Deep Learning-Based Event Prediction for Text Analysis

1st Muhammad Waseem
School of Computer Science and Technology
Beijing Institute of Technology
Beijing, China.
waseempk.csc@yahoo.com

2nd Qasim Umer
Department of Computer Science
Hanyang University
Seoul, South Korea.
qasimumer@hanyang.ac.kr

3rd Choonhwa Lee
Department of Computer Science
Hanyang University
Seoul, South Korea.
lee@hanyang.ac.kr

4th Sungwook Chung
Department of Computer Engineering
Changwon University
Changwon, South Korea.
swchung@changwon.ac.kr

5th Zohaib Latif
Department of Computer Science
Nazarbayev University
Astana, Kazakhstan.
latif.zohaib@nu.edu.kz

***Abstract*—This paper addresses automatic event prediction from unstructured text, specifically event chains. While current approaches employ LSTM for encoding full chains, learning long-range narrative orders, or learning partial orders and long-range narrative orders, none of them consider writer sentiment. To address this, we propose a deep learning-based approach that incorporates writer sentiment. We preprocess the text, extract events, compute sentiment scores using *SentiWordNet*, convert events to digital vectors, and feed them along with sentiment scores into a deep learning-based classifier. This classifier uses hidden states for event pair modeling, with each pair having its associated sentiment. Evaluation results show that our approach significantly surpasses state-of-the-art methods with 29.2% accuracy.**

***Index Terms*—Event Prediction, Deep Learning, Sentiment**

I. Introduction

It is critical to several artificial intelligence applications (i.e., intention recognition, discourse understanding, and dialogue generation) to understand/identify events described in a large text. To encode information into natural language, we usually think that the reader can effortlessly make inferences based on commonsense knowledge and ignore the sentiment involved in the information. Take the event "Inam entered a restaurant" as an example, which can lead to numerous possible inferences. These inferences may include scenarios such as Inam having to wait in a long line, being seated at a table, reading the menu, placing an order, receiving the food, and so forth. Such assumptions based on commonsense knowledge are called *Scripts* which is intuitive for readers and commonly lost in context [1]. Therefore, an automatic understanding of the text for machines and predicting the events from scripts is a very challenging task because machines have no extra knowledge to understand the events in a chain [2].

Early scripts are manually extracted for a specific purpose and it is time-consuming and difficult to construct for long text. To automate the manual task, a number of following researches have been conducted. For example, Chambers and Jurafsky [3] automatically extracted the knowledge of event sequences from the text using coreference resolution that provides the source of information about an entity. Jans et al. [4] improved the method of Pointwise Mutual Information (PMI) using the skip bi-gram probabilities to calculate the relationship between the events. Other researches [5]–[7] further exploited the improved PMI method to induce the chain of events from the text.

Furthermore, neural network models [2], [4], [6], [8]–[10] are recently used for the modeling of event sequences. These neural network approaches are reliable with the earlier statistical models to exploit the relationship between a pair of events. To improve the accuracy of the neural network, Pichotta and Mooney [5] adopted the Long Short-Term Memory (LSTM) [11] to model the sequence of events that gathers much more information of order in contrast to other researches [2], [6], [9]. Later, Wang et al. [12] exploited the LSTM model for the sequence of narrative events and used a dynamic memory network [13] to model relationships between event pairs because LSTM model suffers from a redundancy problem. In contrast to Wang et al. [12], previous approaches [2], [3], [9] give equal weights to events that exist in a text. Shangwen et al. [14] developed a self-attention mechanism to focus on various parts of events in the chain, and the chain of events is presented as a set of event segments. Although

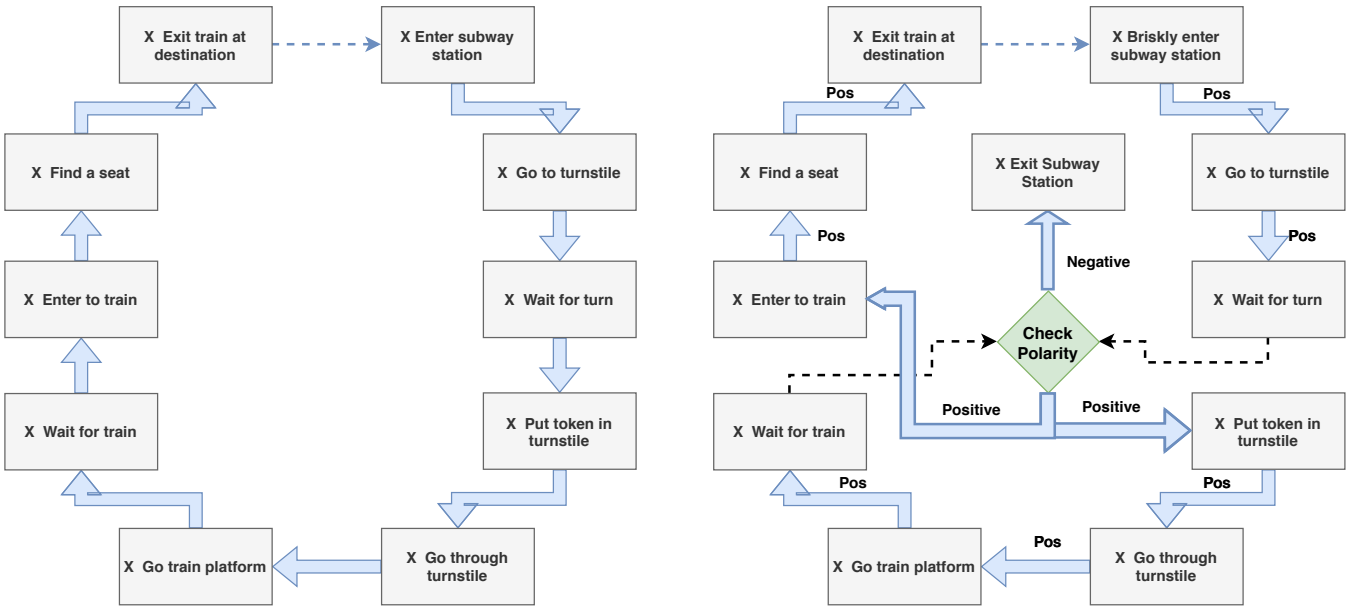

Fig. 1. Event Sequence for a Restaurant Visit

there are a number of approaches for event prediction, none of them considers the sentiment within the text for event prediction. The importance of the sentiment within the text is illustrated in Fig. 1.

To this end, we propose a Deep Learning-based Event Predictive Model (DLEPM) for a narrative production system. First, we leverage natural language techniques for text preprocessing on the gigaword NYT corpus. Second, we extract the events using the state-of-the-art model. Third, we leverage *SentiWordNet* to compute the sentiment score of each event as the author's emotion within an event may reflect the subsequent events. Fourth, we leverage *word2vec* to convert the events into a digital vector. Finally, we pass the digital vector and emotion score of each event to a deep-learning classifier to predict the subsequent events. The evaluation results suggest that DLPM surpasses the state-of-the-art and improves the accuracy score by more than 26 percent.

II. Related Work

Scripts have long been a focal point of AI research [1]. Traditionally, these scripts involve manually encoding the sequence of events within knowledge bases, which can then be utilized for various tasks such as inference. The concept of scripts is also interconnected with studies in linguistics and psychology, occasionally referred to as frameworks (Fillmore, 1982). The researchers [1], [15] proposed "script theory" as a partial solution to the problem of illustration. They suggest that some of our knowledge about hundreds of stereotypical attitudes is organized with routine activities. Chambers and Jurafsky [3] pioneered work in script induction [4], [9], [12], [16]. However, Shangwen et al [14] focused on the modeling of narrative event chains

A. Representation of Scripts

In the realm of event representation, Chambers and Jurafsky defined narrative events as threefold of the "no" model dependency [3]. They employed a syntactic parser and coreference analyzer to organize narrative chains centered around the main actor or protagonist, extracting events that share a common protagonist in the text. They referred to these hierarchies of participants and events as scripts [1] or Fillmore frameworks [17]. Jans et al. adopted a similar simplistic event representation but introduced a new model that yields more accurate predictions on test data [4]. These methods solely focus on modeling the actions of a single participant, specifically referred to as the protagonist.

Chambers and Jurafsky extended their approach to encompass multiple participants by modeling events involving all entities within a document [10]. However, their approach falls short in capturing the interactions between these entities. Balasubramanian et al. highlighted potential weaknesses in representing event chains solely from the perspective of the protagonist and proposed an alternative representation using <arg1, relationship, arg2>, where arg1 and arg2 denote the subject and object, respectively [18]. This alternative approach addresses the limitations and inconsistencies found in the protagonist-centric representation. This representation was inspired by the extraction of open information [19] and provides

more comprehensive functionality for pairs of modeling events. Pichotta and Mooney [5] adopted a similar idea and used a tuple verb (subject, object, entity) to represent an event. A multi-argument event model encodes more representation of the richness of events. They have proven empirically the usefulness of multi arguments in the event. Its representation is used when adding multiple arguments to an event by subsequent work such as Modi el al. [2] and Granroth-Wilding and Clark [9]. Wang et al. [12] follow Pichotta and Mooney [20] to represent their events. We also follow the Pichotta and Mooney [20] in our event representation model with several arguments including adverbs to represent the event. We consider the adverb in event representation because we have to apply sentiment analysis to events.

B. Modeling od Scripts

In the context of scripting modeling, current methods (low-order and high-order models) calculate relationships between event pairs and consider the chronological order of events. However, the previous work used separate representations of events and the estimated relationships of events by statistical counting, e.g., the researchers [3]–[6] exploited PMI, skip bi-gram probabilities, and skip n-grams to calculate event relationships. Being multi-argument structures, count-based methods can suffer from scarcity problems. To resolve this problem, Rudinger et al. [6] learned to integrate events as a byproduct to form a logarithmic language model of events.

Granroth-Wilding and Clark [9] benefited from Mikolov et al. [8] to form a superposition of events and arguments in a pseudo sentence. Modi et al. [2] used the embedding of words to merge the verbs and arguments directly, using a hidden layer to automatically integrate word combinations into structured event vaccinations. Wang et al. [12] followed Modi et al. [2] and use a hidden layer to learn the composition of event arguments by word embedding, thus configuring the composition function in the learning process in the chain of events. By alleviating the problem of the under-representation of events, neural methods can capture temporal order between events that go beyond n-gram omission. Wang et al. [12] integrate the benefits of learning through strong ordering and learning event pairs using hidden LSTM states as a representation of the event properties present in the calculation of event pair relationships. In addition, they use a memory network model to weigh current events, to get better results on the equal weighting method of existing models. However, Wang et al. [12] and Shangwen et al. [14] noted that there is an event-rich relationship chain in the chain of events that can provide benefits for predicting the next event. Consequently, they develop a self-attention mechanism to extract events from different segments and represent the chain of events as a combination of event segments.

Hu et al. [21] proposed an LSTM hierarchical model that incorporates word sequences and event sequences to predict what will happen next. Regneri el al. [22] compiled natural language descriptions of specific event sequences for volunteer texts and build a temporal script graph. Orr et al. [7] use hidden Markov models to create a transition graph and make script conclusions based on the probability of state transition. Lee et al. [23] designed a narrative event evolutionary graph (NEEG) to represent the rich relationships between events and provided a scaled-graph neural network (SGNN) to model interactions between events and knowledge of the representation of events.

C. Evaluation of Scripts

Chambers and Jurafsky [3] proposed the Narrative close task, which requests a missing event in a series of gap-related events. This task was adopted by several subsequent works [4]–[6] to compare the results with Chambers and Jurafsky [3]. One issue of the narrative cloze test is that sometimes there may be many sensible answers, but there is only one standard answer, which can make the evaluation of system results manually expensive. To solve this problem, Modi et al. [2] proposed an Adversarial Narrative Cloze (ANC), consisting of a series of pairs of real and corrupt events. Granroth-Wilding and Clark [9] proposed a Multi-CNC (MCNC), which involves selecting the next event most likely from a group of candidates from a series of events. Wang et al. [12] and Shangwin et al. [14] both chose MCNC to compare different models. We also choose MCNC to measure our assessment and compare the impact of different models to predict the script event. Although there are a number of available approaches for event prediction, DLEPM differs in that it considers the sentiment within the text to improve the accuracy of event prediction.

III. Approach

A. Extraction of Event Chains

DLEPM is inspired by the existing work [3], [9], [10]. We utilize the New York Times (NYT) segment of the Gigaword corpus[1] to extract events. Following the methodology of Chambers and Jurafsky [10], we employ Stanford CoreNLP for Part-of-Speech (POS) tagging, dependency parsing, and co-reference resolution. Similar to the approach taken by Mark Granroth-Wilding and Stephen Clark [9], we extract the predicate adjective, lemmatized verb, and their dependency relationship. We exclude stop-events that have English stop-words as predicates. Note that we used *nyt_eng* documents for our experiments.

[1]https://catalog.ldc.upenn.edu/LDC2003T05

B. Baseline Models

To compare the performance of DLEPM, we extract the event chain using state-of-the-art models.

1) Chambers & Jurafsky 08: We first consider the C&J08 model. It represents each event e by its predicate $p(e)$ and the dependency relation $d(e)$ to entity e of the chain, which can be represented as $pg(e) = (p(e), d(e))$. For example, C&J08 extracts the event chain from the text "*Robbers made a big score, fleeing after stealing more than $300,000 ... a gun approached and ordered them to lie down ...* " as *(service, subj), (report, subj), (put, subj), (lie_down, subj), ...*, where entities and predicates in the given text are *Wells Fargo armored-truck guards, ...* and *service(x_0, ATMs), report(x_0), ...*, respectively.

For each given pair of events, we use multiple choice narrative cloze (PPMI) as a relatedness score and compute the co-occurrence in the same chains while training. To predict the next event ne, the score of the event $s(ne)$ is computed by adding its PPMI with the n events that can be defined as

$$s(ne) = \sum_{j=0}^{n-1} \text{ppmi}\left(pg(ne), pg\left(e_j\right)\right) \tag{1}$$

2) Bigram: To find out the probability of each predicate, we exploit *bigram* [4] that considers maximum likelihood. We exploit *bigram* as its performance against narrative cloze is significant rather than C&J08. However, we employ Laplace smoothing to compute the unigram probabilities for the unobserved event ne that can be defined as

$$s(ne) = \frac{1}{n} \sum_{j=0}^{n-1} P\left(pg(ne) | pg\left(e_j\right)\right) \tag{2}$$

3) Distributional Vectors: We exploit latent semantic indexing (LSI) [24] for the representation of events as a vector space model to assign a score to the individual event and to overcome the limitation (assigning a score to pairs of events) of C&J08. We create a matrix to compute the frequency count of predicates in which each row represents a predicate and each cell represents the frequency count of the predicate. We set the dimensionality to 300 for the dense representation of each predicate. We compute the cosine similarity of the predicate of the next event ne that can be defined as

$$s(ne) = \text{cosine}\left(\sum_{j=0}^{n-1} S_{pg(e_j)}, S_{pg(ne)}\right) \tag{3}$$

4) Word2vec Representations: We exploit *word2vec* [8], [25] to convert the predicate information into digital vectors. We generate digital vectors from *verb only* and *verb+argument*. *word2vec* is introduced for efficient learning embeddings for deep learning models. Note that, we use the default dimension

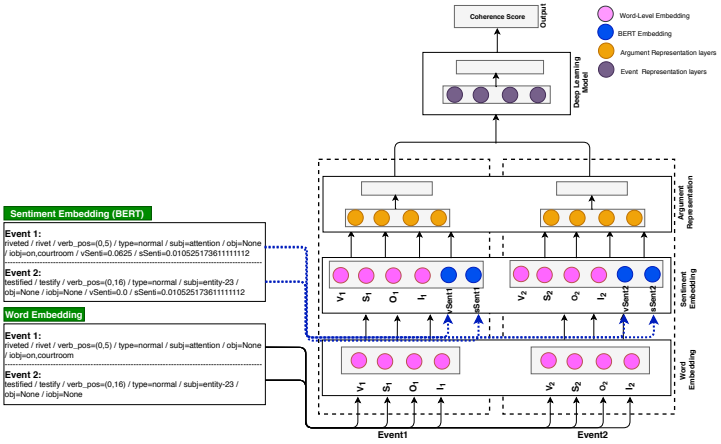

Fig. 2. Overview of DLEPM

300 for the vectors. Such embeddings provide the relationship between two events. The *verb only* model (noted as *Mikolov-Verb*) only considers verb of the event ne for the representation, adds the given events w, and computes the cosine similarity that can be defined as

$$s(ne) = \text{cosine}\left(\sum_{j=0}^{n-1} w_{p(e_j)}, w_{p(ne)}\right) \tag{4}$$

In contrast to *Mikolov-Verb*, the *verb+argument* model (noted as Mikolov-Verb-Arg) add the vectors from w for its verb and each of its argument for the representation of ne.

$$s(ne) = \text{cosine}\left(\sum_{j=0}^{n-1} w_{p(e_j)} + w_{a0(e_j)} + w_{a1(e_j)} + w_{a2(e_j)} \; W_{p(ne)} + W_{a0(ne)} + W_{a1(ne)} + W_{a2(ne)}\right) \tag{5}$$

We also learn embeddings from event chains in which vectors of events that occur in similar chains are close together. We exploit a *skipgram* model (noted as *W2V-Pred*) with settings: *window size = 5* and *vector size = 300*. Furthermore, we add argument words for vector representation (noted as W2V-Pred-Arg) with settings: *window size = 15* and *vector size = 300*. Thus, the event representation ne can be computed by adding the vectors of the predicate and each of the arguments that can be defined as

$$s(ne) = \text{cosine}\left(\sum_{j=0}^{n-1} W_{p(e_j):d(e_j)} + W_{\text{arg:a0 }(e_j)} + W_{\text{arg:a1 }(e_j)} + W_{\text{arg:a2 }(e_j)} \; W_{p(ne):d(ne)} + W_{\text{arg:a0 }(ne)} + W_{\text{arg:a1 }(ne)} + W_{\text{arg }a2(ne)}\right) \tag{6}$$

5) Event Compositional Representations: We exploit *W2V-Pred-Arg* to learn the embeddings. In training,

both word vectors and argument composition *Event-Comp* function weights are modified for a pair of events but their parameters are tied. Therefore, we learn the single event representation that minimizes the object function and can be defined as

$$\frac{1}{m}\sum_{i=1}^{m} -\log\left(p_i \times \text{coh}\left(e0_i, e1_i\right) + \right. \\ \left. (1-p_i) \times (1-\text{coh}\left(e0_i, e1_i\right)\right) + \lambda L(\theta) \tag{7}$$

where, pi = 1 represents positive next event and 0 for negative, $coh(e0_i, e1_i)$ represents the output of the model for the i^{th} training pair and $L(\theta)$ represents the regularization-term on all weights.

C. Deep Learning-Based Event Prediction Model (DLEPM)

1) Sentiment Based Event Compositional Representations: An overview of DLEPM is shown in Fig. 2. We train a deep learning-based network (*Senti-Event-Comp*) to learn the composition of predicates, the sentiment of the predicates, and arguments for an event representation. We exploit the *W2V-Pred-Arg* that contains the large vocabulary of word vectors corresponding to words of predicate and argument. We concatenate the vectors (computed from *word2vec*) related to the positions of the predicate and argument of an event, and the sentiment (a scalar vector) that is computed from *SentiWordNet* using the predicate of the event. We input the concatenated vectors as input to the first layer. Note that, we use zero vector for empty arguments and non-vocabulary words. We train the model by passing two events at the same time to the layers, where each middle layer has a *tanh* (activation function), whereas the final layer has a *sigmoid* (activation function). The final layer generates a coherence score (coh). It defines whether the given pair of events are from the same chain or not. It can be defined as

$$\frac{1}{m}\sum_{i=1}^{m} -\log\left(p_i \times \text{coh}\left(e0_i, e1_i\right) + \right. \\ \left. (1-p_i) \times (1-\text{coh}\left(e0_i, e1_i\right)\right) + \lambda L(\theta) \tag{8}$$

where, $pi = 1$ represents the positive event and 0 for the negative (Note that, training of the model needs positive and negative events for pairs of events. To make a positive pair, we randomly select $e1_i$ randomly from the same event chain, whereas we select $e1_i$ randomly from the other event chain), $coh(e0_i, e1_i)$ represents the output of the i^{th} training pair, $L(\theta)$ represents the regularization term. Note that, a pair of input events can be represented as $(\text{v}_1, s_1, o_1, i_i, V_{sen_1}, S_{sen_1}, v_2, s_2, o_2, i_2, V_{sen_2}, S_{sen_2})$.

For the parameter setting of the model, we set *dimension = 300* for *word2vec*, *400 and 200* as sizes of two hidden layers in *Senti-Event-Comp*, *dropout = 0.2*, *epoch = 8*, *learning-rate = 0.01*, λ = *0.01*, and *batch-size = 1000*.

2) Prediction: DLEPM predicts the next events ne using the average of the pairwise scores with the event as follows:

$$s(ne) = \frac{1}{n}\sum_{j=0}^{n-1} \text{coh}\left(ne, e_j\right) \tag{9}$$

TABLE I
THE PERFORMANCE OF DLEPM

Model	Accuracy
Chance Baseline	20.0%
C&J08	22.2%
Bigram	23.9%
Dist-Vecs	23.6%
W2V-Pred	23.6%
Event-Comp	26.9%
DLEPM	29.2%

IV. EVALUATION

A. Results

Table I presents the performance of each of the comparison models on the given testing events. Note that we only used 1 file from 195 data files of *nyt_eng* for our initial experiment. The selected file contains 17,593 files and the data in the file is arranged by day and month of the year 1994.

The results suggest that *C&J08* and *bigram* are better than the chance baseline, however, the improvement is not significant. Moreover, although the performances of *Dist-Vecs* and the models related to the *Mikolov* embeddings are better than the chance baseline, it does not catch the performance of *C&J08*.

Furthermore, the performance of the models that exploit vector representation generated by *word2vec* in contrast to *Mikolov* embeddings, is significantly better than *C&J08*, *Dist-Vecs*, *Mikolov-Verb*, and *Mikolov-Verb-Arg*. In contrast to simple vector representation generated by *word2vec*, a complex composition of arguments and predicates *Event-Comp* achieves further improvement in accuracy as compared other existing models.

However, DLEPM achieves significant improvement in contrast to the best existing model *Event-Comp*. The improvement in accuracy is up to 8.55% = (29.2% - 26.9%) /26.9%.

Based on the preceding analysis, we conclude that the introduction of the sentiment of events significantly increases the performance.

V. CONCLUSION

For the prediction of the next event from the text, we propose a Deep Learning-based Event Prediction Model (DLEPM). We consider *MCNC* task for the comparison of DLEPM with state-of-the-art models.

We notice that although *word2vec* embeddings can be used for the representations of the events for the improvement in accuracy, the sentiment of the events has a significant impact on performance improvement. We input the word embeddings and sentiment of events to learn the coherence function and composition function that is based on sentiment, predicate, and argument for training. The sentiment value of the events used in DLEPM increases the ability to capture the non-linear interactions between predicates and arguments.

In future, we would like to investigate the performance of the mentioned methods by incorporating the POS tagging information in the input as POS tags have the structural information of the text. We are interested to combine the semantic information *word2vec* and the syntactic information *POS* tags for the event prediction.

ACKNOWLEDGMENT

This research was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-00590, Decentralized High Performance Consensus for Large-Scale Blockchains) and the Korea Meteorological Administration Research and Development Program under Grant KMI 2021-01310.

REFERENCES

[1] R. C. Schank and R. P. Abelson, *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. Hillsdale, NJ: L. Erlbaum, 1977.

[2] A. Modi, I. Titov, V. Demberg, A. Sayeed, and M. Pinkal, "Modeling semantic expectation: Using script knowledge for referent prediction," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 31–44, 2017.

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

[4] B. Jans, S. Bethard, I. Vulić, and M. F. Moens, "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*. Association for Computational Linguistics, 2012, pp. 336–344.

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

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

[7] J. Walker Orr, P. Tadepalli, J. Rao Doppa, X. Fern, and T. G. Dietterich, "Learning scripts as hidden markov models," *arXiv preprint arXiv:1809.03680*, 2018.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

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

[10] N. Chambers and D. Jurafsky, "Unsupervised learning of narrative schemas and their participants," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, 2009, pp. 602–610.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[13] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov, "Towards ai-complete question answering: A set of prerequisite toy tasks," *arXiv preprint arXiv:1502.05698*, 2015.

[14] S. Lv, W. Qian, L. Huang, J. Han, and S. Hu, "Sam-net: Integrating event-level and chain-level attentions to predict what happens next," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6802–6809.

[15] R. Grishman and B. M. Sundheim, "Message understanding conference-6: A brief history," in *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996.

[16] K. Pichotta and R. J. Mooney, "Learning statistical scripts with lstm recurrent neural networks," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[17] C. J. Fillmore, "Frame semantics: Linguistics in the morning calm: Selected papers from the sicol," 1982.

[18] N. Balasubramanian, S. Soderland, O. Etzioni *et al.*, "Generating coherent event schemas at scale," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1721–1731.

[19] A. Ritter, O. Etzioni, and S. Clark, "Open domain event extraction from twitter," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1104–1112.

[20] K. Pichotta and R. J. Mooney, "Using sentence-level lstm language models for script inference," *arXiv preprint arXiv:1604.02993*, 2016.

[21] L. Hu, J. Li, L. Nie, X.-L. Li, and C. Shao, "What happens next? future subevent prediction using contextual hierarchical lstm," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[22] M. Regneri, A. Koller, and M. Pinkal, "Learning script knowledge with web experiments," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 979–988.

[23] A. Leeuwenberg and M.-F. Moens, "A survey on temporal reasoning for temporal information extraction from text," *Journal of Artificial Intelligence Research*, vol. 66, pp. 341–380, 2019.

[24] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.