Using Sentence-Level LSTM Language Models for Script Inference

Karl Pichotta
Department of Computer Science
The University of Texas at Austin
pichotta@cs.utexas.edu

Raymond J. Mooney
Department of Computer Science
The University of Texas at Austin
mooney@cs.utexas.edu

Abstract

There is a small but growing body of research on statistical scripts, models of event sequences that allow probabilistic inference of implicit events from documents. These systems operate on structured verb-argument events produced by an NLP pipeline. We compare these systems with recent Recurrent Neural Net models that directly operate on raw tokens to predict sentences, finding the latter to be roughly comparable to the former in terms of predicting missing events in documents.

1 Introduction

Statistical scripts are probabilistic models of event sequences (Chambers and Jurafsky, 2008). A learned script model is capable of processing a document and inferring events that are probable but not explicitly stated. These models operate on automatically extracted structured events (for example, verbs with entity arguments), which are derived from standard NLP tools such as dependency parsers and coreference resolution engines.

Recent work has demonstrated that standard sequence models applied to such extracted event sequences, e.g. discriminative language models (Rudinger et al., 2015) and Long Short Term Memory (LSTM) recurrent neural nets (Pichotta and Mooney, 2016), are able to infer held-out events more accurately than previous approaches. These results call into question the extent to which statistical event inference systems require linguistic preprocessing and syntactic structure. In an attempt to shed light on this issue, we compare existing script models to LSTMs trained as *sentence-level language models* which try to predict the sequence of words in the next sentence from a learned representation of the previous sentences using no linguistic preprocessing.

Some prior statistical script learning systems are focused on knowledge induction. These systems are primarily designed to induce collections of co-occurring event types involving the same entities, and their ability to infer held-out events is not their primary intended purpose (Chambers and Jurafsky, 2008; Ferraro and Van Durme, 2016, *inter alia*). In the present work, we instead investigate the behavior of systems trained to directly optimize performance on the task of predicting subsequent events; in other words, we are investigating statistical models of events in discourse.

Much prior research on statistical script learning has also evaluated on inferring missing events from documents. However, the exact form that this task takes depends on the adopted definition of what constitutes an event: in previous work, events are defined in different ways, with differing degrees of structure. We consider simply using raw text, which requires no explicit syntactic annotation, as our mediating representation, and evaluate how raw text models compare to models of more structured events.

Kiros et al. (2015) introduced *skip-thought vector* models, in which an RNN is trained to encode a sentence within a document into a low-dimensional vector that supports predicting the neighboring sentences in the document. Though the objective function used to train networks maximizes performance on the task of predicting sentences from their neighbors, Kiros et al. (2015) do not evaluate directly on the ability of networks to predict text; they instead demonstrate that the intermediate low-dimensional vector embeddings are useful for other tasks. We directly evaluate the text predictions produced by such sentence-level RNN encoder-decoder models, and measure their utility for the task of predicting subsequent events.

We find that, on the task of predicting the text of held-out sentences, the systems we train to operate on the level of raw text generally outperform the systems we train to predict text mediated by automatically extracted event structures. On the other hand, if we run an NLP pipeline on the automatically generated text and extract structured events from these predictions, we achieve prediction performance roughly comparable to that of systems trained to predict events directly. The difference between word-level and event-level models on the task of event prediction is marginal, indicating that the task of predicting the next event, particularly in an encoder-decoder setup, may not necessarily need to be mediated by explicit event structures. To our knowledge, this is the first effort to evaluate sentence-level RNN language models directly on the task of predicting document text. Our results show that such models are useful for predicting missing information in text; and the fact that they require no linguistic preprocessing makes them more applicable to languages where quality parsing and co-reference tools are not available.

2 Background

2.1 Statistical Script Learning

Scripts, structured models of stereotypical sequences of events, date back to AI research from the 1970s, in particular the seminal work of Schank and Abelson (1977). In this conception, scripts are modeled as temporally ordered sequences of symbolic structured events. These models are nonprobabilistic and brittle, and pose serious problems for automated learning.

In recent years, there has been a growing body of research into statistical script learning systems, which enable statistical inference of implicit events from text. Chambers and Jurafsky (2008; 2009) describe a number of simple event co-occurrence based systems which infer (verb, dependency) pairs related to a particular discourse entity. For example, given the text:

> *Andrew Wiles won the 2016 Abel prize for proving Fermat's last theorem*,

such a system will ideally be able to infer novel facts like (*accept*, subject) or (*publish*, subject) for the entity *Andrew Wiles*, and facts like (*accept*, object) for the entity *Abel prize*. A number of other systems inferring the same types of pair events have been shown to provide superior performance in modeling events in documents (Jans et al., 2012; Rudinger et al., 2015).

Pichotta and Mooney (2014) give a co-occurrence based script system that models and infers more complex multi-argument events from text. For example, in the above example, their model would ideally be able to infer a single event like *accept*(*Wiles*, *prize*), as opposed to the two simpler pairs from which it is composed. They provide evidence that modeling and inferring more complex multi-argument events also yields superior performance on the task of inferring simpler (verb, dependency) pair events. These events are constructed using only coreference information; that is, the learned event co-occurrence models do not directly incorporate noun information.

More recently, Pichotta and Mooney (2016) presented an LSTM-based script inference model which models and infers multi-argument events, improving on previous systems on the task of inferring verbs with arguments. This system can incorporate both noun and coreference information about event arguments. We will use this multi-argument event formulation (formalized below) and compare LSTM models using this event formulation to LSTM models using raw text.

2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are neural nets whose computation graphs have cycles. In particular, RNN sequence models are RNNs which map a sequence of inputs $x_1, \ldots, x_T$ to a sequence of outputs $y_1, \ldots, y_T$ via a learned latent vector whose value at timestep t is a function of its value at the previous timestep $t-1$.

The most basic RNN sequence models, so-called "vanilla RNNs" (Elman, 1990), are described by the following equations:

$$z_t = f(W_{i,z}x_t + W_{z,z}z_{t-1})$$
$$o_t = g(W_{z,o}z_t)$$

where x_t is the vector describing the input at time t; z_t is the vector giving the hidden state at time t; o_t is the vector giving the predicted output at time t; f and g are element-wise nonlinear functions (typically sigmoids, hyperbolic tangent, or rectified linear units); and $W_{i,z}$, $W_{z,z}$, and $W_{z,o}$ are learned matrices describing linear transformations. The recurrency in the computation graph arises from the fact that z_t is a function of z_{t-1}.

The more complex Long Short-Term Memory (LSTM) RNNs (Hochreiter and Schmidhuber,

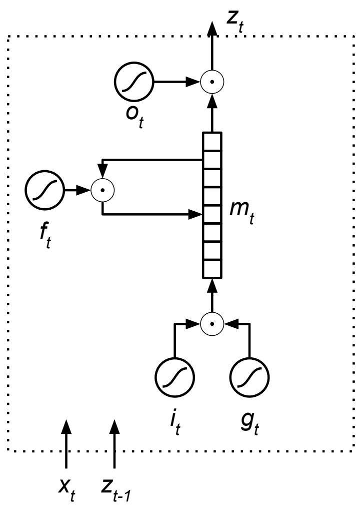

Figure 1: Long Short-Term Memory unit at timestep t. The four nonlinearity nodes (i_t, g_t, f_t, and o_t) all have, as inputs, x_t and z_{t-1}. Small circles with dots are elementwise vector multiplications.

1997) have been shown to perform well on a wide variety of NLP tasks (Sutskever et al., 2014; Hermann et al., 2015; Vinyals et al., 2015, *inter alia*). The LSTM we use is described by:

$$\begin{aligned}
i_t &= \sigma\left(W_{x,i}x_t + W_{z,i}z_{t-1} + b_i\right)\\
f_t &= \sigma\left(W_{x,f}x_t + W_{z,f}z_{t-1} + b_f\right)\\
o_t &= \sigma\left(W_{x,o}x_t + W_{h,i}z_{t-1} + b_o\right)\\
g_t &= \tanh\left(W_{x,m}x_t + W_{z,m}z_{t-1} + b_g\right)\\
m_t &= f_t \circ m_{t-1} + i_t \circ g_t\\
z_t &= o_t \circ \tanh m_t.
\end{aligned}$$

The model is depicted graphically in Figure 1. The memory vector m_t is a function of both its previous value m_{t-1} and the input x_t; the vector z_t is output both to any layers above the unit (which are trained to predict the output values y_t), and is additionally given as input to the LSTM unit at the next timestep $t+1$. The $W_{*,*}$ matrices and b_* vectors are learned model parameters, and $u \circ v$ signifies element-wise multiplication.

2.3 Sentence-Level RNN Language Models

RNN sequence models have recently been shown to be extremely effective for word-level and character-level language models (Mikolov et al., 2011; Jozefowicz et al., 2016). At each timestep, these models take a word or character as input, update a hidden state vector, and predict the next timestep's word or character. There is also a growing body of work on training RNN encoder-decoder models for NLP problems. These systems first encode the entire input into the network's hidden state vector and then, in a second step, decode the entire output from this vector (Sutskever et al., 2014; Vinyals et al., 2015; Serban et al., 2016).

Sentence-level RNN language models, for example the skip-thought vector system of Kiros et al. (2015), conceptually bridge these two approaches. Whereas standard language models are trained to predict the next token in the sequence of tokens, these systems are explicitly trained to predict the next *sentence* in the sequence of sentences. Kiros et al. (2015) train an encoder-decoder model to encode a sentence into a fixed-length vector and subsequently decode both the following and preceding sentence, using Gated Recurrent Units (Chung et al., 2014). In the present work, we train an LSTM model to predict a sentence's successor, which is essentially the forward component of the skip-thought system. Kiros et al. (2015) use the skip-thought system as a means of projecting sentences into low-dimensional vector embeddings, demonstrating the utility of these embeddings on a number of other tasks; in contrast, we will use our trained sentence-level RNN language model directly on the task its objective function optimizes: predicting a sentence's successor.

3 Methodology

3.1 Narrative Cloze Evaluation

The evaluation of inference-focused statistical script systems is not straightforward. Chambers and Jurafsky (2008) introduced the *Narrative Cloze* evaluation, in which a single event is held out from a document and systems are judged by the ability to infer this held-out event given the remaining events. This evaluation has been used by a number of published script systems (Chambers and Jurafsky, 2009; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015). This automated evaluation measures systems' ability to model and predict events as they co-occur in text.

The exact definition of the Narrative Cloze evaluation depends on the formulation of events used in a script system. For example, Chambers and Jurafsky (2008), Jans et al. (2012), and Rudinger et al. (2015) evaluate inference of held-out (verb, dependency) pairs from documents; Pichotta and Mooney (2014) evaluate inference of verbs with coreference information about multiple arguments; and Pichotta and Mooney (2016) evaluate inference of verbs with noun information about multiple arguments. In order to gather human judgments of inference quality, the latter also learn an encoder-decoder LSTM network for transforming verbs and noun arguments into English text to present to annotators for evaluation.

We evaluate instead on the task of directly inferring sequences of words. That is, instead of defining the Narrative Cloze to be the evaluation of predictions of held-out events, we define the task to be the evaluation of predictions of held-out text; in this setup, predictions need not be mediated by noisy, automatically-extracted events. To evaluate inferred text against gold standard text, we argue that the BLEU metric (Papineni et al., 2002), commonly used to evaluate Statistical Machine Translation systems, is a natural evaluation metric. It is an n-gram-level analog to the event-level Narrative Cloze evaluation: whereas the Narrative Cloze evaluates a system on its ability to reconstruct events as they occur in documents, BLEU evaluates a system on how well it reconstructs the n-grams.

This evaluation takes some inspiration from the evaluation of neural encoder-decoder translation models (Sutskever et al., 2014; Bahdanau et al., 2015), which use similar architectures for the task of Machine Translation. That is, the task we present can be thought of as "translating" a sentence into its successor. While we do not claim that BLEU is necessarily the optimal way of evaluating text-level inferences, but we do claim that it is a natural ngram-level analog to the Narrative Cloze task on events.

If a model infers text, we may also evaluate it on the task of inferring events by automatically extracting structured events from its output text (in the same way as events are extracted from natural text). This allows us to compare directly to previous event-based models on the task they are optimized for, namely, predicting structured events.

3.2 Models

Statistical script systems take a sequence of events from a document and infer additional events that are statistically probable. Exactly what constitutes an *event* varies: it may be a (verb, dependency) pair inferred as relating to a particular discourse entity (Chambers and Jurafsky, 2008; Rudinger et al., 2015), a simplex verb (Chambers and Jurafsky, 2009; Orr et al., 2014), or a verb with multiple arguments (Pichotta and Mooney, 2014). In the present work, we adopt a representation of events as verbs with multiple arguments (Balasubramanian et al., 2013; Pichotta and Mooney, 2014; Modi and Titov, 2014). Formally, we define an event to be a variadic tuple (v, s, o, p^*), where v is a verb, s is a noun standing in subject relation to v, o is a noun standing as a direct object to v, and p^* denotes an arbitrary number of (*pobj*, *prep*) pairs, with *prep* a preposition and *pobj* a noun related to the verb v via the preposition *prep*.[1] Any argument except v may be *null*, indicating no noun fills that slot. For example, the text

Napoleon sent the letter to Josephine

would be represented by the event (*sent, Napoleon, letter, (Josephine, to)*). We represent arguments by their grammatical head words.

We evaluate on a number of different neural models which differ in their input and output. All models are LSTM-based encoder-decoder models. These models encode a sentence (either its events or text) into a learned hidden vector state and then, subsequently, decode that vector into its successor sentence (either its events or its text).

Our general system architecture is as follows. At each timestep t, the input token is represented as a learned 100-dimensional embedding vector (learned jointly with the other parameters of the model), such that predictively similar words should get similar embeddings. This embedding is fed as input to the LSTM unit (that is, it will be the vector x_t in Section 2.2, the input to the LSTM). The output of the LSTM unit (called z_t in Section 2.2) is then fed to a softmax layer via a learned linear transformation.

During the encoding phase the network is not trained to produce any output. During the decoding phase the output is a one-hot representation of the subsequent timestep's input token (that is, with a V-word vocabulary, the output will be a V-dimensional vector with one 1 and $V - 1$ zeros). In this way, the network is trained to consume an entire input sequence and, as a second step, iteratively output the subsequent timestep's

[1]This is essentially the event representation of Pichotta and Mooney (2016), but whereas they limited events to having a single prepositional phrase, we allow an arbitrary number, and we do not lemmatize words.

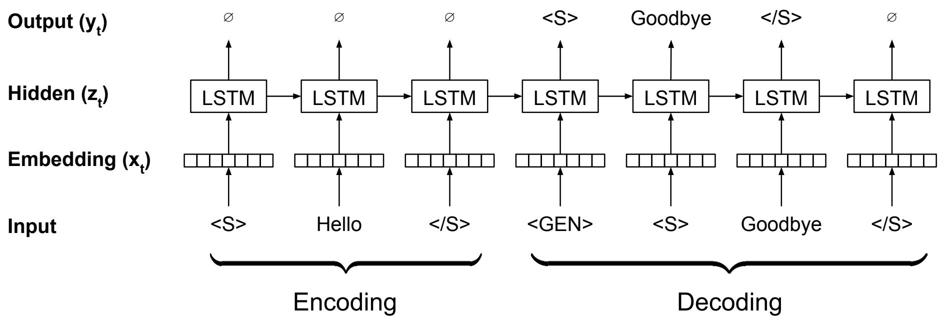

Figure 2: Encoder-Decoder setup predicting the text "Goodbye" from "Hello"

input, which allows the prediction of full output sequences. This setup is pictured diagrammatically in Figure 2, which gives an example of input and output sequence for a token-level encoder-decoder model, encoding the sentence "Hello ." and decoding the successor sentence "Goodbye ." Note that we add beginning-of-sequence and end-of-sequence pseudo-tokens to sentences. This formulation allows a system to be trained which can encode a sentence and then infer a successor sentence by iteratively outputting next-input predictions until the `</S>` end-of-sentence pseudo-token is predicted. We use different LSTMs for encoding and decoding, as the dynamics of the two stages need not be identical.

We notate the different systems as follows. Let s_1 be the input sentence and s_2 its successor sentence. Let t_1 denote the sequence of raw tokens in s_1, and t_2 the tokens of s_2. Further, let e_1 and e_2 be the sequence of structured events occurring in s_1 and s_2, respectively (described in more detail in Section 4.1), and let $e_2[0]$ denote the first event of e_2. The different systems we compare are named systematically as follows:

- The system $t_1 \to t_2$ is trained to encode a sentence's tokens and decode its successor's tokens.
- The system $e_1 \to e_2$ is trained to encode a sentence's events and decode its successor's events.
- The system $e_1 \to e_2 \to t_2$ is trained to encode a sentence's events, decode its successor's events, and then encode the latter and subsequently decode the successor's text.

We will not explicitly enumerate all systems, but other systems are defined analogously, with the schema $X \to Y$ describing a system which is trained to encode X and subsequently decode Y, and $X \to Y \to Z$ indicating a system which is trained to encode X, decode Y, and subsequently encode Y and decode Z. Note that in a system $X \to Y \to Z$, only X is provided as input.

We also present results for systems of the form $X \overset{a}{\to} Y$, which signifies that the system is trained to decode Y from X with the addition of an attention mechanism. We use the attention mechanism of Vinyals et al. (2015). In short, these models have additional parameters which can learn soft alignments between positions of encoded inputs and positions in decoded outputs. Attention mechanisms have recently been shown to be quite empirically valuable in many complex sequence prediction tasks. For more details on the model, see Vinyals et al. (2015).

Figure 3 gives a diagrammatic representation of the different system setups. Text systems infer successor text and, optionally, parse that text and extract events from it; event sequences infer successor events and, optionally, expand inferred events into text.

Note that the system $t_1 \to t_2$, in which both the encoding and decoding steps operate on raw text, is essentially a one-directional version of the skip-thought system of Kiros et al. (2015).[2] Further, the system $e_1 \to e_2 \to t_2$, which is trained to take a sentence's event sequence as input, predict its successor's events, and then predict its successor's words, is comparable to the event inference system of Pichotta and Mooney (2016). They use an LSTM sequence model of events in sequence

[2]The system of Kiros et al. (2015), in addition to being trained to predict the next sentence, also contains a backward-directional RNN trained to predict a sentence's predecessor; we condition only on previous text. Kiros et al. (2015) also use Gated Recurrent Units instead of LSTM.

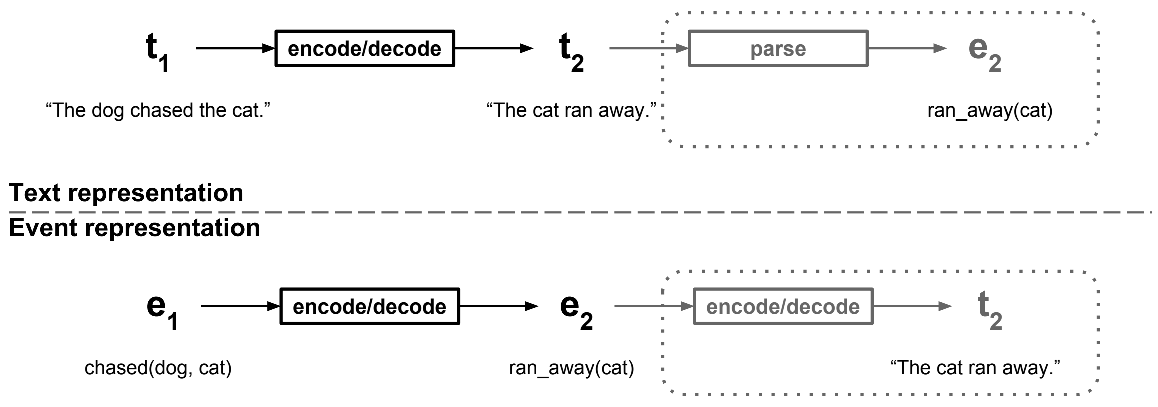

Figure 3: Different system setups for modeling the two-sentence sequence "The dog chased the cat." followed by "The cat ran away." The gray components inside dotted boxes are only present in some systems.

for event inference, and optionally transform inferred events to text using another LSTM; we, on the other hand, use an encoder/decoder setup to infer text directly.

4 Evaluation

4.1 Experimental Details

We train a number of LSTM encoder-decoder networks which vary in their input and output. Models are trained on English Language Wikipedia, with 1% of the documents held out as a validation set. Our test set consists of 10,000 unseen sentences (from articles in neither the training nor validation set). We train models with batch stochastic gradient descent with momentum, minimizing the cross-entropy error of output predictions. All models are implemented in TensorFlow (Abadi et al., 2015). We use a vocabulary of the 50,000 most frequent tokens, replacing all other tokens with an out-of-vocabulary pseudo-token. Learned word embeddings are 100-dimensional, and the latent LSTM vector is 500-dimensional. To extract events from text, we use the Stanford Dependency Parser (De Marneffe et al., 2006; Socher et al., 2013). We use the Moses toolkit (Koehn et al., 2007) to calculate BLEU.[3]

We evaluate the task of predicting held-out text with three metrics. The first metric is **BLEU**, which is standard BLEU (the geometric mean of modified 1-, 2-, 3-, and 4-gram precision against a gold standard, multiplied by a brevity penalty which penalizes short candidates). The second metric we present, **BLEU-BP,** is BLEU without the brevity penalty: in the task of predicting successor sentences, depending on predictions' end use, on-topic brevity is not necessarily undesirable. Evaluations are over top system inferences (that is, decoding is done by taking the argmax). Finally, we also present values for unigram precision (**1G P**), one of the components of BLEU.

We also evaluate on the task of predicting held-out verb-argument events, either directly or via inferred text. We use two evaluation metrics for this task. First, the **Accuracy** metric measures the percentage of a system's most confident guesses that are totally correct. That is, for each held-out event, a system makes its single most confident guess for that event, and we calculate the total percentage of such guesses which are totally correct. Some authors (e.g. Jans et al. (2012), Pichotta and Mooney (2016)) present results on the "Recall at k" metric, judging gold-standard recall against a list of top k event inferences; this metric is equivalent to "Recall at 1." This is quite a stringent metric, as an inference is only counted correct if the verb and all arguments are correct. To relax this requirement, we also present results on what we call the **Partial Credit** metric, which is the percentage of held-out event components identical to the respective components in a system's top inference.[4]

4.2 Experimental Evaluation

Table 1 gives the results of evaluating predicted successor sentence text against the gold standard using BLEU. The baseline system $t_1 \rightarrow t_1$ sim-

[3]Via the script `multi-bleu.pl`.

[4]This metric was used in Pichotta and Mooney (2014), but there it was called *Accuracy*. In the present work, we use "accuracy" only to mean Recall at 1.

System	BLEU	BLEU-BP	1G P
$t_1 \rightarrow t_1$	1.88	1.88	22.6
$e_1 \rightarrow e_2 \rightarrow t_2$	0.34	0.66	19.9
$e_1 \overset{a}{\rightarrow} e_2 \rightarrow t_2$	0.30	0.39	15.8
$t_1 \rightarrow t_2$	**5.20**	7.84	30.9
$t_1 \overset{a}{\rightarrow} t_2$	4.68	**8.09**	**32.2**

Table 1: Successor text predictions evaluated with BLEU.

System	Accuracy	Partial Credit
Most common	0.2	26.5
$e_1 \rightarrow e_2[0]$	**2.3**	26.7
$e_1 \overset{a}{\rightarrow} e_2[0]$	2.2	25.6
$t_1 \rightarrow t_2 \rightarrow e_2[0]$	2.0	**30.3**
$t_1 \overset{a}{\rightarrow} t_2 \rightarrow e_2[0]$	2.0	27.7

Table 2: Next event prediction accuracy (numbers are percentages: maximum value is 100).

ply reproduces the input sentence as its own successor.[5] Below this are systems which make predictions from event information, with systems which make predictions from raw text underneath. Transformations written $X \overset{a}{\rightarrow} Y$ are, recall, encoder-decoder LSTMs with attention.

Note, first, that the text-level models outperform other models on BLEU. In particular, the two-step model $e_1 \rightarrow e_2 \rightarrow t_2$ (and comparable model with attention) which first predicts successor events and then, as a separate step, expands these events into text, performs quite poorly. This is perhaps due to the fact that the translation from text to events is lossy, so reconstructing raw sentence tokens is not straightforward.

The BLEU-BP scores, which are BLEU without the brevity penalty, are noticeably higher in the text-level models than the raw BLEU scores. This is in part because these models seem to produce shorter sentences, as illustrated below in section 4.4.

The attention mechanism does not obviously benefit either text or event level prediction encoder-decoder models. This could be because there is not an obvious alignment structure between contiguous spans of raw text (or events) in natural documents.

These results provide evidence that, if the Narrative Cloze task is defined to evaluate prediction of held-out text from a document, then sentence-level RNN language models provide superior performance to RNN models operating at the event level. In other words, linguistic pre-processing does not obviously benefit encoder-decoder models trained to predict succeeding text.

Table 2 gives results on the task of predicting the next verb with its nominal arguments; that is, whereas Table 1 gave results on a text analog to the Narrative Cloze evaluation (BLEU), Table 2 gives results on the verb-with-arguments prediction version. In the $t_1 \rightarrow t_2 \rightarrow e_2[0]$ system (and the comparable system with attention), events are extracted from automatically generated text by parsing output text and applying the same event extractor to this parse used to extract events from raw text.[6] The row labeled **Most common** in Table 2 gives performance for the baseline system which always guesses the most common event in the training set.

The LSTM models trained to directly predict events are roughly comparable to systems which operate on raw text, performing slightly worse on accuracy and slightly better when taking partial credit into account. As with the previous comparisons with BLEU, the attention mechanism does not provide an obvious improvement when decoding inferences, perhaps, again, because the event inference problem lacks a clear alignment structure.

These systems infer their most probable guesses of $e_2[0]$, the first event in the succeeding sentence. In order for a system prediction to be counted as correct, it must have the correct strings for grammatical head words of all components of the correct event. Note also that we judge only against a system's single most confident prediction (as opposed to some prior work (Jans et al., 2012; Pichotta and Mooney, 2014) which takes the top k predictions—the numbers presented here are therefore noticeably lower). We do this mainly for computational reasons: namely, a beam search over a full sentence's text would be quite computationally expensive.

4.3 Adding Additional Context

The results given above are for systems which encode information about one sentence and decode

[5]"$t_1 \rightarrow t_1$" is minor abuse of notation, as the system is not an encoder/decoder but a simple identity function.

[6]This is also a minor abuse of notation, as the second transformation uses a statistical parser rather than an encoder/decoder.

information about its successor. This is within the spirit of the skip-gram system of Kiros et al. (2015), but we may wish to condition on more of the document. To investigate this, we perform an experiment varying the number of previous sentences input during the encoding step of $t_1 \to t_2$ text-level models without attention. We train three different models, which take either one, three, or five sentences as input, respectively, and are trained to output the successor sentence.

Num Prev Sents	**BLEU**	**BLEU-BP**	**1G P**
1	5.80	8.59	29.4
3	5.82	**9.35**	**31.2**
5	**6.83**	6.83	21.4

Table 3: Varying the amount of context in text-level models. "Num Prev Sents" is the number of previous sentences supplied during encoding.

Table 3 gives the results of running these models on 10,000 sentences from the validation set. As can be seen, in the training setup we investigate, more additional context sentences have a mixed effect, depending on the metric. This is perhaps due in part to the fact that we kept hyperparameters fixed between experiments, and a different hyperparameter regime would benefit predictions from longer input sequences. More investigation could prove fruitful.

4.4 Qualitative Analysis

Figure 4 gives some example automatic next-sentence text predictions, along with the input sentence and the gold-standard next sentence. Note that gold-standard successor sentences frequently introduce new details not obviously inferrable from previous text. Top system predictions, on the other hand, are frequently fairly short. This is likely due part to the fact that the cross-entropy loss does not directly penalize short sentences and part to the fact that many details in gold-standard successor text are inherently difficult to predict.

4.5 Discussion

The general low magnitude of the BLEU scores presented in Table 1, especially in comparison to the scores typically reported in Machine Translation results, indicates the difficulty of the task. In open-domain text, a sentence is typically not straightforwardly predictable from preceding text; if it were, it would likely not be stated.

On the task of verb-argument prediction in Table 2, the difference between $t_1 \to t_2$ and $e_1 \to e_2[0]$ is fairly marginal. This raises the general question of how much explicit syntactic analysis is required for the task of event inference, particularly in the encoder/decoder setup. These results provide evidence that a sentence-level RNN language model which operates on raw tokens can predict what comes next in a document as well or nearly as well as an event-mediated script model.

5 Future Work

There are a number of further extensions to this work. First, in this work (and, more generally, Neural Machine Translation research), though generated text is evaluated using BLEU, systems are optimized for per-token cross-entropy error, which is a different objective (Luong et al. (2016) give an example of a system which improves cross-entropy error but reduces BLEU score in the Neural Machine Translation context). Finding differentiable objective functions that more directly target more complex evaluation metrics like BLEU is an interesting future research direction.

Relatedly, though we argue that BLEU is a natural token-sequence-level analog to the verb-argument formulation of the Narrative Cloze task, it is not obviously the best metric for evaluating inferences of text, and comparing these automated metrics with human judgments is an important direction of future work. Pichotta and Mooney (2016) present results on crowdsourced human evaluation of script inferences that could be repeated for our RNN models.

Though we focus here on forward-direction models predicting successor sentences, bidirectional encoder-decoder models, which predict sentences from both previous and subsequent text, are another interesting future research direction.

6 Related Work

The use of scripts in AI dates back to the 1970s (Minsky, 1974; Schank and Abelson, 1977); in this conception, scripts were composed of complex events with no probabilistic semantics, which were difficult to learn automatically. In recent years, a growing body of research has investigated learning probabilistic co-occurrence models with simpler events. Chambers and Jurafsky (2008) propose a model of co-occurrence of (verb, dependency) pairs, which can be used to infer such

Input:	As of October 1 , 2008 , ⟨OOV⟩ changed its company name to Panasonic Corporation.
Gold:	⟨OOV⟩ products that were branded "National" in Japan are currently marketed under the "Panasonic" brand.
Predicted:	The company's name is now ⟨OOV⟩.
Input:	White died two days after Curly Bill shot him.
Gold:	Before dying, White testified that he thought the pistol had accidentally discharged and that he did not believe that Curly Bill shot him on purpose.
Predicted:	He was buried at ⟨OOV⟩ Cemetery.
Input:	The foundation stone was laid in 1867.
Gold:	The members of the predominantly Irish working class parish managed to save £700 towards construction, a large sum at the time.
Predicted:	The ⟨OOV⟩ was founded in the early 20th century.
Input:	Soldiers arrive to tell him that ⟨OOV⟩ has been seen in camp and they call for his capture and death.
Gold:	⟨OOV⟩ agrees .
Predicted:	⟨OOV⟩ is killed by the ⟨OOV⟩.

Figure 4: Sample next-sentence text predictions. ⟨OOV⟩ is the out-of-vocabulary pseudo-token, which frequently replaces proper names.

pairs from documents; Jans et al. (2012) give a superior model in the same general framework. Chambers and Jurafsky (2009) give a method of generalizing from single sequences of pair events to collections of such sequences. Rudinger et al. (2015) apply a discriminative language model to the (verb, dependency) sequence modeling task, raising the question of to what extent event inference can be performed with standard language models applied to event sequences. Pichotta and Mooney (2014) describe a method of learning a co-occurrence based model of verbs with multiple coreference-based entity arguments.

There is a body of related work focused on learning models of co-occurring events to automatically induce templates of complex events comprising multiple verbs and arguments, aimed ultimately at maximizing coherency of templates (Chambers, 2013; Cheung et al., 2013; Balasubramanian et al., 2013). Ferraro and Van Durme (2016) give a model integrating various levels of event information of increasing abstraction, evaluating both on coherence of induced templates and log-likelihood of predictions of held-out events. McIntyre and Lapata (2010) describe a system that learns a model of co-occurring events and uses this model to automatically generate stories via a Genetic Algorithm.

There have been a number of recent published neural models for various event- and discourse-related tasks. Pichotta and Mooney (2016) show that an LSTM event sequence model outperforms previous co-occurrence methods for predicting verbs with arguments. Granroth-Wilding and Clark (2016) describe a feedforward neural network which composes verbs and arguments into low-dimensional vectors, evaluating on a multiple-choice version of the Narrative Cloze task. Modi and Titov (2014) describe a feedforward network which is trained to predict event orderings. Kiros et al. (2015) give a method of embedding sentences in low-dimensional space such that embeddings are predictive of neighboring sentences. Li et al. (2014) and Ji and Eisenstein (2015), use RNNs for discourse parsing; Liu et al. (2016) use a Convolutional Neural Network for implicit discourse relation classification.

7 Conclusion

We have given what we believe to be the first systematic evaluation of sentence-level RNN language models on the task of predicting held-out document text. We have found that models operating on raw text perform roughly comparably to identical models operating on predicate-argument event structures when predicting the latter, and that text models provide superior predictions of raw text. This provides evidence that, for the task of held-out event prediction, encoder/decoder models mediated by automatically extracted events may not be learning appreciably more structure than systems trained on raw tokens alone.

Acknowledgments

Thanks to Stephen Roller, Amelia Harrison, and the UT NLP group for their help and feedback. Thanks also to the anonymous reviewers for their very helpful suggestions. This research was supported in part by the DARPA DEFT program under AFRL grant FA8750-13-2-0026.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR 2015)*.

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)*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 789–797.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-09)*, pages 602–610.

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)*.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-13)*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning Workshop*.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources & Evaluation (LREC-2006)*, volume 6, pages 449–454.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14:179–211.

Francis Ferraro and Benjamin Van Durme. 2016. A unified Bayesian model of scripts, frames and language. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? Event prediction using a compositional neural network model. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*.

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

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 (EACL-12)*, pages 336–344.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributional semantics for discourse relations. *Transactions of the Association for Computational Linguistics (TACL)*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07) Companion Volume: Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2061–2069, October.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the 4th International Conference on Learning Representations (ICLR-16)*.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 1562–1572.

Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Cernockỳ. 2011. Empirical evaluation and combination of advanced language modeling techniques. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association 2011 (INTERSPEECH 2011)*, pages 605–608.

Marvin Minsky. 1974. A framework for representing knowledge. Technical report, MIT-AI Laboratory.

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, MD, USA.

J Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, and Thomas G Dietterich. 2014. Learning scripts as Hidden Markov Models. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 311–318.

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)*, pages 220–229.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

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-15)*.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum and Associates.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS-14)*, pages 3104–3112.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*, pages 2755–2763.

A Supplemental Material

Our Wikipedia dump from which the training, development, and test sets are constructed is from Jan 2, 2014. We parse text using version 3.3.1 of the Stanford CoreNLP system. We use a vocab consisting of the 50,000 most common tokens, replacing all others with an Out-of-vocabulary pseudotoken. We train using batch stochastic gradient descent with momentum with a batch size of 10 sequences, using an initial learning rate of 0.1, damping the learning rate by 0.99 any time the previous hundred updates' average test error is greater than any of the average losses in the previous ten groups of hundred updates. Our momentum parameter is 0.95. Our embedding vectors are 100-dimensional, and our LSTM hidden state is 500-dimensional. We train all models for 300k batch updates (with the exception of the models compared in §4.3, all of which we train for 150k batch updates, as training is appreciably slower with longer input sequences). Training takes approximately 36 hours on an NVIDIA Titan Black GPU.