b_r) \quad (4)$$

$$\hat{h}_i^{(R)} = \tanh(W_h^{(x)} x_i + W_h^{(f)} f_i + U_h^{(R)} (r_i^{(R)} \circ \ddot{h}_i^{(R)}) + b_h) \quad (5)$$

where $\ddot{h}_i^{(R)} = \sum_{e_j \in R(i)} \alpha^{(R)} h_j$ is the aggregated hidden state (i.e., embedding) of e_is predecessors who has relation R with e_i. The embedding vectors of predecessors in $R(i)$ are aggregated using the first layer attention weight $\alpha^{(R)}$. σ is a sigmoid function, and tanh is a hyperbolic tangent function. $\circ$ denotes the Hadamard product. $W^{(x)} \in \mathbb{R}^{d \times n}$, $W^{(f)} \in \mathbb{R}^{d \times n'}$, $U^{(R)} \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$ are the model parameters.

The attention vector $\alpha^{(R)}$ derived for each single type of relation R is the event attention. It is used to distinguish the influences from different events of the same relation type, where each dimension is the attention weight for a specific event in $R(i)$. It is easy to derive that, for every single event $e_j \in R(i)$, the dimensionality of $\alpha_j^{(R)}$ equals to $|R(i)|$. Particularly, $\alpha_j^{(R)}$ is learnt as follows:

$$\beta_j^{(R)} = \eta^{(R)} \cdot \tanh(Q^{(R)} h_j + b^{(R)}) \quad (6)$$

$$\alpha_j^{(R)} = \frac{exp(\beta_j^{(R)})}{\sum_{v_l \in R(i)} exp(\beta_l^{(R)})} \quad (7)$$

where $\eta^{(R)} \in \mathbb{R}^{d'}$, $Q^{(R)} \in \mathbb{R}^{d' \times d}$ and $b^{(R)} \in \mathbb{R}^{d'}$ are parameters.

The embedding for the event sequence $\{(e_1, t_1), \cdots, (e_i, t_i)\}$ is calculated as

$$h_i = \sum_{R \in \mathcal{R} \cup \{\text{temporal}\}} \tilde{\alpha}_R \left\{ \{(1 - z_i^{(R)}) \circ \ddot{h}_i^{(R)}\} + z_i^{(R)} \circ \hat{h}_i^{(R)} \right\} \quad (8)$$

where $\tilde{\alpha}_R$ is the second layer attention vector used to aggregate embedding from different types of relations.

Specifically, the relation attention $\tilde{\alpha}$ differentiates the contributions from different types of relations. Each dimension of $\tilde{\alpha}$ is the attention weight for a specific relation type $R \in \mathcal{R} \cup \{\text{temporal}\}$, and it is learned as

$$\tilde{\beta}_R = \tilde{\eta} \cdot \tanh(\tilde{Q} \left\{ \{(1 - z_i^{(R)}) \circ \ddot{h}_i^{(R)}\} + z_i^{(R)} \circ \hat{h}_i^{(R)} \right\} + \tilde{b}) \quad (9)$$

$$\tilde{\alpha}_R = \frac{exp(\tilde{\beta}_R)}{\sum_{R \in \mathcal{R}} exp(\tilde{\beta}_l)} \quad (10)$$

where the dimensionality of $\tilde{\alpha}$ equals to $|\mathcal{R}|+1$. $\tilde{\eta} \in \mathbb{R}^{|\mathcal{R}|+1}$, $\tilde{Q} \in \mathbb{R}^{(|\mathcal{R}|+1) \times d}$ and $\tilde{b} \in \mathbb{R}^{|\mathcal{R}|+1}$ are parameters.

Take Fig. 4 as an example, at timestamp t_5, the first layer attention (event attention) vectors includes 2-d $\alpha_5^{(\text{parent})}$ and a 1-d $\alpha_5^{(\text{sibling})}$. $\alpha_{5,1}^{(\text{parent})} + \alpha_{5,2}^{(\text{parent})} = 1$ and they are used to distinguish the different impacts of e_1 and e_2, which are both the parents of current event e_5. Similarly, $\alpha_5^{(\text{sibling})}$ learns the impact of e_3 being e_5's sibling and $\alpha_5^{(\text{temporal})}$ learns the impact of e_5's temporal predecessor event e_4. As there is just one sibling event and one temporal predecessor in the example sequence, $\alpha_{5,1}^{(\text{sibling})} = \alpha_{5,1}^{(\text{temporal})} = 1$ in Fig. 4. The second layer attention (relation attention) is a 3-d vector $\widetilde{\alpha}$. The three dimensions learn the impacts of three relation types, including parent event (e_1, e_2), sibling event (e_3) and temporal predecessor event (e_4). Consequently, the hidden state at t_5 is learned as

$$\begin{aligned} h_5 = &\widetilde{\alpha}_{\text{par}} \left\{ \{(1 - z_5^{(\text{par})}) \circ \ddot{h}_5^{(\text{par})}\} + z_5^{(\text{par})} \circ \hat{h}_5^{(\text{par})} \right\} \\ &+\widetilde{\alpha}_{\text{sib}} \left\{ \{(1 - z_5^{(\text{sib})}) \circ \ddot{h}_5^{(\text{sib})}\} + z_5^{(\text{sib})} \circ \hat{h}_5^{(\text{sib})} \right\} \\ &+\widetilde{\alpha}_{\text{tem}} \left\{ \{(1 - z_5^{(\text{tem})}) \circ \ddot{h}_5^{(\text{tem})}\} + z_5^{(\text{tem})} \circ \hat{h}_5^{(\text{tem})} \right\} \end{aligned}$$

where $\ddot{h}_5^{(\text{par})} = \alpha_{5,1}^{(\text{par})} h_1 + \alpha_{5,2}^{(\text{par})} h_2$, $\ddot{h}_5^{(\text{sib})} = \alpha_{5,1}^{(\text{sib})} h_3$ and $\ddot{h}_5^{(\text{tem})} = \alpha_{5,1}^{(\text{tem})} h_4$.

Note that sometimes multiple events may occur at the same timestamp in real-world scenarios. To deal with the possible co-occurrence of multiple events at one timestamp, we take the following two measures. (1) We process each event individually and only consider the related events happen before the current timestamp, i.e., at time t_i, we only consider $\{e_j | \tau(d_{i,j}) \in \mathcal{R}), t_j < t_i\}$; 2) We use a temporal attention $\alpha^{(\text{temporal})}$ to capture the impacts of different temporal predecessor events that happen one timestamp before.

In the end, the hidden state h_i learned at time t_i (8) is the embedding of the historical event sequence until event marker-time pair (e_i, t_i). The learned embedding can then be fit into an RTPP model to predict the marker and time of the next event (i.e., (e_{i+1}, t_{i+1}). Next, we introduce the procedure of how to fit the event embedding into an RTPP model for the next event marker and time prediction.

MRS-RMTPP

With the above multi-relation structure RNN model ready, we now present our multi-relation structure RNN–based recurrent marked temporal point process model (MRS-RMTPP) for the next event marker and time prediction.

As shown in Eq. 1, to model the probability of observing an event marker-time pair (e_i, t_i), we need to calculate the probability of observing both a specific event marker $f(e_i)$ and the conditional density function $f(t_i|\mathcal{H}_{t_i})$. Both probabilities can be derived based on the embedding learned from our structure RNN model.

For event marker modeling, we derive the multinomial distribution of the next event makers by the below softmax function:

$$f(e_i = k | \{h_j | j \in R(i)\}) = \frac{\exp\left(\sum_{R\in\mathcal{R}} V_k^{e(R)} \ddot{h}_i^{(R)} + b_k^y\right)}{\sum_{k=1}^{K} \exp\left(\sum_{R\in\mathcal{R}} V_k^{e(R)} \ddot{h}_i^{(R)} + b_k^y\right)} \tag{11}$$

where $R \in \mathcal{R} \cup \{\text{temporal}\}$. $\ddot{h}_i^{(R)} = \sum_{v_j \in R(i)} \alpha^{(R)} h_j$ is the aggregated hidden state of e_i's predecessors who has relation R with e_i (same as in Eqs. 3–5). K is the number of event markers. $V^{e(R)} \in \mathbb{R}^{K\times d}$ is the maker related parameter matrix for relation R and $V_k^{e(R)}$ is the kth row of $V^{e(R)}$.

For event time modeling, we adopt the Gaussian function as in [31]. The conditional density function of our MRS-RMTPP model is calculated as

$$\begin{aligned} f(t|\mathcal{H}_{t_i}) &= f(t|\{h_j | j \in R(i)\}) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(t - \sum_{R\in\mathcal{R}} V^{t(R)} \ddot{h}_i^{(R)} + b^t)^2}{2\sigma^2}\right) \end{aligned} \tag{12}$$

where $\ddot{h}_i^{(R)} = \sum_{v_j \in R(i)} \alpha^{(R)} h_j$. $V^{t(R)} \in \mathbb{R}^{1\times d}$ is the time parameter matrix for relation R. $b^t \in \mathbb{R}^1$ represents a background density score. Note that we consider different types of dependencies in addition to the temporal predecessor event. Hence, instead of conditioning the intensity function on the single temporal predecessor event h_{i-1} as in existing RTPP work (e.g., [11, 12, 28]), our intensity function is conditioned on a set of all related predecessor events $\{h_j | j \in R(i)\}$.

The prediction of the next event time is the expectation of Eq. 12 :

$$t_i^* = \sum_{R\in\mathcal{R}} V^{t(R)} \ddot{h}_i^{(R)} + b^t \tag{13}$$

The above function leverages two influence factors to compute the occurrence probability for next event, including the past historical events and a background density score. In particular, the first term $\sum_{R\in\mathcal{R}} V^{t(R)} \ddot{h}_i^{(R)}$ embeds the accumulative influence from the past related events with the consideration of both their marker and time information. The second term provides a background density value for the next event appearing.

Note that another way for time modeling is to calculate the density function as in [11]:

$$\begin{aligned} f(t|\mathcal{H}_{t_i}) &= \lambda(t|\mathcal{H}_{t_i}) \exp\left(-\int_{t_{i-1}}^{t} \lambda(\tau|\mathcal{H}_{t_i}) d\tau\right) \\ &= \exp\left\{ \sum_{R\in\mathcal{R}} V^{(R)} \ddot{h}_i^{(R)} + w^t(t - t_{i-1}) + b^t \right. \\ &\quad + \frac{1}{w^t} \exp\left(\sum_{R\in\mathcal{R}} V^{(R)} \ddot{h}_i^{(R)} + b^t\right) \\ &\quad \left. - \frac{1}{w^t} \exp\left(\sum_{R\in\mathcal{R}} V^{(R)} \ddot{h}_i^{(R)} + w^t(t - t_{i-1}) + b^t\right) \right\} \end{aligned} \tag{14}$$

The prediction of the next event time is then estimated as

$$t_i^* = \int_{t_{i-1}}^{\infty} t \cdot f(t|\mathcal{H}_{t_i})dt \tag{15}$$

However, the experiment results ("Results on Next Event Time Prediction") show that adopting Gaussian function as the density function (12) outperforms the method in Eq. 14. One possible reason could be that the expectation of a Gaussian function is easy to derive (as in Eq. 13), while Eq. 15 does not have an analytic solution. In [11], the authors calculate Eq. 15 by applying the numerical integration techniques [32] for one-dimensional functions. However, this may affect the accurate of the time prediction.

End-to-End Learning

Given a set of M training event sequences $\{\mathcal{S}_m\}_{m=1}^M$ where each sequence $\mathcal{S} = \{(e_1, t_1), \cdots, (e_N, t_N)\}$ is an ordered list of event marker-time pairs, our model is trained to maximize the below joint log-likelihood of observing $\{\mathcal{S}_m\}_{m=1}^M$.

$$\begin{aligned}\ell(\{\mathcal{S}_m\}_{m=1}^M) &= \log\left(\prod_m f(\{(e_i, t_i)\}_{i=1}^N)\right)\\ &= \log\left(\prod_m \prod_i f(e_i|\mathcal{H}_{t_i}) f(t_i|\mathcal{H}_{t_i})\right)\\ &= \sum_m \sum_i \big(\log f(e_i|\{h_j|j \in R(i)\})\\ &\quad + \log f(t_i|\{h_j|j \in R(i)\})\big)\end{aligned} \tag{16}$$

where $f(e_i|\{h_j|j \in R(i)\})$ and $f(t_i|\{h_j|j \in R(i)\})$ are formulated in Eqs. 11 and 12 respectively.

The model parameters are learned by maximizing the above log-likelihood. We adopt back-propagation through time (BPTT) [33] for training. At every iteration, for every event marker-time pair (e_i, t_i), we first learn an event marker embedding as $x_i = W^{(e)}e_i + b^{(e)}$, which together with a 134-d time feature f_i extracted based on t_i, are input into our multi-relation structure GRU model. An event sequence embedding h_i is then generated at each timestamp, which embeds the influence of the historical events up to time t_i. The learned h_i is further used to model both event marker modeling (11) and event time modeling (12) in our MRS-RMTPP model. We apply stochastic gradient descent (SGD) with mini-batch and the model parameters are updated by Adam [34].

Results

In this section, we compare our proposed MRS-RMTPP with the state-of-the-art baselines on two public real-world social network datasets and one real-world ATS log dataset. We design the evaluation experiments to (1) verify the effectiveness of our proposed model in terms of both event marker prediction and event time prediction, (2) validate the importance of modeling ontology dependencies and implicit cross dependencies in the event embedding, (3) verify the effectiveness of our proposed multi-layer attention schema for our multi-relation structure RNN.

Dataset

We evaluate our model on the following three datasets. The statistics of the datasets are listed in Table 2.

- Twitter [36] contains the diffusion of URLs on Twitter in 2010, as well as the follower relationships among users. An event sequence is a diffusion cascade path for a URL, where events are the users who diffuse the URL ordered by time. Note that we did not use other tweet features, e.g., text content, image, etc. The objective is to predict who is the next user that will diffuse a URL and when it will happen. The explicit ontology dependency is given by the follower relationships among users, and there are three types of relations: follower, non-follower and temporal immediate predecessor.
- Memes [37] contain the diffusion of memes in April 2009 over online news websites. Similar to Twitter, an event sequence is a diffusion cascade path. We follow the settings in [4] to generate a link between two websites if one website appears earlier than the other in any cascade, and this serves as the explicit ontology dependency. There are also three types of relations: linked predecessor, non-linked predecessor and temporal immediate predecessor.
- SMRT is a set of ATS log data collected from the SMRT corporation, which operates several Mass Rapid Transit systems in Singapore. The dataset contains the ATS log for 64 circle line trains from August 1st to September 15, 2016. We form the event logs for one train in one day as an event sequence, and remove the sequence with length equals one. Each event is an alarm/error reported by the system during the train operation. The ontology

Table 2 Statistics of datasets

	Twitter	Memes	SMRT
# event markers	944	3871	77
# relations	830	312,7237	2147
# rvent sequences	3335	161,544	15,812
# timestamps	3859	188,364	11,068
Avg. sequence length	8.91	8.35	17.7
# relation types	3	3	4

Table 3 Comparison of all methods

Methods	Temp. depend.	Onto. depend.	Implicit cross depend.	Multi-relation types	Time modeling
RMTPP [35]	•				Eq. 14
Intensity RNN [31]	•				Eq. 12
CYAN-RNN [13]	•		•		Eq. 14
Topo-LSTM [4]	•	•			
Ours	•	•	•	•	Eq. 12

structure is provided by SMRT, which indicates a tree-like dependencies structure as illustrated in Fig. 3. There are four types of relations: parent, sibling, self and temporal immediate predecessor.

For all datasets, we sort all event sequences by the time they happened, and then select the first 80% of event sequences for training and last 20% of event sequences for testing. This is to simulate the online processing in the real world, when predictions are conducted based on the history data observed so far. We further split the training sequences and use the last 20% of the training sequences for validation. For each validation and test event sequence $\{(e_1, t_1), \cdots, (e_N, t_N)\}$, the predictions are made for every (e_i, t_i) given $\{(e_1, t_1), \cdots, (e_{i-1}, t_{i-1})\}$ where $i = 2, \cdots, N$.

Baseline

We compared with the following four state-of-the-art baselines. They are either the latest recurrent temporal point process models: RMTPP, intensity RNN, CYAN-RNN; or the representative RNN model that considers the explicit ontology dependence: Topo-LSTM. We compare the factors considered in all methods in Table 3.

- RMTPP [35] uses a recurrent neural network to model the nonlinear dependency over both of the markers and the timings from the historical events. Only temporal dependency among events (i.e., the temporal immediate predecessor of each event) is considered.
- Intensity RNN [31] separately models the background time series pattern and the long-range dependency in the historical events using two RNNs. Unlike RMTPP which only sample transient time series features when an event happens, intensity RNN captures both the regularly updated features and the long-range dynamics in asynchronous events. However, it also considers the temporal dependency only.
- CYAN-RNN [13] adds an attention layer on top of their RNN model to capture the cross-dependence in cascade. But, they do not consider the explicit ontology dependency.
- Topo-LSTM [4] uses diffusion topologies (an explicit ontology dependence) to describe the cascade structure and then models it as a dynamic DAG using their proposed topological LSTM model. They do not model the implicit cross-dependence, nor do they model the event timing. Hence, they cannot predict the timestamp for the next event.

Parameters and Environment

Follow the settings in [4], we set the learning rate as 0.001 and use Adam [34] to update model parameters. We set the batch size as 256 and the number of epochs to a large value and apply an early stopping mechanism which is triggered when the model performance evaluated on the validation data set is not further improved.

For all embedding dimensions, we tune the values in the range of {32, 64, 128, 256, 512} using the validation data set and choose the best parameters for all methods. Specifically, the embedding dimension is $d = 256$ for Twitter and memes and $d = 64$ for SMRT. The event marker feature dimension is $n = 256$ for Twitter and memes and $n = 64$ for SMRT. The time-dependent feature dimension is $n' = 134$ (a concatenate vector indicates the information of "month of the year," "day of the month," "day of the week," "hour of the day," and "minute of the hour" as in [35]) for all the three datasets. The attention dimension $d' = 128$ for Twitter and memes and $d' = 32$ for SMRT. The dimension values are usually larger in Twitter and memes because the number of event markers in these two datasets is much larger than that in SMRT (Table 2). We omit the parameters tuning result figures due to the space limitation.

We run all experiments on a Linux server (Ubuntu 14.04.5 LTS with Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz*32, 128G memory, and NVIDIA Corporation GK210GL [Tesla K80]). And we use Theano [38] and python for the model implementation. Our code is available online.[1]

Evaluation Metrics

We evaluate the performance in terms of both event marker prediction and event time prediction. For the

[1]https://github.com/LaineyCai/MRSRMTPP

evaluation of event marker prediction, we adopt mean average precision (MAP)@**K**. At each timestamp t_i, where $i = 1, \cdots, N-1$, we predict the next event (e_{i+1}, t_{i+1}) given $\{(e_1, t_1), \cdots, (e_i, t_i)\}$. MAP is then calculated as the mean of the average event marker prediction precision value of each i for all test event sequences. Specifically,

$$\text{MAP}@K = \frac{(\sum_{q=1}^{Q} \text{Ave}P@K(q))}{Q} \tag{17}$$

where Q is the total number of predictions made in the test. AveP@K is defined as $\text{Ave}P@K = \frac{\sum_{i=1}^{K}\text{Precision}(i)\times\text{rel}(i)}{\text{numberofevents}}$, where Precision$(i)$ is the precision at the top i events in the prediction. rel(i) equals to one if the ith event actually happens in the next timestamp, zero otherwise. Depends on the number of event markers in each dataset, we set $K = \{20, 40, 60, 80, 100\}$ in Twitter and memes and $K = \{2, 4, 6, 8, 10\}$ in SMRT. The larger MAP is, the better.

In terms of the event time prediction evaluation, we use root mean square error (RMSE) [39] between the estimated time t_i^* and the ground-truth t_i. Formally,

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{Q}(t_i^* - t_i)^2}{Q}} \tag{18}$$

The smaller RMSE is, the better.

Results on Next Event Marker Prediction

The comparison of our model and the baselines in terms of next event marker prediction are illustrated in Fig. 5. From the figure, we can see that, because we comprehensively model different factors in event sequences, our proposed model consistently outperforms the baselines across all the three datasets. Specifically, on average, our model improves the MAP of the baselines by 12.8% to 17.2% relatively in Twitter, 7% to 23% relatively in memes, and 4.1% to 20.4% relatively in SMRT. The performance of RMTPP is not satisfying, as it adopts the standard RNN for event embedding, which does not model the explicit ontology dependencies nor does it model the implicit cross dependencies. Intensity RNN considers both the background time pattern and the long-range dependencies, hence the MAP is improved by 5.4% m and 11.8% (SMRT) on average.

- Validation of modeling implicit cross dependencies and ontology dependencies: As shown in Fig. 5, compared to intensity RNN, both CYAN-RNN and Topo-LSTM derive a higher MAP, but due to different factors. CYAN-RNN adds an attention layer on top of the RNN model to capture the implicit cross-dependence in the event sequences, which improves MAP of intensity RNN by 0.9% to 9.2%. In contrast, Topo-LSTM explicitly models the ontology dependence among social network users for information cascade embedding, deriving a 1.8% to 8.5% higher MAP. This shows that both ontology dependence and implicit cross dependency are important factors when modeling the event sequences. And as shown in Fig. 5, the implicit cross dependency is more important in Twitter and memes (CYAN-RNN outperforms Topo-LSTM by 0.9%), while the ontology dependence is more important in SMRT (Topo-LSTM beats CYAN-RNN by 2.9%). The reason could be that the ontology structure in the SMRT dataset is a more refined knowledge of the relations among different components in the system, which is strong auxiliary information to guide the modeling of system alarms. On the contrary, the social user following structure contains a lot of noises. E.g., there are many artificial followers in social networks. In such settings, the automatically learned dependencies (i.e., implicit cross dependencies) are more flexible and adaptive in event sequence modeling. Finally, by comprehensively considering both ontology dependencies and the implicit cross dependencies, our model outperforms CYAN-RNN by 5.8% (SMRT) to 13% (Twitter) and Topo-LSTM by 3.8% (SMRT) to 14.3% (Twitter) relatively.

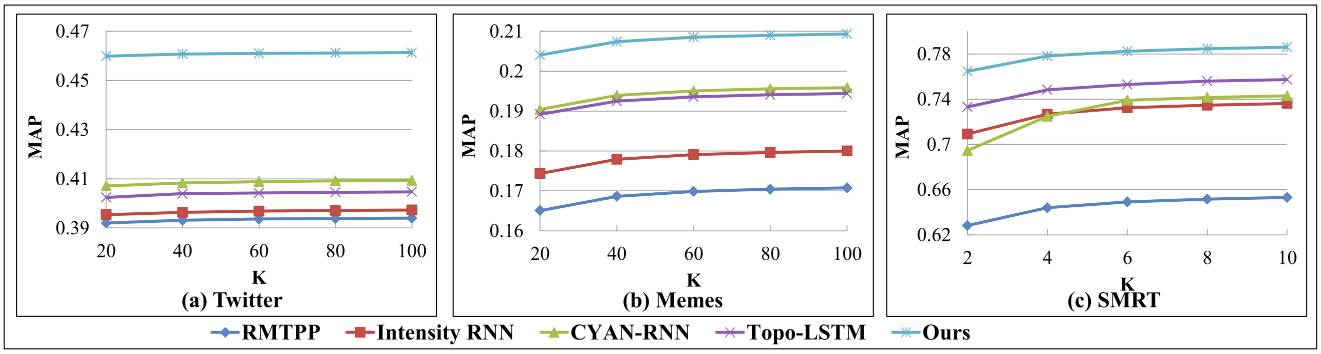


Fig. 5 Results on next event marker prediction

Table 4 Effect of different relation types

Relation type	MAP@2	MAP@4	MAP@6	MAP@8	MAP@10
Temporal	0.7092	0.7268	0.7324	0.7346	0.7362
Temporal + parent	0.7389	0.7510	0.7555	0.7571	0.7585
Temporal + sibling	0.7327	0.7443	0.7483	0.7503	0.7525
Temporal + self	0.7350	0.7471	0.7517	0.7541	0.7554
Ours	0.7647	0.7781	0.7823	0.7846	0.7860

- Validation of the proposed hierarchical Attention Mechanism: Topo-LSTM equally considers the influences from all the related events indicated in the ontology structure. However, not all these events are relevant to the next event. And even if the next event is triggered by several past events, their contributions to the occurrence of the next event are not necessarily the same. Our hierarchical attention mechanism helps to automatically learn the weights of past events from the historical data. The inter-relation and intra-relation attention weights capture the influences from different types and relations and within each type of relation respectively. As shown in Fig. 5, with the help of our proposed hierarchical attention, our MRS-RMTPP consistently predicts the next event marker more accurately compared to Topo-LSTM. Specifically, the MAP@K for next event marker prediction is improved by 14.0% to 14.3% in Twitter, 7.7% to 7.8% in memes, and 3.8% to 4.3% in SMRT.
- Effect of different relation types: We further study the effect of different relation types in SMRT to validate the importance of distinguishing different types of relations. As shown in Table 4, parent relation is more important than sibling and self. This means that when an event happens, it is more likely to see its parent event happens next, rather than that of its sibling or itself. This is understandable as a broadcast signal error is more likely to trigger a broader/higher-level signal (e.g., train signal) rather than other not-so-relevant error (e.g., emergency brake error). Moreover, self seems to be a more relevant relation type than sibling. This is because an event in SMRT represents an error, and this error may occur repeatedly until it is manually fixed. Hence, the occurrence of an event is very likely to be followed by itself in an event sequence. Table 4 shows that in general, different types of relations have varying degrees of influence to the next event. Using a layer of attention to learn the impact weights from different relation types helps to provide more intelligible results to SMRT staff for maintenance service and analysis. Finally, as shown in the last row, comprehensively considering different relation types helps achieve a MAP of 0.76 (MAP@2) to 0.79 (MAP@10) for the next event marker prediction.

Results on Next Event Time Prediction

Figure 6 reports result on next event time prediction. As introduced in "MRS-RMTPP" and Table 3, there are mainly two ways to model event time. One is Eq. 12 adopted by intensity RNN [31] and our model, and the other is Eq. 14 adopted by RMTPP [35] and CYAN-RNN [13]. Figure 6 shows that, in general, modeling time using Eq. 12 is better than using Eq. 14. The reason could be that the expectation of a Gaussian function is easy to derive (as in Eq. 13), while Eq. 15 does not have an analytic solution. Du et al. [35] and [13] use the numerical integration techniques to approximate the expectation value, which may affect the accuracy of time prediction.

More specifically, by modeling density function using a Gaussian function, intensity RNN outperforms RMTPP (by 11.1% to 30.9% relatively) and CYAN-RNN (by 0.72% to 11.1% relatively) across all the three datasets in terms of RMSE. Our proposed MRS-RMTPP further improves the RMSE of intensity RNN by 8.3% on average. Note that, both intensity RNN and our model adopt Eq. 12 as the event time density function. Because we comprehensively consider temporal, ontology and implicit cross dependencies in the event sequence embedding, our learned embedding can capture the dynamics from the history event sequences more accurately. This leads to a more representative event sequence embedding, and thus enables us to make better time prediction for the next event. Similar trend can be found in the other pair of methods. i.e., RMTPP and CYAN-RNN both use Eq. 14 for time modeling. However, the RMSE of CYAN-RNN is on average

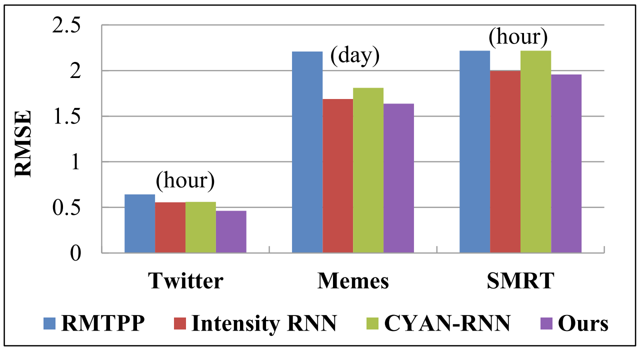


Fig. 6 Results on next event time prediction

12.3% smaller than that of RMTPP, because CYAN-RNN models the implicit cross event dependencies in addition to the temporal dependencies considered by RMTPP.

Conclusions

In this work, we propose a novel recurrent marked temporal process to model the event sequences with both ontology relation structure and temporal structure. We design a multi-relation structure RNN model to embed the dynamics that govern the generation process of the events, and the learned event sequence embedding is then used to model the density function of the temporal point process. Due to its capability of capturing the two structures in the history event sequences, our proposed MRS-RMTPP outperforms the state-of-the-art baselines by 3.8% to 24.4% (MAP@K) and 1.9% to 38.6% (RMSE) on two real-world social network datasets and one real-world ATS log dataset.

Funding Information This study was financially supported in part by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate. This research is also financially supported in part by the National Research Foundation, Prime Ministers Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program.

Compliance with Ethical Standards

Conflict of Interests This research was mainly conducted when Hongyun Cai and Vincent W. Zheng worked at Advanced Digital Sciences Center. This research was supported in part by the National Research Foundation (NRF), Prime Ministers Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate. It is also supported in part by the National Research Foundation, Prime Ministers Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Dong H, Ning B, Cai B, Hou Z. Automatic train control system development and simulation for high-speed railways. IEEE Circuits Syst Mag. 2010;10(2):6–18.
2. Wang H, Feng R, Leung AC, Tsang KF. Lagrange programming neural network approaches for robust time-of-arrival localization. Cogn Comput. 2018;10(1):23–34.
3. Zhang H, Wu L, Song Y, Su C, Wang Q, Su F. An online sequential learning non-parametric value-at-risk model for high-dimensional time series. Cogn Comput. 2018;10(2):187–200.
4. Wang J, Zheng VW, Liu Z, Chang KC. Topological recurrent neural network for diffusion prediction. In: ICDM; 2017.
5. Daley DJ, Vere-Jones D. An introduction to the theory of point processes, vol II. 2nd ed. Probability and its applications (New York), General theory and structure. 2008.
6. Kingman JFC, Vol. 3. Poisson processes. Oxford: Oxford University Press; 1993.
7. HAWKES AG. Spectra of some self-exciting and mutually exciting point processes. Biometrika. 1971;58(1):83–90.
8. Isham V, Westcott M. A self-correcting point process. Adv Appl Probab. 1979;37:629–46.
9. Engle R, Duration RJR. Autoregressive conditional a new model for irregularly spaced transaction data. Econometrica. 1998;66(5):1127–62.
10. Grossberg S. REcurrent neural networks. Scholarpedia. 2013;8 (2):1888.
11. Du N, Dai H, Trivedi R, Upadhyay U, Gomez-Rodriguez M, Song L. Recurrent marked temporal point processes: embedding event history to vector. In: KDD; 2016. p. 1555–64.
12. Xiao S, Yan J, Yang X, Zha H, Chu SM. Modeling the intensity function of point process via recurrent neural networks. In: AAAI; 2017. p. 1597–603.
13. Wang Y, Shen H, Liu S, Gao J, Cheng X. Cascade dynamics modeling with attention-based recurrent neural network. In: IJCAI; 2017. p. 2985–91.
14. Li Y, Yang L, Xu B, Wang J, Lin H. Improving user attribute classification with text and social network attention. Cogn Comput. 2019;11(4):459–68.
15. Ma Y, Peng H, Khan T, Cambria E, Hussain A. Sentic LSTM: a hybrid network for targeted Aspect-Based sentiment analysis. Cogn Comput. 2018;10(4):639–50.
16. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
17. Cho K, van Merriënboer B, Gülçehre Ç, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: EMNLP; 2014. p. 1724–34.
18. Yang H, Cheung LP. Implicit heterogeneous features embedding in deep knowledge tracing. Cogn Comput. 2018;10(1):3–14.
19. Lauren P, Qu G, Yang J, Watta P, Huang G, Lendasse A. Generating word embeddings from an extreme learning machine for sentiment analysis and sequence labeling tasks. Cogn Comput. 2018;10(4):625–38.
20. Tai KS, Socher R, Manning CD. Improved semantic representations from tree-structured long short-term memory networks. In: ACL; 2015. p. 1556–66.
21. Zheng J, Cai F, Chen W, Feng C, Chen H. Hierarchical neural representation for document classification. Cogn Comput. 2019;11(2):317–27.
22. Zhang B, Yin X. SSDM2: a two-stage semantic sequential dependence model framework for biomedical question answering. Cogn Comput. 2018;10(1):73–83.
23. Shuai B, Zuo Z, Wang B, Wang G. DAG-recurrent neural networks for scene labeling. In: CVPR; 2016. p. 3620–29.
24. Liu Z, Zheng VW, Zhao Z, Zhu F, Chang KC, Wu M, et al. Semantic proximity search on heterogeneous graph by proximity embedding. In: AAAI; 2017. p. 154–60.
25. Liu Z, Zheng VW, Zhao Z, Yang H, Chang KCC, Wu M, et al. Subgraph-augmented path embedding for semantic user search on heterogeneous social network. In: WWW; 2018.
26. Brillinger DR, Guttorp PM, Schoenberg FP. In: Point processes, temporal. American Cancer Society; 2013.
27. Aalen O, Borgan O, Gjessing H. Survival and event history analysis: a process point of view statistics for biology and health. 2008.
28. Mei H, Eisner J. The neural Hawkes process: a neurally self-modulating multivariate point process. In: NIPS; 2017.

29. Song H, Rajan D, Thiagarajan JJ, Spanias A. Attend and diagnose: clinical time series analysis using attention models. In: AAAI; 2018. p. 4091–98.
30. Ma F, Chitta R, Zhou J, You Q, Sun T, Gao J. Dipole: diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In: KDD; 2017. p. 1903–11.
31. Xiao S, Yan J, Yang X, Zha H, Chu SM. Modeling the intensity function of point process via recurrent neural networks. In: AAAI; 2017. p. 1597–1603.
32. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C. Cambridge: Cambridge University Press; 1992.
33. Werbos PJ. Backpropagation through time: what it does and how to do it. Proceedings of the IEEE. 1990;78(10):1550–60.
34. Kingma DP, Ba J. Adam: a method for stochastic optimization. CoRR arXiv:1412.6980. 2014.
35. Du N, Dai H, Trivedi R, Upadhyay U, Gomez-Rodriguez M, Song L. Recurrent marked temporal point processes: embedding event history to vector. In: 1555–64; 2016.
36. Hodas NO, Lerman K. The simple rules of social contagion. CoRR arXiv:1308.5015. 2013.
37. Leskovec J, Backstrom L, Kleinberg J. Meme-tracking and the dynamics of the news cycle. 2009.
38. Team TD. Theano: a python framework for fast computation of mathematical expressions. CoRR arXiv:1605.02688. 2016.
39. Hyndman RJ, Koehler AB. Another look at measures of forecast accuracy. International Journal of Forecasting. 2006;22(4): 679–88.

Affiliations

Hongyun Cai[1] · Thanh Tung Nguyen[2] · Yan Li[3] · Vincent W. Zheng[4] · Binbin Chen[5] · Gao Cong[2] · Xiaoli Li[2]

Hongyun Cai
laineycai@tencent.com

Thanh Tung Nguyen
ng0155ng@e.ntu.edu.sg

Yan Li
li.yan@adsc.com.sg

Binbin Chen
binbin_chen@sutd.edu.sg

Gao Cong
gaocong@ntu.edu.sg

Xiaoli Li
xlli@ntu.edu.sg

[1] Tencent, Keji Middle 1st Rd, Shenzhen, China

[2] Nanyang Technological University, 50 Nanyang Avenue, Singapore, 639798, Singapore

[3] Advanced Digital Sciences Center, 1 Create Way, 14-02 Create Tower, Singapore, 138602, Singapore

[4] WeBank, Shahexilu 1819, Shenzhen, China

[5] Singapore University of Technology and Design & Advanced Digital Sciences Center, 1 Create Way, 14-02 Create Tower, Singapore, 138602, Singapore