Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

What Happens Next? Future Subevent Prediction Using Contextual Hierarchical LSTM

Linmei Hu,[1] **Juanzi Li,**[1] **Liqiang Nie,**[2] **Xiao-Li Li,**[3] **Chao Shao**[1]
[1] Department of Computer Science and Technology, Tsinghua University, China
[2] School of Computer Science and Technology, Shandong University, China
[3] Institute for Infocomm Research, A*STAR, Singapore
{hulinmei1991, lijuanzi2008, nieliqiang}@gmail.com, xlli@i2r.a-star.edu.sg, birdlinux@gmail.com

Abstract

Events are typically composed of a sequence of subevents. Predicting a future subevent of an event is of great importance for many real-world applications. Most previous work on event prediction relied on hand-crafted features and can only predict events that already exist in the training data. In this paper, we develop an end-to-end model which directly takes the texts describing previous subevents as input and automatically generates a short text describing a possible future subevent. Our model captures the two-level sequential structure of a subevent sequence, namely, the word sequence for each subevent and the temporal order of subevents. In addition, our model incorporates the topics of the past subevents to make context-aware prediction of future subevents. Extensive experiments on a real-world dataset demonstrate the superiority of our model over several state-of-the-art methods.

1 Introduction

As defined in (Allan et al. 1998), an event refers to a particular thing that happens at a specific time and place. Typically, an event is composed of a sequence of subevents. For instance, an *earthquake* event may consist of sequential subevents such as *omen*, *quake*, *damages*, *rescue efforts*, *lack of food and water*, *chaos* and *outbreak of epidemics*. A subevent is usually described by a news article. Over years, tens of thousands of news events have been reported and archived with the progression of their corresponding subevents.

As Mark Twain said "the past does not repeat itself, but it often rhymes". In the spirit of this reflection, in this paper, given a sequence of a few subevents, we aim to automatically predict the future subevent by leveraging large-scale historical events. Take the event "*Egyptian revolution of 2011*" as an example, given its sequential subevents described by the headlines of news articles[1], namely, "*Conflict occurred again in Egypt on 22nd, and people plan to hold a million-people march''*, "*Egypt's military will deliver a speech to respond to the conflict between demonstrators and police*", our model outputs "*protests*", "*burst*", "*chaos*", "*cause*", "*deaths*", "*injuries*" word by word, which describes a possible future subevent. It is consistent with a later news report "*The Egyptian protest has caused 32 people dead and more than 2000 people injured*". This demonstrates our model can predict a future subevent that has not yet happened. Clearly, predicting future subevents in advance is of great importance to governments, companies and news agencies. Governments can benefit from it by taking proactive actions to avoid damages and deaths based on predicted subevents. Companies can better control their potential crisis of public relations. News agencies can pay their close attention to news topics that the public are particularly interested in.

This problem is, however, challenging due to the following three challenges. Firstly, it is hard to accurately represent a subevent. Constructing hand-crafted features for subevent representation often needs domain knowledge and cannot be well generalized to other applications. Secondly, events usually exhibit sequential structures at two levels: 1) words describing a particular subevent are semantically positioned in order, and 2) multiple subevents belonging to the same event temporally progress in a sequential order. It is hard, if not impossible, to build a unified predictive model to seamlessly capture the two-level dependencies. Finally, the topic of the future subevent is likely to be closely related to the topics of previous subevents. How can we incorporate the topics, i.e. context-aware features, in our model?

Although great effort has been dedicated to event detection using social media (Sakaki, Okazaki, and Matsuo 2010) and search engines (Ginsberg et al. 2009), a relatively few work have been proposed to predict future events. Different from existing work that focused on target (known) event prediction (Radinsky, Davidovich, and Markovitch 2012), we focus on non-targeted (unknown) event prediction. For example, Radinsky et al. (2012) extracted causality relations between two events and generalized them using ontology for prediction. Granroth-Wilding et al. (2016) extracted event chains (Chambers and Jurafsky 2008) from texts and learned the coherence score of two events using a compositional neural network. Manshadi et al. (2008) learned a probabilistic language model of the event sequences. Pichotta and Mooney (2016) described a model for statistical script learning using Long Short-Term Memory (LSTM). All these work need hand-crafted features to represent events and can only predict the future events from a given set of candidate

[1] The headlines and model outputs are actually in Chinese. We have translated them into English.

events that exist in the training set. In this work, we predict future subevents with a generative model. We can generate subevents which do not exist in the training data. In addition, we automatically represent subevents by embeddings, which are generic and can be applied to other applications.

In this paper, we propose a novel model, **C**ontext-aware **H**ierarchical **L**ong **S**hort-**T**erm **M**emory (CH-LSTM) for future subevent prediction, which takes text descriptions of previous subevents (e.g. news headlines) as input and generates a short text describing the future subevent as our output. Our CH-LSTM model has a *two-level* hierarchical LSTM architecture. Its first level is to encode a subevent, which sequentially consumes the words describing the subevent one by one and then maps each subevent into an *embedding*. Its second level, on the other hand, is to encode the observed subevent sequences where we also incorporate contextual *topic* features to enhance the semantic information. In particular, we represent a subevent with a concatenation of an embedding vector from words and an embedding vector from its topic. With the two-level LSTM architecture, each subevent sequence is mapped into an embedding. Finally, the embedding is fed into another LSTM for (word by word) next subevent generation. Our CH-LSTM model, considering sequential structures at two levels of abstraction and incorporating additional contextual features, is thus more capable of predicting the next subevent.

In summary, our contributions are threefold:

1) To our best knowledge, this is the first work on future (unknown) subevent prediction by automatically generating a short text (a sequence of words) to describe it.
2) We propose a novel neural CH-LSTM model, which is able to capture the two-level sequential structures of subevent sequences and to incorporate additional topic information for subevent prediction.
3) Experimental results on a large-scale real world dataset demonstrate our model significantly outperforms several state-of-the-art methods in future subevent prediction.

2 Our Proposed Methodology

Let us first define some notations. An *event* $\mathbb{E}$=$(s_1, ..., s_M)$ is considered as a sequence of M *subevents*. Each subevent s_m in $\mathbb{E}$ $(1 \leq m \leq M)$ is denoted by its description text (e.g., news headline) which is a sequence of words, i.e. s_m=$(w_{m,1}, ..., w_{m,N_m})$, where $w_{m,n} \in \mathcal{V}$ denotes the n-*th* word in the m-*th* subevent s_m, and $\mathcal{V}$ is the vocabulary.

Given a sequence of observed subevents $(s_1, ..., s_{m-1})$, we aim to learn a probability distribution over all possible texts of the next subevent s_m. This can be defined by a language model:

$$P(s_m|s_{1:m-1}) = \Pi_{n=1}^{N_m} P(w_{m,n}|w_{m,1:n-1}, s_{1:m-1}). \quad (1)$$

Obviously, a naive method is to address this problem by considering an event $\mathbb{E}$ as a whole word sequence by concatenating all its subevents $s_1, ..., s_M$ together. Then we can apply traditional *n-grams* to compute conditional probability tables for each word token given its n preceding tokens. However, it suffers from the curse of dimensionality and is intractable for realistic vocabulary size (Serban et al. 2016). Recently, the RNN (Recurrent Neural Network) based language models (Mikolov et al., 2010) have been proposed to learn long *n-gram* contexts. Nevertheless, RNN suffers from the problem of vanishing or exploding gradient. To tackle this problem, variants of RNN, such as LSTM (Hochreiter and Schmidhuber 1997) and Gated Recurrent Unit (Cho et al. 2014) were designed, which have been shown to be superior to RNN models (Kim et al. 2016).

In the light of this, we build our models based on LSTM. In the following subsections, we first introduce a basic LSTM prediction model for our future subevent prediction task and then present our proposed CH-LSTM model. It captures the two-level sequential structure of a subevent sequence (i.e. a sequence of words for each subevent and a sequence of subevents) and incorporates the topics of the subevents. Our CH-LSTM model can automatically represent subevents by embedding vectors, which are generic and can be applied to different applications.

2.1 LSTM Prediction Model

In this subsection, we first provide a quick review of the LSTM and then present a basic LSTM prediction model for our problem of future subevent prediction.

Long-Short Term Memory (LSTM). LSTM (Hochreiter and Schmidhuber 1997) is a special kind of RNN, capable of learning long-term dependencies. It is defined as follows: given a temporal sequence of inputs $(\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_N)$, LSTM associates each timestep with an input $\mathbf{i}_n$, memory cell $\mathbf{c}_n$ and output gate $\mathbf{o}_n$. Let $\mathbf{h}_n$ denote the vector computed by LSTM model at time n, σ denote the sigmoid function and $\odot$ denote the element-wise product. The vector representation $\mathbf{h}_n$ for each time-step n is given by:

$$\begin{aligned}
\mathbf{i}_n &= \sigma(\mathbf{W}_{iw}\mathbf{w}_n + \mathbf{W}_{ih}\mathbf{h}_{n-1} + \mathbf{b}_i),\\
\mathbf{f}_n &= \sigma(\mathbf{W}_{fw}\mathbf{w}_n + \mathbf{W}_{fh}\mathbf{h}_{n-1} + \mathbf{b}_f),\\
\mathbf{o}_n &= \sigma(\mathbf{W}_{ow}\mathbf{w}_n + \mathbf{W}_{oh}\mathbf{h}_{n-1} + \mathbf{b}_o),\\
\mathbf{g}_n &= tanh(\mathbf{W}_{gw}\mathbf{w}_n + \mathbf{W}_{gh}\mathbf{h}_{n-1} + \mathbf{b}_g),\\
\mathbf{c}_n &= \mathbf{f}_n \odot \mathbf{c}_{n-1} + \mathbf{i}_n \odot \mathbf{g}_n,\\
\mathbf{h}_n &= \mathbf{o}_n \odot tanh(\mathbf{c}_n),
\end{aligned} \quad (2)$$

where $\mathbf{W}_{*w}$ is the transformation matrix from the input to LSTM states, $\mathbf{W}_{*h}$ is the recurrent transformation matrix between the recurrent states $\mathbf{h}_n$, and $\mathbf{b}_*$ is the bias vector. As shown in Eqn.(2), $\mathbf{c}_n$ is a summation of the previous memory cell $\mathbf{c}_{n-1}$ modulated by the forget gate $\mathbf{f}_n$ and $\mathbf{g}_n$, a function of previous hidden state and the current input modulated by the input gate $\mathbf{i}_n$. Initially, $\mathbf{h}_0$ and $\mathbf{c}_0$ are set to zero vectors. We denoted Eqn.(2) as LSTM(.) function.

LSTM Prediction Model. Our basic LSTM prediction model based on LSTM for future subevent prediction is illustrated in Figure 1. For an event $\mathbb{E}$, we consider all its subevents $(s_1, s_2, ..., s_M)$ as one whole sequence of words by concatenating their description texts together. Note that we associate each subevent with a special token $\langle$end$\rangle$ at its end location. The model takes the word tokens represented by one-hot encoding as input, converts them to embedding vectors $\mathbf{w}_{m,n}$, and consumes them one at a time

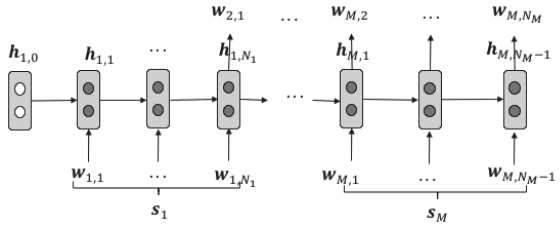

Figure 1: Illustration of our basic LSTM prediction model for our problem. The model encodes the input word sequence describing previous subevents into an embedding and decodes a word sequence describing the possible future subevent out of the embedding.

through LSTM(.) function. After consuming words of m-1 subevents, the model can generate the m-*th* subevent word by word. It currently assigns confidence to $w_{m,n}$ that comes next with a softmax function, which is formally defined as,

$$P(w_{m,n}|w_{m,1:n-1}, s_{1:m-1}) = \frac{\exp(\mathbf{O}_{w_{m,n}}^T \mathbf{h}_{m,n-1})}{\exp(\sum_v \mathbf{O}_v^T \mathbf{h}_{m,n-1})}, \quad (3)$$

where the matrix $\mathbf{O} \in \mathcal{R}^{D\times|\mathcal{V}|}$ (D denotes the dimension of the hidden vectors) represents the output word embeddings. Therefore, each possible next word is projected into another dense vector and compared to the hidden state $\mathbf{h}_{m,n-1} \in \mathcal{R}^D$ computed from Eqn.(2). The initial step $\mathbf{h}_{1,0}$=$\{0\}$ and $\mathbf{h}_{m,0}$=$\mathbf{h}_{m-1,N_{m-1}}$.

Once $\langle$end$\rangle$ is predicted, the model accomplishes prediction with the generated text. However, this model does not learn subevent representations and leads to *long-range dependencies* of words which are usually difficult to capture (Sordoni et al. 2015). In addition, it fails to consider the two-level sequential structure of a subevent sequence and does not incorporate meaningful contextual topic features of subevents which are potentially useful for prediction.

2.2 Contextual Hierarchical LSTM Model

To address the limitations of the LSTM prediction model, we propose CH-LSTM model, illustrated in Figure 2. It considers the sequential structures of events at multiple levels of abstraction (i.e., word-level sequence and subevent-level sequence) and incorporates additional contextual features i.e., the topics of the subevents. It thus alleviates the long-range dependencies in LSTM prediction model and better characterizes the semantic association among subevents. CH-LSTM model is composed of two LSTM encoders for subevent-level encoding and event-level encoding respectively, and one LSTM decoder for next subevent prediction. The subevent-level LSTM encoder first maps each subevent (a sequence of words) to an embedding. The event-level LSTM encoder then keeps track of the past subevents by processing each subevent vector iteratively. After processing m-1 subevents, the hidden state of the event-level LSTM represents a summary of the event up to the (m-1)-*th* subevent, which is subsequently fed to the decoder LSTM for predicting the next subevent s_m word by word. We provide its detailed 3-step process as follows.

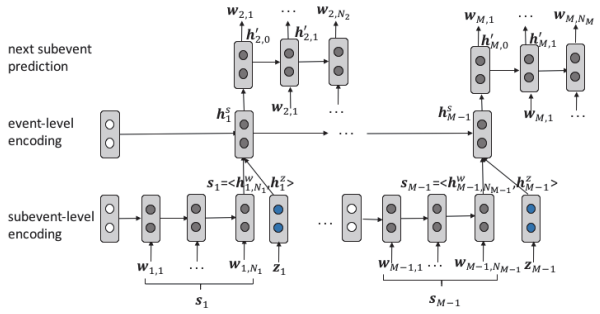

Figure 2: Illustration of our proposed CH-LSTM model. It is composed of three turns. Each subevent represented by a word sequence is first encoded into a dense vector by subevent-level encoding. Afterwards, it is combined with its topic embedding vector and fed into the event-level encoding which accounts for subevent sequences. At last, a decoder LSTM is used to decode the words in the next subevent.

Subevent-Level Encoding. For a subevent s_m = $(w_{m,1}, ..., w_{m,N_m})$, the subevent-level LSTM encoder reads the words within a subevent sequentially and updates its hidden state according to:

$$\mathbf{h}_{m,n}^w = \text{LSTM}_{enc}^w(\mathbf{h}_{m,n-1}^w, \mathbf{w}_{m,n}), n = 1, ..., N_m, \quad (4)$$

where LSTM_{enc}^w denotes the LSTM(.) function (Eqn.(2)) for encoding word sequences and $\mathbf{h}_{m,n}^w \in \mathcal{R}^D$ denotes the recurrent state. Initially, $\mathbf{h}_{m,0}^w$=$\{0\}$. The last recurrent state $\mathbf{h}_{m,N_m}^w$ is a vector storing order-sensitive information about all the words within the subevent s_m. We thus can represent the subevent $\mathbf{s}_m$=$\mathbf{h}_{m,N_m}^w$. In summary, the subevent LSTM encoder maps a subevent to a fixed-length embedding vector, without using hand-crafted features.

Event-Level Encoding. The event-level LSTM encoder takes as input the sequence of subevent embeddings $(\mathbf{s}_1, \mathbf{s}_2, ...\mathbf{s}_m)$ outputted from the subevent-level encoding and computes the sequence of event-level recurrent states:

$$\mathbf{h}_m^s = \text{LSTM}_{enc}^s(\mathbf{h}_{m-1}^s, \mathbf{s}_m), m = 1, ..., M, \quad (5)$$

where LSTM_{enc}^s denotes the LSTM(.) function (Eqn.(2)) for encoding subevent sequences, $\mathbf{h}_m^s \in \mathcal{R}^D$ is the event-level recurrent state, and the initial time step $\mathbf{h}_0^s$=$\{0\}$. The event-level recurrent state $\mathbf{h}_m^s$ summarizes the subevents $(s_1, s_2, ..., s_m)$ that have occurred so far.

To leverage the semantic association among subevents, we further incorporate the *topics* of the subevents as additional contextual features. We employ the widely used Latent Dirichlet Allocation (LDA) to generate topic distributions $\{\theta_m\}_{m=1:M}$ of the subevents, where θ_m=$\{\theta_{m,k}\}_{k=1:K}$ is a K-dimensional vector. K denotes the number of topics. We take the topic z_m=$\arg\max_k \theta_{m,k}$ with the largest probability as the topic of the subevent s_m. The topic z_m is also represented by a one-hot vector and converted to an embedding vector $\mathbf{h}_m^z$ by the model. Hereafter, we represent a subevent $\mathbf{s}_m$=$\langle \mathbf{h}_{m,N_m}^w, \mathbf{h}_m^z \rangle$, which is a concatenation of the hidden vectors from words and its topic.

In this way, we incorporate the contextual features in our model, which is likely to boost next subevent prediction.

Next-Subevent Prediction. After encoding a subevent sequence $\mathbf{s}_{1:m-1}$ with the above two-level LSTM architecture, a LSTM decoder is applied to predict the next subevent $\mathbf{s}_m$ word by word. Formally, we aim to estimate the probability $P(\mathbf{s}_m|\mathbf{s}_1, ..., \mathbf{s}_{m-1})$ using Eqn.(1).

The desired condition on previous subevents is obtained by initializing the recurrent state of the LSTM decoder with the event-level encoding, i.e., $\mathbf{h}'_{m,0}$=$\mathbf{h}^s_{m-1}$, where $\mathbf{h}'_{m,0}$ is the initial recurrent state of the decoder. Similar to encoding, the decoding is in the form of:

$$\mathbf{h}'_{m,n} = \text{LSTM}_{dec}(\mathbf{h}'_{m,n-1}, \mathbf{w}_{m,n}), n = 1, ..., N_m, \quad (6)$$

where LSTM_{dec} is the LSTM(.) function (Eqn.(2)) for decoding a word sequence describing the next possible subevent. In a LSTM decoder, each recurrent state $\mathbf{h}'_{m,n-1}$ is used to compute the probability of the next word $\mathbf{w}_{m,n}$ with a softmax layer as shown in Eqn.(3). Once $\langle$end$\rangle$ is predicted, the decoder terminates.

2.3 Training and Testing

For both the basic LSTM prediction model and our proposed CH-LSTM model, their model parameters $\mathbf{W}$ are learned by maximizing the log-likelihood of subevents $\{s_m\}_{m\in\{2:M\}}$ given previous ones $s_{1:m-1}$, defined by the probabilities estimated in Eqn.(1) and Eqn.(3),

$$\begin{aligned}\mathcal{L} &= \sum_{e=1}^{E_{train}} \sum_{m=2}^{M} logP(s^e_m|s^e_{1:m-1}) \\ &= \sum_{e=1}^{E_{train}} \sum_{m=2}^{M} \sum_{n=1}^{N^e_m} P(w^e_{m,n}|w^e_{m,1:n-1}, s^e_{1:m-1}),\end{aligned} \quad (7)$$

where the superscript e denotes the *e-th* event in the training set. E_{train} is the total number of events in the training set. Batch gradient descent was adopted for optimization.

When dealing with the testing data, we can adopt a beam-search (Sordoni et al. 2015) for forward prediction until $\langle$end$\rangle$ is predicted in the decoding procedure. Let we assume that the beam-search size is 1.

3 Experiments

3.1 Dataset

There is no benchmark dataset for our future subevent prediction task. We therefore crawled a large-scale Chinese news event dataset containing 15,254 news series from Sina News[2]. Each news series consists of a sequence of news articles in temporal order reporting on the same event, and the average number of articles for all news series is 50. We only used headlines of articles in our experiments as they summarize the key content and main idea of the articles.

We further processed the data as follows: 1) We used a window of size 5 to segment each new series with more than five articles into non-overlapping events (partitions) where news articles inside an event are treated as subevents. We choose window size 5, as we observe from the data, there is little dependency beyond more than 5 continuous subevents. After segmentation, we obtained 155,358 events in total. 2) We then used ICTCLAS[3] to perform tokenization for our event data and subsequently removed the stopwords and the words that occur in less than 100 documents to reduce data sparsity. We finally got a vocabulary of 4,515 unique words including a special end-of-subevent symbol $\langle$end$\rangle$. On average, each subevent contains about five words.

After preprocessing, we randomly split all the events into three parts: 80% for training, 10% for validation and the remaining 10% for test. The statistics of our final dataset for experiments are summarized in Table 1.

Table 1: Statistics of Dataset.

	Training	Validation	Test
Events	124,288	15,535	15,535
Subevents	607,090	75,802	75,957

3.2 Experimental Setting and Evaluation Metrics

We determined the hyper-parameters in our models through experiments on the training set and validation set. Particularly, we chose the parameter values which led to the best results on validation set and apply these optimal values in our model to evaluate our proposed method in the independent test set. The optimal parameter values are given as follows.

1) LSTM parameters and word embedding were initialized from a uniform distribution between [-0.08, 0.08]; 2) Learning rate = 0.1; 3) Batch size = 32; 4) Dropout rate = 0.2; 5) The dimension of word embeddings and topic embeddings = 100, and the dimension of hidden vector D = 400; 6) The number of hidden layers of the LSTM networks = 2; 7) The topic number = 1,000.

Evaluation Metrics. Two criteria, namely *perplexity* and *word error-rate*, were employed to measure the performance of our models. Perplexity is a standard metric in information theory (Shannon 2001). It measures how well a model fits the data and thus can perform better prediction. Lower perplexity indicates a better model. Formally, the per-word perplexity of a model is defined as follows:

$$\text{Perp} = \exp(-\frac{1}{N_w} \sum_{e=1}^{E_{test}} logP(s^e_M|s^e_{1:M-1})) , \quad (8)$$

where N_w is the total number of words in the test set, E_{test} is the number of events in the test set, s^e_M is the M*-th*, i.e. the last subevent to be predicted in the e*-th* event.

Following (Serban et al. 2016), we also employed the word classification error (also known as word error-rate). This is defined as the number of words that are predicted incorrectly divided by the total number of words in the

[2] http://news.sina.com.cn/zt/

[3] http://ictclas.nlpir.org/

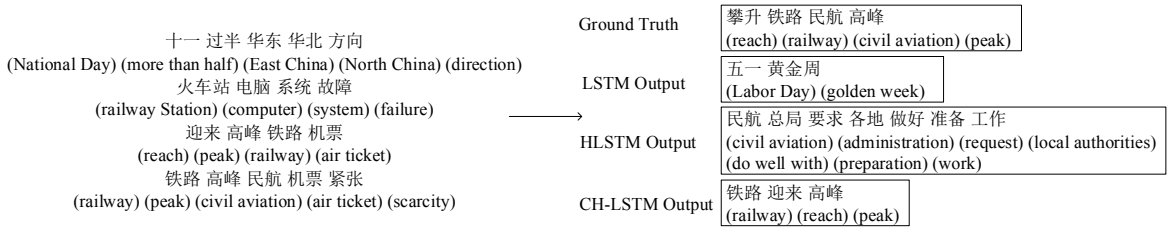

Figure 3: An example of model outputs. The left shows the observed previous four subevents. The right shows the ground truth and predicted next subevents of different models.

dataset[4]:

$$\text{Error\_Rate} = \frac{1}{N_w} \sum_{e=1}^{E_{test}} \sum_{n=1}^{N_M^e} I(w_{M,n}^{'e} \neq w_{M,n}^e), \qquad (9)$$

where I is an indicator function – it equals to 1 when a predicted word is not true, i.e., $w_{M,n}^{'e} \neq w_{M,n}^e$; otherwise 0. $w_{M,n}^{'e}$ denotes the predicted n-*th* word in the subevent s_M^e and $w_{M,n}^e$ denotes the true n-*th* word in the subevent. N_M^e is the total number of words in s_M^e. As we aim to predict more accurate words for the future subevent, we prefer a model with low word classification error.

3.3 Experimental Results

We benchmarked our proposed CH-LSTM models against both state-of-the-art language models and neural network baselines based on the same dataset as illustrated in Table 1. Particularly, we compared with three well-established *n-gram language models* (we set n=5 in this paper), including backoff n-gram, Modified Kneser-Ney and Witten-Bell Discounting n-gram model, which were implemented based on SRILM tool (Stolcke and others 2002). We also compared our model with LSTM prediction model and HLSTM model (namely our proposed CH-LSTM model without additional topic features) under the same parameter setting.

Table 2: The average perplexity and word error-rate of five runs on test set.

Model	Perp	Error_Rate
Backoff N-Gram	264.07	93.03%
Modified Kneser-Ney	257.24	93.06%
Witten-Bell Discounting N-Gram	255.48	92.60%
LSTM	201.59 ± 0.38	75.22% ± 0.02%
HLSTM	129.44 ± 0.23	71.06% ± 0.02%
CH-LSTM	**127.74 ± 0.21**	**70.02% ± 0.01%**

Our results are presented in Table 2. All neural models outperform state-of-the-art n-gram models substantially w.r.t. both word perplexity and word classification error. The extremely high word error rate of n-gram models demonstrate it is not capable for future subevent prediction. Comparatively, the neural models can better deal with it. The last two lines of Table 2 confirm that considering two-level sequential structures of events achieves significant gains in both measures. It allows a gain of around 70 perplexity points and 5 percentage points of word error-rate compared to the LSTM prediction model, demonstrating the effectiveness of our proposed methods. From Table 2, we also observe that CH-LSTM model further improves the performance of HLSTM w.r.t. both metrics, indicating the importance of incorporating subevent topics for next subevent prediction, as the topic of the future subevent is likely to be closely related to the topics of previous subevents.

We used beam-search for our neural models to approximate the most probable next subevent, s_M, given the previous subevents $s_{1:M-1}$. As a case study on travel issues on China's National day (around 5 hundred million people travel during this holiday season), we show the outputs of 3 different models in Figure 3. We observe that the output of LSTM is totally not relevant. While HLSTM can achieve better results (it still varies from the true output), our proposed CH-LSTM model generates almost same output with the ground truth.

Finally, we visualized the intermediate results (i.e., embeddings of all the subevents in the test dataset) based on our CH-LSTM model. The visualization is implemented with t-sne (Maaten and Hinton 2008). We observe from Figure 4, topically similar subevents are close in the embedding space. The embedding representation can generalize to test subevents very well even if they have not been seen in the training data, as long as their words appear in the model vocabulary. This equips our model with capability for subevent representation which could be further used for other applications such as subevent similarity computation.

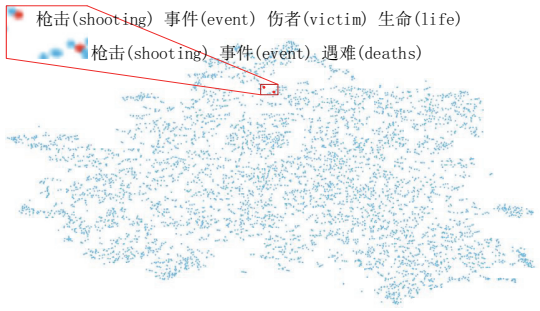

Figure 4: Visualization of subevent embedding. Topically similar subevents are close in the embedding space.

[4]For a word prediction to be counted as correct, both the word and its position must be correct.

3.4 Next Subevent Ranking

In this section, we focus on comparisons of our models on the *next subevent ranking* task, where we are given a sequence of subevents and the goal is to find the most probable next subevent from a set of candidate subevents. Formally, given a sequence of subevents $s_{1:M-1}$, we aim to find the most likely next subevent s_M from a candidate set of next subevents S, such that:

$$s_M = \arg\max_{s \in S} logP(s|s_{1:M-1}), \tag{10}$$

We ran next subevent ranking experiments with a dataset generated from our test set. Particularly, following (Ghosh et al. 2016), we randomly divided the test dataset (15,535 events) into 311 non-overlapping subsets with each containing 50 events except the last one which contains remaining 35 events (i.e. 15,535=311*50+35). For each event (consisting of a few subevents) in a subset, we aim to choose the best last subevent given its previous subevents. We treated the last subevents of all the events in the corresponding subset as candidate subevents. The metric hits@n, i.e. the proportion of correct subevents ranked in the top n candidate subevents is used to evaluate different models. Table 3 shows the performance of different models on this task.

Table 3: Performance comparison among models for next subevent ranking.

Model	hits@1	hits@5	hits@10
Random	2.00%±0.10	10.00%±0.15	20.00%±0.20
LSTM	21.96%± 0.12	49.73%±0.16	66.31%±0.21
HLSTM	25.11%±0.10	54.49%±0.17	70.22%±0.18
CH-LSTM	**25.79 %± 0.10**	**55.68%±0.18**	**71.57%±0.20**

As we can see from Table 3, all the models significantly outperform the random method for subevent ranking task. In addition, our proposed HLSTM model and CH-LSTM model achieve significant improvements consistently, compared to LSTM prediction model, which further validates the importance of considering hierarchical structure and topic information for predicting related future subevents.

4 Related Work

Our work is mainly related to event prediction and RNN language modeling. We compare with them as follows.

4.1 Event Prediction

Event prediction aims to predict in advance of the occurrence of a future event. Some work focused on learning relations of two events for prediction. For example, Radinsky et al. (2012) extracted generalized causality relations of two events in the form of "x causes y" from past news and applied the templates on a present news event to predict the next possible event. Granroth-Wilding et al. (2016) extract event knowledge about typical sequences of events from text (2008) and learned the coherence score of two events using a compositional neural network. They aim to predict the strength of association between two events, which could be used for predicting whether an event is likely to be the next event.

Other work focused on learning event sequences for next subevent prediction. Radinsky et al. (2013) mined chains of events from a large-scale news articles to predict in advance of the occurrence of target events. In this work, we focus on non-targeted event prediction. Manshadi et al. (2008) learned a probabilistic model of event sequences from Internet Web log stories based on n-gram language models. Pichotta and Mooney (2016) described a model for statistical script learning using LSTM, whereby a script encodes knowledge of prototypical sequences of events. They represent an event by a predicate with several arguments and can only predict the future events from a given set of candidate events which already exist in the training set. Different from existing work, in this work, we propose an end-to-end generative model which does not need hand-crafted features and can generate new events that have not been seen in the training set. It embeds both word sequence inside subevents and subevent sequences automatically for next subevent prediction.

4.2 RNN Language Models

Neural language models encompass a rich family of neural network architectures for language modeling. Some example architectures include feed-forward (Bengio et al. 2006), and RNN (Mikolov et al. 2010). Due to the vanishing gradient problem of RNN, it is difficult for RNN to learn long-term dependencies. To address the problem, variants such as LSTM (Hochreiter and Schmidhuber 1997) and Gated Recurrent Unit (GRU) (Cho et al. 2014) were presented and widely applied in language modeling. Recent research efforts further improved the RNN models by exploiting hierarchical structures (Li, Luong, and Jurafsky 2015) for a wide range of applications such as web query suggestion (Sordoni et al. 2015), movie dialogue modeling (Serban et al. 2016) and video representation (Pan et al. 2016). Some other efforts focused on improving the RNN models using an attention mechanism (Xu et al. 2015; Li, Luong, and Jurafsky 2015) and additional contextual features (Ghosh et al. 2016; Mikolov and Zweig 2012). Differently, we present a novel hierarchical LSTM network combining topic information of subevents for future subevent prediction, which is different from the proposed models in the above work.

5 Conclusion and Future Work

In this paper, we propose to predict a future subevent by generating a short text describing it and present a novel model, CH-LSTM, which is an end-to-end solution that takes the texts describing previous subevents as input and outputs the text describing the next possible subevent. Our model does not need hand-crafted features and well represent subevents by embeddings, which are generic and can be applied to other domains. Our experiments have demonstrated that CH-LSTM model outperforms state-of-the-art n-gram language models and neural models, as it can capture two-level sequential structures of a subevent sequence and enhance the semantics by incorporating the topics of subevents. In future

work, we will explore the extensions of our model such as using an attention mechanism for improving future subevent prediction.

Acknowledgments

The work is supported by 973 Program (No. 2014CB340504), NSFC-ANR (No. 61261130588), NSFC key project (No. 61533018), Key Technologies Research and Development Program of China (No. 2014BAK04B03), Fund of Online Education Research Center, Ministry of Education (No. 2016ZD102), Tsinghua University Initiative Scientific Research Program (No. 20131089256), and THUNUS NExT Co-Lab. Besides, we gratefully acknowledge the assistance of Hanwang Zhang (NUS), Shizhu He (Institute of Automation, Chinese Academy of Sciences) for improving the work.

References

Allan, J.; Carbonell, J. G.; Doddington, G.; Yamron, J.; and Yang, Y. 1998. Topic detection and tracking pilot study final report.

Bengio, Y.; Schwenk, H.; Senécal, J.-S.; Morin, F.; and Gauvain, J.-L. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*. Springer. 137–186.

Chambers, N., and Jurafsky, D. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, volume 94305, 789–797. ACL.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1724–1734. ACL.

Ghosh, S.; Vinyals, O.; Strope, B.; Roy, S.; Dean, T.; and Heck, L. 2016. Contextual LSTM (CLSTM) models for large scale NLP tasks. In *KDD Workshop on Large-scale Deep Learning for Data Mining (DL-KDD)*. ACM.

Ginsberg, J.; Mohebbi, M. H.; Patel, R. S.; Brammer, L.; Smolinski, M. S.; and Brilliant, L. 2009. Detecting influenza epidemics using search engine query data. *Nature* 457(7232):1012–1014.

Granroth-Wilding, M., and Clark, S. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the 30th Conference on Artificial Intelligence*, 2727–2733. AAAI.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2016. Character-aware neural language models. In *Proceedings of the 30th Conference on Artificial Intelligence*, 2741–2749. AAAI.

Li, J.; Luong, M.-T.; and Jurafsky, D. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 1106–1115. ACL.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Manshadi, M.; Swanson, R.; and Gordon, A. S. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference*, 159–164. FLAIRS.

Mikolov, T., and Zweig, G. 2012. Context dependent recurrent neural network language model. In *The 4th IEEE Workshop on Spoken Language Technology (SLT)*, 234–239.

Mikolov, T.; Karafiát, M.; Burget, L.; Cernockỳ, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *International Conference on Spoken Language Processing*, volume 2, 3.

Pan, P.; Xu, Z.; Yang, Y.; Wu, F.; and Zhuang, Y. 2016. Hierarchical recurrent neural encoder for video representation with application to captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Pichotta, K., and Mooney, R. J. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the 30th Conference on Artificial Intelligence*, 2800–2806. AAAI.

Radinsky, K., and Horvitz, E. 2013. Mining the web to predict future events. In *Proceedings of the 6th ACM international conference on Web search and data mining*, 255–264. ACM.

Radinsky, K.; Davidovich, S.; and Markovitch, S. 2012. Learning causality for news events prediction. In *Proceedings of the 21st international conference on World Wide Web*, 909–918. ACM.

Sakaki, T.; Okazaki, M.; and Matsuo, Y. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, 851–860. ACM.

Serban, I. V.; Sordoni, A.; Bengio, Y.; Courville, A.; and Pineau, J. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th Conference on Artificial Intelligence*. AAAI.

Shannon, C. E. 2001. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review* 5(1):3–55.

Sordoni, A.; Bengio, Y.; Vahabi, H.; Lioma, C.; Grue Simonsen, J.; and Nie, J.-Y. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th Conference on Information and Knowledge Management*, 553–562. ACM.

Stolcke, A., et al. 2002. Srilm-an extensible language modeling toolkit. In *Interspeech*, volume 2002, 2002.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, 2048–2057. JMLR Workshop and Conference Proceedings.