Statistical Script Learning with Multi-Argument Events

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

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

Abstract

Scripts represent knowledge of stereotypical event sequences that can aid text understanding. Initial statistical methods have been developed to learn probabilistic scripts from raw text corpora; however, they utilize a very impoverished representation of events, consisting of a verb and one dependent argument. We present a script learning approach that employs events with multiple arguments. Unlike previous work, we model the interactions between multiple entities in a script. Experiments on a large corpus using the task of inferring held-out events (the "narrative cloze evaluation") demonstrate that modeling multi-argument events improves predictive accuracy.

1 Introduction

Scripts encode knowledge of stereotypical events, including information about their typical ordered sequences of sub-events and corresponding arguments (Schank and Abelson, 1977). The classic example is the "restaurant script," which encodes knowledge about what normally happens when dining out. Such knowledge can be used to improve text understanding by supporting inference of missing actions and events, as well as resolution of lexical and syntactic ambiguities and anaphora (Rahman and Ng, 2012). For example, given the text "John went to Olive Garden and ordered lasagna. He left a big tip and left," an inference that scripts would ideally allow us to make is "John ate lasagna."

There is a small body of recent research on automatically learning probabilistic models of scripts from large corpora of raw text (Manshadi et al., 2008; Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; Jans et al., 2012). However, this work uses a very impoverished representation of events that only includes a verb and a single dependent entity. We propose a more complex multi-argument event representation for use in statistical script models, capable of directly capturing interactions between multiple entities. We present a method for learning such a model, and provide experimental evidence that modeling entity interactions allows for better prediction of events in documents, compared to previous single-entity "chain" models. We also compare to a competitive baseline not used in previous work, and introduce a novel evaluation metric.

2 Background

The idea of representing stereotypical event sequences for textual inference originates in the seminal work of Schank and Abelson (1977). Early scripts were manually engineered for specific domains; however, Mooney and DeJong (1985) present an early knowledge-based method for learning scripts from a single document. These early scripts (and methods for learning them) were non-statistical and fairly brittle.

Chambers and Jurafsky (2008) introduced a method for learning statistical scripts that, using a much simpler event representation that allows for efficient learning and inference. Jans et al. (2012) use the same simple event representation, but introduce a new model that more accurately predicts test data. These methods only model the actions of a single participant, called the *protagonist*. Chambers and Jurafsky (2009) extended their approach to the multi-participant case, modeling the events in which all of the entities in a document are involved; however, their method cannot represent interactions between multiple entities.

Balasubramanian et al. (2012; 2013) describe the Rel-gram system, a Markov model similar to that of Jans et al. (2012), but with tuples instead of (verb, dependency) pairs. Our approach is sim-

Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 220–229,
Gothenburg, Sweden, April 26-30 2014.

ilar, but instead of modeling a distribution over co-occurring verbs and nominal arguments, we model interactions between entities directly by incorporating coreference information into the model.

Previous statistical script learning systems proceed broadly as follows. For a document D:

1. Run a dependency parser on D, to match up verbs with their argument NPs.
2. Run a coreference resolver on D to determine which NPs likely refer to the same entity.
3. Construct a sequence of event objects, using syntactic and coreference information.

One can then build a statistical model of the event sequences produced by Step 3. Such a model may be evaluated using the narrative cloze evaluation, described in Section 4.1, in which we hold out an event from a sequence and attempt to infer it.

The major difference between the current work and previous work is that the event sequences produced in Step 3 are of a different sort from those in other models. Our events are more structured, as described in Section 3.1, and we produce one event sequence per document, instead of one event sequence per entity. This requires a different statistical model, as described in Section 3.2.

3 Script Models

In Section 3.1, we describe the multi-argument events we use as the basis of our script models. Section 3.2 describes a script model using these events, and Section 3.3 describes the baseline systems to which we compare.

3.1 Multi-Argument Events

Statistical scripts are models of stereotypical sequences of *events*. In Chambers and Jurafsky (2008; 2009) and Jans et al. (2012), events are (verb, dependency) pairs, forming "chains," grouped according to the entity involved. For example, the text

(1) Mary emailed Jim and he responded to her immediately.

yields two chains. First, there is a chain for Mary:

(*email*, subject)
(*respond*, object)

indicating that Mary was the subject of an *emailing* event and the object of a *responding* event. Second, there is a chain for Jim:

(*email*, object)
(*respond*, subject)

indicating that Jim was the object of an *emailing* event and the subject of a *responding* event. Thus, one document produces many chains, each corresponding to an entity. Note that a single verb may produce multiple pair events, each present in a chain corresponding to one of the verb's arguments. Note also that there is no connection between the different events produced by a verb: there is nothing connecting (*email*, subject) in Mary's chain with (*email*, object) in Jim's chain.

We propose a richer event representation, in which a document is represented as a single sequence of event tuples, the arguments of which are entities. Each entity may be mentioned in many events, and, unlike previous work, each event may involve multiple entities. For example, sentence (1) will produce a single two-event sequence, the first event representing Mary emailing Jim, and the second representing Jim responding to Mary.

Formally, an **entity** is represented by a constant, and noun phrases are mapped to entities, where two noun phrases are mapped to the same constant if and only if they corefer. A **multi-argument event** is a relational atom $v(e_s, e_o, e_p)$, where v is a verb lemma, and e_s, e_o, and e_p are possibly-null entities. The first entity, e_s, stands in a subject relation to the verb v; the second, e_o, is the direct object of v; the third e_p stands in a prepositional relation to v. One of these entities is **null** (written as "$\cdot$") if and only if no noun phrase stands in the appropriate relation to v. For example, *Mary hopped* would be represented as $\mathit{hop}(\mathit{mary}, \cdot, \cdot)$, while *Mary gave the book to John* would be $\mathit{give}(\mathit{mary}, \mathit{book}, \mathit{john})$. In this formulation, Example (1) produces the sequence

$\mathit{email}(m, j, \cdot)$
$\mathit{respond}(j, m, \cdot)$

where *m* and *j* are entity constants representing all mentions of Mary and Jim, respectively. Note that this formulation is capable of capturing interactions between entities: we directly encode the fact that after one person emails another, the latter responds to the former. In contrast, pair events can capture only that after an entity emails, they are responded to (or after they are emailed, they respond). Multi-argument events capture more of the basic event structure of text, and are therefore well-suited as a representation for scripts.

3.2 Multi-argument Statistical Scripts

We now describe our script model. Section 3.2.1 describes our method of estimating a joint probability distribution over pairs of events, modeling event co-occurrence, and Section 3.2.2 shows how this co-occurrence probability can be used to infer new events from a set of known events.

3.2.1 Estimating Joint Probabilities

Suppose we have a sequence of multi-argument events, each of which is a verb with entities as arguments. We are interested in predicting which event is most likely to have happened at some point in the sequence. Our model will require a conditional probability $P(a|a')$, the probability of seeing event a after event a', given we have observed a'. However, as described below, directly estimating this probability is more complicated than in previous work because events now have additional structure.

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 a Maximum Likelihood 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

(2) *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*.

Algorithm 1 Learning with entity substitution

```
for a1, a2 ∈ evs do
    N(a1, a2) ← 0
end for
for D ∈ documents do
    for a1, a2 ∈ coocurEvs(D) do
        for σ ∈ subs(a1, a2) do
            N(σ(a1), σ(a2)) += 1
        end for
    end for
end for
```

One approach to this problem would be to deploy one of many previously described Statistical Relational Learning methods, for example Logical Hidden Markov Models (Kersting et al., 2006) or Relational Markov Models (Anderson et al., 2002). These methods can learn various statistical relationships between relational logical atoms with variables, of the sort considered here. However, we investigate a simpler option.

The most important relationship between the entities in two multi-argument events concerns their overlapping entities. For example, to describe the relationship between the three entities in (2), 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 identity of the non-overlapping entities *mary* and *question* is not important for capturing the relationship between the two events.

We note that 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.

Algorithm 1 gives the pseudocode for the learn-

ing method. This populates a co-occurrence matrix 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 between them.[1] 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, as

$$P(a_1, a_2) = \frac{N(a_1, a_2)}{\sum_{a_1', a_2'} N(a_1', a_2')}. \quad (3)$$

We may then define the conditional probability of seeing a_2 after a_1, given an observation of a_1:

$$\begin{aligned} P(a_2|a_1) &= \frac{P(a_1, a_2)}{\sum_{a'} P(a_1, a')} \\ &= \frac{N(a_1, a_2)}{\sum_{a'} N(a_1, a')}. \end{aligned} \quad (4)$$

3.2.2 Inferring Events

Suppose we have a sequence of multi-argument events extracted from a document. A natural task for a statistical script model is to infer what other events likely occurred, given the events explicitly stated in a document. Chambers and Jurafsky (2008; 2009) treat the events involving an entity as an unordered set, inferring the most likely additional event, with no relative ordering between the inferred event and known events. We adopt the model of Jans et al. (2012), which was demonstrated to give better empirical performance. This model takes an ordered sequence of events and a position in that sequence, and guesses events that likely occurred at that position. In that work, events are (verb, dependency) pairs, and an event sequence consists of all such pairs involving a particular entity. We use this model in the multi-argument event setting, in which a document produces a single sequence of multi-argument events.

Let A be an ordered list of events, and let p be an integer between 1 and $|A|$, the length of A. For $i = 1, \ldots, |A|$, define a_i to be the ith element of A. We follow Jans et al. (2012) by scoring a candidate event a according to its probability of following all of the events before position p, and preceding all events after position p. That is, we rank candidate events a by maximizing $S(a)$, defined as

$$S(a) = \sum_{i=1}^{p-1} \log P(a|a_i) + \sum_{i=p}^{|A|} \log P(a_i|a) \quad (5)$$

with conditional probabilities $P(a|a')$ calculated using (4). Each event in $a_i \in A$ independently contributes to a candidate a's score; the ordering between a and a_i is taken into account, but the ordering between the different events $a_i \in A$ does not directly affect a's score.

3.3 Baseline Systems

We describe the baseline systems against which we compare the performance of the multi-argument script system described in section 3.2. These systems infer new events (either multi-argument or pair events) given the events contained in a document.

Performance of these systems is measured using the narrative cloze task, in which we hold out a single event (either a multi-argument or pair event), and rate a system by its ability to infer this event, given the other events in a document. The narrative cloze task is described in detail in Section 4.1.

3.3.1 Random Model

The simplest baseline we compare to is the **random baseline**, which outputs randomly selected events observed during training. This model can guess either multi-argument or pair events.

3.3.2 Unigram Model

The **unigram** system guesses events ordered by prior probability, as calculated from the training set. If scripts are viewed as n-gram models

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

over events, this baseline corresponds to a bag-of-words unigram model. In this model, events are assumed to occur independently, drawn from a single distribution. This model can be used to guess either multi-argument or pair events.

3.3.3 Single Protagonist Model

We refer to the system of Jans et al. (2012) as the **single protagonist** system. This model takes a single sequence of (verb, dependency) pair events, all relating to a single entity. It then produces a list of pair events, giving the model's top predictions for additional events involving the entity. This model maximizes the objective given in (5), with the sequence A (and the candidate guesses a) comprised of pair events.

3.3.4 Multiple Protagonist Model

The **multiple protagonist** system infers multi-argument events. While this method is not described in previous work, it is the most direct way of guessing a full multi-argument event using a single protagonist script model.

The multiple protagonist system uses a single-protagonist model, which models pair events, 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 describe the most direct method of using 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, as described in Section 3.3.2, over (verb, dependency) pair events.

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 scores a multi-argument event a by combining one independent single-protagonist model for every entity in a.

This model is 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 Evaluation

4.1 Evaluation Task

We follow previous work in using the narrative cloze task to evaluate statistical scripts (Chambers and Jurafsky, 2008; Chambers and Jurafsky, 2009; Jans et al., 2012). The task is as follows: given a sequence of events $a_1, \ldots, a_n$ from a document, hold out some event a_p and attempt to predict that event, given the other events in the sequence. As we cannot automatically evaluate the prediction of truly unmentioned events in a document, this evaluation acts as a straightforward proxy.

In the aforementioned work, the cloze task is to guess a pair event, given the other events in which the held-out pair's entity occurs. In Section 4.2.2, we evaluate directly on this task of guessing pair events. However, in Section 4.2.1, 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 model described in Section 3.2.1, however, only models 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 structure from the original sequence, but retains the important pairwise entity relationships between the held-out event and the other events.

4.2 Experimental Evaluation

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.[2] We train on approximately 1.1M articles from years 1994-2006 of the NYT portion of 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. To remove duplicate documents, we hash the first 500 characters of each article and remove any articles with hash collisions. 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.[3]

We evaluate performance using the following two metrics:

1. **Recall at 10**: 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.[4] 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; a pair event has two 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. This is a novel evaluation metric for the script learning task.

These metrics target a system's most confident predicted events: we argue that a script system is best evaluated by its top inferences.

In Section 4.2.1, we evaluate on the task of inferring multi-argument events. In Section 4.2.2, we evaluate on the task of guessing pair events.

4.2.1 System Comparison on Multi-argument Events

We first compare system performance on inferring multi-argument events, evaluated on the narrative cloze task as described in Section 4.1, using the corpora and metrics described in Section 4.2. We compare against three baselines: the uninformed **random** baseline from Section 3.3.1, the **unigram** system from 3.3.2, and the **multiple protagonist** system from Section 3.3.4.

The **joint** system guesses the held-out event, given the other events in the document that involve the entities in that held-out tuple. The system orders candidate events a by their scores $S(a)$, as given in Equation (5). This is the primary system described in this paper, modeling full multi-argument events directly.

Table 1 gives the recall at 10 ("R@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%

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

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

[4] 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.

Method	R@10	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.

Method	R@10	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.

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.

4.2.2 System Comparison on Pair Events

In Section 4.2.1, we adapted 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 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.

The **random**, **unigram**, and **single protagonist** systems are pair-event systems described in Sections 3.3.1, 3.3.2, and 3.3.3, respectively. The **joint pair** system takes the multi-argument events guessed by the joint system of Section 4.2.1 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.

Table 2 gives the comparative results for these four systems. 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 Section 4.2.1, such that the extracted pair event relates to an entity with at least one additional known pair event. Evaluating by recall at 10, 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. 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 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).

5 Related Work

The procedural encoding of common situations for automated reasoning dates back decades. The frames of Minsky (1974), schemas of Rumelhart (1975), and scripts of Schank and Abelson (1977) are early examples. These models use quite complex representations for events, with many different relations between events. They are not statistical, and use separate models for different scenarios (e.g. the "restaurant script" is different from the "bank script"). Generally, they require humans to encode procedural information by hand; see, however, Mooney and DeJong (1985) for an early method for learning scripts automatically from a document. Miikkulainen (1990; 1993) gives a hierarchical Neural Network system which stores sequences of events from text in episodic memory, capable of simple question answering.

Regneri et al. (2010) and Li et al. (2012) give methods for using crowdsourcing to create situation-specific scripts. These methods

help alleviate the bottleneck of the knowledge-engineering required for traditionally conceived script systems. These systems are precision-oriented: they create small, highly accurate scripts for very limited scenarios. The current work, in contrast, focuses on building high-recall models of general event sequences. There are also a number of systems addressing the related problem of modeling domain-specific human-human dialog for building dialog systems (Bangalore et al., 2006; Chotimongkol, 2008; Boyer et al., 2009).

There have been a number of recent approaches to learning statistical scripts. Chambers and Jurafsky (2008) and Jans et al. (2012) give methods for learning models of (verb, dependency) pairs, as described above. Manshadi et al. (2008) give an n-gram model for sequences of verbs and their patients. McIntyre and Lapata (2009; 2010) use script objects learned from corpora of fairy tales to automatically generate stories. Chambers and Jurafsky (2009) extend their previous model to incorporate multiple entities, but do not directly model the different arguments of an event. Bamman et al. (2013) learn latent character personas from film summaries, associating character types with stereotypical actions; they focus on identifying persona types, rather than event inference.

Manshadi et al. (2008) and Balasubramanian et al. (2012; 2013) give approaches similar to the current work for modeling sequences of events as n-grams. These methods differ from the current work in that they do not model entities directly, instead modeling co-occurrence of particular nouns standing as arguments to particular verbs. Lewis and Steedman (2013) build clusters of relations similar to these events, finding such clusters helpful to question answering and textual inference.

There has also been recent interest in the related problem of automatically learning event frames (Bejan, 2008; Chambers and Jurafsky, 2011; Cheung et al., 2013; Chambers, 2013). These approaches focus on identifying frames for information extraction tasks, as opposed to inferring events directly. Balasubramanian et al. (2013) give an event frame identification method, developed in parallel with the current work, using sequences of tuples similar to our multi-argument events, noting coherence issues with pair events. Their formulation differs from ours primarily in that they do not incorporate coreference information into their event co-occurrence distribution, and evaluate using human judgments of frame coherence rather than a narrative cloze test.

6 Future Work

We have evaluated only one type of multi-argument event inference, in which a script infers an event given a set of entities and the events involving those entities. We claim that this is the most natural adaptation of the cloze evaluation to the multi-argument event setting. However, other types of inferences would be useful as well for question-answering. Additional script inferences, and their applications to question answering, are worth investigating more fully.

The evaluation methodology used here has two serious benefits: it is totally automatic, and it does not require labeled data. The cloze evaluation is intuitively reasonable: a good script system should be able to predict stated events as having taken place. Basic pragmatic reasoning, however, tells us that the most obvious inferable events are not typically stated in text. This evaluation thus fails to capture some of the most important commonsense inferences. Further investigation into evaluation methodologies for script systems is needed.

7 Conclusion

We described multi-argument events for statistical scripts, which can directly encode the pairwise entity relationships between events in a document. We described a script model that can handle the important aspects of the additional complexity introduced by these events, and a baseline model that can infer multi-argument events using single-protagonist chains instead of directly modeling full relations. We introduced the novel unigram baseline model for comparison, as well as the novel accuracy metric, and provided empirical evidence that modeling full multi-argument events provides more predictive power than modeling event chains individually.

Acknowledgments

Thanks to Katrin Erk, Amelia Harrison, and the DEFT group at UT Austin for helpful discussions. Thanks also to the anonymous reviewers for their helpful comments. This research was supported in part by the DARPA DEFT program under AFRL grant FA8750-13-2-0026. Some of our experiments were run on the Mastodon Cluster, supported by NSF Grant EIA-0303609.

References

Corin R Anderson, Pedro Domingos, and Daniel S Weld. 2002. Relational Markov models and their application to adaptive web navigation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, pages 143–152.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2012. 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)*, pages 101–105.

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

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*, pages 352–361.

Srinivas Bangalore, Giuseppe Di Fabbrizio, and Amanda Stent. 2006. Learning the structure of task-driven human–human dialogs. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-06)*, pages 201–208.

Cosmin Adrian Bejan. 2008. Unsupervised discovery of event scenarios from texts. In *Prodeedings of the 21st International Florida Artificial Intelligence Research Society Conference (FLAIRS-2008)*, pages 124–129.

Kristy Elizabeth Boyer, Robert Phillips, Eun Young Ha, Michael D. Wallis, Mladen A. Vouk, and James C. Lester. 2009. Modeling dialogue structure with adjacency pair analysis and Hidden Markov Models. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Paper (NAACL-HLT-09 Short)*, pages 49–52.

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

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 602–610.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT-11)*, pages 976–986.

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

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

Ananlada Chotimongkol. 2008. *Learning the structure of task-oriented conversations from the corpus of in-domain dialogs*. Ph.D. thesis, Carnegie Mellon University.

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

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

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL-12)*, pages 336–344.

Kristian Kersting, Luc De Raedt, and Tapani Raiko. 2006. Logical Hidden Markov Models. *Journal of Artificial Intelligence Research*, 25:425–456.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

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

Mehdi Manshadi, Reid Swanson, and Andrew S Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *Prodeedings of the 21st International Florida Artificial Intelligence Research Society Conference (FLAIRS-2008)*, pages 159–164.

Neil McIntyre and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 217–225.

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

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

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

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

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

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2010)*, pages 492–501.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: the Winograd schema challenge. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-12)*, pages 777–789.

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

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

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