SAM-Net: Integrating Event-Level and Chain-Level Attentions to Predict What Happens Next

Shangwen Lv,[1,2] **Wanhui Qian,**[1,2] **Longtao Huang,**[1*] **Jizhong Han,**[1] **Songlin Hu**[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{lvshangwen, qianwanhui, huanglongtao, hanjizhong, husonglin}@iie.ac.cn

Abstract

Scripts represent knowledge of event sequences that can help text understanding. Script event prediction requires to measure the relation between an existing chain and the subsequent event. The dominant approaches either focus on the effects of individual events, or the influence of the chain sequence. However, only considering individual events will lose much semantic relations within the event chain, and only considering the sequence of the chain will introduce much noise. With our observations, both the individual events and the event segments within the chain can facilitate the prediction of the subsequent event. This paper develops self attention mechanism to focus on diverse event segments within the chain and the event chain is represented as a set of event segments. We utilize the event-level attention to model the relations between subsequent events and individual events. Then, we propose the chain-level attention to model the relations between subsequent events and event segments within the chain. Finally, we integrate event-level and chain-level attentions to interact with the chain to predict what happens next. Comprehensive experiment results on the widely used New York Times corpus demonstrate that our model achieves better results than other state-of-the-art baselines by adopting the evaluation of Multi-Choice Narrative Cloze task.

Introduction

When people encode information into natural language, they usually assume that readers are able to seamlessly make inferences based on commonsense knowledge. For example, given an event "Smith entered a restaurant", people can make a number of probable inferences: he got seated, he read the menu, he ordered food, and so on. Such commonsense knowledge is called *scripts* (Schank and Abelson 1977), which is intuitive to human readers and usually omitted in the context. This brings great challenges to automatically understand language texts for machines because machines have no extra information to infer other events within the chain (Modi 2016). Figure 1 shows a restaurant visiting script.

A script system constructs the structure of abstract events and involves the actions and the entities that participate in them. Script systems can support many understanding applications , such as automated storytelling (Swanson and Gordon 2008) or inference of missing events (Chambers and Jurafsky 2008), etc.

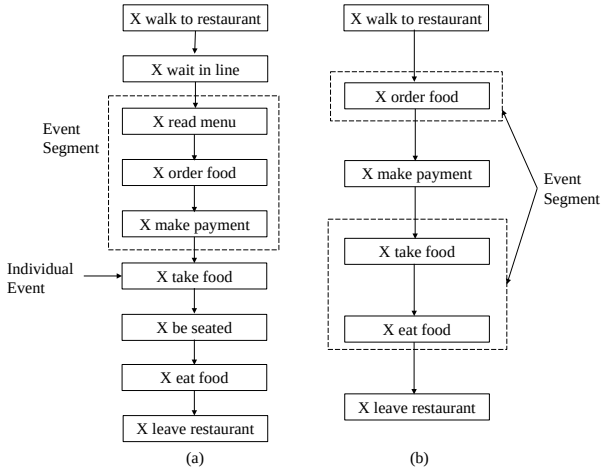

Figure 1: Script for restaurant visiting. An event segment is a part of the script and it can have various lengths of events. The event segment can be continuous or discontinuous.

Our work follows a recent line of *script learning*. Early scripts were manually engineered for specific domains and time-consuming to construct (Schank and Abelson 1977). Chambers and Jurafsky (2008) described methods of automatically learning scripts from large text corpus. Following their works, many researchers focused on statistical methods to learn scripts, such as PMI (Chambers and Jurafsky 2008) or skip-grams (Balasubramanian et al. 2013), etc. However, they relied on count-based methods to model event pair relations and suffered from sparsity issues.

There is another line of works which introduces event embedding to tackle the sparsity problems. Granroth-Wilding and Clark (2016) adopted word embeddings (Mikolov et al. 2013) to represent the verb and its arguments in an event. They adopted a Siamese Network instead of PMI to calculate the relation score between subsequent events and individual events within the chain. Pichotta and Mooney (2016a) supported that individual events models were incapable of

*Correspondence to Longtao Huang.

expressing interactions between entities and they adopted Long Short-Term Memory (LSTM) Model (Hochreiter and Schmidhuber 1997) to model event sequences. (Wang, Zhang, and Chang 2017) argued that the LSTM model suffered from over-fitting problem and they utilized LSTM to model narrative event orders and adopted Dynamic Memory Network (Weston, Chopra, and Bordes 2014) to model the event pair relations. However, they did not fully utilize the event segments within the chain to predict script events, which have richer semantic information than individual events and introduce less noise than chain sequence modeling.

In Figure 1(a), we assume the event to predict is "X eat food". We can see that the individual event "X take food" has a strong relation with "X eat food". We also observe that there are many continuous event segments within the chain, such as <"X read menu", "X order food", "X make payment">, <"X walk to restaurant", "X wait in line"> etc. The event segment <"X read menu", "X order food", "X make payment"> can also strongly predict the event "X eat food". The event segments contain a set of individual events which are related to each other and segments can be continuous or discontinuous. A discontinuous event segment <"X order food", "X take food", "X eat food"> is shown in Figure 1(b).

We can see that the subsequent event can have strong relation with some individual events or event segments within the chain. Only considering individual events will lose much semantic information within the chain. Only considering the full sequence in the chain to predict the subsequent event may introduce much noise. The event "X wait in line" has minor relation with the predicted event but sequence models also take it into consideration. This paper selects individual events and diverse event segments within the chain to interact with the subsequent events.

Ultimately, this paper aims to provide solutions for open-domain script learning. We focus on the task of predicting the subsequent events with the existing event chain. This is close to the narrative cloze task in (Chambers and Jurafsky 2008) and (Granroth-Wilding and Clark 2016). In particular, we address two challenges in this task: (1) An event chain is a sequence of events and events can be more sparse than words in sentences. The challenge is how to represent event chains accurately. (2) Individual events within the chain have semantic relations with the subsequent events. The segments in the chain also have an effect on predicting the subsequent events. The challenge is how to integrate them together to represent the relations between the existing event chain and subsequent events.

For the first challenge, we develop self attention mechanism (Lin et al. 2017) to capture diverse event segments from the event chain. Then we adopt DenseNet (Huang et al. 2017) to perform feature extraction to reduce redundancy and get a vector representation for the event chain. For the second challenge, we first adopt attentions (Luong, Pham, and Manning 2015) to capture the relation between subsequent events and individual events to get an *event-level* attentional representation. Then we perform attentions between subsequent events and the event segments to obtain a *chain-level* attentional representation. Finally, we integrate *event-level* and *chain-level* attentional representations to interact deeply with the chain representation to predict what happen next.

Our contributions can be summarized as follows:

- We develop self attention mechanism to capture diverse event segments within the chain and adopt DensetNet to extract features for the event chain.
- We integrate event-level and chain-level attentions to interact with the chain representation to model the relations between the existing event chain and subsequent events.
- We achieve the best results compared with other state-of-the-art baselines on the evaluation of Multi-Choice Narrative Cloze task.

Problem Definition

A script in this paper is a sequence of events. An event e is a structure $v(e_s, e_o, e_p)$, where v is the verb describing the event, e_s is the subject, e_o is the object and e_p is a prepositional object. e_s, e_o, e_p are called event arguments. For example, we can extract an event structure *bring (Tom, book, to Mary)* from the sentence "Tom brought the book to Mary".

As shown in Figure 2, given a sequence of events $<e_1, e_2, e_3, ..., e_n>$ and a set of choice events $\{e_{c_1}, e_{c_2}, ..., e_{c_m}\}$, our work is to predict the event e_{c_i} which is most likely to happen next. We call the given event sequences as *event chain* and the next event e_{c_i} as *choice event*. Following (Granroth-Wilding and Clark 2016), (Wang, Zhang, and Chang 2017) and (Li, Ding, and Liu 2018), we choose the Multi-Choice Narrative Cloze (MCNC) as our evaluation method. We select e_{c_i} from a set of given events with the same protagonist, where only one event is the right answer. The event which gets the highest probability is considered to be the output. We adopt accuracy metric to compare the results of different models.

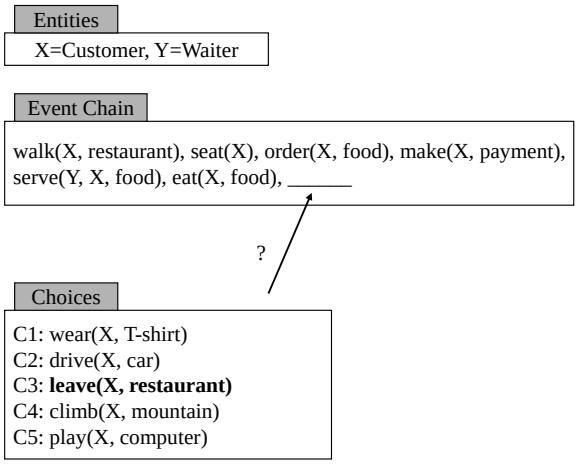

Figure 2: Multiple choice narrative cloze. The likely event to happen next is marked in bold.

Model

In this section, we describe our script event prediction model which is composed of the following three components: (1) event representation layer, (2) attentional representation layer and (3) prediction layer. The overall architecture of the proposed model is shown in Figure 3.

Event Representation Layer

In the event representation layer, we represent an event as a composition function of its components. An event has four component v, e_s, e_o and e_p. Each component is represented as a $d-$dimensional vector by using a pre-trained word embedding from GloVe (Pennington, Socher, and Manning 2014) and will be updated during the training process. For those words out of vocabulary (OOV), we represent them with zero vectors. For those events with less than 4 components, we will set NULL to the corresponding components. For the event "Tom loves Mary", we set v=love, e_s=Tom, e_o=Mary, e_p=NULL and the NULL component will be represented by zero vectors.

We denote the word embeddings of v, e_s, e_o, e_p as $\mathbf{v}(v), \mathbf{v}(e_s), \mathbf{v}(e_o), \mathbf{v}(e_p)$, respectively. In this paper, we follow (Wang, Zhang, and Chang 2017) and represent the event using a $tanh$ composition layer:

$$\begin{aligned}\mathbf{v}(e) =& tanh(\mathbf{W}_e^v \cdot \mathbf{v}(v) + \mathbf{W}_e^s \cdot \mathbf{v}(e_s) + \mathbf{W}_e^o \cdot \mathbf{v}(e_o) \\ &+ \mathbf{W}_e^p \cdot \mathbf{v}\,(e_p) + \mathbf{b}_e)\,,\end{aligned} \tag{1}$$

where $\mathbf{W}_e^v, \mathbf{W}_e^s, \mathbf{W}_e^o, \mathbf{W}_e^p \in \mathbb{R}^{d\times d}, \mathbf{b}_e \in \mathbb{R}^d$ are model parameters and $\mathbf{v}(e) \in \mathbb{R}^d$.

Attention Representation Layer

In the attention representation layer, we model the relation between choice event and individual events in the chain to get the event-level attention. Then we develop self attention mechanism (Lin et al. 2017) to extract diverse event segments within the event chain. Next we model the relations between choice event and the chain to get an chain-level attention representation.

Event-Level Attention

The individual events in the chain have different semantic relations with the choice event and contribute differently to the choice event. We adopt an attention mechanism (Luong, Pham, and Manning 2015) to specify different summation weights to events in the chainas follows:

$$\begin{aligned}\mathbf{v}_E &= \mathrm{ReLU}(\mathbf{V}\mathbf{a}_e)\,, \\ \mathbf{a}_e &= \mathrm{softmax}(\mathbf{V}^T\mathbf{W}_1\mathbf{v}(e_{c_i}))\,,\end{aligned} \tag{2}$$

where $\mathbf{v}_E \in \mathbb{R}^u$ is the *event-level* context representation and $\mathbf{V}$=($\mathbf{v}(e_1)$, $\mathbf{v}(e_2)$, ... , $\mathbf{v}(e_n)$), $\mathbf{a} \in \mathbb{R}^n$ is the normalized attention weight vector, $\mathbf{W}_1 \in \mathbb{R}^{u\times u}$ is the model parameter.

Narrative Event Orders

Given the event chain $<e_1, e_2, ..., e_n>$, we model the narrative event orders of the chain by utilizing a standard LSTM. We feed $\mathbf{v}(e_1)$,$\mathbf{v}(e_2)$, ... , $\mathbf{v}(e_n)$ to LSTM recurrently and calculate the hidden state vectors $\mathbf{h}_i$ of LSTM:

$$\mathbf{h}_i = \mathrm{LSTM}(\mathbf{v}(e_i), \mathbf{h}_{i-1})\,. \tag{3}$$

Each hidden state vector $\mathbf{h}_i \in \mathbb{R}^u$ contains the narrative event order information from e_1 to e_i. The initial hidden state vectors $\mathbf{h}_0$ is initialized randomly and we can obtain a sequence of hidden state vectors $<\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n>$ for the event chain.

There are m choice events $\{e_{c_1}, e_{c_2}, ..., e_{c_m}\}$ for each chain. For each choice event, we append it to the event chain respectively and get the hidden state vectors using the same way from $\mathbf{h}_1$ to $\mathbf{h}_n$. Then we obtain the hidden state vector $\mathbf{h}_{c_i}$ for the choice event e_{c_i} by feeding $\mathbf{v}(e_{c_i})$ as follows:

$$\mathbf{h}_{c_i} = \mathrm{LSTM}(\mathbf{v}(e_{c_i}), \mathbf{h}_n)\,. \tag{4}$$

We perform this operation m times and obtain $\mathbf{h}_{c_1}, \mathbf{h}_{c_2}$, ..., $\mathbf{h}_{c_m}$. Each $\mathbf{h}_{c_i}$ contains the narrative event orders of the event chain $<e_1, e_2, ..., e_n>$ and the choice event e_{c_i}.

For simplicity, we denote all the n hidden state of the event chain as $\mathbf{H}$, which has size n-by-u and u is the size of the hidden state vector in LSTM:

$$\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n)\,. \tag{5}$$

Event Segments within Event Chain

In this part, we develop self attention mechanism (Lin et al. 2017) to extract diverse event segments within the event chain.

$\mathbf{H}$ contains all the hidden state vectors for events in the event chain. We can follow (Lin et al. 2017) and introduce a single attention vector $\mathbf{a}$ which contains the summation weight of $\mathbf{H}$ to extract common components within the event chains. However, it does not interact e_i and e_j directly and cannot extract event segments directly from the chain. Inspired by their ideas, we develop the self attention mechanism and interact e_i with $e_1, e_2, ..., e_{i-1}, e_{i+1}, ..., e_n$ respectively to capture the relation between e_i and the other events in the chain. The events which have strong relations with e_i will be extracted and form an event segment with e_i. We perform self attention mechanism as follow:

$$\begin{aligned}\mathbf{M} &= \mathbf{H}\mathbf{A}_c\,, \\ \mathbf{A}_c &= \mathrm{softmax}(\mathrm{ReLU}(\mathbf{H}^T\mathbf{W}_2\mathbf{H}))\,,\end{aligned} \tag{6}$$

where $\mathbf{M} \in \mathbb{R}^{u\times n}$ contains diverse event segments within the chain. The column i is the event segment which contains e_i and other related events. $\mathbf{A}_c \in \mathbb{R}^{n\times n}$ is the self attention matrix, $\mathbf{W}_2 \in \mathbb{R}^{u\times u}$ is the model parameter.

Chain-Level Attention

$\mathbf{M}$ contains diverse event segments within the chain and each event segment can contribute to predicting the subsequent event differently. In order to reflect the relations between subsequent events between event segments, we perform an attention (Luong, Pham, and Manning 2015) mechanism to match $\mathbf{h}_{c_i}$ to the corresponding event segments, such as matching the choice event "X eat food" to the event segment <"X read menu","X order food">. Then we can get an chain-level context representation which represents the relation between subsequent events and the event segments. We perform chain-level attention as follows:

$$\begin{aligned}\mathbf{v}_R &= \mathrm{ReLU}(\mathbf{M}\mathbf{a}_s)\,, \\ \mathbf{a}_s &= \mathrm{softmax}(\mathbf{H}^T\mathbf{W}_3\mathbf{V}(e_{c_i}))\,,\end{aligned} \tag{7}$$

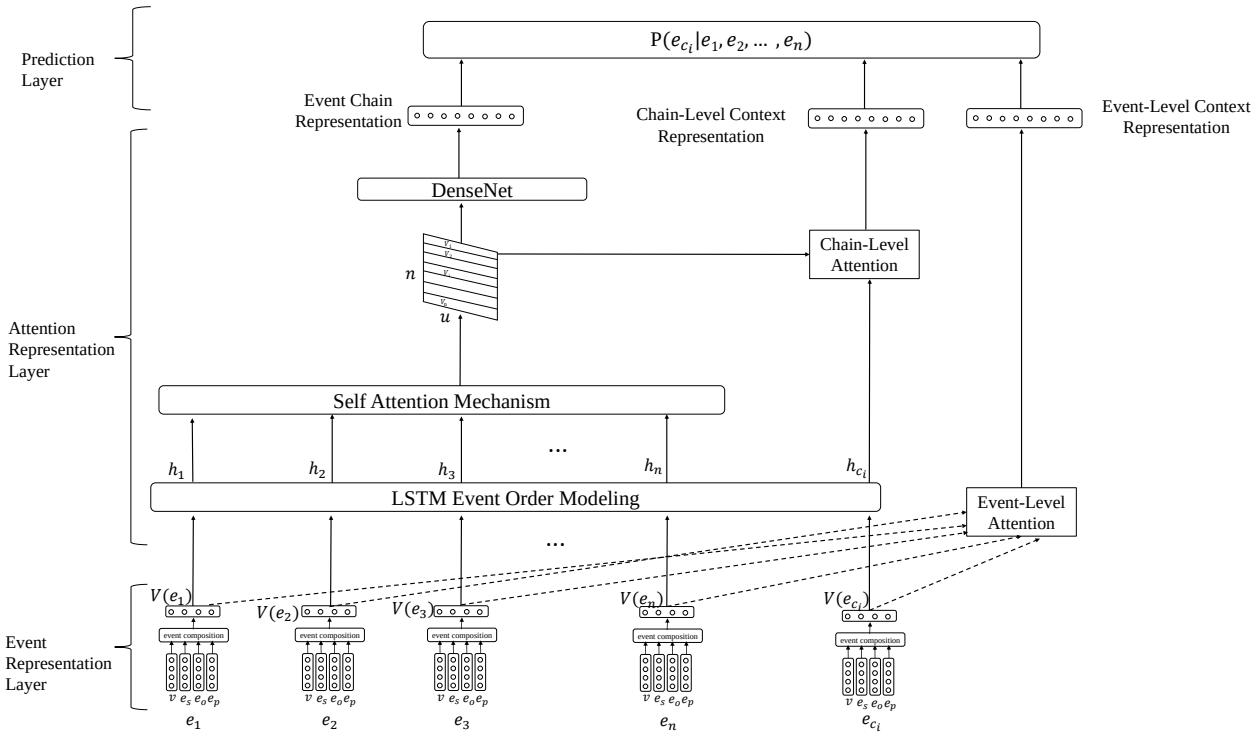

Figure 3: Overall Model Structure

where $\mathbf{v}_R \in \mathbb{R}^u$ is the *chain-level* context representation, $\mathbf{a}_s \in \mathbb{R}^u$ is the normalized attention summation weight, $\mathbf{W}_3 \in \mathbb{R}^{u \times u}$ is the model parameter.

Chain Feature Extraction

The extracted event segments may extract similar event segments and cause redundancy problem, so we adopt DenseNet (Huang et al. 2017) to perform feature extraction to filter out redundancy. DenseNet allows to create direct connections from one layer to all subsequent layers and the inputs of the ℓ^{th} layer are the feature maps of all preceding layers:

$$\mathbf{x}_\ell = \mathrm{F}_\ell([\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{\ell-1}]), \tag{8}$$

where F_ℓ is the non-linear transformation of the ℓ^{th} layer and $[\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{\ell-1}]$ is the concatenation of the feature-maps produced in layers $0, 1, ..., \ell - 1$.

Following (Huang et al. 2017), we define $\mathrm{F}_\ell(\bullet)$ as a composite function of three consecutive operations: a batch normalization (BN) (Ioffe and Szegedy 2015), a rectified linear unit (ReLU) (Glorot, Bordes, and Bengio 2011) and a 1×3 convolution (Conv). A dense block contains an operation chain (BN-Relu-Conv)$\times3$. The transformation between two dense blocks is the transition layer and it contains an operation chain (Conv-Pooling). The overall structure of DenseNet in this paper is as follows: Input-Conv-Dense Block 1-Transition Layer 1-Dense Block 2-Transition Layer 2-DenseBlock 3-Pooling-Linear.

The input of DenseNet is **M**. The output of the DenseNet is a feature vector representation for event chain $\mathbf{v}_C \in \mathbb{R}^u$.

We have obtained the event chain feature vector $\mathbf{v}_C$ and the choice event feature vector $\mathbf{h}_{c_i}, \mathbf{h}_{c_2}, ..., \mathbf{h}_{c_m}$. In the next part, we will interact $\mathbf{v}_E, \mathbf{v}_R$ with $\mathbf{v}_C$ to select the choice event which has the maximum probability.

Prediction Layer

In the prediction layer, we integrate the obtained event-level and chain-level attentional representations to interact with the chain representation. We can get the relation scores with all the event choices, then we select the event choice with the highest score as the event to happen next. We calculate relation score **s** as follow:

$$\begin{aligned} \mathbf{r}_1 &= \mathrm{ReLU}(\mathbf{W}_{11}\mathbf{v}_C + \mathbf{W}_{12}\mathbf{v}_E + \mathbf{b}_1), \\ \mathbf{r}_2 &= \mathrm{ReLU}(\mathbf{W}_{21}\mathbf{v}_C + \mathbf{W}_{22}\mathbf{v}_R + \mathbf{b}_2), \\ \mathbf{s} &= \mathrm{similarity}(\mathbf{r}_1, \mathbf{r}_2). \end{aligned} \tag{9}$$

where $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{R}^u, \mathbf{W}_{11}, \mathbf{W}_{12}, \mathbf{W}_{21}, \mathbf{W}_{22} \in \mathbb{R}^{u\times u}, \mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^u$.

When calculating the similarity between $\mathbf{r}_1$ and $\mathbf{r}_2$, there are multiple choices and we select the following methods to compare:

- **Cosine Similarity** is the cosine similarity of two vectors: $\mathrm{cosine}(\mathbf{r}_1, \mathbf{r}_2) = \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{||\mathbf{r}_1|| ||\mathbf{r}_2||}$.
- **Dot Similarity** is the dot product of two vectors: $\mathrm{dot}(\mathbf{r}_1, \mathbf{r}_2) = \mathbf{r}_1 \cdot \mathbf{r}_2$.
- **Fusion Similarity** α is the fusion of two vectors: $\alpha(\mathbf{r}_1, \mathbf{r}_2) = \mathbf{w}_1 \cdot \mathbf{r}_1 + \mathbf{w}_2 \cdot \mathbf{r}_2 + b$.

- **Fusion Similarity** β is the fusion of two vectors: $\alpha(\mathbf{r}_1, \mathbf{r}_2) = \mathbf{w}_1 \cdot \mathbf{r}_1 + \mathbf{w}_2 \cdot \mathbf{r}_2 + \mathbf{w}_3 \cdot (\mathbf{r}_1 \odot \mathbf{r}_2) + b$.

In Fusion Similarity α and β, $\odot$ is the element-wise multiplication, $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \in \mathbb{R}^u$, $b \in \mathrm{R}$.

There are five choice events for each event chain, and we will perform softmax to get the final score for each choice as follows:

$$\mathrm{P}(e_{c_i}|e_1, e_2, ..., e_n) = \frac{\exp(\mathbf{s}_i)}{\sum_j \exp(\mathbf{s}_j)} \quad (10)$$

We select the choice event which get the maximum probability as the predicted event:

$$\text{predicted event} = \arg\max_i \mathrm{P}(e_{c_i}|e_1, e_2, ..., e_n) \quad (11)$$

Training

Given an event chain and a set of event choice, our goal is to minimize the cross-entropy loss between the right answers and the predicted answers. The loss function of event chain prediction is defined as follows:

$$L(\Theta) = -\frac{1}{N}\sum_{k=1}^{N} \log \mathrm{P}(e_{c_i}|e_1, e_2, ..., e_n) + \frac{\lambda}{2}||\Theta||_2^2, \quad (12)$$

where $\mathrm{P}(e_{c_i}|e_1, e_2, ..., e_n)$ is the predicted relation score corresponding to the right answer e_{c_i} for the event chain k, λ is the parameter for L2 regularization and Θ is the set of model parameters.

Experiments

In this section, we evaluate the effectiveness of our proposed model with several baseline methods. Accuracy is adopted as the evaluation metric.

Dataset

Following (Granroth-Wilding and Clark 2016) and (Wang, Zhang, and Chang 2017), we extract events from the NYT portion of the Gigaword corpus (Graff et al. 2003). The C&C tools (Curran, Clark, and Bos 2007) are used for POS tagging and dependency parsing, and OpenNLP[1] is used for coreference resolution. The training set consists of 1.01M event chains. We adopt 10,000 event chains as development set and 10,000 event chains as the test set. There are 5 choice events for every event chain and only one of them is correct.

Baselines

We compare our model with the following baseline models:

- **PMI** (Chambers and Jurafsky 2008) is the co-occurrence based model which adopts Pointwise Mutual Information (PMI) to score each candidate event e_{c_i} by the sum of the PMI scores with given events $e_1, e_2, ..., e_n$.
- **Bigram** (Jans et al. 2012) is the counting-based skip-gram models which calculates event pair relations based on bigram probabilities and is trained using maximum likelihood estimation.

[1]https://opennlp.apache.org/

Method	Accuracy(%)
Random	0.2
PMI	31.25
Bigram	28.76
Word2vec	42.10
LSTM	46.75
Event-Comp	49.66
PairLSTM	50.34
SGNN	52.32
SAM-Net	**54.48**

Table 1: Results of script event prediction on the test set.

- **Word2vec** (Mikolov et al. 2013) learns word embeddings from large text corpora and the learned word embeddings for verbs and arguments are used to calculate pairwise event relation scores.
- **LSTM** (Pichotta and Mooney 2016a) adopts the hidden state $\mathbf{h}_{c_i}$ of the choice event and $\mathbf{h}_i$ to predict the output.
- **Event-Comp** (Granroth-Wilding and Clark 2016) is the neural network model which learns event representation by calculating pair-wise event scores using a Siamese network.
- **PairLSTM** (Wang, Zhang, and Chang 2017) is the model which integrates event order information and pairwise event relations to predict output event.
- **SGNN** (Li, Ding, and Liu 2018) constructs a narrative event evolutionary graph (NEEG) to describe event evolutionary patterns. It adopts scaled graph neural network (SGNN) to model event interactions and learn event representations.

Experiment Configuration

We tune the hyper-parameters in our proposed *SAM-Net* model using the development set. We set batch size to 128 and regularization weight to $\lambda = 10^{-5}$. We adopt Adam Optimizer (Kingma and Ba 2014) to optimize our model and the initial learning rate is set to 10^{-4}. We adopt the Glove (Pennington, Socher, and Manning 2014) pre-trained word embeddings and the dimension is set to 100. The size of LSTM hidden state is set to 128. The parameters are initialized with Xavier Initialization (Glorot and Bengio 2010).

Experiment Results and Analyses

Experiment results are shown in Table 1. From the results, we can have the following observations:

(1) Word2vec, Event-Comp and other neural models achieve significantly better results than count-based models PMI and Bigram. The main reason is that count-based models have sparsity issues and low-dimensional embeddings for events are more effective to represent the semantic relationships among events.

(2) Comparisons between Event-Comp, PairLSTM and LSTM demonstrate that only considering the strong order relation in the event chains is not adequate enough to represent the relations between existing chains and choice events.

This is because only adopting LSTM model to represent the event chains may cause over-fitting problems.

(3) Comparisons between Event-Comp, LSTM and our proposed model SAM-Net demonstrate that SAM-Net can better represent the relations between choice events and the event chain. SAM-Net can extract diverse event segments within the chain and capture the relations from event-level and chain-level attention representations to provide more accurate semantic relations. The result confirms our intuition in the introduction.

(4) SGNN constructs event graphs to represent the relations among events and it achieves better results than other baselines. In this paper, by integrating event-level and chain-level attention representations we can obtain better results than the single SGNN model.

(5) Our proposed SAM-Net model achieves the best performance on the script event prediction task, which is an absolute 2.16% improvement over the best baselines.

Comparative Experiments

We conduct experiments to compare the four relation score functions in the prediction layer to investigate further into their influence on the final results. The results are shown in Table 2. We can conclude that different relation score functions can have influence on the final results. The Fusion Similarity β can get the best results because the fusion similarity can fuse the vectors to interact more adequately. We select Fusion Similarity β in our model to compare with other baselines.

Method	**Accuracy(%)**
Cosine	53.23
Dot	53.45
Fusion α	54.22
Fusion β	**54.48**

Table 2: Effects of different relation score functions.

Quanlitative Study

We provide a qualitative analysis with the following example. Given an existing event chain shown in Table 3. The right choice event is "act(X, shop teacher,-)".

(1) The event-level attention aims to calculate the relation between the choice event and individual events. The normalized attention vector is as follows: [0.04,0.06,0.03,0.08,0.02,0.07,0.23,0.47]. We can see that events 7 and 8 have important influence on the choice event. The event-level attention tends to capture the semantic relation between events. The higher attention weight means the event has more similar event components with the choice event, e.g., 'play' and 'act'.

(2) Self attention aims to discover event segments within the existing event chain. The event segments extracted from the event chain are as follows (according to the self attention matrix): <1,2,6>, <1,2,3>, <3,8>, <4,5>, <5,6>, <4,5,6>, <5,6,7,8>, <4,7,8>. With our observations, 6 out of 8 event segments are meaningful. The event segments <1,2,6> and <3,8> does not bring much benefit and they might introduce noise in the model. We will propose new methods to generate more meaningful event segments in future works.

Event Order	Event
1	marry(X, Roseanne, -)
2	dismiss(X, -, -)
3	split(X, Roseanne, -)
4	rebound(X, -, in the films)
5	use(X, goodwill, against sexist)
6	win(X, Golden Globe Awards, -)
7	display(X, nasty side, -)
8	play(X, Roscoe Bigger, -)

Table 3: An event chain example. X=Arnold and '-' means NULL components.

(3) The chain-level attention aims to capture the relation between choice event and the event segments. The normalized attention vector is as follows: [0.06,0.02,0.03,0.11,0.09,0.12,0.27,0.30]. We can see that the choice event has a significant relation with the event segments <5,6,7,8> and <4,7,8>. The results confirm our intuition that the event segments have significant influence on the script event prediction.

Ablation Study

In this part, we perform ablation study to get a better insight into each component of the model, as shown in Table 4. *SAM-Net* is our proposed model. The notation '-' means removing this part or replacing this part with simpler operations.

Influence of Event Representation: We first study the event representation part. *-Event Representation* means that we do not adopt the event representation in our model and replace it with simpler <verb, dependency> representation from (Chambers and Jurafsky 2008). The result from Table 4 demonstrates that the richer event representation which contains e_o and e_p is more effective in modeling the semantic relations between events.

Influence of Event-level Attention: We remove the event-level attention and only utilize the chain-level attentional representation to predict the next event. From the result, we can conclude that the relations between choice events and events in the chain does have effect on the final results.

Influence of Self Attention: *-Self Attention* means that we drop Self Attention Mechanism. Since DenseNet needs to perform on a 2-D or 3-D representation, we stack all the hidden states of LSTM to form a matrix and feed it to DenseNet. When we drop self attention part, we cannot perform chain-level attention and we also drop the chain-level attention. *-Self Attention* causes a 4.94 drop, which means that the event segments within the event chain does have significant influence on the final results. And the result confirms our intuition in the introduction part.

Influence of Chain-Level Attention: We remove the chain-level attention and adopt the event-level attentional

representation to predict the next event. The result demonstrates that the relations between choice events and event segments within the chain is essential to the final results, which confirms our intuition in introduction.

Influence of DenseNet: Finally, we study the effect of DenseNet. *-DenseNet* means that we get event chain representation by averaging the matrix representation from Self Attention. The result demonstrates that DenseNet has an important effect on the final results because DenseNet can extract important features for the event chain and get a better representation.

Method	Accuracy(%)	△
SAM-Net	**54.48**	-
-Event Representation	52.35	-1.23
-Event-level Attention	53.25	-2.13
-Self Attention	49.54	-4.94
-Chain-Level Attention	51.38	-3.04
-DenseNet	52.96	-1.52

Table 4: Ablation study of network structure.

Related Work

Script Event Representation: Chambers and Jurafsky (2008) represented events as *<event, dependency>* pairs, where the *event* was typically a verb and the *dependency* was the typed dependency relations between the event and the protagonist such as 'subject' and 'object'. In (Balasubramanian et al. 2013), the researchers observed that the protagonist representation suffered from weakness such as the lack of coherence and they proposed a new representation *<Arg1, Relation, Arg2>*, where *Arg1* and *Arg2* were the subject and objectq in the event. Similar to this idea, Pichotta and Mooney (2014) proposed a richer multi-argument event representation $v(e_s, e_o, e_p)$, where v was the verb, e_s was the subject, e_o was the object and e_p was the entity with prepositional relation with v. Subsequent works like (Pichotta and Mooney 2016a) and (Granroth-Wilding and Clark 2016) adopted the representation. Following (Wang, Zhang, and Chang 2017), we also adopt the multi-argument event representation in this paper.

Script Modeling: Chambers and Jurafsky (2008) adopted PMI to calculate the event relations and Jans et al. (2012) used skip-gram probability to improve the performance. In (Pichotta and Mooney 2014) and (Rudinger et al. 2015), the researchers modeled event co-occurrence and estimated a joint probability distribution over pairs of events. However, the count-based models suffered from sparsity issues. Granroth-Wilding and Clark (2016) employed event embeddings to solve this problem. They represented components in an event as word embeddings and obtained the event embeddings by a composition neural network. In (Pichotta and Mooney 2016a) and (Pichotta and Mooney 2016b), the researchers adopted LSTM (Long Shor-Term Memory) neural networks to model event representation and event relations. Following their works, Wang, Zhang, and Chang (2017) adopted the dynamic memory network model to model event orders and event pair coherence relations.

We observe that there are rich event segments relations within the event chain that can bring benefit to predict the next event, so we develop Self Attention Mechanism (Lin et al. 2017) to extract diverse event segments and represent the event chain as a combination of event segments.

Furthermore, Hu et al. (2017) proposed a hierarchical LSTM model which incorporated the word sequence and event sequence to predict what happens next. Mostafazadeh et al. (2016) proposed the story close task to predict the ending of a story sequence. There are also many researchers constructing script graphs to reflect the relation among events. Regneri, Koller, and Pinkal (2010) collected natural language descriptions of script-specific event sequences from volunteers and built a temporal script graph. Orr et al. (2014) employed Hidden Markov Models to construct a transition graph and performed script inference according to the state transition probability. In (Li, Ding, and Liu 2018), the researchers constructed a narrative event evolutionary graph (NEEG) to represent rich relations among events and presented a scaled graph neural network (SGNN) to model event interactions and learn event representations.

Script Evaluation: Chambers and Jurafsky (2008) proposed the Narrative Cloze Test where they removed some events and adopted the model to predict the missing event. Modi (2016) proposed Adversarial Narrative Cloze (ANC), which aimed to distinguish right and wrong event chains. Following their works, Granroth-Wilding and Clark (2016) proposed Multi-Choice Narrative Cloze (MCNC) task which was designed to select the most likely next event from a set of candidate events. In this paper, we choose MCNC for our evaluation metric and compare the effect of different models to predict script event.

Conclusion and Future Work

We utilize the event-level attention to represent the relations between subsequent events and individual events in the chain. Then, we develop the self attention mechanism to extract diverse event segments within an event chain. We also propose the chain-level attention to capture the relations between subsequent event and event segments. Finally, we integrate event-level and chain-level attentions to predict what happens next. Standard evaluation shows that our model achieves the best results compared to other baselines. In future works, we will propose methods to extract event segments more accurately and will focus on the event generation problem.

Acknowledgement

This research is supported in part by the National Key Research and Development Program of China (No. 2017YFB1010000) and the National Natural Science Foundation of China (No. 61702500).

References

Balasubramanian, N.; Soderland, S.; Etzioni, O.; et al. 2013. Generating coherent event schemas at scale. In *Proceedings*

of the 2013 Conference on Empirical Methods in Natural Language Processing, 1721–1731.

Chambers, N., and Jurafsky, D. 2008. Unsupervised learning of narrative event chains. *Proceedings of ACL-08: HLT* 789–797.

Curran, J. R.; Clark, S.; and Bos, J. 2007. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, 33–36. Association for Computational Linguistics.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323.

Graff, D.; Kong, J.; Chen, K.; and Maeda, K. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia* 4(1):34.

Granroth-Wilding, M., and Clark, S. 2016. What happens next? event prediction using a compositional neural network model. In *AAAI Conference on Artificial Intelligence*.

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

Hu, L.; Li, J.; Nie, L.; Li, X.-L.; and Shao, C. 2017. What happens next? future subevent prediction using contextual hierarchical lstm. In *AAAI*, 3450–3456.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*, volume 1, 3.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jans, B.; Bethard, S.; Vulić, I.; and Moens, M. F. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 336–344. Association for Computational Linguistics.

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

Li, Z.; Ding, X.; and Liu, T. 2018. Constructing narrative event evolutionary graph for script event prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, 4201–4207.

Lin, Z.; Feng, M.; Santos, C. N. d.; Yu, M.; Xiang, B.; Zhou, B.; and Bengio, Y. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 3111–3119.

Modi, A. 2016. Event embeddings for semantic script modeling. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 75–83.

Mostafazadeh, N.; Chambers, N.; He, X.; Parikh, D.; Batra, D.; Vanderwende, L.; Kohli, P.; and Allen, J. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Orr, J. W.; Tadepalli, P.; Doppa, J. R.; Fern, X.; and Dietterich, T. G. 2014. Learning scripts as hidden markov models. In *AAAI*, 1565–1571.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Pichotta, K., and Mooney, R. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, 220–229.

Pichotta, K., and Mooney, R. J. 2016a. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*, 2800–2806.

Pichotta, K., and Mooney, R. J. 2016b. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Regneri, M.; Koller, A.; and Pinkal, M. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 979–988. Association for Computational Linguistics.

Rudinger, R.; Rastogi, P.; Ferraro, F.; and Van Durme, B. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1681–1686.

Schank, R. C., and Abelson, R. P. 1977. Scripts, plans, goals and understanding: An inquiry into human knowledge structures. *Lawrence Erlbaum Associates, Hillsdale, NJ*.

Swanson, R., and Gordon, A. S. 2008. Say anything: A massively collaborative open domain story writing companion. In *Joint International Conference on Interactive Digital Storytelling*, 32–40. Springer.

Wang, Z.; Zhang, Y.; and Chang, C.-Y. 2017. Integrating order information and event relation for script event prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 57–67.

Weston, J.; Chopra, S.; and Bordes, A. 2014. Memory networks. *CoRR* abs/1410.3916.