Event Embeddings for Semantic Script Modeling

Ashutosh Modi
Saarland University,
Germany
ashutosh@coli.uni-sb.de

Abstract

Semantic scripts is a conceptual representation which defines how events are organized into higher level activities. Practically all the previous approaches to inducing script knowledge from text relied on count-based techniques (e.g., generative models) and have not attempted to compositionally model events. In this work, we introduce a neural network model which relies on distributed compositional representations of events. The model captures statistical dependencies between events in a scenario, overcomes some of the shortcomings of previous approaches (e.g., by more effectively dealing with data sparsity) and outperforms count-based counterparts on the narrative cloze task.

1 Introduction

It is generally believed that the lack of knowledge on how individual events are organized into higher-level scenarios is one of the major obstacles for natural language understanding. Texts often do not provide a detailed specification of underlying events as writers rely on the ability of humans to read between the lines or, more specifically, on their common sense knowledge of underlying scenarios. For example, going to a restaurant involves entering the restaurant, getting seated, making an order and so on. Consequently, when describing a visit to a restaurant, a writer will not specify all the events, as they are obvious to the reader. This kind of knowledge is typically refereed to as semantic scripts (Schank and Abelson, 1977), and, in this work, will aim to capture some aspects of this knowledge within our probabilistic model.

Early work on scripts focused on manual construction of knowledge bases and rule-based systems for inference using these knowledge bases (Schank and Abelson, 1977). More recent approaches relied on automatically learning script knowledge either from crowd-sourced or naturally-occurring texts (Chambers and Jurafsky, 2008; Regneri et al., 2010; Modi and Titov, 2014; Frermann et al., 2014; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015a).

Most of these methods represent events as verbal predicates along with tuples of their immediate arguments (i.e. syntactic dependents of the predicate). These approaches model statistical dependencies between events (or, more formally, mentions of events) in a document, often restricting their model to capturing dependencies only between events sharing at least one entity (a common protagonist). We generally follow this tradition in our approach.

Much of this previous work has focused on count-based techniques using, for example, either the generative framework (Frermann et al., 2014) or relying on information-theoretic measures such as pointwise mutual information (PMI) (Chambers and Jurafsky, 2008). Some of these techniques treat predicate-argument structures as an atomic whole (e.g., Pichotta and Mooney (2014)), in other words their probability estimates are based on co-occurrences of entire (predicate, arguments) tuples. Clearly such methods fail to adequately take into account compositional nature of expressions used to refer to events and suffer from data sparsity.

In this work our goal is to overcome the shortcomings of the count-based methods described above by representing events as real-valued vectors (*event embeddings*), with the embeddings computed in a compositional way relying on the predicate and its arguments. These embeddings capture semantic properties of events: events which differ in surface forms of their constituents but are semantically similar will get similar embeddings. The event embeddings are used and estimated within our probabilistic model of semantic scripts. We evaluate our model on predicting left-out events (the *narrative cloze task*) where it outperforms existing count-based methods.

2 Background

The general idea in the previous count based methods is to collect events sequences for an entity from the corpus (referred as a script). An *entity* is typically a noun/pronoun describing a person, location or temporal construct mentioned in a document. A document is parsed using a statistical dependency parser. Then, the document is processed with a coreference resolution system, linking all the mentions of an entity in the document. Information from the parser and the coreference system is used to collect all the events corresponding to an entity. Different systems differ on how they represent an event. We later explain in detail these event representation differences. The process described above is repeated for all the documents in the corpus to collect event chains for each of the entities. The collected event sequences are used to build different statistical script models. These script models are typically evaluated using a narrative cloze test as explained in section 3. In the cloze test, an event is removed from an event chain and the task is to predict the missing event.

As described above different script models differ in, how they represent an event. Chambers and Jurafsky (2008), Jans et al. (2012) and Rudinger et al. (2015a) represent an event as verb dependency type (for example subject, object etc) pair. Using dependency parser and coreference system, they collect verbs governing entity mentions, this chain of verbs along with corresponding dependency, forms the event chain. Recently, Pichotta and Mooney (2014) extended the concept of an event to include multiple arguments. In their model, an event is a tuple $v(e_s, e_o, e_p)$, where, entities e_s, e_o and e_p are arguments with subject, object and prepositional relation with the governing verb v. A multi-argument event model encodes a richer representation of events. They have empirically show the advantages of having multiple arguments in an event. In our work, we follow the event definition of Pichotta and Mooney (2014) and include multiple arguments in an event.

One of the disadvantage of the count based models described above is poor event representations. Due to these impoverished representations, these models fail to take into account compositional nature of an event and suffer from sparsity issues. These models treat verb-argument pair as one unit and collect chains of verb-arguments pair observed during training. Verb-arguments combinations never observed during training are assigned zero (or very small, if model is smoothed) probability, even if these are semantically similar to the ones in training. These models fail to account for semantic similarity between individual components (verbs and arguments) of an event. For example, events *cook(John,spaghetti,dinner)* and *prepared(Mary,pasta,dinner)* are semantically very similar but count based models would not take this into account unless both events occur in similar context. Due to sparsity issues, these models can fail. This can be exemplified as follows. Suppose the following text is observed during model training :

John cooked spaghetti for dinner. Later, John ate dinner with his girlfriend. After dinner, John took a dog for a walk. After 30 minutes, John came home. After a while, John slept on the bed.

Event sequence (script) corresponding to the above story is:

cook(john,spaghetti,dinner)→eat(john,dinner,girlfriend)→
take(john,dog,walk)→come(john,home)→sleep(john,bed)

Suppose during testing the following event sequence is observed :

prepared(mary,pasta,dinner)→eat(mary,dinner,boyfriend)→
take(mary,cat,walk)→ ? →sleep(mary, couch)

The model is required to guess the missing event marked with '?'. A count-based model would fail if it never encountered the same events during training. It would fail to take into account the semantic similarity between words prepared and cook, dog and cat.

A related disadvantage of a count based script models is that they suffer from the curse of dimensionality (Bengio et al., 2001). Since these methods are based on co-occurrence counts of events, the number of instances required to model the joint probability distribution of events grows exponentially. For example, if event vocabulary size is 10 and number of events occurring in a chain are 5, then number of instances required to model the joint distribution of events is $5^{10} - 1$. This is so because the number of instances required are directly proportional to number of free parameters in the model.

To counter the shortcomings of count based script models, we propose a script model based on distributed representations (Bengio et al., 2001; Turian et al., 2010; Collobert et al., 2011). Our model tries to overcome the curse of dimensionality and sparsity by representing events as vector of real values. Both verbs and arguments are represented as a vector of real values (a.k.a *embeddings*). Verb and argument embeddings are composed to get event vector (event embedding). The model automatically learns these embeddings from the data itself and in the process encodes semantic properties in the event representations.

3 Tasks Definition

One of the standard tasks used for evaluating script models is Narrative Cloze (Chambers and Jurafsky, 2008; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015a). Origins of narrative cloze lie in psychology where it was used to assess child's ability to fill in missing word in a sentence (Taylor, 1953). In our setting the cloze task is described as follows : given a sequence of events with an event removed from the sequence, guess the missing event. For example, given the sequence cook(john,spaghetti,dinner)→eat(john,dinner,girlfriend)→ take(john,dog,walk)→?→sleep(john,bed) , predict the event that should come at position marked by ?

Narrative cloze task evaluates models for exact correctness of the prediction. It penalizes predictions even if they are semantically plausible. It would be more realistic to evaluate script models on a task that gives credit for predicting semantically plausible alternatives as well. We propose adversarial narrative cloze task. In this task, the model is shown two event sequences, one is the correct event sequence and another is same sequence but with one event replaced by a random event. The task is to guess which of the two, is the correct event sequence. For example, given two sequences below, the model should be able to distinguish the correct event sequence from the incorrect one. Interestingly, Manshadi et al. (2008) also propose a similar task for evaluating event based language model and they refer to it as event ordering task. As explained in section 5, we evaluate our model on both the tasks: narrative cloze and adversarial narrative cloze.

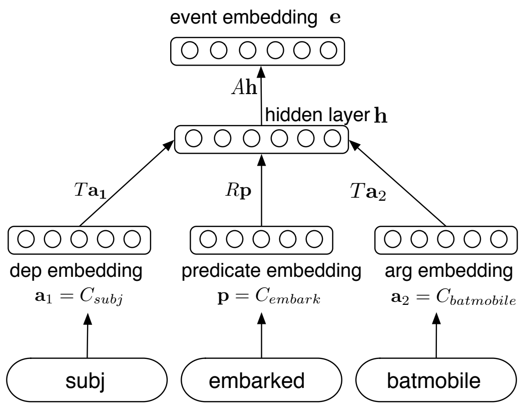
event embedding $\mathbf{e}$
$A\mathbf{h}$
hidden layer $\mathbf{h}$
$T\mathbf{a}_1$
$R\mathbf{p}$
$T\mathbf{a}_2$
dep embedding
$\mathbf{a}_1 = C_{subj}$
predicate embedding
$\mathbf{p} = C_{embark}$
arg embedding
$\mathbf{a}_2 = C_{batmobile}$
subj
embarked
batmobile

Figure 1: Computation of an event representation for a predicate with dependency and an argument (*subj (batman) embarked batmobile*), an arbitrary number of arguments is supported by our approach.

Correct:
cook(john,spaghetti,dinner)→eat(john,dinner,girlfriend) →take(john,dog,walk)→**come(john,home)**→sleep(john,bed)
Incorrect:
cook(john,spaghetti,dinner)→eat(john,dinner,girlfriend) →take(john,dog,walk)→**play(john,tennis)**→sleep(john,bed)

4 Script Model

We propose a probabilistic model for learning a sequence of events corresponding to a script. The proposed model predicts the event incrementally. It first predicts a verbal predicate, followed by protagonist position (since the protagonist argument is already known) and followed by remaining arguments. We believe this is more natural way of predicting the event as opposed to predicting the complete event, treating it as an atomic unit. The information about the predicate influences the possible arguments that could come next due to selectional preferences of the verb.

As done in previous work, (Chambers and Jurafsky, 2008; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015a) each event in a sequence of events has a common entity (protagonist) as one of the argument. We represent an event as a tuple $v(d, a^{(1)}, a^{(2)})$ where v is the verbal predicate, d is the position (subj, obj or prep) of the protagonist, $a^{(1)}$ and $a^{(2)}$ are the other dependent arguments of the verb. We marked absent argument as 'NULL'.

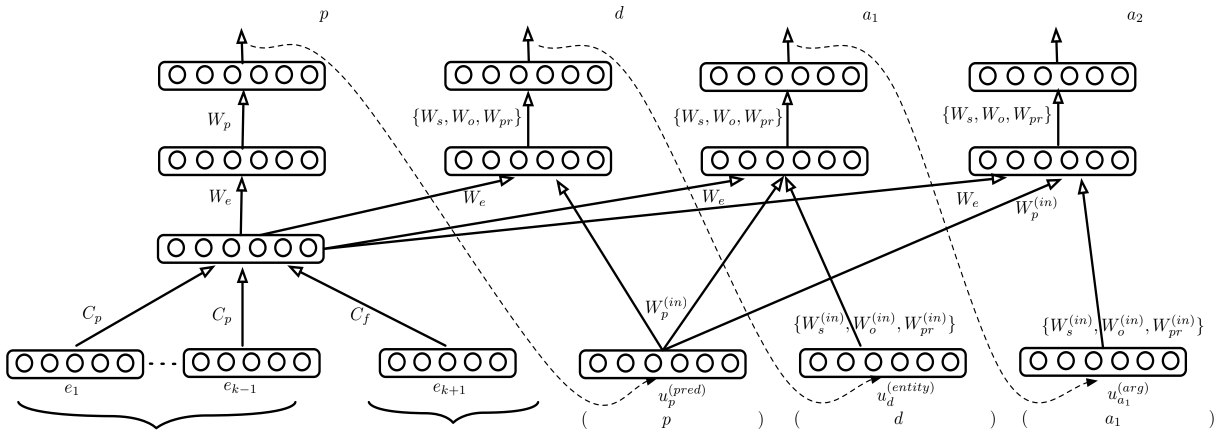

Figure 2: Model for learning event sequence. Here, we are given sequence of events $e_1, e_2, ..., e_{k-1}, e_k, e_{k+1}$. Event e_k is removed from the sequence and it is predicted incrementally.

4.1 Event Representation

For event representation, one could use a sophisticated compositional model based on recursive neural networks (Socher et al., 2012) , we take a simpler approach and choose a feedforward network based compositional model as this is easier to train and more robust to choice of hyper-parameters. Our event representations model is inspired from the event ordering model of Modi and Titov (2014). Their model uses distributed word representations for representing verb and argument lemmas constituting the event. Distributed word representations (also known as word embeddings) encode semantic and syntactic properties of a word in a vector of real values (Bengio et al., 2001; Turian et al., 2010; Collobert et al., 2011). Word embeddings have been shown to be beneficial in many NLP applications Turian et al. (2010); Collobert et al. (2011).

The event model is a simple compositional model representing an event. The model is shown in Figure 1. Given an event, $e = (v, d, a_1, a_2)$, (here v is the predicate lemma, d the dependency and a_1, a_2 are corresponding argument lemmas), each lemma (and dependency) is mapped to a vector using a lookup matrix C. For example, a particular row number of C, corresponding to index of a predicate in the vocabulary, gives the embedding for the predicate. These constituent embeddings are projected into same space by multiplying with respective projection matrices R (for predicates) and T (for arguments). Hidden layer h is obtained by applying a nonlinear activation function ($tanh$ in our case). Final event representation e is obtained by projecting the hidden layer using a matrix A. Formally, event representation is given by $e = A{*}tanh(T{*}C_{a_1,:}{+}R{*}C_{v,:}{+}T{*}C_{a_2,:}){+}b$. All the projection matrices (R, T, A) and lookup matrix C are learned during training. We also experimented with different matrices T for subject and object positions, in order to take into account the positional information. Empirically, this had negligible effect on the final results.

4.2 Event Sequence Model

A good script model should capture the meaning as well as the statistical dependencies between events in an event sequence. More importantly, the model should be able to learn these representations from unlabeled script sequences available in abundance.

We propose a neural network based probabilistic model for event sequences, for learning event sequence as well as the event representations. The model is shown in Figure 2. The model is trained by predicting a missing event in an event sequence. During training, a window (size = 5 = 3*2 + 1) is moved over all events in each event sequence corresponding to each entity. The event in window's center is the event to be predicted and events on the left and right of the window are the context events. As explained earlier, the missing event is predicted incrementally, beginning with a predicate, followed by the protagonist position, followed by other participants in the event.

In order to get an intuition how our model predicts an event, consider the following event sequence in a script, with a missing event : ($e_1 \rightarrow e_2 \cdots \rightarrow e_{k-1} \rightarrow ? \rightarrow e_{k+1} \rightarrow \ldots e_n$). We would like to predict the missing event, say e_k. The event model is used to obtain event representations for each event in the context. These event representations are then composed into context representation by summing the representation for each of the event in the context. We sum the representations, as this formulation works well in practice. The desired event e_k is predicted incrementally, beginning with the predicate p for e_k. The context embedding is used to predict the verbal predicate via a hidden layer followed by a multiclass logistic regression (softmax) classification. Next, the protagonist position d (subject, object etc) is predicted. For predicting d, the context embedding and the predicate embedding (corresponding to the predicate predicted in previous step) are linearly combined to be given as input to a hidden layer. This is followed by regular softmax prediction. Similarly, arguments are predicted. For each of the argument, predicate embedding and the previous prediction (position or argument) are linearly combined with the context embedding. If at each prediction stage we used gold predicate/position/argument embedding for linearly combining with context embedding, our model would not be robust to wrong predictions during testing. Using the embeddings corresponding to predicted unit would make the model robust against noise and would help the model to partially recover from wrong predictions during testing.

We train the model by minimizing the negative likelihood function for the event prediction. Formally, we minimize the objective function $-J(\Theta)$ as shown in equation 1 and 2. As shown in equation 3, we factorize the event distribution into constituents, making appropriate independence assumptions as explained earlier. Each of the factor is a multiclass logistic regression (softmax) function. Equation 4 illustrates the probability distribution for the predicate given the context. Here, u_{v_i} is the word embedding for the predicate v_i, E is the context embedding and b_{v_i} is the bias. Probability distributions for arguments has similar form and are not shown here due to space constraints.

$\Theta = \{C, T, R, A, C_p, C_f, W_e, W_p, W_s, W_o, W_{pr}, W_p^{(in)}, W_s^{(in)}, W_o^{(in)}, W_{pr}^{(in)}, B\}$ is the parameter vector to be learned. Parameters are learned using mini-batch (size=1000) stochastic gradient descent with adagrad (Duchi et al., 2011) learning schedule. During training, the error incurred during predictions at each stage are back-propagated to update the parameters for the model including the embeddings for predicates and arguments (matrix C).

We regularize the parameters of the model using L2 regularization (regularization parameter = 0.01). All the hidden layers have a dropout factor of 0.5. We trained a word2vec model on train set documents to learn word embeddings. Predicate and arguments vectors are initialized using the learned word embeddings. Predicate and argument embeddings have dimensionality of 50 and hidden layers have dimensionality of 50. All the hyper-parameters were tuned using a dev set.

$$\Theta^* = argmin_\Theta - J(\Theta) \tag{1}$$

$$\begin{aligned} J(\Theta) &= \prod_{i=1}^{N} \underbrace{p(e_i \mid e_1, ..., e_{i-1}, e_{i+1}, e_k, \Theta)}_{prob.\ of\ an\ event\ given\ context} \\ &= \prod_{i=1}^{N} p(e_i \mid \underbrace{\mathbf{e}}_{\text{context events}}, \Theta) \end{aligned} \tag{2}$$

$$\begin{aligned} p(e_i \mid \mathbf{e}, \Theta) &= p(v_i, d_i, a_i^{(1)}, a_i^{(2)} \mid \mathbf{e}, \Theta) \\ &= \underbrace{p(v_i \mid \mathbf{e}, \Theta)}_{verb\ prob.} * \underbrace{p(d_i \mid v_i, \mathbf{e}, \Theta)}_{dependency\ prob.} * \\ &\quad \underbrace{p(a_i^{(1)} \mid v_i, d_i, \mathbf{e}, \Theta)}_{first\ arg\ prob.} * \\ &\quad \underbrace{p(a_i^{(2)} \mid v_i, a_i^{(1)}, \mathbf{e}, \Theta)}_{second\ arg\ prob.} \end{aligned} \tag{3}$$

$$p(v_i \mid \mathbf{e}, \Theta) = \frac{exp(u_{v_i}^T (W_p \tanh(W_e E)) + b_{v_i})}{\sum_k exp(u_k^T (W_p \tanh(W_e E)) + b_k)} \tag{4}$$

5 Experiments and Analysis

5.1 Data

There is no standard dataset for evaluating script models. We experimented with movies summary corpus[1] (Bamman et al., 2014). The corpus is created by extracting 42,306 movie summaries from November, 2012 dump of Wikipedia[2]. Each document in the corpus concisely describes a movie plot along with descriptions of various characters involved in the plot. Average length of a document in the corpus is 176 words. But more popular movies have much more elaborate descriptions going up to length of 1,000 words. The corpus has been processed by the Stanford Corenlp pipeline (Manning et al., 2014). The texts in the corpus were tokenized and annotated with POS

[1]http://www.cs.cmu.edu/~ark/personas/
[2]http://dumps.wikimedia.org/enwiki/

Data Set	No. of Scripts	No. of Unique Events
Train Set	104,041	856,823
Dev Set	15,169	119,302
Test Set	29,943	231,539

Table 1: Data statistics

tags, dependencies, NER and coreference (coref) information. Since, each of the document in the corpus is about a movie, the scripts in this corpus involve interesting interactions between different entities (actors/objects). In order to study and explore the above mentioned rich script structure, we selected this corpus for our experiments. Nevertheless, our model is domain agnostic, the experiments performed on this corpus are generalizable to any other corpus as well.

As mentioned in section 2, we extract scripts corresponding to each of the entities using the dependency annotations and coref information. The corpus documents are divided randomly into three parts: train (~70%), development (~10%) and test (~20%). Data statistics about the data set are given in Table 1. As a preprocessing step, low frequency (< 100) predicates and arguments are mapped to a special UNK (unknown) symbol. Similarly, arguments consisting of only digits are mapped to a NUMB (number) symbol. There are 703 unique predicate lemmas and $46,644$ unique argument lemmas in the train set. Average length of a script is 10 events. During testing, predicate and arguments not observed during training are mapped to the same UNK symbol.

5.2 Baselines Systems

We compare our model against two baseline models: **Unigram** model and **MultiProtagonist** model.

A unigram model is a simple but competitive script model. This model predicts an event by sampling from unigram event frequency distribution of the train set. The events are predicted independent of the context.

MultiProtagonist (M-Pr) is the model proposed by Pichotta and Mooney (2014) and described as *joint* model in Pichotta and Mooney (2014). The model calculates conditional probability of an event given another context event ($P(e_2 \mid e_1$) by counting the co-occurrence counts of the events in the corpus. The model predicts the missing event given the context events by maximizing the sum of log conditional probabilities of an event w.r.t each of the context events i.e.

$e^* = \text{argmax}_e \sum_{i=1}^{k-1} log\, P(e \mid e_i) + \sum_{i=k+1}^{K} log\, P(e_i \mid e)$

For evaluation and comparison purposes, we reimplemented both baselines on our dataset. In the experiments described next, we refer our model as NNSM (Neural Network based Script Model).

5.3 Evaluation Metrics

We evaluated models for narrative cloze task with three metrics **Recall@50** and **Accuracy** and **Event Perplexity**. Recall@50 is the standard metric used for evaluating script models (Jans et al., 2012; Pichotta and Mooney, 2014). The idea here is to evaluate top 50 predictions of a script model on a test script with a missing event. The metric is calculated as fraction of the predictions containing the gold held-out event. Its value lies in the range 0 (worst) and 1(best). Accuracy is a new metric introduced by Pichotta and Mooney (2014). This metric evaluates the event prediction, taking into account prediction of each constituent. Specifically, it is defined as average of the accuracy of the predicate, the dependency, the first argument and the second argument predictions. This is a more robust metric as it does not treat an event as an atomic unit. This is in contrast to Recall@50 which penalizes semantically correct guesses and awards only events which have exactly the same surface form.

The baseline models and our model are probabilistic by nature. Taking inspiration from language modeling community, we propose a new metric Event Perplexity. We define event perplexity as $2^{-\frac{1}{N}\sum_i \log_2 p(e_i \mid \mathbf{e}_{(\mathbf{context},\mathbf{i})})}$. The perplexity measure, like the accuracy takes into account the constituents of an event and is a good indicator of the model predictions.

5.4 Narrative Cloze Evaluation

Narrative Cloze task was tested on 29,943 test set scripts. The results are shown in Table 2 and 3. We evaluated with two versions of the cloze task. In the first version, events are the predicate argument tuple as defined before. Second version, evaluates on predicates only i.e. an event is not a tuple but only a predicate. Our model, NNSM outperforms both the unigram and M-Pr models on both the versions of the task with all the metrics. This further strengthens our hypothesis of

Model	R@50	Accuracy	Event Perplexity
Unigram	0.32	34.26%	298.45
M-Pr	0.31	35.67%	276.54
NNSM$_{\text{full}}$	0.37	44.36%	256.41

Table 2: Model evaluation on test set for narrative cloze task against the baselines

Model	R@50	Accuracy	Event Perplexity
Unigram$_{\text{pred}}$	0.27	23.79%	264.16
M-Pr$_{\text{pred}}$	0.27	24.04%	260.34
NNSM$_{\text{pred}}$	0.49	33.40%	247.64

Table 3: Model evaluation on predicate only event test set for narrative cloze task against the baselines

having distributed representation for events rather than atomic representations. Unigram, although a simple model, is competitive with the M-Pr model.

We performed an interesting experiment. We evaluated another simplistic baseline model, most frequent event. This baseline model predicts by sampling from top-5 most frequent predicate/argument in the full event narrative cloze task. Surprisingly, the accuracy reported by this simple baseline is 45.04% which is slightly more than our best performing NNSM model and much more than the M-Pr baseline. This simple looking baseline is hard to beat by both count-based methods and NNSM. None of the previous methods have been evaluated against this baseline. We propose using this baseline for evaluation of script models. We think this outperformance is due to skewed distribution of the predicate and arguments in the corpus. As we found empirically, these distributions have a very long tail and this makes it hard for the models to beat the most frequent baseline.

5.5 Adversarial Narrative Cloze Evaluation

Similar to narrative cloze, adversarial narrative cloze task was evaluated on 29,943 test set scripts. In each of the event sequence an event was replaced by a random event. The results for the adversarial narrative cloze task are shown in Table 4 and 5. As evident from the results Unigram model is as good as random. In this task as well, our model outperforms the count based M-Pr model by 2.3% and 2.9% for full and pred model respectively.

Model	Accuracy
Unigram	50.07%
M-Pr	53.04%
NNSM $_{\text{full}}$	55.32%

Table 4: Model evaluation on test set for adversarial narrative cloze task against the baselines

Model	Accuracy
Unigram$_{\text{pred}}$	49.97%
M-Pr$_{\text{pred}}$	55.09%
NNSM$_{\text{pred}}$	57.94%

Table 5: Model evaluation on predicate only test set for adversarial narrative cloze task against the baselines

6 Related Work

Work on scripts dates back to 70's beginning with introduction of Frames by Minsky (1974), schemas by Rumelhart (1975) and scripts by Schank and Abelson (1977). These early formulations were not statistical in nature and used hand-crafted complex rules for modeling relations between events. These formulations were limited to few scenarios and did not generalize well.

Miikkulainen (1990) proposed DISCERN system for learning script knowledge. Their neural network based model read event sequences and stored them in episodic memory. The model was capable of generating expanded paraphrases of narratives and was able to answer simple questions. Similarly, Lee et al. (1992) proposed DYNASTY (DYNAmic STory understanding sYstem). This system was also based on distributed representations. It predicted missing events in event sequences and performed script based information retrieval. Their system was limited to only few scenarios and did not generalize well.

As mentioned previously, in past few years a number of count based systems for script learning have been proposed for learning script knowledge in unsupervised fashion. (Chambers and Jurafsky, 2008; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015a). Recently, Regneri et al. (2010), Modi et al. (2016) and Wanzare et al. (2016) used crowd-sourcing methods for acquiring script knowledge. They in the process created script databases which are used to develop automatic script learning systems. McIntyre and Lapata (2009) developed a system for generating stories, they learned an object based script model on fairy tales.

Orr et al. (2014) proposed hidden markov model (HMM) approach to learn scripts. The model clusters event descriptions into different event types and then learns an HMM over the sequence of event types. Again this model treats an event as an atomic unit and the inference algorithm may not generalize well, as the number of event types increases. Similarly, Frermann et al. (2014) proposed non-parametric Bayesian model for learning script ordering. The inference procedure may not scale well, as number of event chains increases.

Manshadi et al. (2008) proposed language model based approach to learning event sequences, in their approach as well, events are treated as atomic units (a predicate-argument tuple). Recently, Rudinger et al. (2015b) have proposed a neural network approach to learn scripts by learning a bilinear distributed representation based language model over events. Their model is non-compositional in nature and they also consider events as an atomic unit and directly learn distributed representation for events. Granroth-Wilding and Clark (2015) also propose a compositional neural network based model for events. Our model is more general than their model. They learn event representations by modeling pair wise event scores for calculating compatibility between two events. This score is then used to predict the missing event by selecting an event that maximizes the average score between the event and the context events.

7 Conclusion and Future Work

In this paper we proposed a probabilistic compositional model for scripts. As shown in experiments, our model outperforms the existing co-occurrence count based methods. This further reinforces our hypothesis of having more richer compositional representations for events. Current tasks to evaluate script models are crude, in the sense that they penalize semantically plausible events. In the future, we propose to create a standard data set of event sequence pairs (correct sequence vs incorrect sequence). The replaced event in the incorrect sequence should not be a random event but rather a semantically close but incorrect event. Models evaluated on this data set would give a better indication of script learning capability of the model. Another area which needs further investigation is related to developing models which can learn long tail event distributions. Current models do not capture this well and hence do not perform better than most frequent event baseline on accuracy task.

In this paper, we proposed a very simple compositional feed forward neural network model. In the future we plan to explore more sophisticated recurrent neural network (RNN) based models. Recently, RNN based models have shown success in variety of applications (Graves, 2012).

Acknowledgments

We would like to thank anonymous reviewers for their helpful comments. This research was funded by the German Research Foundation (DFG) as part of SFB 1102 'Information Density and Linguistic Encoding'.

References

David Bamman, Brendan O'Connor, and Noah A Smith. 2014. Learning latent personas of film characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. page 352.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In *Proceedings of NIPS*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL*.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

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

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *EACL, Gothenberg, Sweden*.

Mark Granroth-Wilding and Stephen Clark. 2015. What happens next? event prediction using a compositional neural network model. In *Proceedings of AAAI*.

Alex Graves. 2012. *Supervised sequence labelling*. Springer.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 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*. Association for Computational Linguistics, Stroudsburg, PA, USA, EACL '12, pages 336–344.

Geunbae Lee, Margot Flowers, and Michael G Dyer. 1992. Learning distributed representations of conceptual knowledge and their application to script-based story processing. In *Connectionist Natural Language Processing*, Springer, pages 215–247.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Mehdi Manshadi, Reid Swanson, and Andrew S Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *FLAIRS Conference*. pages 159–164.

Neil McIntyre and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. 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 1-Volume 1*. Association for Computational Linguistics, pages 217–225.

Risto Miikkulainen. 1990. *DISCERN: A Distributed Artificial Neural Network Model Of Script Processing And Memory*. Ph.D. thesis, University of California.

Risto Miikkulainen. 1995. Script-based inference and memory retrieval in subsymbolic story processing. *Applied Intelligence* 5(2):137–163.

Marvin Minsky. 1974. A framework for representing knowledge .

Ashutosh Modi, Tatjana Anikina, and Manfred Pinkal. 2016. Inscript: Narrative texts annotated with script information. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 16), Portorož, Slovenia.*.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *CoNLL*. volume 14, pages 49–57.

John Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, and Thomas G Dietterich. 2014. Learning scripts as hidden markov models. In *AAAI*. pages 1565–1571.

Karl Pichotta and Raymond J Mooney. 2014. Statistical script learning with multi-argument events. In *EACL*. volume 14, pages 220–229.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of ACL*.

Rachel Rudinger, Vera Demberg, Ashutosh Modi, Benjamin Van Durme, and Manfred Pinkal. 2015a. Learning to predict script events from domain-specific text. *Lexical and Computational Semantics (* SEM 2015)* page 205.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015b. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*.

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

R. C Schank and R. P Abelson. 1977. *Scripts, Plans, Goals, and Understanding*. Lawrence Erlbaum Associates, Potomac, Maryland.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*.

Wilson L Taylor. 1953. Cloze procedure: a new tool for measuring readability. *Journalism and Mass Communication Quarterly* 30(4):415.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*.

Lilian Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2016. Descript: A crowdsourced database for the acquisition of high-quality script knowledge. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 16), Portorož, Slovenia.*.