# Integrating Order Information and Event Relation for Script Event Prediction*

**Zhongqing Wang**[†,‡], **Yue Zhang**[‡] and **Ching-Yun Chang**[‡]
[†]Soochow University, Suzhou, China
[‡]Singapore University of Technology and Design, Singapore
wangzq.antony@gmail.com, yue_zhang@sutd.edu.sg,
chang.frannie@gmail.com

## Abstract

There has been a recent line of work automatically learning scripts from unstructured texts, by modeling narrative event chains. While the dominant approach group events using event pair relations, LSTMs have been used to encode full chains of narrative events. The latter has the advantage of learning long-range *temporal orders*[1], yet the former is more adaptive to partial orders. We propose a neural model that leverages the advantages of both methods, by using LSTM hidden states as features for event pair modelling. A dynamic memory network is utilized to automatically induce weights on existing events for inferring a subsequent event. Standard evaluation shows that our method significantly outperforms both methods above, giving the best results reported so far.

## 1 Introduction

Frequently recurring sequences of events in prototypical scenarios, such as visiting a restaurant and driving to work, are a useful source of world knowledge. Two examples are shown in Figure 1, which are different variations of the "restaurant visiting" scenario, where events are partially ordered and can be flexible. Such knowledge is useful for natural language understanding because texts typically do not include event details when mentioning a scenario. For example, the reader is expected to infer that the narrator could have been driving or cycling given the text "I got flat tire". Another typical use of event chain knowledge is to help infer what is likely to happen next given a previous event sequence in a scenario. We investigate the modeling of stereotypical event chains, which is remotely similar to language modeling, but with events being more sparse and flexibly ordered than words.

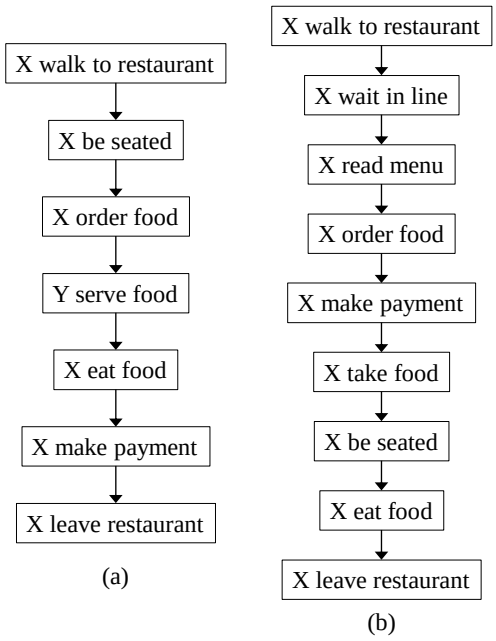


Figure 1: Event sequences for restaurant visiting.

Our work follows a recent line of NLP research on *script learning*. Stereotypical knowledge about partially-ordered *events*, together with their participant *roles* such as "customer", "waiter", and "table", is conventionally referred to as *scripts* (Schank et al., 1977). NLP algorithms have been investigated for automatically inducing scripts from unstructured texts (Mooney and DeJong, 1985; Chambers and Jurafsky, 2008). In particular, Chambers and Jurafsky (2008) made a first attempt to learn scripts from test inducing event

*This work has been done when the first author worked at SUTD.

[1]The term "temporal order" is used throughout this work to indicate the narrative order in texts, following Chambers and Jurafsky (2008). Strictly speaking, the event order we extract is the narrative order.

chains by grouping events based on their narrative coherence, calculated based on Pairwise Mutual Information (PMI). Jans et al. (2012) showed that the method can be improved by calculating event relations using skip bi-gram probabilities, which explicitly model the temporal order of pairs event. Jans et al. (2012)'s model is adopted by a line of subsequent methods on inducing event chains from text (Orr et al., 2014; Pichotta and Mooney, 2014; Rudinger et al., 2015).

While the above methods are statistical, neural network models have recently been used for event sequence modeling. Granroth-Wilding and Clark (2016) used a Siamese Network instead of PMI to calculate the coherence between two events. Rudinger et al. (2015) extended the idea of Jans et al. (2012) by using a log-bilinear neural language model (Mnih and Hinton, 2007) to calculate event probabilities. By learning embeddings for reducing sparsity, the above models give much better results compared to the models of Chambers and Jurafsky (2008) and Jans et al. (2012). Similar in spirit, Modi (2016) predicted the probability of an event belonging to a certain event chain by modeling known events in the chain as a bag of vectors, showing that it outperforms discrete statistical methods. These neural methods are consistent with the earlier statistical models in leveraging event-pair relations.

Pichotta and Mooney (2016) experimented with LSTM for script learning, using an existing sequence of events to predict the probability of a next event, which outperformed strong discrete baselines. One advantage of LSTMs is that they can encode unbounded time sequences without losing long-term historical information. LSTMs capture significantly more order information compared to the methods of Granroth-Wilding and Clark (2016), Rudinger et al. (2015), and Modi (2016), which model the temporal order of only pairs of events. On the other hand, a strong-order LSTM model can also suffer the disadvantage of over-fitting, given the flexible order of event chains in a script, as demonstrated by the cases of Figure 1. In this aspect, event-pair models are more adaptive for flexible orders. However, no direct comparisons have been reported between LSTM and various existing neural network methods that model event-pairs.

We make such comparisons using the same benchmark, finding that the method of Pichotta and Mooney (2016) does not necessarily outperform event-pair models, such as Granroth-Wilding and Clark (2016). LSTM temporal ordering and event pair modeling have their respective strength. To leverage the advantages of both methods, we propose to integrate chain temporal order information into event relation measuring. In particular, we calculate event pair relations by representing events in a chain using LSTM hidden states, which encode temporal information. The LSTM over-fitting issue is mitigated by using the temporal-order in a chain as a *feature* for event pair modeling, rather than the direct model *output*. In addition, observing that the importance of existing events can vary for inferring a subsequent event, we use a dynamic memory network model to automatically induce event weights for each event for inferring the next event. In contrast, previous methods give equal weights to existing events (Chambers and Jurafsky, 2008; Modi, 2016; Granroth-Wilding and Clark, 2016).

Results on a multi-choice narrative cloze benchmark show that our model significantly outperforms both Granroth-Wilding and Clark (2016) and Pichotta and Mooney (2016), improving the state-of-the-art accuracy from 49.57% to 55.12%. Our contributions can be summarized as follows:

- We make a systematic comparison of LSTM and pair-based event sequence learning methods using the same benchmarks.
- We propose a novel dynamic memory network model, which combines the advantages of both LSTM temporal order learning and traditional event pair coherence learning.
- We obtain the best results in the standard multi-choice narrative cloze test.

Our code is released at `https://github.com/wangzq870305/event_chain`.

## 2 Related Work

*Scripts* have been a traditional subject in AI research (Schank et al., 1977), where event sequences are manually encoded in knowledge bases, and used for end tasks such as inference. They are also connected with research in linguistics and psychology, and sometimes referred to as *frames* (Minsky, 1975; Fillmore, 1982) and *schemata* (Rumelhart, 1975). The same concept

is also studied as *templates* in information extraction (Sundheim, 1991). Chambers and Jurafsky (2008) pioneered the recent line of work on script induction (Jans et al., 2012; Pichotta and Mooney, 2016; Granroth-Wilding and Clark, 2016), where the focus is on modeling narrative event chains, a crucial subtask for script modeling from raw text. Below we summarize such investigations.

With respect to **event representation**, Chambers and Jurafsky (2008) casted narrative events as triples of the form $\langle event, dependency \rangle$, where the event is typically represented by a verb and the dependency represents typed dependency relations between the event and a protagonist, such as "subject" and "object". Chambers and Jurafsky (2008) organized narrative chains around a central actor, or protagonist, mining events that share a common protagonist from texts by using a syntactic parser and a coreference resolver. Balasubramanian et al. (2013) observed that the protagonist representation of event chains can suffer from weaknesses such as lack of coherence, and proposed to represent events as $\langle arg_1, relation, arg_2 \rangle$, where $arg_1$ and $arg_2$ represent the subject and object, respectively. Such representation is inspired by open information extraction (Mausam et al., 2012), and offers richer features for event pair modeling. Pichotta and Mooney (2014) adpoted a similar idea, using $v(e_s, e_o, e_p)$ to represent an event, where $v$ is a verb lemma, $e_s$ is the subject, $e_o$ is the object, and $e_p$ is an entity with prepositional relation to $v$. Their representation is used by subsequent work such as Modi (2016) and Granroth-Wilding and Clark (2016). We follow Pichotta and Mooney (2016) in our event representation form.

With respect to **modeling**, existing methods can be classified into two main categories, namely *weak-order* models, which calculate relations between pairs of events, and *strong-order* models, which consider the temporal order of events in a full sequence. Event-pair models have so far been the dominant method in the literature. Earlier work used discrete event representations and estimated event relations by statistical counting. As mentioned earlier, Chambers and Jurafsky (2008) used PMI to calculate event relations, and Jans et al. (2012) used skip bigram probabilites to the same end, which is order-sensitive. Most subsequent methods followed Jans et al. (2012) in using skip n-grams (Pichotta and Mooney, 2014; Rudinger et al., 2015).

Events being multi-argument structures, counting-based methods can suffer from sparsity issues. Recent work employed embeddings to address this disadvantage. Rudinger et al. (2015) learned event embeddings as a by-product of training a log-bilinear language model for events; Granroth-Wilding and Clark (2016) leveraged the skip-gram model of Mikolov et al. (2013) for training the embeddings of event and arguments by ordering them into a pseudo sentence. Modi (2016) utilized *word embeddings* of verbs and arguments directly, using a hidden layer to automatically consolidate word embedding into a single structured *event embeddings*. We follow Modi (2016) and use a hidden layer to learn event argument compositions given word embeddings, training the composition function as a part of the event chain learning process.

Mitigating the sparsity issue of event representations, neural methods can capture temporal orders between events beyond skip n-grams. Our model integrates the advantages of strong-order learning and event-pair learning by using LSTM hidden states as feature representation of existing events in the calculation of event pair relationships. In addition, we use a memory network model to weigh existing events, which gives better results compared to the equal weighting method of existing models.

With respect to **evaluation**, Chambers and Jurafsky (2008) proposed the *Narrative Cloze Test*, which asks for a missing event in a given event chain with a gap. The task has been adopted by various subsequent work for comparing results with Chambers and Jurafsky (2008) (Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015). One issue of the narrative cloze test is that there can sometimes be multiple plausible answers, but only one gold-standard answer, which can make it overly expensive to manually evaluate system outputs. To address this issue, Modi (2016) proposed the *Adversarial Narrative Cloze* (ANC) task, which is to discriminate between pairs of real and corrupted event chains. Granroth-Wilding and Clark (2016) proposed the *Multi-Choice Narrative Cloze* (MCNC) task, which is to choose the most likely next event from a set of candidates given a chain of events. We choose MCNC for comparing different models.

**Other related work** includes learning temporal relations of events (Modi and Titov, 2014; Uz-

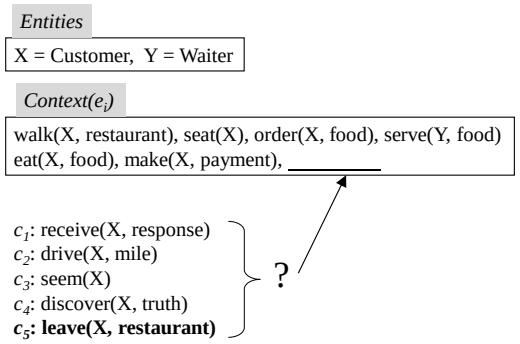


Figure 2: Multiple choice narrative cloze. The gold subsequent event is marked in bold.

Zaman et al., 2013; Abend et al., 2015), evaluated using different metrics. There has also been work using graph models to induce frames, which emphasize more on learning event structures and less on temporal orders (Chambers, 2013; Cheung et al., 2013). The above methods focus on one of the two subtasks we consider here. Frermann et al. (2014) used a Bayesian model to jointly cluster web collections of explicit event sequence and learn input event-pair temporal orders. However, their work is under a different input setting (Regneri et al., 2010), not learning event chains from texts. Mostafazadeh et al. (2016) proposed the story close task (SCT), which is to predict the ending given a unfinished story. Our narrative chain prediction task can be regarded as a sub task in the story close task, which can contribute as a major approach. On the other hand, information beyond event chains can be useful for the story close task.

## 3 Problem Definition

As shown in Figure 2, given a chain of narrative events $e_1$, $e_2$, ..., $e_{n-1}$, our work is to predict the likelihood of a next event candidate $e_n$. Formally, an event $e$ is a structure $v(a_0, a_1, a_2)$, where $v$ is a verb describing the event, $a_0$ and $a_1$ are its subject and direct object, respectively, and $a_2$ is a prepositional object. For example, given the sentence "John brought Marry to the restaurant", an event $bring\{John, Marry, to\ the\ restaurant\}$ can be extracted.

We follow the standard script induction setting (Chambers and Jurafsky, 2008; Granroth-Wilding and Clark, 2016), extracting events from a text corpus using a syntactic parser and a named entity resolver. A neural network is used to model chains of extracted events for script learning. In particular, we model the probability of a subsequent event given a chain of events. For evaluation, we solve the multi-choice narrative cloze task: given a chain of events and a set of candidate next events, the most likely candidate is chosen as the output.

## 4 Model

The overall structure of our model is shown in Figure 3, which has three main components. First, given an event $v(a_0, a_1, a_2)$, a representation layer is used to compose the embeddings of $v$, $a_0$, $a_1$, and $a_2$ into a single event vector $e$. Second, a LSTM is used to map a sequence of existing events $e_1$, $e_2$, ..., $e_{n-1}$ into a sequence of hidden vectors $h_1$, $h_2$, ..., $h_{n-1}$, which encode the temporal order. Given a next event candidate $e_c$, the recurrent network takes one further step from $h_{n-1}$ to derive its hidden vector $h_c$, which encodes $e_c$. Third, $h_c$ is paired with $h_1$, $h_2$, ..., $h_{n-1}$ individually, and passed to a dynamic memory network to learn the relatedness score $s$. $s$ is used to denote the connectedness between the candidate subsequent event and the context event chain.

### 4.1 Event Representation

We learn vector representations of standard events by composing pre-trained word embeddings of its verb and arguments. The skipgram model (Mikolov et al., 2013) is used to train word vectors. For arguments that consist of more than one word, we use the averaged word for the representation. OOV words are represented simply using zero vectors. For events with less than 3 arguments, such as "John fell", where $v$ = fall, $a_0$ = John, $a_1$ = NULL, and $a_2$ = NULL, the NULL arguments are represented using all-zero vectors.

Denoting the embeddings of $v$, $a_0$, $a_1$, and $a_2$ as $e(v)$, $e(a_0)$, $e(a_1)$, and $e(a_2)$, respectively, the embedding of $e$ is calculated using a $\tanh$ composition layer

$$\begin{aligned} e(e) = \tanh(&W_e^v \cdot e(v) + W_e^0 \cdot e(a_0) + \\ &W_e^1 \cdot e(a_1) + W_e^2 \cdot e(a_2) + b_e) \end{aligned} \quad (1)$$

Here $W_e^v$, $W_e^0$, $W_e^1$, $W_e^2$, and $b$ are model parameters, which are randomly initialized and tuned during the training of the main network.

### 4.2 Modeling Temporal Orders

Given the embeddings of the existing chain of events $e_1$, $e_2$, ..., $e_{n-1}$, we use a standard LSTM (Hochreiter and Schmidhuber, 1997) without

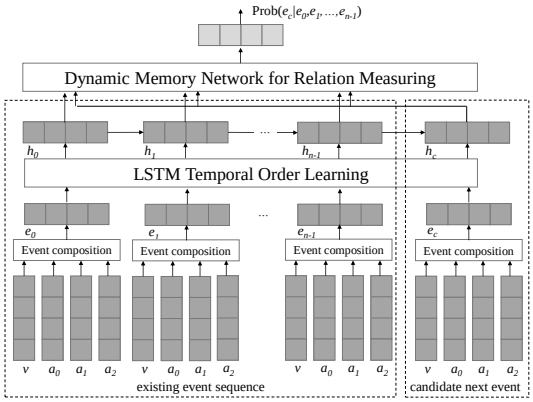


Figure 3: Overview of proposed model.

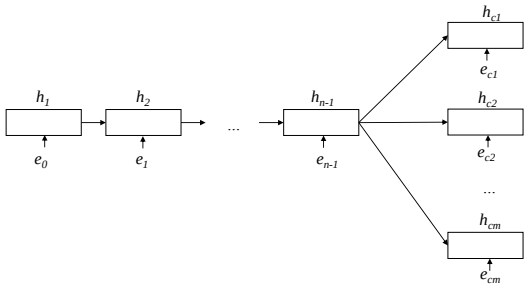

Figure 4: Temporal order modeling.

coupled input and forget gates or peephole connections to model the temporal order. We obtain a sequence of hidden state vectors $h_1$, $h_2$, ..., $h_{n-1}$ by recurrently feeding $e(e_1)$, $e(e_2)$, ..., $e(e_{n-1})$ as inputs to the LSTM, where $h_i = \text{LSTM}(e(e_i), h_{i-1})$. The initial state $h_s$ and all stand LSTM parameters are randomly initialized and tuned during training.

Now for each candidate next event $e_c$, we obtain its vector representation $e(e_c)$ in the same way as for $e_1$ to $e_{n-1}$. $e(e_c)$ is then appended to the existing event chain to obtain a temporal-order-sensitive feature vector $h_c$, by advancing the recurrent encoding process for one step from $h_{n-1}$: $h_c = \text{LSTM}(e(e_c), h_{n-1})$. With multiple next event candidates $e_c^1$, $e_c^2$, ..., $e_c^m$ ($m \in [1, \infty]$), $m$ feature vectors are obtained, as shown in Figure 4, each being used as a basis for estimating the probability of the corresponding event candidate.

### 4.3 Modeling Pairwise Event Relations

After obtaining the hidden states for events, we model event pair relations using these hidden state vectors. A straightforward approach to model the relation between two events is using a Siamese network (Granroth-Wilding and Clark, 2016). The order-sensitive LSTM features for existing events $h_1$, $h_2$, ..., $h_{n-1}$ and the candidate event $h_c$ are used as event representations. Given a pair of events $h_i$ ($i \in [1..n-1]$) and $h_c$, the relatedness score is calculated by

$$s_i = \text{sigmoid}(W_{si} h_i + W_{sc} h_c + b_s), \quad (2)$$

where $W_{si}$, $W_{sc}$ and $b_s$ are model parameters.

Given the relation score $s_i$ between $h_c$ and each existing event $h_i$, the likelihood of $e_c$ given $e_1$, $e_2$, ..., $e_{n-1}$ can be calculated as the average of $s_i$:

$$s = \frac{\sum_{i=1}^{n-1} s_i}{n-1} \quad (3)$$

**Weighting existing events.** The drawback of above approach is that it considers the contribution of each event on the chain is same. However, given a chain of existing events, some are more informative for inferring a subsequent event than others. For example, given the events "wait in queue", "getting seated" and "order food", "order food" is more relevant for inferring "eat food" compared with the other two given events. Given information over the full event chain, this link can be more evident since the scenario is likely restaurant visiting.

We use an attentional neural network to calculate the relative importance of each existing event according to the subsequent event candidate, using $h_i$ ($i \in [1..n-1]$) and $h_c$ for event representations:

$$u_i = \tanh(W_{ei} h_i + W_c h_c + b_u) \quad (4)$$

$$\alpha_i = \frac{\exp(u_i)}{\sum_j \exp(u_j)} \quad (5)$$

where $\alpha_i \in [0, 1]$ is the weight of $h_i$, and $\sum_i \alpha_i^t = 1$. $W_{ei}$, $W_c$, and $b_u$ are model parameters.

After obtaining the weight $\alpha_i$ of each existing event $h_i$, the relatedness of $e_c$ with the existing events can be calculated as:

$$s = \sum_{i=1}^{n-1} \alpha_i \cdot s_i \quad (6)$$

**Multi-layer attention using Deep memory network.** Memory network (Weston et al., 2014; Mikolov et al., 2014) has been used for exploring deep semantic information for semantic tasks. Such as question answering (Sukhbaatar et al., 2015; Kumar et al., 2016) and reading comprehension (Hermann et al., 2015; Weston et al., 2015). Our task is analogous to such semantic tasks in

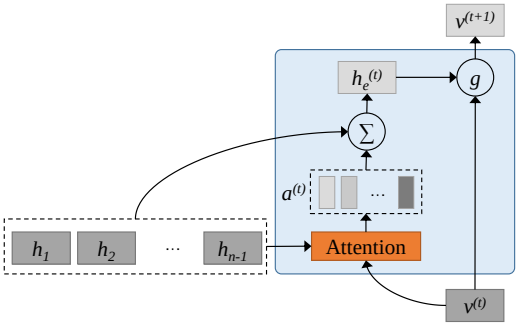


Figure 5: Memory network at hop $t$. $h_i$ is the hidden variable of the existing event chain, $v^t$ is the semantic representation between context events and candidate event. $a^t$ is the weight of context events, and $g$ is the gated recurrent network on Eq.10.

the sense that deep semantic information can be necessary for making the most rational inference. Hence, we are motivated to use a deep memory network model to refine event weight and event relation calculation by recurrently modeling more abstract representations of the scenario. Different from the previous researches, we use the memory network to model the event chain, refining the attention mechanism used to explore the pair-wise relation between events.

The memory model consists of multiple dynamic computational layers (hops). For the first layer (hop 1), the weights $\alpha$ for existing events $e_1$, $e_2$, ..., $e_{n-1}$ can be calculated using the same attention mechanism as Eq.4 and Eq.5. Given the weights $\alpha$, we build a consolidated representation of context event chain $e_1$, $e_2$, ..., $e_{n-1}$ as a weighted sum of $h_1$, $h_2$, ..., $h_{n-1}$:

$$h_e = \sum_{i-1}^{n-1} \alpha_i \cdot h_i \tag{7}$$

The event candidate $h_c$ and the new representation of the existing chain $h_e$ can be further integrated to deduce a deeper representation of the full event chain hypothesis to the next layer (hop 2), denoted as $v$. $v$ contains deeper semantic information compared with $h_c$, which encode the temporal order of the event chain $[h_1, h_2, ..., h_{n-1}, h_c]$ without differentiating the weights of each event. As a result, in the next hop, better event weights can potentially be deduced by using $v$ instead of $h_c$ in the calculation of attention:

$$u_i^t = \tanh(W_{ei} h_i + W_v v^t + b_u) \tag{8}$$

$$\alpha_i^t = \frac{\exp(u_i^t)}{\sum_j \exp(u_j^t)} \tag{9}$$

In the same way, we stack multiple hops and repeat the steps multiple times, so that more abstract evidences can be extracted according to the chain of existing events. The above process can be performed recurrently, by taking $h_c$ as an initial scenario representation $v_0$, and then repeatedly calculating $h_e^t$ given $h_1$, $h_2$, ..., $h_{n-1}$ and $v^t$, and using $h_e^t$ and $v_t$ to find a deeper scenario representation $v^{t+1}$. Following Chung et al. (2014) and Tran et al. (2016), a gated recurrent network is used to this end:

$$\begin{aligned} z &= \sigma(W_z h_e^t + U_z v^t) \\ r &= \sigma(W_r h_e^t + U_r v^t) \\ \hat{h} &= \tanh(W h_e^t + U(r \odot v^t)) \\ v^{t+1} &= (1 - z) \odot v^t + z \odot \hat{h} \end{aligned} \tag{10}$$

At any step, if the value of $|v^{t+1} - v^t|$ is less than the threshold $\mu$, we consider that the progress has reached convergence. Figure 5 shows an overview of the memory network at hop $t$.

### 4.4 Training

Given a set of event chains, each with a gold-standard subsequent event and a number of non-subsequent events, our training objective is to minimize the cross-entropy loss between the gold subsequent event and the set of non-subsequent events. The loss function of event chain prediction is that:

$$L(\Theta) = \sum_{i=1}^{N} (s_i - y_i)^2 + \frac{\lambda}{2} ||\Theta||^2 \tag{11}$$

where $s_i$ is the relation score, $y_i$ is the label of the candidate ($y_i = 1$ for positive sample, and $y_i = 0$ for negative sample), $\Theta$ is the set of model parameters and $\lambda$ is a parameter for L2 regularization. We apply online training, where model parameters are optimized by using AdaGrad (Duchi et al., 2011). We train word embedding using the *Skip-gram* algorithm (Mikolov et al., 2013)[2].

## 5 Experiments

### 5.1 Datasets

Following Granroth-Wilding and Clark (2016), we extract events from the NYT portion of the Gigaword corpus (Graff et al., 2003). The C&C

[2]https://code.google.com/p/word2vec/

tools (Curran et al., 2007) are used for POS tagging and dependency parsing, and OpenNLP[3] for phrase structure parsing and coreference resolution. The training set consists of 1,500,000 event chains. We follow Granroth-Wilding and Clark (2016) and use 10,000 event chains as the test set, and 1,000 event chains for development. There are 5 choices of output event for event input chain, which are given by Granroth-Wilding and Clark (2016). This dataset is referred to as **G&C16**.

We also adapt the Chambers and Jurafsky (2008)'s dataset to the multiple choice setting, and use this dataset as the second benchmark. The dataset contains 69 documents, with 346 multiple choice event chain samples. We randomly sample 4 negative subsequent events for each event chain to make multiple-choice candidates. This dataset is referred to as **C&J08**. For both datasets, accuracy (Acc.) of the chosen subsequent event is used to measure the performance of our model.

### 5.2 Hyper-parameters

There are several important hyper-parameters in our models, and we tune their values using the development dataset. We set the regularization weight $\lambda = 10^{-8}$ and the initial learning rate to 0.01. The size of word vectors is set to 300, and the size of hidden vectors in LSTM to 128. In order to avoid over-fitting, dropout (Hinton et al., 2012) is used for word embedding with a ratio of 0.2. The neighbor similarity threshold $\eta$ is set to 0.25. The threshold $\mu$ of the memory network sets to 0.1.

### 5.3 Development Experiments

We conduct a set of development experiments on the G&C16 development set to study the influence of event argument representations and network configurations of the proposed *MemNet* model.

#### 5.3.1 Influence of Event Structure

Existing literature discussed various structures to denote events, such as $v(a_0, a_1)$ and $v(a_0, a_1, a_2)$. We investigate the influence of integrating argument values of the subject $a_0$, object $a_1$ and preposition $a_2$, by doing ablation experiments on the development data. The results are shown in Table 1, where the system using all arguments gives a 54.36% accuracy. By removing $a_2$, which exists in 17.6% of the events in our developmental data, the accuracy drops to 54.02%. In contrast, by removing $a_0$ and $a_1$, which exist in 87.6% and 64.6% of the events in the development data, respectively, the accuracies drop to 53.43% and 53.57%, respectively, which demonstrates the relative importance of $a_0$ (i.e., the subject) and $a_1$ (i.e., the object) for event modelling. While most previous work (Chambers and Jurafsky, 2008; Balasubramanian et al., 2013; Pichotta and Mooney, 2014) modelled only $a_0$ and $a_1$, recent work (Pichotta and Mooney, 2016; Granroth-Wilding and Clark, 2016) modelled $a_2$ also.

| Method | Acc. (%) |
|---|---|
| MemNet | **54.36** |
| -$verb$ | 42.63 |
| -$(a_0, a_1)$ | 52.32 |
| -$(a_0)$ | 53.43 |
| -$(a_1)$ | 53.57 |
| -$(a_2)$ | 54.02 |

Table 1: Influence of event arguments.

By removing both $a_1$ and $a_2$, the accuracy drops further to 53.32%. Interestingly, by removing the verb while keeping only the arguments, the accuracy drops to 42.63%. While this demonstrates the central value of the verb in denoting a event, it also suggests that the arguments themselves play a useful role in inferring the stereotypical scenario.

#### 5.3.2 Influence of Network Configurations

We study the influence of various network configurations by performing ablation experiments, as shown in Table 2. *MemNet* is the full model of this paper; *-LSTM* denotes ablation of the LSTM layer, using $e(e_1)$, $e(e_2)$, ..., $e(e_{n-1})$ instead of $h_1$, $h_2$, ..., $h_{n-1}$ to represent events; *-Hop* denotes ablation of the dynamic network model, using only attention mechanism to calculate the weights of each existing event; *-Attention* denotes ablation of the attention mechanism, using the same weight on each existing event when inferring $e_c$. The model "*-Attention, -LSTM*" is hence similar to the method of Granroth-Wilding and Clark (2016), although we used a different way of deriving event embeddings. The model "*LSTM-only*" shows a based by using LSTM hidden vector $h_{n-1}$ to directly predict the next event, which is similar to the method of Pichotta and Mooney (2016).

**Influence of Temporal Order.** By comparing "MemNet" and "-LSTM", and comparing "-

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

| Method | Acc. (%) |
|---|---|
| MemNet | **54.36** |
| -Hop | 52.03 |
| -Attention | 50.76 |
| -LSTM | 51.72 |
| -Hop,-LSTM | 50.65 |
| -Attention,-LSTM | 48.26 |
| LSTM-Only | 46.72 |

Table 2: Analysis of network structure.

Attention" with "-Attention, -LSTM", one can find that temporal order information over the whole event chain does have significant influence on the results ($p - value < 0.01$ using $t$-test). On the other hand, using LSTM to directly predict the subsequent event ("LSTM-only") does not give better accuracies compared to model event pairs ("-Attention, -LSTM"). This confirms our intuition that strong-oder modelling and event-pair modelling each have their own strength.

**Influence of Attention.** Comparison between "-Attention" and "-Hop", and between "-Attention, -LSTM" and "-Hop, -LSTM" shows that giving different weights to different events does lead to improving results. Our analysis in Section 4.3 gives more intuitions to this observation. Finally, comparison between "-Hop" and "MemNet" and between "-Hop, -LSTM" and "-LSTM" shows that a multi-hop deep memory network can indeed enhance the model with single level attention by offering more effective semantic representation of the scenarios.

### 5.4 Final Results

Table 3 shows the final results on the C&C 16 and C&J08 datasets, respectively. We compare the results of our final model with the following baselines:

- **PMI** is the co-occurrence based model of Chambers and Jurafsky (2008), who calculate event pair relations based on Pointwise Mutual Information (PMI), scoring each candidate event $e_c$ by the sum of PMI scores between the given events $e_0$, $e_1$, ..., $e_{n-1}$ and the candidate.

- **Bigram** is the counting based model of Jans et al. (2012), calculating event pair relations based on skip bigram probabilities, trained using maximum likelihood estimation.

| Method | G&C16 | C&J08 |
|---|---|---|
| PMI | 30.52 | 30.92 |
| Bigram | 29.67 | 25.43 |
| Event-Comp | 49.57 | 43.28 |
| RNN | 45.74 | 43.17 |
| MemNet | **55.12** | **46.67** |

Table 3: Final results.

- **Event-Comp** is the neural event relation model proposed by Granroth-Wilding and Clark (2016). They learn event representations by calculating pair-wise event scores using a Siamese network.

- **RNN** is the method of Pichotta and Mooney (2016), who model event chains by directly using $h_c$ in Section 4.2 to predict the output, rather than taking them as features for event pair relation modeling.

- **MemNet** is the proposed deep memory network model.

Our reimplementation of PMI and Bigrams follows (Granroth-Wilding and Clark, 2016). It can be seen from the table that the statistical counting-based models PMI and Bigram significantly underperform the neural network models Event-Comp, RNN and MemNet, which is largely due to their sparsity and lack of semantic representation power. Under our event representation, Bigram does not outperform PMI significantly either, although considering the order of event pairs. This is likely due to sparsity of events when all arguments are considered.

Direct comparison between Event-Comp and RNN shows that the event-pair model gives comparable results to the strong-order LSTM model. Although Granroth-Wilding and Clark (2016) and Pichotta and Mooney (2016) both compared with statistical baselines, they did not make direct comparisons between their methods, which represent two different approaches to the task. Our results show that they each have their unique advantages, which confirm our intuition in the introduction. By considering both pairwise relations and chain temporal orders, our method significantly outperform both Event-Comp and RNN ($p - value < 0.01$ using $t$-test), giving the best reported results on both datasets.

## 6 Conclusion

We proposed a dynamic memory network to integrate chain order information into event relation measuring, calculating event pair relations by representing events in a chain using LSTM hidden states, which encode temporal orders, and using a dynamic memory model to automatically induce event weights for each event. Standard evaluation showed that our method significantly outperforms state-of-the-art event pair models and event chain models, giving the best results reported so far.

## Acknowledgments

The corresponding author is Yue Zhang. We are grateful for the help of Fei Dong for his initial discussion. We thank our anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by the Temasek Lab grant IGDST1403012 at Singapore University of Technology and Design.

## References

Omri Abend, Shay B. Cohen, and Mark Steedman. 2015. Lexical event ordering with an edge-factored model. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1161–1171.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1721–1731.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1797–1807.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 789–797.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 837–846.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

James R. Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with c&c and boxer. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Charles Fillmore. 1982. Frame semantics. *Linguistics in the morning calm*, pages 111–137.

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 49–57.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2727–2733.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Bram Jans, Steven Bethard, Ivan Vulic, and Marie-Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *EACL 2012, 13th Conference of the European Chapter of*

*the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*, pages 336–344.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1378–1387.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 523–534.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michaël Mathieu, and Marc'Aurelio Ranzato. 2014. Learning longer memory in recurrent neural networks. *CoRR*, abs/1412.7753.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Marvin Minsky. 1975. A framework for representing knowledge.

Andriy Mnih and Geoffrey E. Hinton. 2007. Three new graphical models for statistical language modelling. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 641–648.

Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 75–83.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*, pages 49–57.

Raymond Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. *Urbana*, 51:61801.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 839–849.

John Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, and Thomas G. Dietterich. 2014. Learning scripts as hidden markov models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1565–1571.

Karl Pichotta and Raymond J. Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 220–229.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2800–2806.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 979–988.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1681–1686.

David E Rumelhart. 1975. Notes on a schema for stories. *Representation and understanding: Studies in cognitive science*, 211(236):45.

Roger Schank, Roger Schank, and Robert P Abelson. 1977. Scripts, plans, goals and understanding; an inquiry into human knowledge structures. Technical report.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Beth Sundheim. 1991. Third message understanding evaluation and conference (muc-3): Phase 1 status report. In *HLT*.

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language

modeling. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 321–331.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James F. Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*, pages 1–9.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.