Statistical Script Learning with Recurrent Neural Nets

Karl Pichotta
University of Texas at Austin
pichotta@cs.utexas.edu

Doctoral Dissertation Proposal

December 17, 2015

Abstract

Statistical Scripts are probabilistic models of sequences of events. For example, a script model might encode the information that the event "Smith met with the President" should strongly predict the event "Smith spoke to the President." We present a number of results improving the state of the art of learning statistical scripts for inferring implicit events. First, we demonstrate that incorporating multiple arguments into events, yielding a more complex event representation than is used in previous work, helps to improve a co-occurrence-based script system's predictive power. Second, we improve on these results with a Recurrent Neural Network script sequence model which uses a Long Short-Term Memory component. We evaluate in two ways: first, we evaluate systems' ability to infer held-out events from documents (the "Narrative Cloze" evaluation); second, we evaluate novel event inferences by collecting human judgments.

We propose a number of further extensions to this work. First, we propose a number of new probabilistic script models leveraging recent advances in Neural Network training. These include recurrent sequence models with different hidden unit structure and Convolutional Neural Network models. Second, we propose integrating more lexical and linguistic information into events. Third, we propose incorporating discourse relations between spans of text into event co-occurrence models, either as output by an off-the-shelf discourse parser or learned automatically. Finally, we propose investigating the interface between models of event co-occurrence and coreference resolution, in particular by integrating script information into general coreference systems.

Contents

1 Introduction

Natural Language documents are written with the clear assumption that readers will be able to seamlessly perform many different types of inferences in order to make sense of the text. If automated systems are to answer questions about Natural Language documents, they must be able to make the same sorts of inferences. Consider the following example:

(1) Following the Battle of Actium, Octavian invaded Egypt. As he approached Alexandria, Antony's armies deserted to Octavian on August 1, 30 BC.[1]

From Example 1, we would like to be able to infer that "Octavian defeated Antony," which is not explicitly stated in the document (it is assumed that the reader will infer this from the fact that the latter's armies deserted). What we mean by "we would like to infer" this fact is that a competent automated Question Answering system should be able to answer "yes" to the query "Did Octavian defeat Antony?" A Statistical Script model encodes statistical information about prototypical sequences of events; ideally, for example, such a system will encode the fact that if X's armies desert to Y, then it is very probable that Y defeats X. This would allow an automated system to make the inference we desire in Example 1.

In this proposal, we describe a number of improvements on the state of the art of statistical script learning for implicit event inference, first by enriching the structure of the events modeled by scripts (Section 3) and, second, by applying Recurrent Neural Nets to the task (Section 4). We go on to propose a number of areas of further research. First, we propose improving the Neural Net models used for script learning (Section 5.1). Second, we propose further enrichments of events to encode more useful information (Section 5.2). Third, we propose incorporating discourse relations between events, expanding from a single global notion of event co-occurrence to more nuanced types of co-occurrence depending on discursive structure (Section 5.3). Finally, we propose integrating script information into noun coreference systems (Section 5.4).

2 Background and Related Work

This proposal concerns advances in the learning of statistical scripts, in particular by leveraging recent advancements in the art of training large neural nets on large amounts of data. We first give a background on prior work in statistical script learning, and then provide a short summary of Recurrent Neural Networks relevant to our methods. We conclude with a short survey of recent applications of Neural Networks to discourse processing.

2.1 Statistical Script Learning

The modeling of sequences in events for reasoning in AI dates back to the 1970s. Minsky (1974) and Rumelhart (1975) provide early methods for incorporating hand-constructed notions of co-occurring events into reasoning systems. Schank and Abelson (1977) provide a particularly de-

[1] Unless specified otherwise, all examples in this paper are taken from English Language Wikipedia, sometimes with minor modifications for readability.

tailed and influential analysis of structured scripts for understanding situations in AI. These approaches are non-probabilistic and depend on complicated hand-structured world information, which results in brittle systems that cannot straightforwardly generalize to situations differing significantly from the ones encoded. Around the same time, there was a related effort to describe narratives using Story Grammars (Mandler and Johnson, 1977; Thorndyke, 1977), which are essentially Context-Free Grammars describing the structures of stories, analogous to the more familiar word-level CFGs describing the syntactic structure of sentences.

Mooney and DeJong (1985) present a non-probabilistic method of learning script structures automatically from text. This is a first step to helping obviate the need for hand-engineering knowledge structures for situation understanding. Miikkulainen (1993) proposes a Neural Network system which stores events in episodic memory and is capable of answering simple questions.

The literature about scripts following Schank and Abelson (1977) typically uses quite complex notions of events to capture the subtleties of interacting events, and the script objects themselves are typically not learned and disjoint across different situations. For example, there may be a "restaurant script," giving the stereotypical description of a diner at a restaurant, and a distinct "workday at office" script, and the two are unrelated. These scripts are non-probabilistic (that is, there is no notion of probability associated with any states or transitions). As a means of describing situations as expressed in documents, these scripts will have high precision and low recall: a document may invoke the events in a hand-written script exactly as it was written, but any variation on this rigid structure is difficult to handle. Further, since these objects are non-probabilistic, there is no way of resolving ambiguity in event inference probabilistically. For example, consider the following (constructed) examples:

(2) Nancy commutes to her job in New York City.

(3) Nancy commutes to her job in upstate New York.

Under a reasonable conception of script knowledge, we should be able to infer either *Nancy drives to her job* or *Nancy takes the subway to her job* from either sentence; however, given the differing commuting scenarios of the two locations (as, ideally, expressed in a sufficiently large text corpus), taking the subway is much more likely in the former example than in the latter. This sort of information is difficult to encode and learn without a probabilistic framework that learns from data, and is crucial for making probable inferences.

These difficulties may be addressed by simplifying event representations and adding probabilities. This is somewhat analogous to the history of automatic parsers, where theoretically satisfying lexical and grammatical frameworks with complex structure and non-probabilistic semantics are not typically used by state-of-the-art parsers, which instead use greatly simplified lexical representations with learned probabilities. Simplifying event representations enables tractable statistical learning of script models, along with probabilistic event inference.

Chambers and Jurafsky (2008) give a method of learning co-occurrence statistics between simple events in which entities engage, learning their model from a large corpus of text. Their approach follows the following general method, explained in more detail below, for learning a script model:

1. Syntactically parse a large corpus of documents.

2. Run a coreference resolution engine on each document to determine which noun phrases refer to the same entity.

3. Extract events (consisting of verbs and their entity arguments) from each document.

4. Aggregate statistics on which events frequently co-occur, involving the same entity in the same document.

The syntactic analysis in step (1) will output information on, e.g., how verbs relate to various noun phrases within a sentence, and which nouns are grammatical heads of noun phrases. The coreference resolution engine in step (2) will output which noun phrases across a document refer to the same entity. For example, in the sentence *Chaucer travelled to Picardy the next year; in 1373 he visited Florence*, a coreference system will annotate that *Chaucer* and *he* refer to the same entity. In step (3), Chambers and Jurafsky (2008) treat events as (verb, dependency) pairs, dividing these events into sets based on the entity participating. So in the above example, there is one entity (*Chaucer*) which engages in two events, (*travel*, subject) and (*visit*, subject), indicating that the same entity was grammatical subject of both of these verbs.

In step (4), a co-occurrence statistic $N(a, b)$ is calculated for all pairs of events a and b, where $N(a, b)$ is the total number of event sets from step (3) where both a and b occur, across all documents. For example, we would expect (*eat*, subject) and (*drink*, subject) to have a high N value, since entities that are mentioned being the subject of *eat* are also likely to be subjects of *drink*; however, we would expect (*eat*, object) and (*drink*, object) to be lower, since things which are eaten are not typically also drunk.

In order to infer novel (verb, dependency) event pairs which a document's entity is likely to have engaged in, Chambers and Jurafsky (2008) pick events maximizing Pointwise Mutual Information (PMI) with that entity's observed events. That is, if an entity e is involved in events $a_1, \ldots, a_\ell$ in a document, then novel events b are inferred by maximizing the objective

$$S_{pmi}(b) = \sum_{i=1}^{\ell} \text{PMI}(a_i, b) \tag{1}$$

with PMI defined in the usual way:

$$\begin{aligned} \text{PMI}(a, b) &= \log \frac{P(a, b)}{P(a)P(b)} \\ &\propto \log \frac{N(a, b)}{\left(\sum_x N(a, x)\right)\left(\sum_x N(b, x)\right)} \end{aligned}$$

Events inferred for an entity will be those which co-occur more frequently than chance with the events the entity is observed as having engaged in.

In order to evaluate a system's inferences, Chambers and Jurafsky (2008), like some subsequent work (including our work described below), use what they call the *Narrative Cloze* test, in which an observed event is held out and a system is judged by its ability to infer this held-out event, given the remaining observed events. This is somewhat like the standard use of perplexity or cross-entropy

when measuring sequence model performance: a model is judged quantitatively by its ability to statistically model observed data.

Bejan (2008) and Manshadi et al. (2008) also described systems with somewhat similar ideas roughly contemporaneously. Bejan (2008) represents events as single words, and uses Latent Dirichlet Allocation (Blei et al., 2003) to build an unsupervised generative model (where a "topic" becomes instead a "scenario" describing a notion of event co-occurrence). Manshadi et al. (2008) represent events as (verb, noun) pairs, with "verb" the main verb of a sentence, and "noun" the head noun of the verb's patient argument, based on a classifier trained on PropBank (Palmer et al., 2005). Each of these pairs is treated as an item in an event vocabulary, and a language model is trained on this the sequence of such events. Systems are evaluated on two tasks: (1) differentiating between sequences of events with their observed document order and randomly shuffled sequences of the same events; and (2) differentiating between a sequence and the same sequence with its last event replaced by a random imposter event.

Chambers and Jurafsky (2009) extend the methods of Chambers and Jurafsky (2008) in a number of ways. First, they incorporate the noun identity of arguments into their event inference objective, providing performance gains under the Narrative Cloze evaluation. We will similarly demonstrate below, in Section 4, that incorporating noun information into a more complex script model provides significant performance improvements. Second, they account for all of a document's entities when inferring novel events, rather than just a single entity; however, the cost of doing so is that they infer only bare verbs rather than more structured (verb, dependency) pairs.

Jans et al. (2012) describe a model for sequences of (verb, dependency) events, showing improvements on the Narrative Cloze evaluation over the method of Chambers and Jurafsky (2008). Unlike the latter, they take the relative ordering between events in a document into account. That is, during learning and inference, Chambers and Jurafsky (2008) treat the collection of events in which an entity engages as an unordered set, resulting in $N(a, b) = N(b, a)$ for all events a, b; on the other hand, Jans et al. (2012) account for the document order of events, so in general $N(a, b) \neq N(b, a)$. When accounting for event ordering, the task under evaluation becomes "infer an event at a position t in the observed sequence of events," rather than "infer an event co-occurring with the observed set of events." They infer such events by maximizing the objective

$$S_{bigram}(b) = \sum_{i=1}^{t} \log P(b|a_i) + \sum_{i=t+1}^{\ell} \log P(a_i|b) \tag{2}$$

where ℓ is the length of the event chain from which we are inferring novel events, and $P(b|a)$ is the learned bigram probability of observing event b in a sequence after event a:

$$\begin{aligned} P(b|a) &= \frac{P(a, b)}{P(a)} \\ &= \frac{N(a, b)}{\sum_x N(a, x)} \end{aligned}$$

where $N(a, b)$ is the *2-skip bigram count*, defined to be the number of times event b is observed following a in a training corpus with at most two intervening events. Using this objective, they

demonstrate substantial improvements over the method of Chambers and Jurafsky (2008) under the Narrative Cloze evaluation.

Rudinger et al. (2015b) demonstrate improved results in modeling chains of (verb, dependency) pair events by applying the log-bilinear language model of Mnih and Hinton (2007) to the task. This model learns, for each event type a, two dense vectors in $\mathbb{R}^d$: t_a, representing the event a when it occurs as a target event to be inferred, and c_a, representing a when it occurs as a context event used to infer other events. It also learns a real-valued bias $b_a \in \mathbb{R}$ to represent a's prior probability. The probability of an event b following a sequence of events $a_1, \ldots a_k$ is represented as a log-linear model:

$$p(b|a_1, \ldots, a_k) = \frac{1}{Z}\exp\left(t_b^T \hat{t}_{\overline{a}} + b_b\right) \tag{3}$$

with Z the partition function normalizing the distribution, and $\hat{t}_{\overline{a}}$ being a sum of the context vectors in the k-element context window, modulated pointwise by a collection of final learned vectors:

$$\hat{t}_{\overline{a}} = \sum_{i=1}^{k} m_i \circ c_{b_i}$$

where $x \circ y$ is the elementwise product of two identically-sized vectors x and y, and m_i is a vector weighting the relative importance of different dimensions when preceding an inferred event by i event. They demonstrate superior performance to Chambers and Jurafsky (2008) and Jans et al. (2012) on the Narrative Cloze evaluation.

In addition to prior work directly focused on modeling sequences of events for the goal of event inference, there are a number of related threads of prior work which we describe here more briefly. First, there is a body of work focusing on automatically inducing structured collections of events intended to be useful for information extraction. Chambers and Jurafsky (2011) gives a method for unsupervised learning of event templates, evaluating on the MUC-4 terrorism corpus. They are able to perform slot-filling[2] in an unsupervised manner, achieving system performance comparable to supervised systems. Cheung et al. (2013), Chambers (2013), and Nguyen et al. (2015) describe different generative models aimed at learning event templates and evaluated on the MUC template filling task.

There is also a body of work on learning models of co-occcurring events with the aim of interpretability. Balasubramanian et al. (2013) give an unsupervised method of learning collections of events that annotators on a crowdsourcing platform judge to be coherent. That is, where previously described methods evaluate on ability to infer held-out events or perform a slot-filling task, their method is evaluated on human coherence judgments on grounded instances of event templates. Bamman et al. (2013) describe a generative model of sequences of actions characterizing characters in films. Bamman and Smith (2014) describe a method for learning biographical models from Wikipedia, evaluating quantitatively on the task of predicting the age of a person at life events.

[2]A learned event template will be a data structure encoding, for example, the fact that bombing events have locations, perpetrators, and destroyed targets; slot-filling will identify the entity a particular text indicates is, e.g., the perprtrator.

There is also a body of work aimed at producing small, high-precision models of real-world situations from smaller corpora. Regneri et al. (2010) and Li et al. (2012) provide methods of learning directed graphs of events from human-elicited event sequences describing specific situations (for example, "visiting a doctor" or "going on a date in a movie theater"). Frermann et al. (2014) describe a hierarchical Bayesian model which is able to outperform that of Regneri et al. (2010) on the task of properly ordering event pairs. Orr et al. (2014) describe a Hidden Markov Model system which learns event structure from human-generated narratives of different household tasks. In these systems, events are either simple verbs or snippets of text. Rudinger et al. (2015a) apply a number of simple models of (verb, dependency) event pairs to a corpus of 143 short blog posts about dining experiences.

McIntyre and Lapata (2009) and McIntyre and Lapata (2010) give systems which learn event structure in an unsupervised fashion for the end goal of automatic story generation, evaluated by collecting human judgments of the generated stories. Rahman and Ng (2012) and Peng et al. (2015) find event co-occurence features to be beneficial for a limited coreference resolution problem. Adel and Schütze (2014) demonstrate that event co-occurrence information is empirically useful for the task of antonym detection.

2.2 Recurrent Neural Networks

Neural Networks (NNs) are a general class of (statistical or non-statistical) models which date back to the 1940s and 1950s (McCulloch and Pitts, 1943; Rosenblatt, 1958). Neural nets are, abstractly, functions which apply a sequence of linear and nonlinear transformations to some input data. The transformation coefficients are learned using some method which optimizes parameters to get the NN's output to match, as closely as possible, some target distribution generating a training set. Neural Nets are often conceptualized as directed graphs describing their computations, with designated input nodes and output nodes. In this respect they cosmetically resemble Probabilistic Graphical Models (PGMs), but the nodes in the graph need not (and typically do not) have strict probabilistic semantics, being instead learned deterministic functions; because of this, there is not a large overlap in the methods used to train NNs and PGMs.

Figure 1 depicts the simplest nontrivial NN with latent learned features, a feedforward NN with one hidden layer. The net is "feedforward" because the computation graph has no cycles. The intermediate hidden layer is a series of z totally latent nonlinearities (of a fixed function type, for example, sigmoids or rectified linear units) applied to a linear transformation of the input layer. The coefficients parametrizing both the linear transformations (represented by arrows between nodes) and nonlinear transformations (represented by H_i nodes) are learned from data, frequently with first-order gradient-based methods (e.g. backpropagation).

Recurrent Neural Nets (RNNs) are Neural Nets whose computation graphs contain cycles. In particular, RNN sequence models are RNNs which learn to map an arbitrarily long input sequence $x_1, \ldots, x_n$ to an output sequence $o_i, \ldots, o_n$ via a learned intermediate hidden state $z_i, \ldots, z_n$. Suppose that, for $t = 1, \ldots, n$, $x_t \in \mathbb{R}^N$, $o_t \in \mathbb{R}^M$, and $z_t \in \mathbb{R}^H$. The most basic RNN sequence

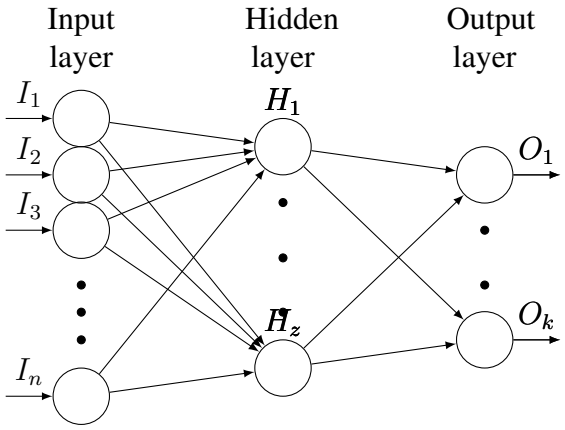

Figure 1: Single Hidden Layer Feedforward Network.

models (so-called "vanilla RNNs") 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, chosen as part of the model design process); and $W_{i,z}$, $W_{z,z}$, and $W_{z,o}$ are the appropriately-sized weight matrices describing the linear transformations of the input-to-hidden, hidden-to-hidden, and hidden-to-output connections, respectively. The cycle in the computation graph arises from the fact that z_t is a function of z_{t-1}. That is, vectors z_t and z_{t-1} are computed by the same dynamics matrix $W_{z,z}$ and the same nonlinear function f, and to calculate z_t we need z_{t-1} as input.

Vanilla RNNs are notoriously difficult to train on account of the so-called vanishing and exploding gradient problem (Hochreiter et al., 2001), the phenomenon that the gradient signal used to train the network will likely either approach zero ("vanish") or diverge ("explode") as it is propagated back through timesteps during learning, leading to instability. The spectral radius (the magnitude of the largest eigenvalue) of the hidden-to-hidden dynamics matrix $W_{z,z}$ should be around 1 for stability of learning (Sutskever et al., 2013), and this property is not necessarily straightforward to initialize or maintain. Further, long-distance data dependencies (in which some timestep's input is highly predictive of output at some much later timestep) are not well modeled by vanilla RNNs (Hochreiter et al., 2001).

Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997) sidestep both of these problems by introducing a more complicated hidden unit. The LSTM formulation we use,

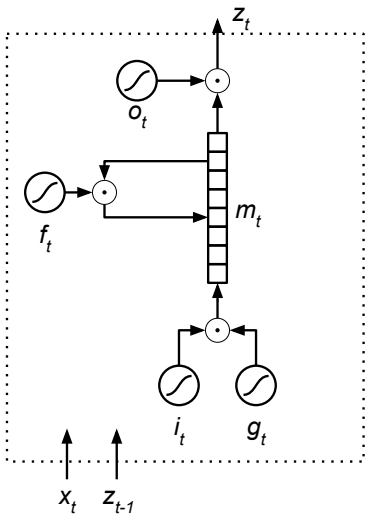

Figure 2: Graphical Depiction of Long Short-Term Memory unit at timestep t. The four nonlinearity nodes (i_t, g_t, f_t, and o_t) all have, as inputs, x_t and z_{t-1}. Small circles with dots are elementwise vector multiplications. Though all of o_t, f_t, i_t, g_t, and m_t are vectors, we only depict m_t, the memory, as a vector, for clarity.

from Zaremba and Sutskever (2014),[3] is described by the following equations (explained below):

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

These equations are depicted graphically in Figure 2. Here, as with the vanilla RNN above, we have an input vector $x_t \in \mathbb{R}^N$, an output vector $o_t \in \mathbb{R}^M$, and a hidden state vector $z_t \in \mathbb{R}^H$. Now, however, we have three additional vectors in $\mathbb{R}^H$: f_t, g_t, and m_t, the forget gate, input modulation gate, and memory cell, respectively. The vectors b_i, b_f, b_o, and b_g are constant bias vectors. The functions σ and $\tanh$ are the sigmoid and hyperbolic tangent, defined by

$$
\begin{aligned}
\sigma(x) &= \frac{1}{1+e^{-x}} \\
\tanh x &= \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1.
\end{aligned}
$$

[3] We use this formulation of LSTM because it is the one implemented in the library we used, Caffe (Jia et al., 2014), as described below in Section 4; it is a minor modification of the architecture of Graves et al. (2013), which is used in much recent published research using LSTMs.

Note that $0 \leq \sigma(x) \leq 1$ and $-1 \leq \tanh x \leq 1$. The operator $v \circ w$ denotes element-wise multiplication between two identically-sized vectors v and w. The memory cell m_t is multiplied element-wise by the forget vector f_t, whose values are between 0 and 1, calculated from the current input and the previous hidden state; this mechanism allows the network to learn to "forget" or "remember" information in the hidden state, based on the input and hidden state. The input at state t is also fed directly into m_t, modulated by the vector g_t, whose values are between -1 and 1. The only directly recurrent variable is the memory cell m_t. Note that since the LSTM unit is simply a composition of easily differentiable functions, we may use standard gradient-based methods (e.g. backpropagation) to train all of the parameters.

2.3 Neural Networks for NLP and Discourse

In recent years, LSTMs have been applied quite successfully to a number of difficult natural language problems, for example, Machine Translation (Kalchbrenner and Blunsom, 2013b; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015), Speech Recognition (Graves et al., 2013) Language Modeling (Sundermeyer et al., 2012; Kim et al., 2016) and captioning images and videos (Donahue et al., 2015; Venugopalan et al., 2015b,a).

There is a small but growing body of work applying various Neural models to various tasks in discourse processing, computationally modeling nonlocal phenomena in documents above the sentence level. Li et al. (2014), Ji and Eisenstein (2015), and Ji and Eisenstein (2014) use RNNs for discourse parsing, the task of determining how spans of text in documents relate to each other from a discourse perspective. Li and Hovy (2014) and Modi and Titov (2014) present neural models of event ordering, optimizing for the binary decision of deciding if one event precedes another in text. It is not immediately obvious how to infer novel events from these models.

Kalchbrenner and Blunsom (2013a) use RNNs to classify speech acts in dialog (classifying utterances as opinions, yes/no questions, opinions, and so on). Weston et al. (2015) describe a model with a long-term memory component, which they use to answer questions about short generated stories. Kiros et al. (2015) describe a method of mapping sentences to low-dimensional embeddings (which they call "skip-thought vectors") such that two contextually similar sentences will have similar embeddings. They do this by training an RNN to predict the previous and subsequent sentences. This is conceptually somewhat similar to the system we present in Section 4 below, but they do not evaluate directly on their system's predictive power.

3 Statistical Script Learning with Multi-Argument Events

Here, we describe a method of learning co-occurrence-based statistical scripts with more complex events modeling interactions between entities, described in more detail in Pichotta and Mooney (2014). The statistical model is a simple Markov-like co-occurrence model; Section 4 describes a more complex Neural Net system (with superior performance) which models similar events.

Napoleon	Marie Louise	Elba
(remain_married, subj)	(remain_married, prep)	
(not_join, obj)	(not_join, subj)	(not_join, prep)
(not_see, obj)	(not_see, subj)	

remain_married(n, ml, $\cdot$)
not_join(ml, n, e)
not_see(ml, n, $\cdot$)

Figure 3: (Top) Pair event representation of text in Example 4. (Bottom) Multi-argument event representation of the text.

3.1 Methods

Statistical scripts are models of co-occurring events learned from large corpora. In this setting, the precise formulation of what constitutes an "event" becomes crucial. Prior work focuses on inferring (verb, dependency) pair events (Chambers and Jurafsky, 2008; Jans et al., 2012; Rudinger et al., 2015b), (verb, noun) pair events (Manshadi et al., 2008) or simplex verb events (Bejan, 2008; Chambers and Jurafsky, 2009; Orr et al., 2014). These events are simple enough to enable straightforward learning and inference algorithms (and they are simple enough that the total number of events remains manageable); however, they are incapable of expressing fundamental aspects of event structure. Consider, for example, the following example:

(4) Napoleon remained married to Marie Louise until his death, though she did not join him in exile on Elba and thereafter never saw her husband again.

A representation of this sentence with (verb, dependency) pair events is given in Figure 3 (Top). Each column gives the sequence of pair events for a different entity in the discourse (so Napoleon and Marie Louise each engage in three different events, while Elba engages in one). Some crucial aspects of the event structure, e.g. that Napoleon and Marie Louise are married to each other (and that one didn't see the other), are not captured by this pair representation. That is, the (remain_married, subj) event and the (remain _married, obj) event are totally unrelated to each other: the pairwise interaction between the entities Napoleon and Marie Louise cannot be captured with pair events, even in principle. From (3), we may wish to infer something like "Napoleon stayed on Elba" or "Napoleon sent letters to Marie Louise;" however, it is not obvious how to represent these events with multiple arguments in this framework.

We therefore enrich our event representation by introducing *multi-argument events*, described in Pichotta and Mooney (2014). These multi-argument events are more complex than pair events, but still simple enough to enable tractable learning and inference. Figure 3 (Bottom) gives the multi-argument event representation of Example 4. There are three entity variables n, ml, and e, representing Napoleon, Marie Louise, and Elba, respectively, and these variables serve as arguments to the multi-argument predicates based on verbs. This formulation is capable of capturing relationships between different entities.

Formally, we define a multi-argument event to be a relational atom of the form $v(e_s, e_o, e_p)$, where v is a predicate and e_s, e_o, and e_p are entity variables standing in subject, direct object, and prepositional relations to the predicate v. In the present work, v will be a verb lemma (possibly phrasal), and e_s, e_o, and e_p will take values of different entities observed in documents, depending on the syntactic relation those entities have to the verb v. Any of e_s, e_o, or e_p may be *null*, indicating that no noun phrase stands in that particular relation to v. We represent null arguments with a dot "$\cdot$". For example, "Napoleon was exiled to Elba" could be represented as exile($\cdot$, n, e).

In this section, we describe the simple multi-argument event co-occurrence model from Pichotta and Mooney (2014) (in Section 4 we describe a superior model which also uses relational events). A simple objective function we will use to infer novel events is (2) from Section 2.1, which requires only a conditional distribution $P(a_2|a_1)$ describing the probability of observing event a_2 after having observed event a_1. By definition, we have

$$P(a_2|a_1) = \frac{P(a_1, a_2)}{P(a_1)}$$

where $P(a_1, a_2)$ is the probability of seeing a_1 and a_2, in order. The most straightforward way to estimate $P(a_1, a_2)$ is, if possible, by counting the number of times we observe a_1 and a_2 co-occurring and normalizing the function to sum to 1 over all pairs (a_1, a_2). For Chambers and Jurafsky (2008; 2009) and Jans et al. (2012), such an estimate is straightforward to arrive at: events are (verb, dependency) pairs, and two events co-occur when they are in the same event chain, relating to the same entity (Jans et al. (2012) further require a_1 and a_2 to be near each other). One need simply traverse a training corpus and count the number of times each pair (a_1, a_2) co-occurs. The Rel-grams of Balasubramanian et al. (2012; 2013) admit a similar strategy: to arrive at a joint distribution of pairwise co-occurrence, one can simply count co-occurrence of ground relations in a corpus and normalize.

However, given two multi-argument events of the form $v(e_s, e_o, e_p)$, this strategy will not suffice. For example, if during training we observe the two co-occurring events

(5) *ask*(*mary*, *bob*, *question*)
answer(*bob*, $\cdot$, $\cdot$)

we would like this to lend evidence to the co-occurrence of events *ask*(x, y, z) and *answer*$(y, \cdot, \cdot)$ for all distinct entities x, y, and z. If we were to simply keep the entities as they are and calculate raw co-occurrence counts, we would get evidence only for $x =$ *mary*, $y =$ *bob*, and $z =$ *question*, resulting in poor generalization.

A good deal of the relationship between the entities in two multi-argument events may be captured by paying attention to their overlapping entities. For example, to describe the relationship between the three entities in (5), it is most important to note that the object of the first event is identical with the subject of the second (namely, both are *bob*). The exact identity of the non-overlapping entities *mary* and *question* is not incredibly important for capturing the relationship between the two events.[4]

[4]Though note these intuitions break down in the presence of idiomatic singleton entities: in the event pay(*bob*, *attention*, *mary*), it is crucial to note that the direct object is *attention*; that is, "x paid y to z" fails to capture the crucial

Algorithm 1 Learning with entity substitution

```
1: for a_1, a_2 ∈ evs do
2:     N(a_1, a_2) ← 0
3: end for
4: for D ∈ documents do
5:     for a_1, a_2 ∈ coocurEvs(D) do
6:         for σ ∈ subs(a_1, a_2) do
7:             N(σ(a_1), σ(a_2)) += 1
8:         end for
9:     end for
10: end for
```

Two multi-argument events $v(e_s, e_o, e_p)$ and $v'(e'_s, e'_o, e'_p)$, share at most three entities. We thus introduce four variables x, y, z, and O. The three variables x, y, and z represent arbitrary distinct entities, and the fourth, O, stands for "Other," for entities not shared between the two events. We can rewrite the entities in our two multi-argument events using these variables, with the constraint that two identical (i.e. coreferent) entities must be mapped to the same variable in $\{x, y, z\}$, and no two distinct entities may map to the same variable in $\{x, y, z\}$. This formulation simplifies calculations while still capturing pairwise entity relationships between events.

To learn a co-occurrence-based model for novel event inference, we must count the number of times two events a_1 and a_2 co-occur. Call this count $N(a_1, a_2)$. The joint co-occurrence probability then becomes simply

$$P(a_1, a_2) = \frac{N(a_1, a_2)}{\sum_{a'_1, a'_2} N(a'_1, a'_2)}, \tag{4}$$

from which we may straightforwardly calculate a conditional probability $P(a_2|a_1)$. To calculate $N(a_1, a_2)$ in such a way that pairwise relationships are maintained and exact entity identity is abstracted away, we, upon observing a_2 following a_1 in a training document, "hallucinate" observing co-occurrences of all events a'_1 and a'_2, where a'_1 is a_1 with any subset of its entities rewritten, a'_2 is a_2 with any subset of its entities rewritten, subject to the constraint that entities overlapping between a_1 and a_2 are rewritten consistently with variables $\{x, y, z\}$.

Algorithm 1 gives this method in more formal pseudocode. The algorithm populates a co-occurrence table N, where entry $N(a_1, a_2)$ gives the co-occurrence count of events a_1 and a_2. The variable `evs` in line 1 is the set of all events in our model, which are of the form $v(e_s, e_o, e_p)$, with v a verb lemma and $e_s, e_o, e_p \in \{x, y, z, O\}$. The variable `documents` in line 4 is the collection of documents in our training corpus. The function `cooccurEvs` in line 5 takes a document D and returns all ordered pairs of co-occurring events in D, where, following the 2-skip bigram model of Jans et al. (2012), and similar to Balasubramanian et al. (2012; 2013), two events a_1 and a_2 are said to co-occur if they occur in order, in the same document, with at most two intervening events

fact that y is "attention" and interacts idiomatically with the verb. The system presented below in Section 4 better handles this.

between them.[5] The function `subs` in line 6 takes two events and returns all variable substitutions σ mapping from entities mentioned in the events a_1 and a_2 to the set $\{x, y, z, O\}$, such that two coreferent entities map to the same element of $\{x, y, z\}$. A substitution σ applied to an event $v(e_s, e_o, e_p)$, as in line 7, is defined as $v(\sigma(e_s), \sigma(e_o), \sigma(e_p))$, with the null entity mapped to itself.

Once we have calculated $N(a_1, a_2)$ using Algorithm 1, we may define $P(a_1, a_2)$ for two events a_1 and a_2, giving an estimate for the probability of observing a_2 occurring after a_1, from (4), and then define the conditional probability of seeing a_2 after having seen a_1 as:

$$\begin{aligned} P(a_2|a_1) &= \frac{P(a_1, a_2)}{\sum_{a'} P(a_1, a')} \\ &= \frac{N(a_1, a_2)}{\sum_{a'} N(a_1, a')}. \end{aligned} \tag{5}$$

3.2 Experiments

We evaluate four systems on the task of inferring held-out relational events from unseen test documents. That is, we extract a sequence of events from an unseen test document, hold one out, and judge systems by their ability to infer this event. These four systems are:

1. **Random**: This system guesses randomly selected events observed during training.

2. **Unigram**: This system ignores the observed events in test documents and infers events according to their observed frequency (that is, its most confident inference is always the most common event, the next inference is the second-most-common event, and so on).

3. **Multiple Protagonist**: This is the most direct way of guessing a full multi-argument event using a single protagonist pair-event model (a co-occurrence model of pair-events relating to a single entity). The multiple protagonist system uses a single-protagonist model to predict multi-argument events, given a sequence of known multi-argument events.

   Suppose we have a non-empty set E of entities mentioned in the known events. We use a single-protagonist system to infer additional multi-argument events involving E. A multi-argument event $a = v(e_s, e_o, e_p)$ represents three pairs: (v, e_s), (v, e_o), and (v, e_p). The multiple protagonist model scores an event a according to the score the single protagonist model assigns to these three pairs individually.

   For entity $e \in E$ in some multi-argument event in a document, we first extract the sequence of (verb, dependency) pairs corresponding to e from all known multi-argument events. For a pair d, we calculate the score $S_e(d)$, the score the single protagonist system assigns the pair d, given the known pairs corresponding to e. If e has no known pairs corresponding to it (in the cloze evaluation described below, this will happen if e occurs only in the held-out event), we fall back to calculating $S_e(d)$ with a unigram model over (verb, dependency) pair-events.

[5] Other notions of co-occurrence could easily be substituted here.

We then rank a multi-argument event $a = v(e_s, e_o, e_p)$, with $e_s, e_o, e_p \in E$, with the following objective function:

$$M(a) = S_{e_s}((v, \text{subj})) + S_{e_o}((v, \text{obj})) + S_{e_p}((v, \text{prep})) \quad (6)$$

where, for null entity e, we define $S_e(d) = 0$ for all d. In the cloze evaluation, E will be the entities in the held-out event. Each entity in a contributes independently to the score $M(a)$, based on the known (verb, dependency) pairs involving that entity.

This model is somewhat similar to the multi-participant narrative schemas described in Chambers and Jurafsky (2009), but whereas they infer bare verbs, we infer an entire multi-argument event.

4. **Joint**: Finally, we evaluate the system described in Section 3.1, which directly models the multiple entities serving as event arguments.

We follow previous work in using the narrative cloze task to evaluate statistical scripts (Chambers and Jurafsky, 2008, 2009; Jans et al., 2012; Rudinger et al., 2015b), evaluating a system on its ability to infer a held-out event given the other events in an unseen test document. In other work, the cloze task is to guess a pair event, given the other events in which the held-out pair's entity occurs. We will evaluate directly on this task of guessing pair events shortly. First, however, we evaluate on the task of guessing a multi-argument event, given all other events in a document and the entities mentioned in the held-out event. This is, we argue, the most natural way to adapt the cloze evaluation to the multi-argument event setting: instead of guessing a held-out pair event based on the other events involving its lone entity, we will guess a held-out multi-argument event based on the other events involving any of its entities.

A document may contain arbitrarily many entities. The script models we evaluate, however, only model events involving entities from a closed class of four variables $\{x, y, z, O\}$. We therefore rewrite entities in a document's sequences of events to the variables $\{x, y, z, O\}$ in a way that maintains all pairwise relationships between the held-out event and others. That is, if the held-out event shares an entity with another event, this remains true after rewriting.

We perform entity rewriting relative to a single held-out event, proceeding as follows:

- Any entity in the held-out event that is mentioned at least once in another event gets rewritten consistently to one of x, y, or z, such that distinct entities never get rewritten to the same variable.
- Any entity mentioned only in the held-out event is rewritten as O.
- All entities not present in the held-out event are rewritten as O.

This simplification removes some structure from the original sequence, but retains the pairwise entity relationships between the held-out event and the other events.

For each document, we use the Stanford dependency parser (De Marneffe et al., 2006) to get syntactic information about the document; we then use the Stanford coreference resolution engine (Raghunathan et al., 2010) to get (noisy) equivalence classes of coreferent noun phrases in a document.[6] We train on approximately 1.1M articles from years 1994-2006 of the NYT portion of

[6]We use version 1.3.4 of the Stanford CoreNLP system.

the Gigaword Corpus, Third Edition (Graff et al., 2007), holding out a random subset of the articles from 1999 for development and test sets. Our test set consists of 10,000 randomly selected held-out events, and our development set is 500 disjoint randomly selected held-out events. We use add-one smoothing on all joint probabilities. To reduce the size of our model, we remove all events that occur fewer than 50 times.[7]

We evaluate performance using the following two metrics:

1. **Recall at 10 (R10)**: Following Jans et al. (2012), we measure performance by outputting the top 10 guesses for each held-out event and calculating the percentage of such lists containing the correct answer.[8] This value will be between 0 and 1, with 1 indicating perfect system performance.

2. **Accuracy**: A multi-argument event $v(e_s, e_o, e_p)$ has four components. For a held-out event, we may judge the accuracy of a system's top guess by giving one point for getting each of its components correct and dividing by the number of possible points. We average this value over the test set, yielding a value between 0 and 1, with 1 indicating perfect system performance.

Method	**R10**	**Accuracy**
Random	0.001	0.334
Unigram	0.216	0.507
Multiple Protagonist	0.209	0.504
Joint	**0.245**	**0.549**

Table 1: Results for multi-argument events.

Table 1 gives the Recall at 10 and accuracy scores for the different systems. The unigram system is quite competitive, achieving performance comparable to the multiple protagonist system on accuracy, and superior performance on recall at 10. Evaluating by the recall at 10 metric, the joint system provides a 2.9% absolute (**13.2**% relative) improvement over the unigram system, and a 3.6% absolute (**17.2**% relative) improvement over the multiple protagonist system. These differences are statistically significant ($p < 0.01$) by McNemar's test. By accuracy, the joint system provides a 4.2% absolute (**8.3**% relative) improvement over the unigram model, and a 4.5% absolute (**8.9**% relative) improvement over the multiple protagonist model. Accuracy differences are significant ($p < 0.01$) by a Wilcoxon signed-rank test. These results provide evidence that directly modeling full multi-argument events, as opposed to modeling chains of (verb, dependency) pairs for single entities, allows us to better infer held-out verbs with all participating entities.

The "Multiple Protagonist" system adapts a baseline pair-event system to the task of guessing multi-argument events. We may also do the converse, adapting our multi-argument event system

[7] A manual inspection reveals that the majority of these removed events come from noisy text or parse errors.

[8] Jans et al. (2012) instead use recall at 50, but we observe, as they also report, that the comparative differences between systems using recall at k for various values of k is similar.

to the task of guessing the simpler pair events. That is, we infer a full multi-argument event and extract from it a (subject,verb) pair relating to a particular entity. This allows us to compare directly to previously published methods which infer pair-events.

Method	**R10**	**Accuracy**
Random	0.001	0.495
Unigram	0.297	0.552
Single Protagonist	0.282	0.553
Joint Pair	**0.336**	**0.561**

Table 2: Results for pair events.

Table 2 gives the comparative results for on the task of inferring held-out pair events. The **random** and **unigram** systems are analogous to the identically-named multi-argument systems, but on pair events instead. The **single protagonist** system is our reimplementation of the methods of Jans et al. (2012), maximizing the objective (2) (the "Multiple Protagonist" model pieces together inferences from this single protagonist model). The **joint pair** system takes the multi-argument events guessed by the joint system and converts them to pair events by discarding any information not related to the target entity; that is, if the held-out pair event relates to an entity e, then every occurrence of e as an argument of a guessed multi-argument event will be converted into a single pair event, scored identically to its original multi-argument event. Ties are broken arbitrarily. The test set is constructed by extracting one pair event from each of the 10,000 multi-argument events in the test set used in Table 1, such that the extracted pair event relates to an entity with at least one additional known pair event.

Somewhat to our surprise, on the task of inferring pair events, the joint system provides a 3.9% absolute (**13.1**% relative) improvement over the unigram baseline, and a 5.4% absolute (**19.1**% relative) improvement over the single protagonist system, according to R10. These differences are significant ($p < 0.01$) by McNemar's test. By accuracy, the joint system provides a 0.9% absolute (**1.6**% relative) improvement over the unigram model, and a 0.8% absolute (**1.4**% relative) improvement over the single protagonist model. Accuracy differences are significant ($p < 0.01$) by a Wilcoxon signed-rank test.

These results indicate that modeling more complex multi-argument event sequences allows better inference of simpler pair events. These performance improvements may be due to the fact that the joint model conditions on information not representable in the single protagonist model (namely, all of the events in which a multi-argument event's entities are involved).

4 Script Learning with Recurrent Neural Networks

In this section, we describe an LSTM-based script system, described in more detail in Pichotta and Mooney (2016), which provides superior performance to the simpler co-occurrence-based system from Section 3.

The co-occurrence joint model of Section 3 has a number of shortcomings which the LSTM-based model we present below can in principle address:

1. The Joint model does not decompose events into constituent components: when calculating co-occurrence scores between two events, the events are treated as essentially atomic, so no information is shared between similar events. So "x married y" and "x is married to y" are totally unrelated events (since they have different argument structure).

2. There is no mechanism for generalization beyond the lexical level. For example, "x journeyed to y" and "x traveled to y" are only related insofar as they might co-occur with each other; the fact that "x arrived at y" is quite likely to co-occur with both can only be learned if the latter is observed directly with both of them.

3. The noun identity of entity arguments is entirely ignored. So, for example, "she sits on the chair" and "she sits on the board of directors" will get identical representations. This is a considerable shortcoming, given that the most common verbs (which account for a very large portion of event tokens) are typically very polysemous, and incorporating noun information about arguments can provide useful cues for sense disambiguation.

4. There is only a single notion of "event co-occurrence," so, during inference, the relative position of an observed event to an inferred event (whether it immediately precedes or is relatively far back) is ignored.

5. The model cannot infer events not observed exactly in the training set, nor can it infer events never observed co-occurring with observed events (modulo smoothing effects, which typically do not greatly affect top inferences).

6. Finally, there is a particularly important special case of Point 3 above, namely, that nouns which behave idiomatically with other event components are ignored. For example, "the hurricane made landfall" and "the elves made shoes" will get identical representations, though "make landfall" is an idiomatic light verb construction whose semantics are very poorly captured by only using the verb "make." As such constructions are quite common in English (Butt, 2010), this is an important consideration.

Though the Neural Net model we propose in this section handily outperforms the co-occurrence model from Section 3, the extent to which the performance improvement is actually due to addressing any of these issues in particular is not straightforward to determine. Though we hope to perform a further analysis to tease these issues apart, we leave any such analysis to future work.

4.1 Methods

Motivated by these concerns, we propose using a Recurrent Neural Net model (in particular, an LSTM, described in Section 2.2) to statistically model events in sequence. This model will be capable of incorporating noun information about event arguments and will give verbs and nouns low-dimensional embeddings, capturing predictive similarity between words. Critically, the model's

inferences are generated from a continuous-valued hidden state vector to model the dynamics of event sequences, rather than simple Markov associations between surface forms of events. The qualitative analysis we provide below in Section 4.2 will provide evidence that learned latent states capture fairly long-range dependencies between events.

We frame script learning as an RNN sequence modeling task, using the standard technique of training an RNN sequence model to sequentially predict the next input. At timestep t, the model is trained to predict the input at timestep $t+1$. The sequence modeled is the sequence of 5-component events (in this section we model events as atoms $v(e_s, e_o, e_p, p)$, where we add p, the preposition relating e_p to v). That is, a sequence of N events has $5N$ timesteps.

We differentiate between two types of script systems based on what the models predict. **Noun models** learn to predict events as verb lemmas, noun lemmas, and prepositions. **Entity models** learn to predict verbs, entity IDs, and prepositions, where an *entity ID* is an integer identifying an argument's entity according to a coreference resolution engine. This is similar to the entity-based systems in Section 3; however, where before we rewrore entities relative to a single (held-out) event's entities, here we assign global entity IDs instead and do not perform any entity rewriting.

For example, suppose we observe the two co-occurring events

(*pass, senate, bill*$_1$, $\cdot$, $\cdot$)
(*veto, president, it*$_1$, $\cdot$, $\cdot$)

where subscripts indicate entity IDs. An LSTM noun model will be trained to model the sequence

(*pass, senate, bill,* $\cdot$, $\cdot$, *veto, president, it,* $\cdot$, $\cdot$)

by successively predicting the next element in the sequence (when receiving *pass* as input, it is trained to predict *senate*; in the next timestep it is trained to predict *bill*, and so on). An LSTM entity model will be trained to predict

(*pass,* 0, 1, $\cdot$, $\cdot$, *veto,* 0, 1, $\cdot$, $\cdot$)

where 0 denotes singleton nouns, and 1 is the entity ID for *bill/it*. To infer a five-element event, it suffices to infer five timesteps' output.

We consider four similar model architectures differing in inputs and outputs, depicted in Figure 4 (the inputs and outputs not present in all models have dotted lines). At each timestep t, there are multiple inputs, each of which is a one-hot vector (with one 1 and all other entries 0). First, there is the deterministic 1-of-5 input c_t, indicating which component of the event is input at t: verbs will have $c_t = 1$, subject entities $c_t = 2$, and so on. Next, there is a 1-of-V input w_t, with V the size of the vocabulary, giving the component word at timestep t (this may be a verb, a noun, a preposition, or *null*). Finally, three of the four models have a one-hot e_t input, which gives the entity ID of noun arguments according to a coreference engine. This e_t value has special values for null, singleton entities, and non-entity words (verbs and prepositions). We limit the number of entity IDs to 5,[9] treating all other entities as singletons.

[9] 98% of training sequences involve five or fewer non-singleton entities, so we lose coreference information in only 2% of sequences.

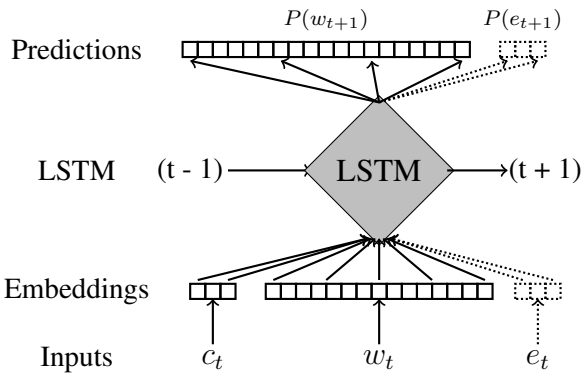

Figure 4: LSTM Script System at timestep t.

One-hot input vectors are mapped to continuous distributed representations, labeled "Embeddings" in Figure 4. These embeddings are learned jointly with the other model parameters. Predictively similar words should get similar embeddings. The embeddings are input to a recurrent LSTM unit, which modifies a latent state vector at each timestep. All models have an output vector from the LSTM, in $\mathbb{R}^V$, which is input to a softmax function, yielding a distribution over predictions for the next w_t value. Additionally, entity models have a second output vector which is input to a softmax predicting the next e_t value. We train all models by minimizing the cross-entropy error at the top softmax layer and backpropagating the error gradient through the network.

We compare four related architectures, which all receive and predict verbs and prepositions but differ in the input and output of entity arguments:

1. **LSTM-noun-noun**, which receives only noun information about arguments and learns to predict argument nouns;
2. **LSTM-ent-ent**, which receives only entity IDs and learns to predict entity IDs;
3. **LSTM-both-noun**, which receives noun and entity IDs and learns to predict nouns;
4. **LSTM-both-ent**, which receives noun and entity IDs and learns to predict entity IDs.

To generate probable event inferences, we perform a five-step beam search over the components (v, e_s, e_o, e_p, p) of events. In steps 2 through 5 of this search, the previous step's output is treated as input. Since the LSTM-both-noun and LSTM-both-ent models require both noun and entity ID information but only predict one of the two, we must generate entity ID information from predicted nouns, and vice versa. When predicting with the LSTM-both-noun model, we call any predicted non-null noun a singleton entity; when predicting with the LSTM-both-ent model, we guess the special Out-Of-Vocabulary token for any predicted non-null entities.

4.2 Experiments

As in Section 3, we evaluate using the Narrative Cloze (we will also provide human judgments of inferences below). We compare the four systems enumerated in Section 4.1 to four baseline

systems:

1. **Unigram**: Like the unigram system in Section 4, this infers events (either with or without nouns, as appropriate) according to their unigram frequency, ignoring the other elements in the test document.

2. **All-bigram**: This is the "Joint" system of Section 3, but without rewriting, and conditioning only on previous events. That is, a closed event vocabulary of the most frequent events is calculated, co-occurrence event statistics are counted from the training corpus as in Section 3, and an event a is scored as an inference at position t by maximizing the objective

$$S(a) = \sum_{i=0}^{t-1} \log P(a|s_i).$$

3. **Rewrite Bigram**: This is the All-bigram system, but we rewrite entity IDs during learning as in Section 3.

4. **2D Rewrite Bigram**: This is the Rewrite bigram system, but it optimizes the objective given in (2), incorporating events after timestep t in addition to past events. This will allow us to compare directly to the system of Section 3.

We use four different metrics, all based on the Narrative Cloze:

1. **Recall at 25 ("R25")**, as above, is the percentage of held-out events that appear in the top 25 system inferences. We relax from 10 to 25 because the task is exceptionally difficult for noun systems, which must infer the verb and head nouns of entity arguments. Note, however, that results are comparatively similar for R10 and R25.

2. **Verb recall at 25 ("R25-V")** is recall at 25, but counting an inference as correct if its verb matches the held-out event's verb (ignoring arguments).

3. **4-Tuple recall at 25 ("R25-4")** is recall at 25, ignoring prepositions. This allows us to compare directly to the methods in Section 3, which do not directly include prepositions. We evaluate LSTM systems by predicting 5-tuples and discarding prepositions, and evaluate baseline systems by directly modeling (v, e_s, e_o, e_p) 4-tuples.

4. **Accuracy with Partial Credit ("Acc")** is like "accuracy" in Section 3.2, but with partial credit for similar words. We compute a system's single most confident inference and calculate, for every component of the held-out event, a similarity score between that component and the respective inferred component. This relaxes the requirement that inferred events match exactly, which is intuitively appealing for systems that predict nouns as well as verbs. Partial credit is computed using WUP similarity (Wu and Palmer, 1994), based on distance in the WordNet hierarchy (Fellbaum, 1998). We assign a similarity score by taking the maximum WUP scores over all Synset pairs (with appropriate parts-of-speech). Accuracy is average WUP score across event components (ignoring OOVs and nulls in the held-out event). This will be between 0 and 1. We use the NLTK implementation of WUP (Bird et al., 2009)

	Entities				Nouns			
System	**R25**	**R25-V**	**R25-4**	**Acc.**	**R25**	**R25-V**	**R25-4**	**Acc.**
Unigram	0.101	0.192	0.109	0.402	0.025	0.202	0.024	0.183
All-Bigram	0.124	0.256	0.140	0.420	0.037	0.224	0.039	0.220
Rewrite Bigram	0.110	0.205	0.125	0.421	-	-	-	-
2D Rewrite Bigram	0.104	0.192	0.114	0.416	-	-	-	-
LSTM-ent-ent	0.145	0.279	0.160	0.450	-	-	-	-
LSTM-both-ent	**0.152**	**0.303**	**0.171**	**0.458**	-	-	-	-
LSTM-noun-noun	-	-	-	-	0.054	0.298	0.057	0.256
LSTM-both-noun	-	-	-	-	**0.061**	**0.300**	**0.062**	**0.260**

Table 3: Narrative Cloze results on entity and noun models, with four metrics (higher scores are better).

We use the Stanford dependency parser (De Marneffe et al., 2006) and coreference system (Raghunathan et al., 2010).[10] We represent noun arguments by their head lemmas. For a training and testing corpus, we use English Language Wikipedia,[11] breaking articles into paragraphs. We switched to Wikipedia from newswire because, first, it is larger (the RNN model is quite complex and needs a large training corpus) and, second, a qualitative analysis indicates it contains a fair amount more narrative text describing events in order than newswire does.

Our training set was approximately 8.9 million event sequences, our validation set was approximately 89,000 event sequences, and our test set was 2,000 events from 411 sequences, such that no test-set article is in the training or validation set. We add a `<s>` beginning-of-sequence pseudo-event and a `</s>` end-of-sequence pseudo-event to every sequence. The event component vocabulary comprises the 2,000 most common verbs, the 8,000 most common nouns, and the top 50 prepositions; all other words are replaced with an Out-Of-Vocabulary (OOV) token. For the unigram and bigram event vocabulary, we select the 10,000 most common events (with either nouns or entity IDs, depending on the system). We apply add-one Laplace smoothing to bigram co-occurrence counts. We use the implementation of LSTM provided by the Caffe library (Jia et al., 2014), training using batch stochastic gradient descent with momentum with a batch size of 20.

Table 3 gives results on the Narrative Cloze evaluation. The LSTM-both-ent system demonstrates a **50.0%** relative improvement (5.7% absolute improvement) over the current best-published system (2D rewritten all-bigram, evaluated using 4-Tuple event recall at 25). Note that the simpler all-bigram system outperforms the rewritten versions. This is probably because there is information encoded in the entity IDs (the relative ordering of entities, and which entities are singletons) that is lost during rewriting. Note also that, on this corpus, the 2D rewritten system, which makes predictions based on subsequent events in addition to previous events, does marginally worse than the system using only previous events. We hypothesize this is because subsequent events are less predictive than previous events on this corpus, and are comparatively overweighted.

[10] We use version 3.3.1 of the Stanford CoreNLP system in these experiments.

[11] `http://en.wikipedia.org/`, dump from Jan 2, 2014.

Compared to the strongest baselines, the best-performing entity system achieves a **22.6%** relative improvement on R25, an **18.4%** relative improvement on verb-only R25, and an **8.8%** relative improvement on accuracy with partial credit. The best-performing noun system achieves a **64.9%** relative improvement on R25, a **33.9%** relative improvement on verb-only R25, and an **18.2%** relative improvement on accuracy with partial credit. LSTM-both-ent is the best entity model, and LSTM-both-noun is the best noun model; that is, the best performing system in both cases is the one which is given both noun and entity information.

The low magnitude of the Narrative Cloze scores in Table 3 reflects the task's difficulty. The evaluation has a number of intuitive shortcomings: first, by their very nature, most obviously inferable facts are not explicitly stated in documents, and so the Narrative Cloze cannot evaluate such inferences. Further, Cloze scores on individual held-out events are not easily interpretable (if a system has difficulty inferring a single held-out event, it is unclear if it is from a system shortcoming or because the held-out event was simply inherently difficult to predict in that context).

Motivated by these concerns, we also evaluate inferences by eliciting human judgments via Amazon Mechanical Turk. Given a text snippet, annotators are asked to rate, on a 5-point Likert scale, the likelihood of inferences, with 5 signifying "Very Likely" and 1 "Very Unlikely/Irrelevant" (uninterpretable events are to be marked "Nonsense"). This provides interpretable scores, and, further, allows us to directly compare entity- and noun-predicting models, which is not straightforward using the Narrative Cloze.

We present annotators with a snippet of text and 5 phrases, 4 of which are automatic script inferences based on the events in the snippet, and one of which is a randomly selected event from the 10,000 most frequent events ("Random"). We transform relational events to English phrases using an LSTM model trained to predict, from extracted event tuples, the original text from which the event was extracted. This network uses a hidden state vector of length 1,000 and a vocabulary of 100k tokens. We elicit three judgments for each inference, treating "nonsense" judgments as 0 scores.

We asked annotators to judge each system's most confident inference not involving one of the ten most frequent verbs in the corpus.[12] We evaluate two noun-predicting systems: LSTM-both-noun and All-bigram-noun, which were the best-performing LSTM and Bigram systems on the Narrative Cloze; we also collect judgments for two entity systems, LSTM-both-ent and All-bigram-ent. We collect judgments on inferences from 100 snippets, each of which is the smallest set of initial sentences from a different paragraph in the test set such that the text contains at least two events.

The "All" column in Table 4 gives average ratings for each system. The "Filtered" column gives the results after removing annotations from annotators whose average "Random" score is higher than 1.0 (this is intended to be a simple quality-control procedure). The LSTM-both-noun system, which predicts verbs and nouns, significantly outperforms all other systems, both with and without filtering ($p < 0.05$, Wilcoxon-Pratt signed-rank test). Incorporating nouns into LSTM models improves inferences; on the other hand, bigram models, which do not decompose events into constituent components, perform worse when directly incorporating nouns, as this increases event co-occurrence sparsity.

[12] *have, make, use, include, know, take, play, call, see, give*.

System	All	Filtered
Random	2.00	0.87
All-Bigram Ent	2.87	2.87
All-Bigram Noun	2.47	2.21
LSTM-both-ent	3.03	3.08
LSTM-both-noun	**3.31**	**3.67**

Table 4: Crowdsourced results (scores range from 0 to 5).

Sequence 1 (two events):		
Event 1:	**(obtain, OOV$_1$, phd, dissertation, with)**	**__ obtained a PhD with a dissertation**
Inference 1:	(study, he, ., university, at)	He studied at a university
Inference 2:	(study, OOV, ., university, at)	__ studied at a university
Inference 3:	(study, he, ., OOV, at)	He studied at __
Event 2:	**(graduate, OOV$_1$, ., university, at)**	**__ graduated at a university**
Inference 1:	(move, he, ., OOV, to)	He moved to __
Inference 2:	(move, OOV, ., OOV, to)	__ moved to __
Inference 3:	(return, he, ., OOV, to)	He returned to __
Sequence 2 (two events):		
Event 1	**(destroy, ., airport$_1$, 1945, in)**	**The airport was destroyed in 1945.**
Inference 1:	(destroy, ., airport, ., .)	The airport was destroyed
Inference 2:	(rebuild, ., airport, ., .)	The airport was rebuilt
Inference 3:	(build, ., airport, ., .)	The airport was built
Event 2	**(open, airport$_1$, ., 1940, in)**	**The airport opened in 1940**
Inference 1:	(rename, ., airport, ., .)	The airport was renamed
Inference 2:	(know, ., ., airport, as)	. . . known as __ airport
Inference 3:	(use, ., airport, ., .)	The airport was used

Figure 5: Sample Narrative Cloze inferences. The right column gives possible English descriptions of the structured events on the left.

((pass, route, creek, north, in); (traverse, it, river, south, to))	The route passes the creek in the North It traverses the river to the South
((issue, ., recommendation, government, from); (guarantee, ., regulation, ., .); (administer, agency, program, ., .); (post, ., correction, website, through); (ensure, standard, ., ., .); (assess, ., transparency, ., .))	A recommendation was issued from the government Regulations were guaranteed The Agency administered the program A correction was posted through a website Standards were ensured Transparency was assessed.
((establish, ., ., citizen, by) (end, ., liberation, ., .) (kill, ., man, ., .) (rebuild, ., camp, initiative, on) (capture, squad, villager, ., .) (give, inhabitant, group, ., .))	Established by citizens, ... the liberation was ended A man was killed The camp was rebuilt on an initiative A squad captured a villager . . . . . . [which] the inhabitants had given the group

Figure 6: Probabilistically generated event sequences. The right column gives possible English descriptions of the structured events on the left.

Figure 5 shows, for two short two-event test sequences, the top 3 inferences the LSTM-both-noun system makes at each position (the inferences following an event are the system's top predictions of immediately subsequent events). Subscripts are entity IDs (singleton entities are unsubscripted). We do not display bigram inferences, because in these examples they are exactly the most-common unigram events, as no observed events are in the event vocabulary. These examples clearly illustrate the importance of incorporating argument noun information: for example, without nouns, *(obtain, OOV$_1$, phd, dissertation, with)* would be represented as, roughly, "someone obtained something with something," from which few reasonable inferences can be made. Note that since the learning objective does not directly encourage diversity of inferences, the LSTM makes a number of roughly synonymous inferences.

To get further intuitions for what these models learn, we can seed a model with a <s> beginning-of-sequence event and generate events by probabilistically sampling from its output predictions until it generates </s> ("ask it to generate a story"). That is, the first event component (a verb) is sampled from the model's learned distribution of first components, the hidden state is updated with this sample, the next component is sampled from the model's predictions, and so on, until a </s> is sampled. Figure 6 gives three probabilistically generated sequences from the LSTM-noun-noun model. These sequences, generated totally from scratch one component at a time, are reasonably coherent, and exhibit clear thematic dependencies across events.

5 Proposed Research

We now describe four general threads of investigation we propose undertaking, ordered by increasing scope and difficulty. First, we propose investigating a number of ways of changing the neural architecture used. Next, we propose a number of ways of adding more information to event rep-

resentations. Next, we propose incorporating discourse relations into script learning. Finally, we propose incorporating learned scripts into coreference resolution systems.

5.1 Improved Probabilistic Models

The LSTM model presented in Section 4 provides clear quantitative and qualitative improvements over the co-occurrence based system presented in Section 3. We propose a number of other models to apply to statistical script learning.

First, there are a number of alternative approaches to RNN sequences models besides LSTM. Gated Recurrent Units (GRUs, Cho et al. (2014)) adaptively "remember" and "forget" using dynamics similar to the LSTM unit, but with fewer parameters. GRUs have been shown to be competitive with LSTMs on a number of tasks (Chung et al., 2014; Jozefowicz et al., 2015). Modulo software engineering efforts, this is straightforward: GRU units can be neatly substituted for LSTM units, and may provide a small incremental improvement. Similarly, Grid LSTM networks (Kalchbrenner et al., 2015) and Gated Feedback Recurrent Neural Networks (Chung et al., 2015) can be substituted into the LSTM framework above conceptually straightforwardly. These units extend LSTM and GRU units, respectively, to have stacked hidden units capable of learning hierarchical representations with gates modulating the communication between the hidden unit layers. These units may provide marginal performance improvements.

Convolutional Neural Networks (CNNs), which convolve a set of learned kernels with input data, have, in recent years, been shown to be successful at a wide variety of image processing tasks (LeCun et al., 1998; Krizhevsky et al., 2012). In this setup, a series of layers consisting of of 2-dimensional convolution kernels (learned linear transformations) are repeatedly swept across a 2D input image, with some elementwise nonlinearities and pooling operations, resulting ultimately in a set of representations invariant under some transformations. CNNs have also found recent applications, with one-dimensional kernels, to a number of Natural Language Processing tasks (Kalchbrenner and Blunsom, 2013a; Kalchbrenner et al., 2014; Kim, 2014; Le and Mikolov, 2014; Zhang et al., 2015). Convolutional architectures provide an alternative for RNN sequence models for the script learning task which may bear investigation, in addition to improved sequence models.

There is also a recent body of work using so-called *attention-based models* for Natural Language Processing (Hermann et al., 2015; Bahdanau et al., 2015). These models incorporate some explicit learned notion of which portions of the input are most useful for making predictions. In the script learning setting, this will mean learning a distribution of explicit weights over observed events (with the weights depending on the exact identity of the events) such that more predictively important observed events have higher weights. The intuition behind this is that the extent to which distant events are predictive of new events will vary based on the exact identify of the events comprising the sequence, and incorporating this directly into a script model could prove empirically beneficial.

5.2 Improved Event Representations

The event representation presented above, in which an event is a fixed-arity tuple of the form (v, e_s, e_o, e_p), has a number of serious limitations. We propose investigating the effectiveness of

different event representations for addressing these limitations.

First, the fixed arity of the relational events presented in Section 3, which are always 5-tuples, is a clear shortcoming. For example, the *travel* event in the sentence

(6) In 1697, Peter the Great traveled incognito to Europe on an 18-month journey with a large Russian delegation to seek the aid of the European monarchs.

cannot be represented, as there are multiple prepositional phrases modifying the verb *travel*. Using the relational events presented in Section 3, at most one prepositional phrase will be represented. We will therefore extend the relational event structure to variadic events, representing the event in (6) as, for example, *(travel, Peter, ·, (in 1697), (to Europe), (on journey), (with delegation))*. Using the co-occurrence-based approach of Section 3, this is highly nontrivial, as it would cause a combinatorial increase in the event vocabulary size; however, the RNN-based approach presented in Section 4 straightforwardly admits this extension without increasing the vocabulary size. While the RNN-based systems presented in Section 4 used fixed-arity relational events so as to be directly comparable to the approach presented in Section 3, future work will relax this limitation.

Second, many non-prepositional modifiers crucially affect the meaning of verb phrases: in (6), for example, *incognito* is an important component of the *travel* event which changes which events may be inferred from the text; however, it is not represented in the multi-argument events presented above. For a stronger example, consider the following:

(7) King Frederick William I nearly executed his son for desertion.

The adverb *nearly* is crucial to modeling this event for the purposes of event inference. Incorporating adverbs into events is straightforward in our RNN-based framework.

Next, head nouns of noun phrases are often insufficient to represent the relevant semantics of entities. Consider the following examples:

(8) Frederick helped transform Prussia from a European backwater to an economically strong and politically reformed state.

(9) Martin Luther wrote to his bishop protesting the sale of indulgences.

From (8), we would like to be able to infer, for example, that *Frederick reformed Prussia*, which will not be possible if the noun phrase *an economically strong and politically reformed state* is represented only with its grammatical head *state*. Similarly, from (9), we would like to be able to infer that *Martin Luther disapproved of indulgences*; this inference cannot be made if the noun phrase *sale of indulgences* is represented simply as *sale*. Implicit event inference is affected not only by the nouns heading observed events' arguments, but also frequently by other modifiers in the argument noun phrases. We will investigate the extent to which this helps empirical performance of event inference.

Third, the tense and aspect of events are not modeled. Whether verbs are in the past, present, or future tense may be useful signals, as well as whether they are in progressive or perfect constructions, or under modal operators. These issues are crucial to formal theories of natural language semantics, and the extent to which they are important to event inferences in the general framework presented here is an empirical one.

Fourth, nominal events (states or events expressed not as verbs but instead as nouns) are ignored. Consider the following examples, each of which contains some (bolded) stative or eventive nouns which are crucial for making inferences:

(10) One means of achieving modernization was the introduction of **taxes** for long beards and robes in September 1698.

(11) After a period of **imprisonment** in Pavia for what was deemed a treasonable **offense**, Boethius was executed in 524.

(12) In the years following his **death**, a series of **civil wars** tore Alexander's empire apart.

Nominal events are very common, and incorporating them into event inference systems will likely prove beneficial.

The extent to which implicit event inferences benefit from extending the events in these ways is clearly an empirical matter. So too is the question of whether events should be augmented only when observed, or whether models should be trained to predict this more complex structure as well. Eliciting human judgments about automatically inferred implicit events, as described in Section 4.2, is a handy way of settling these questions empirically.

Further, insofar as event inference systems may be improved by augmenting events with additional textual information, the extent to which pre-existing linguistic knowledge is required deserves investigation. For example, suppose that augmenting events with knowledge of adverbial modifiers and other adjunct information (which is most directly derived from dependency parses) ends up improving system performance. It may be the case that most of this improvement, or perhaps even more, could be accounted for by simply adding as input, say, a five-word context window surrounding the verb, allowing the RNN to learn which types of verbal modifications are predictive for the task of held-out event inference. An approach like the Skip-thought vectors of Kiros et al. (2015), which are trained to predict the sequence of words in one sentence from the sequence of words in its preceding sentence, is in a sense the limiting case of the latter approach. We will investigate comparing more unstructured events with structured events for implicit event inference.

We thus propose two general non-exclusive approaches to improving event representations: on the one hand, adding more explicit structure to events will capture more of the linguistically important structure needed to represent event semantics. On the other hand, adding more raw data to events will hopefully help disambiguate polysemous words and will provide more information to condition predictions on. A neural net model can be easily modified to take the sorts of events we propose as input; we may also wish to train models to predict more complex events as output.

5.3 Script Learning and Discourse Relations

Statistical Scripts are models of events co-occurring in text, and so are intimately related to more general notions of discourse structure which have been previously studied computationally. Consider the following examples, each of which expresses discursively important relationships between different clauses:

(13) After the war, Hannibal successfully ran for the office of suffete.

(14) The Roman cavalry won an early victory by swiftly routing the Carthaginian horses.

(15) Because the local authorities had forbidden students from forming organizations and clubs, Princip and other members of Young Bosnia met in secret.

The discourse connectives *after*, *by* and *because* in (13), (14), and (15), respectively, express important relationships between nearby events. The methods presented above ignore any explicit or implicit discourse connectives such as these. That is, in the above methods, there is only one type of relationship between events; however, differentiating between different types of event co-occurrence by incorporating more fine-grained information (e.g. by noting that two events are related by the words *after* or *by*) may be a fruitful way of improving statistical script performance. We propose two general approaches to incorporating this type of discourse information into statistical script systems: incorporating off-the-shelf discourse parsers, or learning unsupervised representations of discourse connectives.

5.3.1 Incorporating Discourse Parsers

There are a number of freely-available discourse parsers, trained on different manually annotated discourse treebanks with different annotation conventions. These discourse parsers take arbitrary spans of text and automatically annotate them according to learned models of discourse structure. The RST Discourse Treebank (Carlson et al., 2001) is a corpus of text annotated in the style of Rhetorical Structure Theory (RST, (Mann and Thompson, 1988)). In this corpus, each document is annotated with a single tree describing the structure of the discourse, with spans of text at the leaves and labels from a closed set of types labeling edges. For example, in the following example (from the RST Discourse Treebank), the two marked spans of text are annotated as being related by a *condition* relation, indicating that the second marked span's truth entails the truth of the first:

(16) [S.A. brewing would make a takeover offer for all of Bell Resources] [if it exercises the option.]

The corpus has, for each document, a single tree of such annotations, with the root node representing the entire document, each leaf node representing a span of text, and intermediate nodes representing relations between sections of the document. There are a number of automatic parsers trained on this treebank which provide noisy automatic annotations according to these conventions, of which the currently best-performing is that of Ji and Eisenstein (2014), to the best of our knowledge. This parser's output was shown to be empirically useful for Sentiment Analysis by Bhatia et al. (2015).

In addition to the RST Discourse Treebank, there is one other large corpus of discourse annotations on which parsers have been trained, the Penn Discourse Treebank (PDTB) (Prasad et al., 2008). This corpus has more shallow annotations than the RST Discourse Treebank: relations between some spans of text are labeled according to a fixed set of relations, as between two leaf nodes in the RST discourse treebank, but documents do not have tree structure. The best current PDTB parser is, to the best of our knowledge, that of Wang and Lan (2015). There is also a third annotated discourse corpus, the Discourse Graphbank (Wolf et al., 2005) but, at 135 newswire documents, it is too small to realistically train a broad-coverage discourse parser.

We propose two general approaches to integrating the input of an off-the-shelf discourse parser into statistical script learning systems. First, shallow annotations (pairwise labels between spans of text, such as *condition* in the above example) can be straightforwardly incorporated into the RNN framework presented above: an event's representation can be augmented to indicate that it bears a certain labeled discourse relation with another event, similar to how coreference information between entities was encoded in section 4. Second, the deeper tree-structured discourse structures output by an RST parser can be used to define the topology of a neural net. That is, instead of a chain-structured LSTM modeling the discourse, a tree-structured LSTM (Tai et al., 2015; Zhu et al., 2015) could be used, with the network's tree structure dictated by the RST parser's output. This is similar to the approach used by Bhatia et al. (2015) for sentiment analysis, but they use a simpler RNN and propagated a single scalar latent variable, instead of a vector-valued latent state.

5.3.2 Incorporating Learned Discourse Structure

Off-the-shelf discourse parsers capture one notion of discourse structure, namely, the discourse structure which the designers of the treebank annotation conventions decided was generally important, without a particular end task in mind. However, this is not necessarily the discourse information which is most useful for the task of predicting implicit events from text. Further, the RST Discourse Treebank and the PDTB are both annotations of Wall Street Journal news articles from the Penn Treebank (Marcus et al., 1999), which have a distinct formal register and are largely about finance. It is therefore unclear to what extent these discourse parsers transfer successfully to different domains, and it is unclear to what extent the discourse structure annotated in these treebanks is the most useful structure for this task.

To address these concerns, we propose, as an alternative approach to using off-the-shelf discourse parsers, incorporating raw discourse cues explicitly into the model and learning relevant latent features of these discourse cues in addition to the event representations. One simple approach is to note when a member of a closed set of discourse connectives (e.g. *before*, *after*, *because*) relate events to each other syntactically, and add this additional structure to the input event sequence. One interesting consequence of such an approach is that we would have the ability to make queries of the form "what events can we infer as likely from a sequence of events and, additionally, are related to a particular event via the connective *because*?" This could provide an unsupervised way to infer a limited notion of, e.g., causality in event inference. This is similar in spirit to the approach taken by Narasimhan and Barzilay (2015) to answer short multiple-choice questions about passages. This could form an interesting alternative automated evaluation to the Narrative Cloze.

Alternately, a set of discourse cues could be discovered automatically from a corpus using some number of methods. For example, we could select as our set of discourse connectives the most common prepositions and subordinating conjunctions relating verbs to each other and input the presence of these discourse connectives between events in documents as proposed above.

We propose adding these automatically learned discourse cues as labels relating contiguous events to each other, under the assumption that basic discourse structure (as expressed in the syntactic relations between events in text) is crucial for understanding the semantics of how events relate to each other, and therefore predictively useful for inferring additional events.

5.4 Script Learning and Coreference

Finally, we propose using statistical models of events in sequence to empirically improve coreference resolution, the well-studied task of determining which noun phrases refer to the same entity. To motivate this, consider this example, with noun phrases coreferring to *Voltaire* bolded:

(17) **Voltaire**, pretending to work in Paris as an assistant to a notary, spent much of **his** time writing poetry. When his father found out, he sent **Voltaire** to study law, this time in Caen, Normandy. Nevertheless, **he** continued to write, producing essays and historical studies.

The final occurrence of the pronoun *he* is, in the coreference system we use, misclassified as coreferent with *his father* rather than *Voltaire*. This is likely because the main clause in the previous sentence also has as subject the pronoun *he* which does, indeed, corefer with *his father*, and this is a very strong cue for coreference. In this example, it is actually quite difficult to correctly resolve the final pronoun without a reasoning step concluding that the person mentioned as *continuing to write* in the last sentence is probably the same person mentioned *writing* in the first.

Information of this sort is ideally present in the script systems presented above: assuming events are extracted properly, the RNN script model will ideally encode the fact that it's quite probable that an entity will engage in a "continue to write" activity after engaging in a "write" activity. Conceivably, a statistical script system could be incorporated into a coreference resolution system to help make these sorts of inferences requiring event co-occurrence world knowledge. There are many conceivable methods for this integration; the exact means of incorporating scripts into coreference systems which is the most empirically beneficial is unclear. If the coreference system is a feature-based machine learning classifier, script system probabilities could be integrated directly as features. The Berkeley Coreference system (Durrett and Klein, 2013), for example, is a State-of-the-Art ML-based system which admits the introduction of additional mention features. One way to incorporate an RNN-based script model into this system is to include, as a feature of noun mention pairs, the probabilities which a coreference-aware script model assigns to the sequence of events with the nouns made coreferent and, on the other hand, the probability assigned to the sequence where the nouns do not corefer. There are also a number of conceivable ways of incorporating noun-only RNN model probabilities as features to coreference systems. We propose investigating the issue further.

6 Conclusion

The sorts of inferences made by statistical script systems are a prerequisite for robust Question-Answering systems. Other tasks, for example coreference resolution, semantic role labeling, and, potentially, syntactic parsing, could benefit from mature statistical script systems. We have presented a number of improvements of statistical script systems and have proposed a number of topics for further investigation.

We first gave results indicating that incorporating interactions between entities involved in events can help provide improved implicit event inferences from documents and, second, results indicating that LSTM Recurrent Neural Nets handily outperform simple co-occurrence based systems on the task. We proposed four extensions of further work, ordered in increasing scope. First,

we proposed investigating different probabilistic Neural Net models. Second, we proposed further improvements to event representations to incorporate more textual information. Third, we proposed incorporating discourse connectives, either from an off-the-shelf discourse parser or with simpler formulations, perhaps learned automatically. Finally, we proposed integrating script information with coreference systems.

Ultimately, the utility of statistical script systems will come from their integration into other systems of more interest as NLP end-tasks, for example Question Answering systems. Demonstrating utility for Question Answering would likely require a large corpus of questions and answers about implicit events in text, which would be a complex resource whose creation is beyond the scope of the proposed work. Nonetheless, improving statistical script inferences *per se* is an important first step.

References

Heike Adel and Hinrich Schütze. Using mined coreference chains as a resource for a semantic task. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, 2014.

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

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. Rel-grams: a probabilistic model of relations in text. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction at NAACL-HLT 2012 (AKBC-WEKEX 2012)*, 2012.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*, 2013.

David Bamman and Noah Smith. Unsupervised discovery of biographical structure from text. *Transactions of the Association for Computational Linguistics (TACL)*, 2, 2014.

David Bamman, Brendan O'Connor, and Noah A. Smith. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*, 2013.

Cosmin Adrian Bejan. Unsupervised discovery of event scenarios from texts. In *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference (FLAIRS-08)*, 2008.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. Better document-level sentiment analysis from RST discourse parsing. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP-15)*, 2015.

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

Miriam Butt. The light verb jungle: still hacking away. *Complex Predicates: Cross-Linguistic Perspectives on Event Structure*, pages 48–78, 2010.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, 2001.

Nathanael Chambers. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*, 2013.

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

Nathanael Chambers and Dan Jurafsky. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11)*, 2011.

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

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

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-06)*, volume 6, 2006.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR-15)*, 2015.

Greg Durrett and Dan Klein. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*, 2013.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. 1998.

Lea Frermann, Ivan Titov, and Manfred Pinkal. A hierarchical Bayesian model for unsupervised induction of script knowledge. In *Proceedings of the 14th Conference of the European Chapter of the Association for computational Linguistics (EACL-14)*, 2014.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. *English Gigaword Third Edition*. Linguistic Data Consortium, 2007.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

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

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

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. IEEE Press, 2001.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine 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 (EACL-12)*, 2012.

Yangfeng Ji and Jacob Eisenstein. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*, 2014.

Yangfeng Ji and Jacob Eisenstein. One vector is not enough: Entity-augmented distributional semantics for discourse relations. *Transactions of the Association for Computational Linguistics (TACL)*, 3, 2015.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015.

Nal Kalchbrenner and Phil Blunsom. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality (CVSC-13)*, 2013a.

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*, 2013b.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*, 2014.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, 2014.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS-15)*, 2012.

Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Boyang Li, Stephen Lee-Urban, Darren Scott Appling, and Mark O Riedl. Crowdsourcing narrative intelligence. *Advances in Cognitive Systems*, 2:25–42, 2012.

Jiwei Li and Eduard Hovy. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, 2014.

Jiwei Li, Rumeng Li, and Eduard Hovy. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, 2014.

Jean M. Mandler and Nancy S. Johnson. Remembrance of things parsed: Story structure and recall. *Cognitive Psychology*, 9(1):111–151, 1977.

William C. Mann and Sandra A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

Mehdi Manshadi, Reid Swanson, and Andrew S Gordon. Learning a probabilistic model of event sequences from internet weblog stories. In *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference (FLAIRS-08)*, 2008.

Mitchell Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. *Treebank-3 LDC99T42*. Linguistic Data Consortium, 1999.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

Neil McIntyre and Mirella Lapata. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-09)*, 2009.

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

Risto Miikkulainen. *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory*. MIT Press, 1993.

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

Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on machine Learning (ICML-07)*, 2007.

Ashutosh Modi and Ivan Titov. Inducing neural models of script knowledge. In *Proceedings of the 18th Conference on Computational Natural Language Learning (CoNLL-14)*, 2014.

Raymond J. Mooney and Gerald F. DeJong. Learning schemata for natural language processing. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 681–687, 1985.

Karthik Narasimhan and Regina Barzilay. Machine comprehension with discourse relations. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics (ACL-15)*, 2015.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. Generative event schema induction with entity disambiguation. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics (ACL-15)*, 2015.

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

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1), 2005.

Haoruo Peng, Daniel Khashabi, and Dan Roth. Solving hard coreference problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-15)*, 2015.

Karl Pichotta and Raymond J. Mooney. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for computational Linguistics (EACL-14)*, 2014.

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

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. The Penn discourse treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC-08)*, 2008.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*, 2010.

Altaf Rahman and Vincent Ng. Resolving complex cases of definite pronouns: the Winograd schema challenge. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP-12)*, 2012.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, July 2010.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 1958.

Rachel Rudinger, Vera Demberg, Benjamin Van Durme, and Manfred Pinkal. Learning to predict script events from domain-specific text. In *Proceedings of the 4th Joint Conference on Lexical and Computational Semantics*, 2015a.

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

David Rumelhart. Notes on a schema for stories. *Representation and understanding: Studies in Cognitive Science*, 1975.

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

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH-12)*, 2012.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*, 2015.

Perry W. Thorndyke. Cognitive structures in comprehension and memory of narrative discourse. *Cognitive Psychology*, 9(1):77–110, 1977.

Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence – video to text. In *Proceedings of the 2015 International Conference on Computer Vision (ICCV-15)*, 2015a.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-15)*, 2015b.

Jianxiang Wang and Man Lan. A refined end-to-end discourse parser. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning (CoNLL-15) Shared Task*, 2015.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR-15)*, 2015.

Florian Wolf, Edward Gibson, Amy Fisher, and Meredith Knight. *Discourse Graphbank LDC2005T08*. Linguistic Data Consortium, 2005.

Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94)*, 1994.

Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv*, 2014.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *arXiv preprint arXiv:1509.01626*, 2015.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015.