Copyright

by

Karl Pichotta

2017

The Dissertation Committee for Karl Pichotta

certifies that this is the approved version of the following dissertation:

Advances in Statistical Script Learning

Committee:

Raymond J. Mooney, Supervisor

Nathanael Chambers

Katrin Erk

Peter Stone

Advances in Statistical Script Learning

by

Karl Pichotta

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2017

To my father, who didn’t *quite* make it to see his son get a doctorate, and mother.

Acknowledgments

It takes a village to make a hermit. The, frankly, shocking completion of my doctoral studies would have been inconceivable without the love, support, and antagonism from so many friends, family members, and professional colleagues.

Thanks are due, first and foremost, to my advisor, Ray Mooney, who tolerated (and even encouraged!) my various efforts to cram the (partial) meanings of whole %$!@ing sentences, and even %$!@ing discourses, into single %$!@ing vectors, among other things. My intellectual debt to my advisor is great. His support and guidance have been unfailing, extremely helpful, and, suffice it to say, animated. I owe a debt of gratitude also to Vladimir Lifschitz, who worked patiently with me during my first year of graduate school on problems in formal computational logic. I learned from him a good deal about how to approach, decompose, and solve thorny and complex problems, which, it turns out, is very useful. Though, at this particular point in history, a decade ago feels like a century, I nonetheless also owe a debt of intellectual gratitude to my undergraduate honors thesis advisor from a decade ago, Lauri Karttunen, who gave me a wonderful hands-on tutorial on parsimonious linguistic analysis. I'd be remiss in not also thanking Katrin Erk, who's been consistently supportive of me and my work, and whose various conversations and suggestions over the years have always been extremely helpful.

The two summers I spent at Google, working with John DeNero and Saro Meguerdichian, had a formidable influence on me. I owe both of them, in addition to the other wonderful people I worked with there, a nontrivial debt of gratitude. Also, the various grad student co-conspirators with whom I raised Cain during those summers, including Yonatan Bisk, Sam Bowman, Greg Durrett, Jonny Weese, Alex Rudnick, and Jon Malmaud, were, and continue to be, really helpful to talk shop with, bounce dumb ideas off, and troll both online and offline. More generally, the whole NLP community, whose various scions and roustabouts are too numerous to enumerate, is a really wonderful group, and getting to know (and even become one of) them over the past years has been a wonderful and rewarding experience.

I've really wasted a tremendous amount of the time of my labmates and fellow graduate students over the past years, and without their help, attention, and friendship, there's basically no way I'd have finished my PhD. This set of people includes, but is not limited to, Wesley Tansey, Subhashini Venugopalan, Jesse Thomason, Matt Hausknecht, Yinon Bentor, Nazneen Rajani, Stephen Roller, Dan Garrette, I. Beltagy, Piyush Khandelwal, Chris Brown, Elad Liebman, Joe Reisinger, and Ayan Acharya. This is to say nothing of many other friends whose time I've further wasted in manifold other nonprofessional ways, who are numerous, but, thankfully to them, no doubt, need not be explicitly shamed here.

Finally, thanks to my family, who have been consistently supportive of me for reasons recondite; in particular, thanks to my wife Amelia, absent whom I suspect I'd have long ago moved to an off-the-grid cabin somewhere in the woods (and she, generally *mutatis mutandis*, to the desert), my speech devolving to a strange proto-language, my cosmology descending into a Judge-Holden-style *ur*-violence, &c.

Advances in Statistical Script Learning

Karl Pichotta, Ph.D.
The University of Texas at Austin, 2017

Supervisor: Raymond J. Mooney

When humans encode information into natural language, they do so with the clear assumption that the reader will be able to seamlessly make inferences based on world knowledge. For example, given the sentence “Mrs. Dalloway said she would buy the flowers herself,” one can make a number of probable inferences based on event co-occurrences: she bought flowers, she went to a store, she took the flowers home, and so on.

Observing this, it is clear that many different useful natural language end-tasks could benefit from models of events as they typically co-occur (so-called *script* models). Robust question-answering systems must be able to infer highly-probable implicit events from what is explicitly stated in a text, as must robust information-extraction systems that map from unstructured text to formal assertions about relations expressed in the text. Coreference resolution systems, semantic role labeling, and even syntactic parsing systems could, in principle, benefit from event co-occurrence models.

To this end, we present a number of contributions related to statistical event co-occurrence models. First, we investigate a method of incorporating multiple entities into events in a count-based co-occurrence model. We find that modeling multiple entities interacting across events allows for improved empirical performance on the task of modeling sequences of events in documents.

Second, we give a method of applying Recurrent Neural Network sequence models to the task of predicting held-out predicate-argument structures from documents. This model allows us to easily incorporate entity noun information, and can allow for more complex, higher-arity events than a count-based co-occurrence model. We find the neural model improves performance considerably over the count-based co-occurrence model.

Third, we investigate the performance of a sequence-to-sequence encoder-decoder neural model on the task of predicting held-out predicate-argument events from text. This model does not explicitly model any external syntactic information, and does not require a parser. We find the text-level model to be competitive in predictive performance with an event level model directly mediated by an external syntactic analysis.

Finally, motivated by this result, we investigate incorporating features derived from these models into a baseline noun coreference resolution system. We find that, while our additional features do not appreciably improve top-level performance, we can nonetheless provide empirical improvement on a number of restricted classes of difficult coreference decisions.

Table of Contents

Chapter 1

Introduction

1.1 Motivating Statistical Script Learning

A basic Gricean pragmatic analysis of language dictates that when speakers or writers encode information in natural language, very obviously inferable facts will be left implicit (Grice, 1975). Grice states that language, when used felicitously, generally follows the *cooperative principle*, which he succinctly describes as:

> Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged.[1]

In particular, Grice identifies, as a particular set of guidelines embodying this principle, the two *maxims of quantity*, which are:

1. Make your contribution as informative as is required (for the current purposes of the exchange).
2. Do not make your contribution more informative than is required.[2]

In particular, the second maxim of quantity (which is related to another Gricean maxim, that of *relation*, which states simply "be relevant") implies that the typical

[1] Grice (1975), p.43

[2] Ibid.

text has a good many obviously-inferable facts that are not explicitly stated (assuming speakers act cooperatively).

Consider the following examples, taken from English language Wikipedia:

1. Peter the Great made a temporary peace with the Ottoman Empire that allowed him to keep the captured fort of Azov.[3]
2. The Germans built field fortifications to hold the ground captured in 1914 and the French began siege warfare to break through the German defences and recover the lost territory.[4]
3. Although he became an American citizen in 1945, shortly before his death, Bartók never felt fully at home in the USA.[5]

Suppose we want to build a question-answering system that can answer questions about open-domain text (Harabagiu et al., 2000; Lin and Pantel, 2001; Hermann et al., 2015; Rajpurkar et al., 2016). What follows are some simple questions about the above examples that have clear answers, but which nonetheless require nontrivial inferences from the documents in order to answer properly:

- (About (1)): Did Peter negotiate with the Ottomans? Who captured the fort? From whom?
- (About (2)): Who did the Germans capture ground from?
- (About (3)): Did Bartók move to the US?

[3] https://en.wikipedia.org/wiki/Peter_the_Great

[4] https://en.wikipedia.org/wiki/Battle_of_Verdun

[5] https://en.wikipedia.org/wiki/Bela_Bartok

All of these questions, though they have clear answers, nonetheless require inference of implicit events (e.g. one typically negotiates before making peace, one typically moves to a country before naturalization, and so on). In other words, a sufficiently robust natural language question-answering system must have some model, explicit or implicit, of event co-occurrence, in order to answer questions like the above.

This thesis investigates methods of automatically inducing such *script models* from large corpora of unannotated text. Scripts are models of stereotypical event sequences; in this dissertation, we define a number of different models describing event sequences, and investigate different methods of parametrizing and learning such models. Such models will be evaluated intrinsically on their ability to infer held-out events from documents; we also collect human judgments of the quality of systems' automatically inferred events, and incorporate features derived from script systems into a system performing an extrinsic task.

Models of event co-occurrence information, it bears mentioning, have compelling applications to other tasks besides question answering. Chapter 6 gives an investigation into incorporating features from such models into one external task; we give a more thorough treatment of future applications in Chapter 8

1.2 Summary of Contributions

This thesis investigates models of event co-occurrence information, learned from large bodies of natural language documents. Chapter 2 describes some background and related work useful for understanding the original contributions presented in later chapters. Chapter 3 describes a count-based model of event co-occurrence which models events containing multiple entity arguments, originally described in Pichotta and Mooney (2014). This model, contra prior work, models co-occurring predicates with multiple arguments (as opposed to predicates with single positional arguments). We find that, in this regime, modeling multiple arguments yields better empirical predictions of events in documents, even single-argument events.

Chapter 4 describes a Recurrent Neural Network-based model (Pichotta and Mooney, 2016a) which, in a sense, generalizes the previous chapter's model, in that it allows for the direct incorporation of noun information, allows for a greater number of possible event types (by decomposing the simplex events of Chapter 3 into sequences of constituent components), and generalizes lexically by using word embeddings. This model provides superior performance in modeling sequences of events.

Chapter 5 investigates the use of text-level sequence-to-sequence neural encoder-decoder models for the task of event prediction, as described originally in Pichotta and Mooney (2016b). We find that system performance on predicting events is roughly comparable to that of the analogous event-level systems, and, on the task of predicting the raw text describing events, the systems which directly model text provide superior performance.

The results from Chapter 5 indicate, among other things, that event co-occurrence models should be evaluated on extrinsic end-user tasks. Chapter 6 gives some first steps at such evaluations, investigating a few ways of incorporating features derived from systems described in previous chapters into a baseline learning-based coreference resolution system, which is trained to cluster noun phrases in a document according to whether they refer to the same real-world entity. We find that, though top-level system performance is affected only very marginally, some of the features are empirically useful for restricted classes of difficult coreference decisions.

Chapter 7 describes a subset of the large body of related work, to place the current document in context. In Chapter 8 we describe potential future work, and in Chapter 9 we briefly summarize and conclude.

Chapter 2

Background

This thesis 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 give a background on prior work in statistical script learning, provide a short summary of Recurrent Neural Networks relevant to our methods, and give a short description of the task of coreference resolution.

This chapter is intended to give motivations and background useful for understanding the original contributions presented subsequently. Chapter 7 gives a broader survey of related work.

2.1 Roots in Philosophy, Psychology, and Cognitive Science

The ways in which humans contextualize perceptions and reasoning based on learned world dynamics have been of interest to philosophers and psychologists for at least a century. For example, the types of experience-mediated world knowledge which allow humans to make probable (that is, likely, but not classically necessary) inferences have been explored by philosophers since at least the seminal investigations into abductive reasoning by Charles Sanders Peirce in the late 19th and early 20th centuries. Considering "any inference which does not regard its own conclusion as being necessarily true (though the facts be as the premisses assert)," Peirce writes:

> . . . [W]e begin by asking how often certain described conditions will, in the long run of experience, be followed by a result of a predesignate description; then proceeding to note the results as events of that kind present themselves in experience; and finally, when a considerable number of instances have been collected, inferring that the general character of the whole endless succession of similar events in the course of experience will be approximately of the character observed. . . . [I]f the character manifested by the series up to a certain point is not that character which the entire series possesses, still, as the series goes on, it must eventually tend, however irregularly, towards becoming so; and all the rest of the reasoner's life will be a continuation of this inferential process. This inference does not depend upon any assumption that the series will be endless, or that the future will be like the past, or that nature is uniform, nor upon any material assumption whatever.[1]

The machinery of probabilistic inference from experienced events to non-necessary facts, in Peirce's view, is based on a model built from the (incomplete) chain of events presenting themselves to the experience of a human, with the reasoner's inference machinery approaching a stable solution as the human extends their sequence of experienced events. In the chapters below, we will describe computational models of events learned from data which operate in a somewhat conceptually similar way to this idealized probabilistic reasoner of Peirce's.

In the field of psychology, formal experimental investigation of the man-

[1] Peirce (1902), CP 2.784.

ner in which humans synthesize, elide, and infer during recollection of scenarios dates back at least to the seminal work of Bartlett (1932), who, inspired in part by Head (1920), describes the idea of general "schemata" which mediate perceptual processing, action, and remembering:

> Determination by schemata is the most fundamental of all the ways in which we can be influenced by reactions and experiences which occurred some time in the past. All incoming impulses of a certain kind, or mode, go together to build up an active, organised setting: visual, auditory, various types of cutaneous impulses and the like, at a relatively low level; all the experiences connected by a common interest: in sport, in literature, history, art, science, philosophy and so on, on a higher level. There is not the slightest reason, however, to suppose that each set of incoming impulses, each new group of experiences persists as an isolated member of some passive patchwork. They have to be regarded as constituents of living, momentary settings belonging to the organism, . . . and not as a number of individual events somehow strung together and stored within the organism.[2]

This line of analysis was quite influential to later cognitive psychologists and Computer Scientists investigating the role of schemas in human cognition (Minsky, 1974; Rumelhart, 1978; Brewer, 2000, *inter alia*).

[2] Bartlett (1932), p. 201.

2.2 Statistical Script Learning

2.2.1 Early Script Learning Systems

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 detailed and influential analysis of structured scripts for understanding situations in AI. In this conception, sentences are mapped to structures describing a canonical meaning representation, constructed from a fixed set of primitives called conceptual dependencies (Schank, 1975). These dependencies include, for example, the actions PTRANS (a physical transfer of an object's location), ATRANS (an abstract transfer of of some property of an object), MTRANS (a mental transfer of a though), and PROPEL (to apply a force to), among others, along with a number of conceptual categories, conceptual roles, and syntactic rules describing the possible relationships between these different types of structured objects. A script consists of a hand-written prototypical situation-specific sequence of such conceptual dependency structures, allowing one to infer elided details in natural language text, assuming a translation from text to conceptual dependency structures can be faithfully executed. This allows one to fill in elided details from text; for example, given the passage

> John went into the restaurant. He ordered a hamburger and a coke. He asked the waitress for the check and left.

one can infer, using a script describing what happens while eating at a restaurant, that John ate a hamburger, though it is not explicitly stated

These approaches are non-probabilistic and depend on complicated hand-written structured world information, which results in brittle systems that cannot straightforwardly generalize to situations differing significantly from the ones encoded. Further, determining which situation's script to apply to a given text is nontrivial, and the fixed set of conceptual primitives introduces serious limitations. 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. The system, GENESIS, uses explanation-based learning to generalize a particular natural language text describing a novel situation into a more general, abstract schema which describes a formal plan encoding the script dynamics of the situation type. The system uses classical planning techniques to work backward, from observed actions performed by characters in the document to latent explanations for those actions; it can then group together the whole causal structure consisting of induced explanations and observed states, and generalize into a more abstract schema (such that the input text is an instance of such a schema).

Miikkulainen (1993) presents DISCERN, a Neural Network system which stores story instances in a hierarchical episodic memory and is capable of answering simple questions about simple incomplete stories. This system is trained to paraphrase and answer questions about stories generated automatically from, e.g., restaurant, shopping, and travel scripts. The system uses gradient-based optimization to learn distributed representations for a small vocabulary, allowing it to automatically identify situation-specific role relationships between words; it then uses a hierarchically organized episodic memory component, which it can query to identify the story types of novel stories, and subsequently answer simple questions and generate paraphrases.

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:

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

(2) 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.

2.2.2 Statistical Models of Event Co-Occurrence

The difficulties outlined in the previous section—data sparsity and the lack of large labeled resources make learning models with complex structured events difficult, and implicit event inferences would benefit from probabilistic models—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 (e.g. Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994) or Lexical Functional Grammar (Kaplan and Bresnan, 1982)), which have quite complex lexico-syntactic structure, are not typically used by state-of-the-art parsers, which are probabilistic and 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. This system is capable of inferring novel pair events from documents; for example, given the text:

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

the system will ideally be able to infer novel facts such as (*accept*, subject) or (*publish*, subject) for the entity *Andrew Wiles*, and facts like (*accept*, object) for the entity *Abel prize*.

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. The model is, as they describe it, *protagonist*-centered, meaning that events will only be in the same set if they relate to the same entity. 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 (that is to say, $N(a, b)$ gives the number of times the two pair events a and b relate to the same entity). 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. During learning, the model tracks pair events related to one entity at a time; during inference, it conditions on pair events related to a single entity of interest.

In order to infer implicit (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

$$S_{pmi}(b) = \sum_{i=1}^{\ell} \text{PMI}(a_i, b) \tag{2.1}$$

with PMI defined in the usual way:

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

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 Chapter 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

$$S_{bigram}(b) = \sum_{i=1}^{t} \log P(b|a_i) + \sum_{i=t+1}^{\ell} \log P(a_i|b) \tag{2.2}$$

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:

$$p(b|a_1, \ldots, a_k) = \frac{1}{Z} \exp\left(t_b^T \hat{t_{\overline{a}}} + b_b\right) \tag{2.3}$$

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.

2.2.3 Intrinsic Evaluation of Event Inference Systems

There are a number of ways to evaluate learned script systems, each with its own strengths and weaknesses. As mentioned in Section 2.2.2, one widely deployed evaluation is the *narrative cloze* evaluation (Chambers and Jurafsky, 2008), in which systems are evaluated by their ability to infer held-out events from unseen test documents. Chambers (2017) gives an in-depth analysis of the behavior of the narrative cloze evaluation on a number of systems (including the system presented below in Chapter 3), varying a number of different setup hyperparameters to gauge the robustness of comparative system differences. Among other things, Chambers (2017) finds that systems which make inferences by maximizing the PMI-based objective of (2.1) greatly underperform baseline systems, but outperform baselines if the evaluation setup is restricted by evaluating only on inferring uncommon events; due to the Zipfian distribution of natural language tokens, the evaluation setup can easily be dominated by the common events comprising the distribution's head, and this can greatly affect the performance of objective (2.1) on the narrative cloze task, as this objective takes a high value on low-probability events with high PMI with observed events.

Chambers (2017) observe that, compared to the setup originally introduced by Chambers and Jurafsky (2008), more recent instances of the narrative cloze evaluation setup, including those presented below, differ in that:

(1) They use automatically generated test instances, which in general are biased by noisy automatic parses and coreference information, rather than human-generated test instances.

(2) The evaluation setup is ordered, rather than unordered (that is, the document-relative position of events is used as a feature).

(3) All chains of events are used in evaluation, rather than the single largest chain in a document (presuming a document to be primarily about a single entity).

(4) Events are included in the evaluation according to their unigram probability (that is, an events are included in evaluation sets on a per-token basis rather than a per-type basis, so a verb-dependency structure that occurs multiple

times related to an entity may be included multiple times in the test set).

We will, below, evaluate on both the task of inferring held-out (verb, dependency) events and on the task of inferring more structured events with multiple entity arguments.

The numeric results of the various forms of narrative cloze evaluations in the literature do not admit straightforward interpretation. Further, the Gricean principles of pragmatics described in Section 1.1 dictate that, in general, very obviously inferrable facts are not explicitly stated in text, and are therefore not well-handled by the cloze evaluation. Motivated by these concerns, we will also present multiple human evaluations below, in which crowdsourced workers are solicited for judgments about the conditional likelihood of inferred events, based on input text.

2.2.4 Other Types of Event Collection Models

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[3] in an unsupervised manner, achieving system performance comparable to supervised systems. Cheung et al. (2013), Chambers

[3] 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 perpetrator.

(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-occurring 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.

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. (2015b) find event co-occurrence 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.

2.3 Neural Sequence Modeling

2.3.1 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 (though there is a growing body of work into

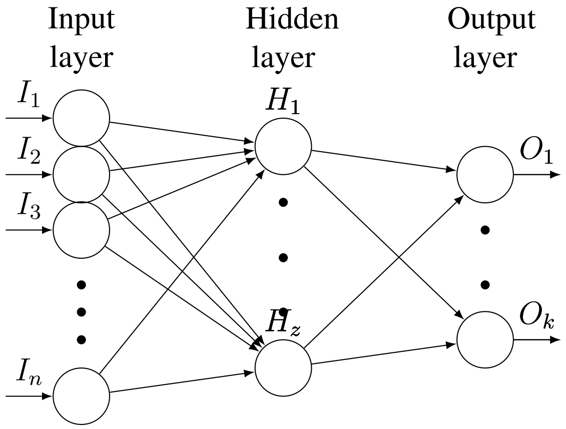


Figure 2.1: Single Hidden Layer Feedforward Network.

differentiable units with probabilistic semantics (Jang et al., 2017; Maddison et al., 2017)).

Figure 2.1 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 models (so-called "vanilla RNNs" (Elman, 1990)) are described by the following equations:

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

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

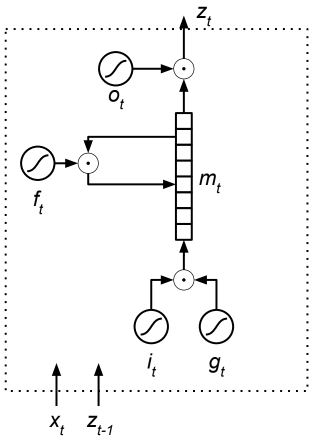


Figure 2.2: Graphical Depiction of Long Short-Term Memory unit at timestep t. The four nonlinearity nodes (i_t, g_t, f_t, and o_t) all have, as inputs, x_t and z_{t-1}. Small circles with dots are elementwise vector multiplications. Though all of o_t, f_t, i_t, g_t, and m_t are vectors, we only depict m_t, the memory, as a vector, for clarity.

(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).

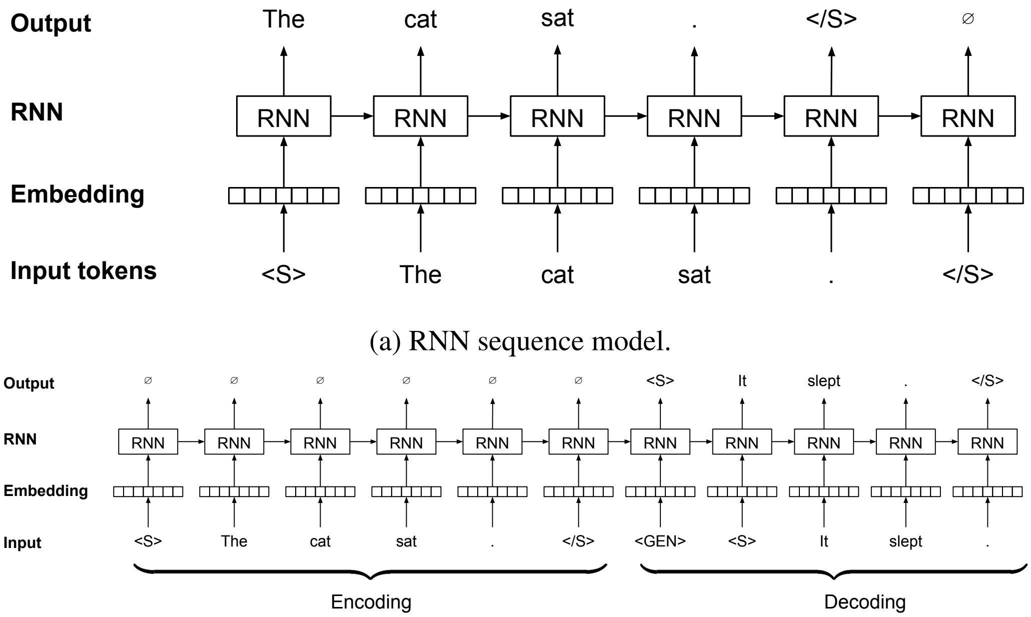
Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997) sidestep both of these problems by introducing a more complicated hidden unit. The LSTM formulation we use, from Zaremba and Sutskever (2014), is described by the following equations (explained below):

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

These equations are depicted graphically in Figure 2.2. 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

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

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;



(a) RNN sequence model.

(b) RNN sequence-to-sequence model.

Figure 2.3: Illustrations of (a) an RNN sequence model modeling the sequence "⟨S⟩ The cat sat . ⟨/S⟩" and (b) an RNN sequence-to-sequence model modeling the decoding of the sequence "⟨S⟩ It slept . ⟨/S⟩" after encoding the same sequence as in (a).

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.

2.3.2 RNN Sequence Models and Sequence-to-Sequence Models

There are (among others) two standard training/testing setups using Recurrent Neural Networks, which we now describe, and which will be used in subsequent chapters. These are the RNN **sequence** model, illustrated in Figure 2.3a, and the RNN **sequence-to-sequence** model, illustrated in Figure 2.3b.

In the sequence model setup of Figure 2.3a, a sequence is modeled by learning to iteratively predict the next element in the sequence, conditioned on all previous elements (Graves, 2013). In the basic setup, the input at a timestep is a one-hot vector identifying the word, and the output is a distribution over next tokens, trained to predict the next timestep's input. During inference, one can argmax or sample from a timestep's output distribution in order to get a single prediction for the next step's input. In the case where the inputs and outputs are words, input one-hot vectors are typically mapped first to low-dimensional continuous embeddings (some architectures also mediate output predictions through tied embeddings, but, below, we do not). Note the RNN weights are tied across timesteps (this is what makes the network recurrent)—that is, the model coefficients controlling the conditional output vectors and hidden-to-hidden dynamics remain fixed across the entire input.

Generally, RNNs can also be used to model sequential data where the input and output do not have the relationship pictured in Figure 2.3a; that is, the output of an RNN need not be the subsequent timestep's input. One common extension to the model in this case is to use a *bidirectional RNN* (sometimes called a biRNN, or a biLSTM in the case of an LSTM hidden unit), which concatenates the hidden states of two RNNs going in different directions before outputting predictions. In this setup, at a timestep t, a forward RNN constructs an RNN output vector conditioned on past inputs $x_1 \ldots, x_{t-1}$, a backward RNN constructs an output vector conditioned on future inputs $x_{t+1}, \ldots, x_N$, and the two are concatenated to form the RNN's final output used for predictions. This is helpful in tasks where future timesteps' inputs are empirically useful for prediction.

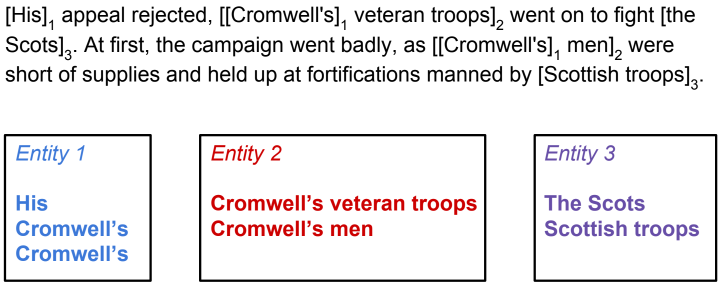
The sequence-to-sequence model pictured in Figure 2.3b, also called an *encoder-decoder* model, is similar to the sequence model, but it has an initial encoding stage with no output signal to predict (Sutskever et al., 2014). This setup is appropriate for sequence-to-sequence transduction, where the task can be defined as mapping from one sequence to another. In the encoding step, the input sequence is consumed, one timestep at a time, which ultimately produces an RNN latent state. In the second decoding stage, an output sequence is produced, one timestep at a time (as in the sequence model setup), conditioned on the hidden state of the initial decoding stage. Typically, separate RNNs are used for encoding and decoding (as the two are different tasks and therefore in principle controlled by distinct dynamics), but some models use the same RNN for both steps. This setup has recently been quite successful in varied tasks across NLP, including Machine Translation (Kalchbrenner and Blunsom, 2013a; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015) parsing (Vinyals et al., 2015b; Choe and Charniak, 2016), dialog systems (Vinyals and Le, 2015), and captioning images and videos (Donahue et al., 2015; Venugopalan et al., 2015b,a), among others.

RNN sequence-to-sequence models are frequently augmented with *attention* mechanisms (Bahdanau et al., 2015; Cho et al., 2015). Attention mechanisms, briefly, parametrize soft alignments between input timesteps and output timesteps in a sequence-to-sequence setup. Whereas a standard encoder-decoder setup, given an input of length n, will produce a single hidden vector after the encoding step, an attention model will instead produce n hidden vectors, maintaining the hidden state at each input encoding step. When decoding, at a given timestep, instead of conditioning updates to the hidden state on a single vector (the previous timestep's hidden state), as a standard encoder-decoder model does, an attention model will also condition on a learned conditional distribution over all the n latent states from the encoding stage, forming the convex sum of the state vectors according to the distribution. That is, at a given output timestep, the attention model will provide a soft alignment between all input steps; the hidden steps will be combined according to this alignment, and the resulting vector will be input to the LSTM decoder unit at that timestep. See Cho et al. (2015) for a survey of a number of attention-based approaches to different problems.

2.4 Coreference Resolution

We now briefly outline the task of nominal coreference resolution. Chapter 6 describes a number of ways of incorporating features derived from statistical scripts into a baseline learning-based coreference resolution system. A more thorough motivation and description of modern coreference resolution systems is given in Section 6.1.

Briefly, the task of coreference resolution is to automatically cluster the noun phrases in a document (including, but not limited to, pronouns) into equivalence



[His]$_1$ appeal rejected, [[Cromwell's]$_1$ veteran troops]$_2$ went on to fight [the Scots]$_3$. At first, the campaign went badly, as [[Cromwell's]$_1$ men]$_2$ were short of supplies and held up at fortifications manned by [Scottish troops]$_3$.

Figure 2.4: Illustration of a nominal coreference clustering. Coreferring noun phrases are indicated in text with subscripts. Note mentions may nest.

classes, such that two noun phrases are in the same class if and only if they corefer identically, that is, they refer to the same entity or entities in the world (Ng, 2017). This task is pictured briefly in Figure 2.4, which gives a short passage along with its coreference clustering, and illustrates a number of the subtleties of the task. First, the pronoun *his* appears before the proper noun describing it, namely *Cromwell*, and is therefore a case of *cataphora* (where a pronoun is used before its coreferent, in opposition to the more common anaphoric case); this phenomenon is not generally well-handled by state-of-the-art coreference resolution systems. Second, note that mentions may be nested—the noun phrase *Cromwell's* is a substring of the phrase *Cromwell's veteran troops*, though the two refer to different entities. Third, note that some non-pronomial noun phrases corefer with other entities (e.g. *troops*, which corefers with *men*), while others do not corefer with anything (e.g. *appeal*, *campaign*). Such nominal coreference decisions are difficult for automatic coreference systems, as the great majority of noun phrases in text do not corefer with other noun phrases.

There are a number of general approaches to learning-based coreference systems, the most common of which are described in Section 6.1. One major system architecture decision concerns exactly how the document-level clustering decision is decomposed into more tractable, local decisions over *candidate mentions*, that is, noun phrases which may corefer. Some systems consider the $\binom{n}{2}$ mention-pairs independently, using either independent decision models or independent ranking models, subsequently combining selected pairwise links into mention clusters (Soon et al., 2001; Durrett and Klein, 2013; Wiseman et al., 2015; Clark and Manning, 2016a). Other models decompose the document-level decision into individual decisions between single mentions and currently-constructed partial clusters of mentions (Luo et al., 2004; Rahman and Ng, 2009). Other systems make decisions at the cluster level—that is, learned parameters control whether to iteratively merge pairs of clusters into a single cluster (Lee et al., 2013; Wiseman et al., 2016; Clark and Manning, 2016b). See Ng (2017) for a survey of systems using these approaches, along with a number of other general approaches.

The intuitive appeal of applying features which encode general world knowledge to the task of coreference resolution is strong. In the example pictured above in Figure 2.4, for example, it requires knowledge of demonym usage to conclude that *the Scots* and *Scottish troops* may corefer—one cannot reasonably cluster these mentions without knowing that these two demonyms both refer to people from Scotland. Statistical scripts, described in Section 2.2.2, encode another type of knowledge which is intuitively appealing to coreference. Consider, for example, the sentence

Rasputin was said to possess the ability to heal through prayer and was

> indeed able to give the boy some relief, in spite of the doctors' prediction that **he** would die.[4]

In this sentence, the word *he* could refer to either *Rasputin* or *the boy*; the decision is difficult to make from the syntactic structure alone (consider the sentence with the word *die* replaced with *make matters worse*). The decision could be aided by the world knowledge that *give X relief* and *predict that X would die* are likely to co-occur for the same entity X (and, indeed, that it is unlikely that *X possesses the ability to heal through prayer* and *predict that X would die* would be mentioned in text for the same entity X). We investigate incorporating script-derived features in Chapter 6, motivated by this observation that information about event co-occurrence can inform coreference decisions. Section 6.4 outlines some of the many prior efforts to incorporate world knowledge in various forms into coreference resolution systems.

[4] https://en.wikipedia.org/wiki/Grigori_Rasputin

Chapter 3

Statistical Scripts 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 Pichotta and Mooney (2014). The statistical model is a simple Markov-like co-occurrence model; Chapter 4 will describe a more complex Neural Net system (with superior performance) which models similar events.

3.1 Background

A short survey of published statistical models of event co-occurrence was given in in Section 2.2.2; we briefly recount some information relevant to this chapter.

There are a number of efforts at building co-occurrence models of (verb, dependency) pair events capable of inferring events. Chambers and Jurafsky (2008) describe a system that infers event pairs based on their having high average PMI with observed event pairs. This is described as a *protagonist*-centered model, in that chains of verb-dependency event pairs in both learning and inference pivot around

The work in this chapter has been published in the following publication:

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 2014)*, pages 220–229, 2014.

All work in this chapter represents original contributions.

a single entity, according to an automatic coreference analysis, and possible entity-entity interactions mediated by events are not modeled. Chambers and Jurafsky (2009) describe an extension to this system which can incorporate noun information about arguments and reason about events involving multiple entities.

Jans et al. (2012) describe a verb-dependency pair event system which infers events at a position by maximizing a function of the probability of textually-prior events succeeding candidate inferences, along with the probability of textually-subsequent events preceding candidate inferences, according to (2.2). Rudinger et al. (2015b) present superior results on inferring such events using a log-bilinear language model.

The systems of Chambers and Jurafsky (2008) and Jans et al. (2012) are directly based on co-occurrence counts of verb-dependency pair events, tabulated from a large corpus: learning consists of tabulating these counts, and inference consists of calculating the value of an objective function using these counts (and subsequently maximizing). In this chapter, we present a novel co-occurrence model in this paradigm.

3.2 Methods

3.2.1 Multi-Argument Events

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. As mentioned above, there is a body of related prior work which focuses on

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

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 various formalizations of events are simple enough to enable tractable learning and inference algorithms to be run on large corpora (and they are simple enough that the total number of events remains manageable). However, these formalisms are incapable of expressing some fundamental aspects of event structure. Consider, for example, the following example:

(3) 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.[1]

A representation of this sentence with (verb, dependency) pair events is given in Figure 3.1 (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.

[1] `https://en.wikipedia.org/wiki/Napoleon`

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 (3.1), 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*. These multi-argument events are more complex than pair events, but still simple enough to enable tractable learning and inference. Figure 3.1 (Bottom) gives the multi-argument event representation of Example 3. 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. Entity information can be derived from gold-standard coreference information if it is available; in the experiments below, we use automatic (noisy) coreference annotations, which can be provided for arbitrarily large corpora. This event formulation is capable of capturing relationships between different entities: two entities may have intra-event interactions, and co-occurring events may involve the same pair of 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 verb 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 work presented in this chapter, 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).

We limit ourselves to verbal predicates when constructing structured events from raw text. Nominal events are widespread (e.g. "explosion," "wedding," "sprint"), but differentiating such nouns from non-eventive nouns (e.g. "computer") is a difficult task without a standard broad-coverage solution, and identifying arguments of nominal events is similarly difficult and noisy (Gerber and Chai, 2010).

3.2.2 Learning Event Co-Occurrence

In this section, we describe the multi-argument event co-occurrence model from Pichotta and Mooney (2014) (in Chapter 4 we describe an empirically superior model which also uses relational events). We use the objective function (2.2) from Section 2.2.2 to infer novel relational events; this objective 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

(4) *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 (4), 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*). We ignore the exact identity of the non-overlapping entities

Algorithm 1 Learning with entity substitution

```
for $a_1, a_2 \in$ evs do
    $N(a_1, a_2) \leftarrow 0$
for $D \in$ documents do
    for $a_1, a_2 \in$ coocurEvs($D$) do
        for $\sigma \in$ subs($a_1, a_2$) do
            $N(\sigma(a_1), \sigma(a_2)) \mathrel{+}= 1$
```

mary and *question*, as it is intuitively less important for capturing the relationship between the two events than knowing the event-event overlap relationships. It is worth noting that 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 fact that y is "attention" and interacts idiomatically with the verb. The system presented below in Chapter 4 better handles this by directly incorporating noun information into an RNN.

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

$$P(a_1, a_2) = \frac{N(a_1, a_2)}{\sum_{a'_1, a'_2} N(a'_1, a'_2)}, \tag{3.1}$$

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 3 is the collection of documents in our training corpus. The function `cooccurEvs` in line 4 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.[2] The function `subs` in line 5 takes two events and returns all variable substitutions σ mapping from

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

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 6, 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 (3.1), and then define the conditional probability of seeing a_2 after having seen a_1 as:

$$\begin{aligned} P(a_2|a_1) &= \frac{P(a_1, a_2)}{\sum_{a'} P(a_1, a')} \\ &= \frac{N(a_1, a_2)}{\sum_{a'} N(a_1, a')}. \end{aligned} \tag{3.2}$$

3.3 Experiments

3.3.1 Evaluation Setup

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. This is the “Narrative Cloze” evaluation described in Section 2.2.2.

The four systems compared 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.

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:

$$M(a) = S_{e_s}((v, \text{subj})) + S_{e_o}((v, \text{obj})) + S_{e_p}((v, \text{prep})) \tag{3.3}$$

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 3.2, 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.

3.3.2 Evaluation Results

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.[3] 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. 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.[4]

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.[5] 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.

Table 3.1 gives the Recall at 10 and accuracy scores for the different sys-

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

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

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

Table 3.1: Results for multi-argument events.

tems. 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 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 3.2: Results for pair events.

Table 3.2 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 (2.2) (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 3.1, such that the extracted pair event relates to an entity with at least one additional known pair event.

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 other events in which a multi-argument event's entities are involved). In the next chapter, we improve upon this co-occurrence model by introducing a neural model over relational events.

Chapter 4

Using Recurrent Neural Networks for Event Inference

In this section, we describe an LSTM-based script system, described in Pichotta and Mooney (2016a), which provides superior performance to the simpler co-occurrence-based system from Chapter 3. It uses a modified version of the multi-argument events presented in the previous chapter, but decomposes these structured events into their constituent components during learning and inference to improve generalizability, and is parametrized by a large number of learned continuous latent parameters, rather than relying directly on discrete corpus counts.

4.1 Motivations and Background

4.1.1 Motivating the System

The co-occurrence joint model of Chapter 3 has a number of shortcomings which the LSTM-based model we present in this chapter can in principle address. These include the following observations about the co-occurrence model of the previous chapter:

The work in this chapter has been published in the following publication:

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.

All work in this chapter represents original contributions.

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).

   This is problematic, first, because there is structure shared between the two events that could be shared during learning (when one is observed, it lends evidence to the other) and, second, because not decomposing events and treating them as simplex objects makes the size of the vocabulary of structured event types prohibitively high if we want to include noun information about arguments.

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. We will use low-dimensional lexical embeddings to allow for this sort of generalization.

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 event representations. This is a considerable shortcoming, given that the most common verbs (which account for a very large portion of event tokens, given the Zipfian distribution of tokens) are typically very polysemous, with many different word senses. The system presented in this chapter will in-

corporate noun information about arguments, which can in principle allow a system to implicitly perform word sense induction and 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. The RNN sequence model we will use is capable of learning more complex latent sequence dynamics.
5. The model cannot infer structured 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). Decomposing events into individual components can address these shortcomings.
6. Finally, there is a particularly important special case of Point 3 above, namely, that nouns frequently behave idiomatically with other event components. For example, "the hurricane made landfall" and "the elves made shoes" will get identical representations in the system of Chapter 3, 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 Chapter 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.

4.1.2 Background

We now briefly review some prior work useful for understanding this chapter. A more thorough treatment of some of this background material can be found in Section 2.3.

Recurrent Neural Nets are described in some detail above, in Section 2.3.1, and RNN sequence models are described in Section 2.3.2. Briefly, RNN sequence models maintain a hidden real-vector-valued state that is updated at each timestep, conditional on the timestep's input and the previous hidden state value (the hidden state, before the first timestep, is initialized to a fixed value, e.g. zero). In our systems, we follow the standard step of mapping high-dimensional categorical input (e.g. words) to low-dimensional continuous vectors (i.e. embeddings) before inputting to the RNN unit. RNN sequence models are trained to iteratively predict the next timesteps' input (as described in Section 2.3.2).

There are a number of related models, many neural, of various notions of event structure and co-occurrence which are described below in Chapter 7. Notably, Rudinger et al. (2015b) give a log-bilinear model of verb-dependency pair event co-occurrence, described in more detail above in Section 2.2.2, which is governed by learned latent continuous embeddings for event pair types. This model is non-recurrent and does not decompose events into subcomponents.

4.2 Methods

Motivated by the concerns outlined above in Section 4.1.1, we propose using a Recurrent Neural Net model (in particular, an LSTM, described in Section 2.3.1) 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 4.3 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), which we linearize into a single flat sequence. 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 Chapter 3; however, where before we rewrote 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 4.1 (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

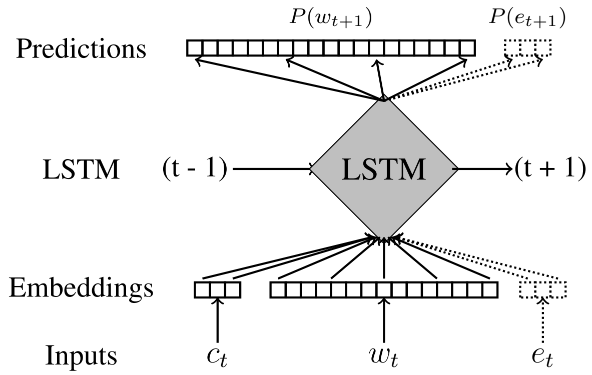


Figure 4.1: LSTM Script System at timestep t.

$c_t = 1$, 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,[1] treating all other entities as singletons.

One-hot input vectors are mapped to continuous distributed representations, labeled "Embeddings" in Figure 4.1. 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 in-

[1] 98% of training sequences involve five or fewer non-singleton entities, so we lose coreference information in only 2% of sequences.

put 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.

4.3 Experiments

4.3.1 Evaluation Setup

Systems Evaluated

We compare the four systems enumerated in Section 4.2 to four baseline systems:

1. **Unigram**: Like the unigram system in Section 3.3, this system 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 3.3, 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 3.3, and an event a is scored as an inference at position t by maximizing the objective

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

   This system conditions on the same events as the neural systems.
3. **Rewrite Bigram**: This is the All-bigram system, but we rewrite entity IDs during learning as in Section 3.3. This allows us to evaluate the contribution of entity rewriting in the current task setup.
4. **2D Rewrite Bigram**: This is the Rewrite bigram system, but it optimizes the objective given in (2.2), incorporating events after timestep t in addition

to past events. This will allow us to compare directly to the system of Chapter 3, and, comparing to the rewrite bigram system, allow us to gauge the utility of conditioning on subsequent events in the current task setup.

Quantitative Evaluation Metrics

As in Chapter 3, we evaluate using the Narrative Cloze (we will also provide human judgments of inferences below).

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 Chapter 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 3.3, 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)

Experimental Details

We use the Stanford dependency parser (De Marneffe et al., 2006) and coreference system (Raghunathan et al., 2010).[2] We represent noun arguments by their head lemmas. For a training and testing corpus, we use English Language Wikipedia,[3] 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 valida-

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

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

tion 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. Since RNNs are quite sensitive to hyperparameter values (Sutskever et al., 2013), we measured validation set performance in different regions of hyperparameter space, ultimately selecting learning rate $\eta = 0.1$, momentum parameter $\mu = 0.98$, LSTM vector length of 1,000, and a Normal $\mathcal{N}(0, 0.1)$ distribution for random initialization (biases are initialized to 0). Event component embeddings have dimension 300. We use ℓ_2 regularization and Dropout (Hinton et al., 2012) with dropout probability 0.5. We clip gradient updates at 10 to prevent exploding gradients (Pascanu et al., 2012) We damp η by 0.9 every 100,000 iterations. We train for 750,000 batch updates, which took between 50 and 60 hours. We use a beam width of 50 in all beam searches.

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

4.3.2 Experimental Results

Automatic Evaluations

Table 4.1 gives results on the Narrative Cloze evaluation. The LSTM-both-ent system demonstrates a **50.0%** relative improvement (5.7% absolute improvement) over the best system of Chapter 3 (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.

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.

Subjective Human Evaluation

The low magnitude of the Narrative Cloze scores in Table 4.1 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.[4] This restriction is motivated by the observation that events with very common verbs occur so frequently that they are almost trivial as reasonable inferences (for the same reason that the unigram system is so competitive at the narrative cloze task); the inference, for example, that an entity *has something* or *knows something* may be strictly true, but is uninteresting as a script inference.

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.

[4] *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 4.2: Crowdsourced results (scores range from 0 to 5).

The “All” column in Table 4.2 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.

Qualitative Analysis

Figure 4.2 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:

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

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

Figure 4.3: Probabilistically generated event sequences. The right column gives possible English descriptions of the structured events on the left.

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 4.3 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.

Chapter 5

Using Sentence-level RNN Models for Event Inference

The work presented above evaluated on inferring missing events from documents. However, the exact form that this task takes depends on the adopted definition of what constitutes an event: formal, structured events may be defined in many different ways, with differing degrees of structure. In this chapter, we investigate using raw text, which requires no explicit syntactic annotation, as our mediating representation, and evaluate how raw text models compare to models of more structured events, as described in Pichotta and Mooney (2016b)

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

The work in this chapter has been published in the following publication:
Karl Pichotta and Raymond J. Mooney. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, Berlin, Germany, 2016.
All work in this chapter represents original contributions.

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

5.1 Background

As described in section 2.3.1 and used in Chapter 4, Recurrent Neural Networks (RNNs) are neural nets whose computation graphs have cycles. In particular, RNN sequence models are RNNs which map a sequence of inputs $x_1, \ldots, x_T$ to a sequence of outputs $y_1, \ldots, y_T$ via a learned latent vector state whose value at timestep t is a function of its value at the previous timestep $t - 1$.

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

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

5.2 Methodology

5.2.1 Narrative Cloze Evaluation

We briefly discuss the Narrative Cloze evaluation, used in previous chapters, as it relates to the current Chapter. Below, we will also present a system evaluation based on human judgments of inferences.

The evaluation of inference-focused statistical script systems is not straightforward. The systems in previous chapters used the Narrative Cloze evaluation (Chambers and Jurafsky, 2008), in which a single event is held out from a document and systems are judged by their ability to infer this held-out event given the remaining events. This automated evaluation measures systems' ability to model and predict events as they co-occur in text.

The exact definition of the Narrative Cloze evaluation depends on the formulation of events used in a script system. For example, Chambers and Jurafsky (2008), Jans et al. (2012), and Rudinger et al. (2015b) evaluate inference of held-out (verb, dependency) pairs from documents; the system of Chapter 3 evaluates inference of verbs with coreference information about multiple arguments; and the system of Chapter 4 evaluates inference of verbs with noun information about multiple arguments. In order to gather human judgments of inference quality, in Chapter 4, we also trained an encoder-decoder LSTM network for transforming verbs and noun arguments into English text to present to annotators for evaluation.

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

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

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

5.2.2 Models

Unidirectional Models

As described in Section 2.2.2, statistical script systems take a sequence of events from a document and infer additional events that are statistically probable. Exactly what constitutes an *event* varies: events may be a (verb, dependency) pairs inferred as relating to a particular discourse entity (Chambers and Jurafsky, 2008; Rudinger et al., 2015b), simplex verbs (Chambers and Jurafsky, 2009; Orr et al., 2014), or verbs with multiple arguments, as in Chapters 3 and 4.

In this chapter, when considering models of formal, structured events, we continue in the vain of Chapters 3 and 4, and adopt a representation of events as verbs with multiple arguments (Balasubramanian et al., 2013; Modi and Titov, 2014). Formally, we define an event to be a variadic tuple (v, s, o, p^*), where v is a verb, s is a noun standing in subject relation to v, o is a noun standing as a direct object to v, and p^* denotes an arbitrary number of (*pobj*, *prep*) pairs, with *prep* a preposition and *pobj* a noun related to the verb v via the preposition *prep*. This is essentially the event representation of Chapter 4, but whereas we then limited events to having a single prepositional phrase, we now allow an arbitrary number, and we do not lemmatize words. Any argument except v may be *null*, indicating no noun fills that slot. For example, the text

Napoleon sent the letter to Josephine

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

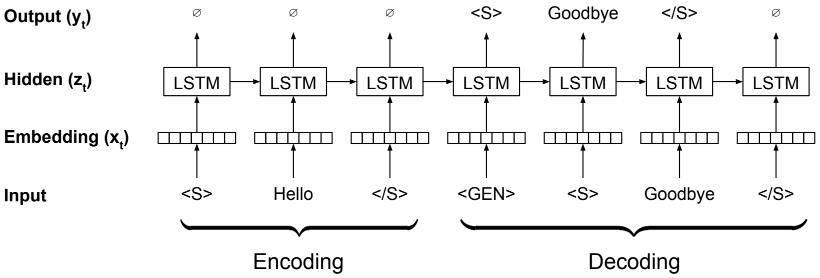


Figure 5.1: Encoder-Decoder setup predicting the text “Goodbye” from “Hello”

We evaluate on a number of different neural models which differ in their input and output. All models are LSTM-based encoder-decoder models, that is to say, sequence-to-sequence models, as described in Section 2.3.2. These models encode a sentence (either its events or text) into a learned hidden vector state and then, subsequently, decode that vector into its successor sentence (either its events or its raw text).

Our general system architecture is as follows. At each timestep t, the input token is represented as a learned embedding vector (learned jointly with the other parameters of the model), such that predictively similar words should get similar embeddings. This embedding is fed as input to the LSTM unit (that is, it will be the vector x_t in Section 2.3.1, the input to the LSTM). The output of the LSTM unit (called z_t in Section 2.3.1) is then fed to a softmax layer via a learned linear transformation.

During the encoding phase the network is not trained to produce any output. During the decoding phase the output is a one-hot representation of the subsequent timestep's input token (that is, with a V-word vocabulary, the output will be a V-dimensional vector with one 1 and $V - 1$ zeros). In this way, the network is trained to consume an entire input sequence and, as a second step, iteratively output the subsequent timestep's input, which allows the prediction of full output sequences. This setup is pictured diagrammatically in Figure 5.1, which gives an example of input and output sequence for a token-level encoder-decoder model, encoding the sentence "Hello ." and decoding the successor sentence "Goodbye ." Note that we add beginning-of-sequence and end-of-sequence pseudo-tokens to sentences. This formulation allows a system to be trained which can encode a sentence and then infer a successor sentence by iteratively outputting next-input predictions until the `</S>` end-of-sentence pseudo-token is predicted. We use different LSTMs for encoding and decoding, as the dynamics of the two stages need not be identical.

We notate the different unidirectional systems as follows. Let s_1 be the input sentence and s_2 its successor sentence. Let t_1 denote the sequence of raw tokens in s_1, and t_2 the tokens of s_2. Further, let e_1 and e_2 be the sequence of structured events occurring in s_1 and s_2, respectively (described in more detail in Section 5.3.1), and let $e_2[0]$ denote the first event of e_2. The different systems we compare are named systematically as follows:

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

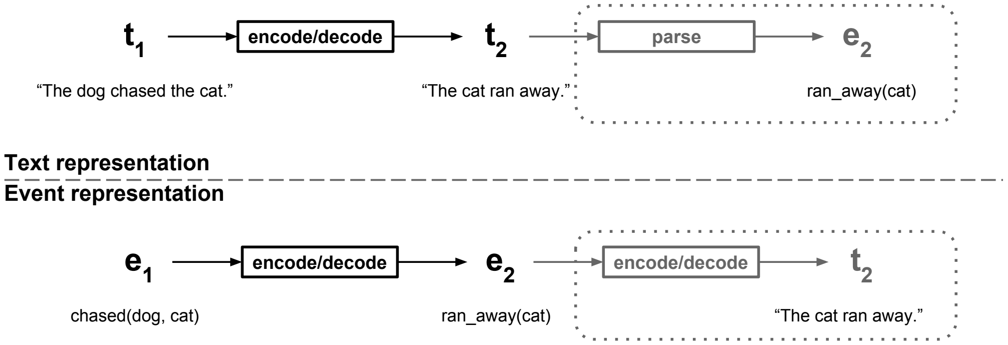


Figure 5.2: Different system setups for modeling the two-sentence sequence "The dog chased the cat." followed by "The cat ran away." The gray components inside dotted boxes are only present in some systems.

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

We also present results for systems of the form $X \overset{a}{\rightarrow} Y$, which signifies that the system is trained to decode Y from X with the addition of an attention mechanism. Attention mechanisms were described in Section 2.3.2. We use the particular attention formulation given by Vinyals et al. (2015b). In short, these models have additional parameters which can learn soft alignments between positions of encoded inputs and positions in decoded outputs. Attention mechanisms have recently been shown to be quite empirically valuable in many complex sequence prediction tasks. For more details on the attention model used, see Vinyals et al. (2015b).

Figure 5.2 gives a diagrammatic representation of the different system setups. Text systems infer successor text and, optionally, parse that text and extract events from it; event sequences infer successor events and, optionally, expand inferred events into text.

Note that the system $t_1 \rightarrow t_2$, in which both the encoding and decoding steps operate on raw text, is essentially a one-directional version of the skip-thought system of Kiros et al. (2015).[1] Further, the system $e_1 \rightarrow e_2 \rightarrow t_2$, which is trained to take a sentence's event sequence as input, predict its successor's events, and then predict its successor's words, is comparable to the event inference system of Chapter 4. There, we use an LSTM sequence model of events in sequence for event inference, and optionally transform inferred events to text using another LSTM; here, on the other hand, we use an encoder/decoder setup to infer text directly.

Bidirectional Models

Future events may also be empirically useful in making inferences—Jans et al. (2012) condition on events subsequent to the position of inference using different parameters from the events prior to the position, as does the system presented in Chapter 3. In addition the unidirectional inference systems presented in the previous chapter, we also consider systems which condition on future document context. When predicting the text of a sentence t_2, in addition to conditioning on the previous sentence t_1, we can also condition on the subsequent sentence t_3. We do this by concatenating the hidden state vector h_1 (obtained from encoding t_1) with the hidden state vector h_3 (obtained from encoding subsequent text t_3), and condition-

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

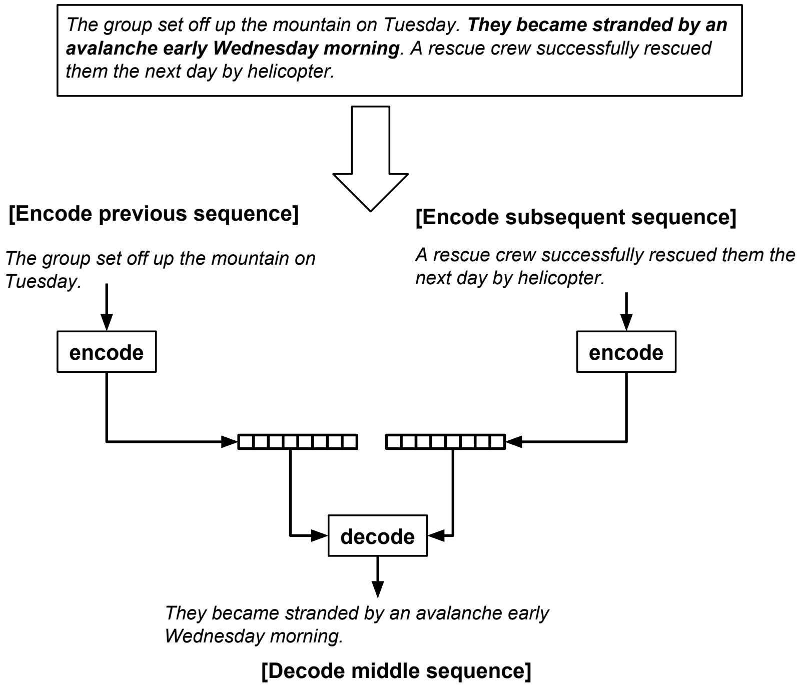

The group set off up the mountain on Tuesday. **They became stranded by an avalanche early Wednesday morning**. A rescue crew successfully rescued them the next day by helicopter.
[Encode previous sequence]
The group set off up the mountain on Tuesday.
[Encode subsequent sequence]
A rescue crew successfully rescued them the next day by helicopter.
encode
encode
decode
They became stranded by an avalanche early Wednesday morning.
[Decode middle sequence]

Figure 5.3: A diagrammatic representation of the bidirectional sequence-to-sequence setup. In this example, the middle sentence in the passage is being decoded based on the encoded representations of the previous and subsequent sentence.

ing on this larger concatenated vector during decoding (during the backward step of learning, we backpropagate error back both directions). We can do the same with event sequences, conditioning on events from both prior and subsequent test. An example of inference in this setup is illustrated in Figure 5.3 This technique is, in principle, applicable to any problem which can be framed as a two-level sequence problem, that is, as a sequence of sequences.

This setup is, in a sense, the sequence-to-sequence analog to the bidirectional RNN sequence models described in Section 2.3.2. Whereas biRNNs typically model a flat sequence in two directions, conditioning on past and future inputs, these systems model a sequence of sequences, conditioning on past and future input sequences.

5.3 Evaluation

5.3.1 Experimental Details

We train a number of LSTM encoder-decoder networks which vary in their input and output. Models are trained on English Language Wikipedia, with 1% of the documents held out as a validation set. Our test set consists of 10,000 unseen sentences (from articles in neither the training nor validation set). We train models with batch stochastic gradient descent with momentum, minimizing the cross-entropy error of output predictions. All models are implemented in TensorFlow (Abadi et al., 2015). We use a vocabulary of the 50,000 most frequent tokens, replacing all other tokens with an out-of-vocabulary pseudo-token. Learned word embeddings are 100-dimensional, and the latent LSTM vector is 500-dimensional. To extract events from text, we use the Stanford Dependency Parser (De Marneffe et al., 2006; Socher et al., 2013). We use the Moses toolkit (Koehn et al., 2007) to calculate BLEU.[2]

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

We also evaluate on the task of predicting held-out verb-argument events, either directly or via inferred text, as in Chapters 3 and 4. We use two evaluation metrics for this task. First, the **Accuracy** metric measures the percentage of a system's most confident guesses that are totally correct. That is, for each held-out event, a system makes its single most confident guess for that event, and we calculate the total percentage of such guesses which are totally correct. This metric is equivalent to "Recall at 1," similar to the recall metrics from Chapters 3 and 4. This is quite a stringent metric, as an inference is only counted correct if the verb and all arguments are correct. To relax this requirement, we also present results on what

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

we call the **Partial Credit** metric, which is the percentage of held-out event components identical to the respective components in a system's top inference. This metric is what Chapter 3 calls "Accuracy;" In this chapter, we use "accuracy" only to mean Recall at 1.

Our Wikipedia dump from which the training, development, and test sets are constructed is from Jan 2, 2014. We parse text using version 3.3.1 of the Stanford CoreNLP system. We use a vocab consisting of the 50,000 most common tokens, replacing all others with an Out-of-vocabulary pseudo-token. We train using batch stochastic gradient descent with momentum with a batch size of 10 sequences, using an initial learning rate of 0.1, damping the learning rate by 0.99 any time the previous hundred updates' average test error is greater than any of the average losses in the previous ten groups of hundred updates. Our momentum parameter is 0.95. Our embedding vectors are 100-dimensional, and our LSTM hidden state is 500-dimensional. We train all models for 300k batch updates (with the exception of the models compared in §5.3.3, all of which we train for 150k batch updates, as training is appreciably slower with longer input sequences). Training takes approximately 36 hours on an NVIDIA Titan Black GPU.

5.3.2 Experimental Evaluation

Table 5.1 gives the results of evaluating predicted successor sentence text against the gold standard using BLEU. The baseline system $t_1 \rightarrow t_1$ simply reproduces the input sentence as its own successor.[3] Below this are systems which

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

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

Table 5.1: Successor text predictions evaluated with BLEU.

make predictions from event information, with systems which make predictions from raw text underneath. Transformations written $X \overset{a}{\rightarrow} Y$ are, recall, encoder-decoder LSTMs with attention.

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

The BLEU-BP scores, which are BLEU without the brevity penalty, are noticeably higher in the text-level models than the raw BLEU scores. This is in part because these models seem to produce shorter sentences, as illustrated below in section 5.3.6.

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

These results provide evidence that, if the Narrative Cloze task is defined to evaluate prediction of held-out text from a document, then sentence-level RNN language models provide superior performance to RNN models operating at the event level.

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

Table 5.2: Next event prediction accuracy (numbers are percentages: maximum value is 100).

Table 5.2 gives results on the task of predicting the next verb with its nominal arguments; that is, whereas Table 5.1 gave results on a text analog to the Narrative Cloze evaluation (BLEU), Table 5.2 gives results on the verb-with-arguments prediction version. In the $t_1 \rightarrow t_2 \rightarrow e_2[0]$ system (and the comparable system with attention), events are extracted from automatically generated text by parsing output text and applying the same event extractor to this parse used to extract events from raw text.[4] The row labeled **Most common** in Table 5.2 gives performance for the baseline system which always guesses the most common event in the training set.

The LSTM models trained to directly predict events are roughly comparable to systems which operate on raw text, performing slightly worse on accuracy and slightly better when taking partial credit into account. As with the previous

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

comparisons with BLEU, the attention mechanism does not provide an obvious improvement when decoding inferences, perhaps, again, because the event inference problem lacks a clear alignment structure.

These systems infer their most probable guesses of $e_2[0]$, the first event in the succeeding sentence. In order for a system prediction to be counted as correct, it must have the correct strings for grammatical head words of all components of the correct event.

5.3.3 Adding Additional Context

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

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

Table 5.3: Varying the amount of context in text-level models. “Num Prev Sents” is the number of previous sentences supplied during encoding.

Table 5.3 gives the results of running these models on 10,000 sentences from the validation set. As can be seen, in the training setup we investigate, more additional context sentences have a mixed effect, depending on the metric. This is perhaps due in part to the fact that we kept hyperparameters fixed between experiments, and a different hyperparameter regime would benefit predictions from longer input sequences. More investigation could prove fruitful.

5.3.4 Evaluating Bidirectional Systems

As described in Section 5.2.2, we also evaluate the performance of bidirectional systems, which condition on both previous and subsequent context when making predictions during the decoding step. We compare bidirectional systems with unidirectional sequence-to-sequence models. We compare the following systems:

- A unidirectional text-level system, as evaluated above, labeled $t_1 \rightarrow t_2$.
- A bidirectional text-level system, labeled $t_1 \rightarrow t_2 \leftarrow t_3$, which conditions on the sequence of words to the left and to the right.
- A unidirectional event-level system, as evaluated above, which is trained to encode events, decode events, and then separately decode text from the decoded events, labeled $e_1 \rightarrow e_2 \rightarrow t_2$.
- A bidirectional event-level system, labeled $(e_1 \wedge e_3) \rightarrow e_2 \rightarrow t_2$, gives results for a bidirectional event model, which encodes the event sequences in sentences to the left and right of the target sentence, conditions on both to predict a sequence of events and expands them into text with a separate sequence-to-sequence decoder (as with previously presented systems).

System	**BLEU**	**BLEU-BP**	**1G P**
$e_1 \rightarrow e_2 \rightarrow t_2$	1.21	2.00	25.07
$(e_1 \wedge e_3) \rightarrow e_2 \rightarrow t_2$	1.30	2.78	28.12
$t_1 \rightarrow t_2$	**3.75**	4.55	25.53
$t_1 \rightarrow t_2 \leftarrow t_3$	3.63	**6.35**	**32.78**

Table 5.4: BLEU Results on bidirectional models compared to unidirectional models. System descriptions are given in the text. BLEU-BP is BLEU without brevity penalty, and "1G P" is unigram precision.

For these experiments, models are trained on the same Wikipedia corpus as in the previous section. All models are trained for 2.5M batch updates (with a batch size of 32); all other hyperparameters are as described in Section 5.3.1. We test only on sentences that have both predecessor and successor sentences (so both unidirectional and bidirectional sentences can make inferences). Thus, the test set in this section is different from that in the previous sections, and the results are slightly different in magnitude.

Table 5.4 gives results on BLEU (along with BLEU without the brevity penalty and 1-gram precision) comparing the one-directional and bidirectional models. Note, first of all, that the bidirectional models generally outperform the unidirectional models. The exception to this is that the one-directional text-level model outperforms the bidirectional one according to raw BLEU; note, however, the relationship reverses when the brevity penalty is removed. The brevity penalty, motivated by machine translation evaluation, the original task the metric was designed for, is a way of balancing precision with recall—a single 4-gram from the target text would maximize BLEU but not have a high-enough recall to be a faithful translation. In this task, maintaining the length relationship between target and candidate texts may not not be as intuitively important. Generally, these results provide evidence that, for the task considered, conditioning on immediately subsequent text or events is empirically beneficial.

5.3.5 Crowdsourced Evaluation

Since the evaluations presented above do not admit straightforward interpretation, we also solicit human judgments of inferred text, collected via Amazon Mechanical Turk.[5] As with the crowdsourced evaluation presented in Section 4.3.2, we present annotators with the text of a predecessor sentence and elicit numeric ratings for different systems' top inferences based on the presented text, with ratings ranging from 1 ("very unlikely/irrelevant") to 5 ("very likely"), with 3 described as "neutral." We also allow annotators to mark a sentence as "nonsense," which we count as 0. We collect 3 annotations apiece for 150 test sentences drawn from the test set from Section 5.3.4. Since the event-level systems sometimes infer an empty sequence of events (because some sentences do not have finite non-copula verbs), we restrict the set of examples we elicit annotations for to those examples for which the event-level systems infer non-empty event sequences.

Table 5.5 gives human evaluation results. The four automated systems are as presented and described in Section 5.3.4. The "Gold" system is the gold-standard sentence, that is, the actual text treated as a reference in Section 5.3.4. The text-level systems are rated more highly than the event-level systems, and comparable bidirectional systems are rated more highly than their unidirectional counterparts.

[5] `http://mturk.com`

System	Judgment (0-5)
$e_1 \to e_2 \to t_2$	1.94
$(e_1 \wedge e_3) \to e_2 \to t_2$	2.00
$t_1 \to t_2$	2.35
$t_1 \to t_2 \leftarrow t_3$	**2.56**
Gold	3.86

Table 5.5: Crowdsourced evaluation results of inference quality for different systems. Higher is better, with min 0 and max 5.

Note that these numbers are generally somewhat lower in magnitude than the previous crowdsourced evaluation results, presented in Table 4.2, though the interface of this task was very similar to the task presented in that chapter. This could be because the tasks were presented slightly differently—in Chapter 4, tasks were presented as "snippets" containing tokens marked as "X," which the annotators were instructed to read as "something" or "someone." This detail was included to account for comparison with the systems of Chapter 3, which contained references to the variable referred to in that chapter as "O," standing for an entity not in the input text. Further, in this evaluation, we presented the gold-standard sentence as one of the inference texts, in addition to the automatically inferred text segments, which may prime annotators to view other inferences as less plausible.

It is worth noting that the gold sentences (that is, the actual text to occur at a given inference position in Wikipedia) received a rating of 3.86 out of 5, with 3 described to annotators as "Neutral," which is lower than may be intuitively expected. The bidirectional text-level system's score of 2.56 is 1.3 points (out of a six-point-total scale) worse. This indicates there is considerable room for improvement. The low gold-standard rating indicates that annotator expectations may be biased in un-

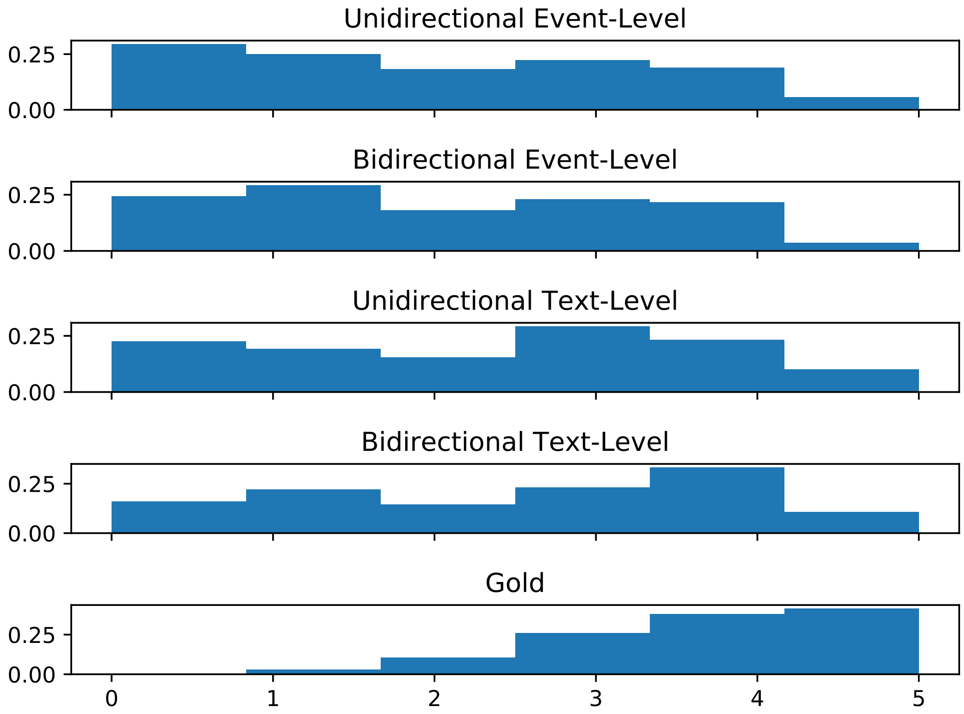


Figure 5.4: Histograms plotting the distribution of human ratings (from 0-5) for inferences from different systems. Horizontal axis gives percentages of ratings at a particular value. Histograms are given in the same top-to-bottom order as in Table 5.5.

expected ways. More detailed experiments could disentangle the possible issues and biases present in these results, but this is left as future work.

Figure 5.4 gives histograms for the distributions of annotations for different systems. The number of very low-rated inferences can be seen to be lower for both bidirectional and text-level systems. The best automatic system lags behind the gold output, particularly in the number of “5” ratings, clearly indicating that there is significant room for improvement.

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

Figure 5.5: Sample next-sentence text predictions. ⟨OOV⟩ is the out-of-vocabulary pseudo-token, which frequently replaces proper names.

5.3.6 Qualitative Analysis

Figure 5.5 gives some example automatic next-sentence text predictions, along with the input sentence and the gold-standard next sentence. Note that gold-standard successor sentences frequently introduce new details not obviously inferable from previous text. Top system predictions, on the other hand, are frequently fairly short. This is likely due in part to the fact that the cross-entropy loss does not directly penalize short sentences and part to the fact that many details in gold-standard successor text are inherently difficult to predict.

Figure 5.6 gives density plots of a number of randomly selected pairs of text-level hidden-state dimensions. That is, we take a number of sentences, aggregate the text-level hidden encoder states for every sentence, and plot the joint densities

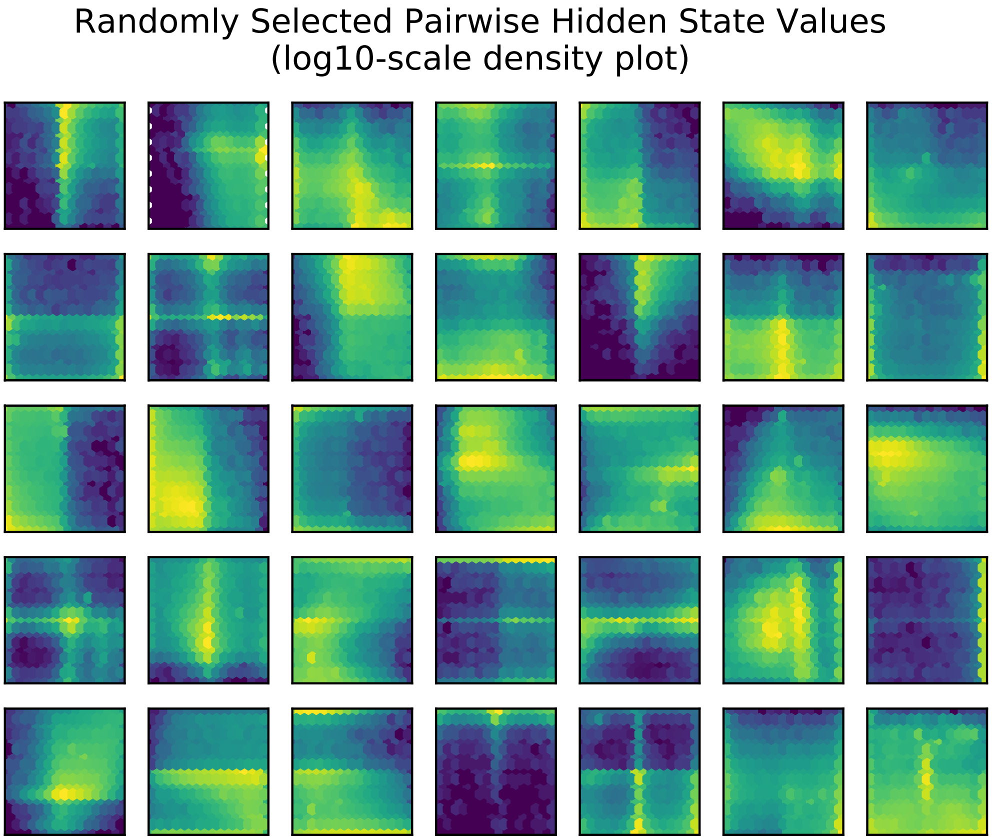


Figure 5.6: Density plots of some randomly selected pairwise text-level hidden state dimensions after encoding input text. We show 35 of a possible $\binom{100}{2} = 4950$ dimension pairs, which range from -1 to 1. Axes are linear; density is colored based on a logarithmic scale, with lighter colors reflecting higher density.

for randomly selected dimension pairs. A spectrum of different types of behavior emerges—some dimension pairs are more obviously independent, with the joint equal to the product of the marginals, and others exhibit more dependent behavior of various types. A standard set of techniques has not yet emerged to analyze and interpret the behavior of latent LSTM activity, so we restricted our analysis above to system output; we note, however, that it is clear from the density plots that different units behave and interact quite differently on the dataset, with varying degrees of independence.

5.3.7 Discussion

The general low magnitude of the BLEU scores presented in Table 5.1, especially in comparison to the scores typically reported in Machine Translation results, indicates the difficulty of the task. In open-domain text, a sentence is typically not straightforwardly predictable from preceding text; if it were, it would likely not be stated. This motivates the human evaluation results given in Table 5.5, which yielded the same comparative system ordering as the automatic BLEU-BP and 1-gram precision results in Table 5.1.

On the task of verb-argument prediction in Table 5.2, the difference between $t_1 \rightarrow t_2$ and $e_1 \rightarrow e_2[0]$ is fairly marginal. This raises the general question of how much explicit syntactic analysis is required for the task of event inference, particularly in the encoder/decoder setup. These results provide evidence that a sentence-level RNN language model which operates on raw tokens can predict what comes next in a document as well or nearly as well as an event-mediated script model.

RNN sequence models have been demonstrated to be capable of learning recursive tree-structured syntax in some settings (Bowman et al., 2015); it is possible that the LSTM text-level networks investigated in this chapter are automatically inducing limited forms of syntactic compositional phenomena (e.g. constituent boundaries, subject-verb or direct-object dependencies, and other types of dependency links) when constructing latent representations, in particular to the extent that they are empirically useful for the prediction task. In other words, one possible conclusion to draw is that, for most cases, easily inducible notions of compositionality suffice to get near to the space of realistic predictions.

The text-level model has access to more lexical clues (adverbs, full NPs of arguments, etc.) than the event-level model, which is based on phrasal head words; these cues likely have a reasonable amount of predictive power. The extent to which different types of lexical cues in input sequences help improve inference is a topic for future study.

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

Chapter 6

Neural Discourse Features for Coreference Resolution

In Chapter 5, we presented evidence that encoder-decoder models operating at the level of raw text perform competitively with analogous systems operating at the level of predicate-argument structures, on the task of predicting the latter (after parsing predicted raw text, if necessary). This raises questions about the general utility of directly modeling predicates and arguments, instead of raw text. To this end, we investigate incorporating features derived from the systems of Chapter 5 into a useful extrinsic task, coreference resolution. We find, using the limited set of features and experimental setups we consider, that the event-level features are more beneficial than the text-level features.

We consider a number of ways of integrating the text- and event-level script systems described in Chapter 5 into a modern feature-based coreference system (Durrett and Klein, 2013). We find that, while the top-level effect of such features is modest, a few specific classes of difficult coreference decisions see improved performance over the baseline system when features derived from the systems described in Chapter 5 are added. We identify and investigate these classes.

6.1 Background

6.1.1 The Task of Coreference Resolution

Coreference resolution is the task of automatically clustering a document's noun phrases into classes such that all and only the noun phrases in a particular class refer to the same entity in the world. For example, given the following passage:

> After a series of court battles, the U.S. Supreme Court unanimously ruled that [the president]$_1$ was obliged to release [the tapes]$_2$ to government investigators. [The tapes]$_2$ revealed that [Nixon]$_1$ had attempted to cover up activities that took place after the break-in.[1]

a coreference resolution system should indicate that *the president* corefers with *Nixon*, and that two two instances of *the tapes* corefer, as indicated by the subscripts. We follow the standard conventions of calling the noun phrases comprising a cluster *mentions*, and calling the referent (that is, the label on a particular cluster of mentions) the *entity*. Note the above example displays one of the more difficult phenomena for current systems to handle: namely, a proper coreference analysis should determine that *Nixon* has an antecedent, even though it is not anaphoric (that is, it is a proper noun, and therefore does not obviously have a coreferring antecedent, as opposed to a pronoun like *he* or *they* that obviously would[2]). Similarly, the nominal mention *the tapes*, the second time it occurs, has an antecedent

[1] `https://en.wikipedia.org/wiki/Watergate_scandal`

[2] Note there is a slight terminology difference in the literature here: NLP/CL papers will sometimes call a Noun Phrase like *Nixon* anaphoric because it has an antecedent, whereas the linguistics literature sometimes uses the term to refer only to pronouns or similar phenomena.

(namely, the same string's first occurrence); generally, most noun phrases in text (e.g. *government investigators* or *activities* above) will be singletons, lacking other coreferring entities.

It bears mentioning briefly that coreference in natural language comprises a much more general and nuanced set of phenomena than described above. We have restricted coreference links above to identity relations—that is, two NPs corefer iff they refer to the exact same entity—but the linguistics literature has analyzed other types of anaphora. For example, *bridging* (Clark, 1975; Prince, 1981; Poesio et al., 2004) is a more general notion of coreferential phenomena that includes set membership, metonymic, and part-whole relationships, among others. For example, consider the following sentence:

> A major fire broke out in the Parthenon shortly after the middle of the third century AD which destroyed the roof and much of the sanctuary's interior.[3]

The antecedent of *the roof* is *the Parthenon*, but the relationship is a part-whole one, rather than an identity one. We do not consider such bridging anaphora in our system. Karttunen (1969) identifies a class of pronomial reference called *paycheck pronouns*, exemplified by his classic example:

> The man who gave his paycheck to his wife was wiser than the man who gave it to his mistress.

In this example, *it* is not (identically) coreferent with *paycheck*; since it refers to a different instance of paycheck (there are two men here, with distinct paychecks),

[3] `https://en.wikipedia.org/wiki/Parthenon`

a formal analysis of the pronoun *it* will have it pick up the definite description introduced by its antecedent, but introducing a new entity with a description that is saliently similar to its antecedent's. Interpretation of such pronouns is well beyond the capabilities of current systems. We follow the majority of work on coreference resolution and restrict our attention to identity relationships.

The task of coreference resolution—clustering noun phrases referring to the same entity in some world—is an intuitively appealing application for incorporating different types of formalized world knowledge. In particular, statistical models of event co-occurrence could be brought to bear to the task. Consider the following example, given in Section 2.4, reproduced again for convenience:

> Rasputin was said to possess the ability to heal through prayer and was indeed able to give the boy some relief, in spite of the doctors' prediction that **he** would die.[4]

Deciding which entity *he* refers to requires world knowledge (the person who may die is the person needing relief, not the person capable of healing). Consider another example:

> While Tsar Nicholas II was away at war, Rasputin's influence over Tsarina Alexandra increased. **He** soon became her confidant and personal adviser....[5]

To resolve *He* to *Rasputin* and not *Tsar Nicholas II*, it is helpful to know that the events *X's influence over Y increased* and *X became Y's confidant* are likely to co-

[4] `https://en.wikipedia.org/wiki/Grigori_Rasputin`

[5] Ibid.

occur for entities X and Y. Indeed, both the Stanford deterministic coreference system (Lee et al., 2013) and the Illinois coref system (Peng et al., 2015a) mis-cluster the pronoun *He* in this example.

In this chapter, we investigate incorporating features from event co-occurrence systems into a feature-based coreference resolution system. A number of other prior approaches to incorporate world knowledge of different types into automatic coreference systems are described in Section 6.4.

6.1.2 Automatic Coreference Resolution Systems

In the last two decades, the state of the art of automatic Noun Phrase (NP) coreference resolution systems has progressed dramatically; however, coreference resolution remains one of the more elusively difficult standard NLP tasks, as it requires a greater amount of extra-linguistic world knowledge than most other tasks in practice. We give a brief description of the field; for a survey describing the partial landscape of considerations and techniques in Machine Learning-based coreference resolution, see Ng (2017).

The task of coreference resolution—labeling NP mentions such that coreferent mentions have identical labels—is fundamentally a clustering problem. For computational reasons, however,[6] the complex structured inference problem of coreference resolution problem is typically framed as a sequence of simpler classification problems. To oversimplify a bit for the purpose of clarity, there are a number of

[6] The number of clusterings of n points is given by the nth Bell number, which is very large indeed; further, it is not obvious what topology to use to induce a distance between mentions for clustering.

broad approaches to decomposing the coreference clustering decision into smaller decisions:

- **Mention-pair models** consider pairs of entity mentions independently, learning a binary model (either a classification model or a ranking model) which labels a candidate pair of mentions as coreferent or non-coreferent (Soon et al., 2001). There is some subtlety in these models concerning training instance creation—if all $\binom{n}{2}$ mention pairs are input as training instances, there is a large class imbalance—so different systems vary on the scheme used to construct training examples.

  These models may either make independent decisions for all mention pairs (Clark and Manning, 2016a), or they may restrict the classifier to choose at most one antecedent for each mention (Denis and Baldridge, 2008; Durrett and Klein, 2013).
- **Entity-mention models** consider, for a given mention, whether it is coreferent with a possible antecedent entity. That is, the local decision is whether to mark a single mention as coreferent with a cluster of mentions (Luo et al., 2004; Rahman and Ng, 2009).
- **Entity-entity models** frame the decision as work by classifying two mention clusters as coreferent or not—that is, the low-level decision made by the classifier is whether to merge two groups of mentions into a single group (Lee et al., 2013; Wiseman et al., 2016; Clark and Manning, 2016b)

Coreference Resolution systems typically have two distinct stages. First, the system will perform an automatic **mention identification** step, in which candidate NP mentions will be identified. That is, the system constructs the list of NPs which it will later cluster, possibly including many singleton mentions that do not corefer (Recasens et al., 2013). Second, the system will **cluster** the identified candidate mentions according to its model of coreference, typically using one of the paradigms listed above.

State-of-the-art systems from the last few decades have typically used machine learning (Soon et al., 2001; Martschat and Strube, 2015; Peng et al., 2015a; Wiseman et al., 2016; Clark and Manning, 2016b), though note the deterministic Stanford coreference system achieves strong results using deterministic handwritten rules (Raghunathan et al., 2010; Lee et al., 2013). Clark and Manning (2016a) give a coreference resolution system that uses reinforcement learning, framing the problem as a series of sequential decisions to merge clusters.

6.2 Methodology

6.2.1 Baseline System

Pure Coreference System

We extend the system of Durrett and Klein (2013) by adding additional features and investigating their effect. This system, while not quite state-of-the-art, is fairly simple, freely available, easily extensible, and has a relatively fast training time. In the remainder of this section, we give a very brief system description; for more details, see Durrett and Klein (2013).

The system starts with a high-recall mention detection system, labeling mentions as either proper mentions of named entities, pronouns, or nominal mentions (that is, non-proper Noun Phrases), as a deterministic handwritten function of the syntactic parse and named entity recognizer output of the document text. The coreference model is a mention-pair model that selects at most one antecedent per mention (a so-called *mention-synchronous* model). That is, for the ith candidate mention in a document, the model defines a random variable that can take i possible values: either an integer in $1 \ldots i - 1$, indicating that the mention at that index is an antecedent of mention i, or the value NEW, indicating that this mention does not have an antecedent.

Supposing there are n mentions in a document, and an individual mention at index i has a random variable A_i (taking i possible values, as described above), and calling all document context X, Durrett and Klein (2013) use a conditional log-linear model

$$P_\theta(A|X) = \frac{1}{Z} \exp \sum_{i=1}^{n} \theta^T \phi(i, A_i, X) \tag{6.1}$$

with ϕ the feature function giving a feature vector for mention A_i in position i in document X, θ the model's learned parameter vector, and Z the appropriate normalizing factor to ensure P is a distribution.

One complication in training this model arises from the fact that, though the sequence $A_1, \ldots, A_n$ uniquely determines a coreference clustering for a sequence of n mentions (by taking the transitive closure), the converse is not true: if mention i is in a coreference cluster, it in general has multiple possible A_I values. This is overcome by marginalizing over all antecedent vectors consistent with the gold standard clustering during training.

The loss function minimized also has a log-linear term penalizing three classes of errors according to different hyperparameter coefficients, chosen to trade off different classes of errors. The three types of penalized errors are (1) false anaphoric decisions (choosing an antecedent instead of NEW); (2) false new decisions (choosing NEW instead of an antecedent); and (3) wrong link errors (choosing the wrong antecedent for an anaphoric mention). This additional structured loss term is combined with the conditional log-linear model into a softmax-margin objective (Gimpel and Smith, 2010). This tradeoff between three error types was also adopted by Wiseman et al. (2015) and Clark and Manning (2016a,b).

Letting $\ell(A, C^*)$ denote this three-term linear sum trading off different error types (with C^* the gold-standard clustering), we define

$$P'_\theta(A|X) \equiv \frac{1}{Z} P(A|X) \exp \ell(A, C^*)$$

(with Z the appropriate normalizing factor). We define $\mathcal{A}(C^*)$ to be the set of antecedent vectors consistent with the gold clustering; the training objective is then:

$$J(\theta) = \sum_{k=1}^{t} \log \left(\sum_{A \in \mathcal{A}(C^*)} P'_\theta(A|X_k) \right) + \lambda \|\theta\|_1, \tag{6.2}$$

with the ℓ_1 regularizer used to induce sparsity. Parameters are learned via AdaGrad (Duchi et al., 2011). See Durrett and Klein (2013) for hyperparameter settings.

We will describe the original system's features briefly, rather than exhaustively, as these are not original contributions and are not central to the analysis; for a complete description, see Durrett and Klein (2013, 2014). The baseline system uses what Durrett and Klein (2013) call the FINAL feature set. Briefly, the system uses a number of simple surface features (e.g. head word identity, preceding/following word identity, mention length), along with a number of slightly more complex features (whether mentions are nested, dependency parent and grandparent, speaker, and number and gender according to Bergsma and Lin (2006)). The full feature list is given in Tables 1 and 6 of Durrett and Klein (2013).

Joint System and Pruning

Though we modify the coreference system of Durrett and Klein (2013), as described above, we carry out experiments with the software described in Durrett and Klein (2014), which jointly trains the coreference resolution system described above, along with a named entity recognition system and an entity linking system. We do not train this full joint system in experiments; we modify only the input features to the coreference component, and only retrain the coreference component.

Notably, during learning and inference, the system constructs a factor graph for each document (see Durrett and Klein (2014) for details of this construction), and the number of edges in this factor graph is quadratic in the number of document mentions. For tractability of learning and inference, the system prunes possible coreference links by, first, training the pure coreference system in isolation and using the model probabilities from this system to trim unlikely mention-mention edges. In particular, it trims edges linking to possible antecedents whose antecedent probability is much lower than the system's most confident guess. That is, the full

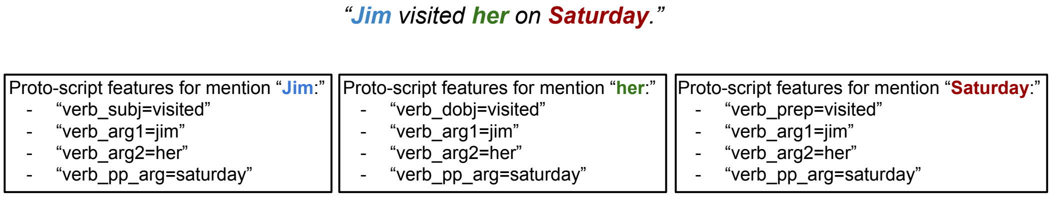


Figure 6.1: Example of proto-script features for different mentions in the sentence "Jim visited her on Saturday."

system, with a coreference-only probability function p_c, prunes antecedent links a_i for mention i in document x for which

$$\log p_c(a_i|x) - \log p_c(a_i^*|x) < k,$$

where $k \in \mathbb{R}$ is a hyperparameter and a_i^* is the coreference system's most confident guess for the mention i's antecedent. We use the system's default value of $k = 5$.

6.2.2 Additional Script Features

We investigate the effects of augmenting the baseline system with a number of statistical-script-related features. We investigate the direct incorporation of features derived from systems described in previous chapters. Some of the baseline features may be expressive enough to tangentially include some related information, as we describe in more detail below.

Proto-script Features

We first investigate the inclusion of what we call **proto-script features**, which are a subset of the dependency structure capturing some of the event-level information of events as defined and used in Chapters 4 and 5. In particular, we use grammatical subject relations, direct object relations, and prepositional relations, derived from automatic dependency parses of the sentences. An example is given in Figure 6.1 for a simple sentence. These features include parent and child verb information, and information about the noun heads of other predicates. They are, essentially, features based on the single-argument event features of Chambers and Jurafsky (2008), described in more detail in Chapter 3.

For a given mention, we identify if it is the subject, direct object, or prepositional object of a verb; if so, we include the (lowercased) verb form as a feature, along with the (lowercased) forms of the head nouns of any other arguments of the verb. We also investigated conjunctions of such features, but found them to be of negligible benefit.

These features are, in principle, capable of capturing selectional preference information about events. If, for example, a mention is the subject of a verb which a candidate antecedent is very seldom the subject of, this information may be informative to a coreference system, and could be captured by such features. They are conceptually similar to the caseframe features described by Bean and Riloff (2004).

Text-Level Neural Features

Next, we investigate the inclusion of text-level encoder hidden features. Chapter 5 describes a neural model that is trained to take input text and predict the subsequent sentence. We encode sentences with coreference decisions into such a model and derive features from the hidden RNN state after this encoding step.

A sentence, after the encoding step of a text-level sequence-to-sequence model, will map to a vector in $\mathbb{R}^d$ (in our case, $\mathbb{R}^{100}$). Given the model used by the full baseline system, it is computationally infeasible to add an additional 100 features to each mention instance. We therefore use Principal Components Analysis (PCA) to reduce the dimensionality of the encoder feature vectors. PCA, briefly, takes the eigenvalue decomposition of the mean-zero design matrix X multiplied by itself, $X^T X$; taking the top k eigenvectors of this decomposition (that is, the eigenvectors with the largest-magnitude eigenvalues) gives a k-dimensional basis which, when points from X are projected into it, the variance of the projected data is maximized. That is, the PCA gives a lower-dimensional representation of the high-dimensional input space which maintains as much of the variance of the data-generating distribution as possible. See, e.g., Bishop (2006) for more description and analysis of PCA.

Once we reduce input vectors to their principal components, we independently quantize the PCA dimensions by taking quantiles such that each quantile has approximately the same number of points in the training set. Each sentence thus gets one feature per PCA dimension. These features are intended to target discourse-level information about what contexts the text containing a candidate mention would be expected to appear in.

Swapped Text-Level Neural Features

Since the baseline coreference resolution system makes pairwise decisions, at training and inference time there are actually two mentions—a candidate mention and one of its candidate antecedents. Inspired by Bansal and Klein (2012), we investigate what we call *swapped text-level neural features*, derived from swapping nominal mentions into the place of pronouns mentions. That is, if a candidate mention is a pronoun[7]), and a candidate antecedent is nominal (that is, a non-pronoun noun phrase), we textually substitute the full noun phrase of the latter into the position of the former, and take the text-level encodings using the same method as with the regular text-level neural features. Bansal and Klein (2012) find this technique to be beneficial when deriving features from a large n-gram corpus.

The motivation of these features is that, if a noun phrase is swapped into the context of one of its true coreferent mentions, the result should be discursively natural, whereas if the NP replaces a mention for a different entity, the resultant sentence may be discursively quite aberrant. These features are intended to target this sort of information.

Event-Level Neural Features

In Chapter 5, we investigated two types of systems, text-level systems and event-level systems. We derived the above text-level features from the former; we also investigate analogous event-level features from the latter.

Given a mention, we examine the automatic dependency structure of its sentence[8] to identify if the mention participates in a predicate-argument structure of the sort modeled by the event-level models in Chapter 5. If it does, we add the

[7] Pronominality is determined by membership in the closed list of pronouns used by the baseline system of Durrett and Klein (2013), comprising 49 word types

[8] Dependency parses are provided by version 1.8.2 of spaCy, available at `https://spacy.io/`

PCA-derived features of its embedding as mention features, as with the text-level neural features described above, in addition to an indicator feature giving what the relationship between the event and mention is (whether it participates as a subject of direct object, for example).

As with the text-level neural features above, these features are meant to target discourse-level information about the context a mention appears in; however, these features are more targeted at events than the text-level features, as they are derived from the structured predicate-argument structures in which a mention appears, which in general contain fewer tokens.

6.3 Evaluation

6.3.1 Dataset

Evaluations are carried out on the CoNLL 2012 shared task dataset (Pradhan et al., 2012), which is based on the OntoNotes corpus (Hovy et al., 2006). The English portion of the training portion of this corpus is approximately 1.3M words, comprising approximately 450k words from newswire, 150k from magazine articles, 200k from broadcast news, 200k from broadcast conversations, and 200k from web data. We use the standard dataset split of 2802 training documents (approx. 1.3M words), 343 development documents (approx. 160k words), and 348 test documents (approx. 170k words).

Figure 6.2 gives the difference between swapped and unswapped text-level sentence embeddings in the training set; that is, for every mention-swapped em-

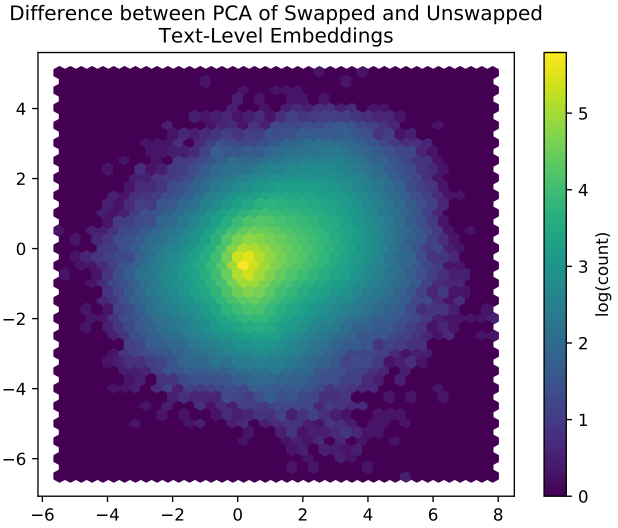


Figure 6.2: A density plot of the difference between the first two PCA dimensions of the swapped embeddings unswapped text-level embeddings of sentences in the CoNLL 2012 training set (first Principal Component on the horizontal axis). Bin densities are in logarithmic (base-10) space. The mean and mode are slightly offset from the origin.

bedding calculated from the dataset (in dimensionality-reduced PCA space), we subtract the corresponding embedding for the mention's sentence (in PCA space) without adulteration. Note the mean and mode are near the origin (the mean is (0.59, -0.38)): most sentences' embeddings are largely unchanged, but the difference the swapping operation makes in script features is relatively symmetric about the mode.

6.3.2 Training and Hyperparameter Details

We use the out-of-the box settings of the baseline coreference system of Durrett and Klein (2013); the software we use is actually the system described in Durrett and Klein (2014), which jointly trains a coreference resolution system, named entity recognition system, and an entity linking system. When training models, we do not retrain the full joint system, training only the coreference portion. The baseline coreference performance figures presented below, then, match those of Durrett and Klein (2014). We maintain all default hyperparameter settings of the baseline system.

For all script features, we use the top 5 principal components, each quantized into 5 quantiles (that is, each dimension is broken into quintiles). For computational tractability, we only consider swapping mentions that are within 100 mentions of each other in the document's mention sequence (that is, we do not consider, as a feature, swap features for mention pairs that are very far from each other).

All proto-script features are lowercased. Following the baseline coreference system's behavior with lexical features, we replace any proto-script feature that occurs fewer than 20 times with the part-of-speech of its headword, to handle the data sparsity introduced by the long-tail of rare words.

6.3.3 Quantitative Evaluations

Evaluation Metrics

Results are given in a number of standard coreference metrics, each measuring different aspects of performance: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and CEAF$_\mathrm{e}$ (Luo, 2005). We also provide the average of the F1 scores of these three metrics (frequently reported as the “CoNLL” score in the literature), as computed from the reference implementation of the CoNLL scorer (Pradhan et al., 2014). We now briefly describe these metrics; for a more in-depth discussion and comparison of them, see Moosavi and Strube (2016).

MUC is a link-focused metric: the target variable being measured is the global per-document set of pairwise links between entity links in clusters, against which one can calculate per-document precision and recall. That is, MUC precision is the number of predicted links present in the gold standard, and recall is the converse. This metric has the bias of naturally preferring large clusters: putting all mentions into a single cluster will yield perfect recall with, depending on the corpus, quite reasonable precision.

The B^3 metric is mention-focused: it computes per-mention precision and recall, and takes the (arithmetic) mean over mentions. For a given mention m with predicted mentions C_m and gold mentions C_m^* the number of correct mentions is given by $c \equiv |C_m \cap C_m^*|$, with the precision given by $c/|C_m|$ and the recall by

System	**MUC**			**B^3**			**CEAF$_e$**			**Avg**
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	
Berkeley	72.33	69.29	70.78	60.94	55.68	58.19	55.45	54.01	54.72	61.23
B+PS	**72.56**	68.80	70.63	**61.56**	55.34	**58.28**	55.65	54.06	54.84	61.25
B+script	72.22	68.72	70.43	60.60	55.22	57.78	55.62	54.03	54.81	61.01
B+swap	72.38	**69.32**	**70.82**	61.05	**55.72**	58.27	55.39	54.08	54.73	**61.27**
B+ev	72.44	69.07	70.71	61.06	55.46	58.13	**55.86**	**54.12**	**54.98**	**61.27**

Table 6.1: Top-level coreference results incorporating various features. Results are given in MUC, B^3, and CEAF$_e$ Precision, Recall, and F1, along with the arithmetic mean of the three F scores. B+PS is baseline with proto-script features; B+script is baseline with text-level features; B+swap is baseline with swapped text-level features; B+ev is baseline with event-level script features.

$c/|C^*_m|$. This metric is not biased toward large clusters like MUC is, and better handles singleton mentions (which do not have gold-standard links, and are therefore left out entirely of the MUC recall metric).

The CEAF$_e$ metric aligns gold and predicted entities (i.e., clusters of mentions) such that each gold cluster aligns to at most one predicted cluster, and vice versa. Once this alignment is made, it is straightforward to calculate precision, recall, and F1. A system with too many clusters suffers low CEAF$_e$ recall, while a system with too few suffers in precision. See Luo (2005) for details on cluster alignment.[9]

Top-Level Results

Top-level system results are given in Table 6.1. We give the Berkeley system results (Durrett and Klein, 2014), along with results of the baseline retrained with features described in Section 6.2.2. The proto-script features (B+PS) generally increase precision and decrease recall, while the text-level script feature hurts both. The swap features give performance comparable to the baseline. The event-level features (B+ev) give a marginal improvement on CEAF_e. Generally, the effects of the additional script features on top-level performance are marginal.

It is worth noting that, generally speaking, the additional features investigated tend to marginally improve precision while leaving recall either approximately the same or slightly worse. Most candidate mentions in text are non-coreferring singletons, and of the $\binom{n}{2}$ mention pairs, most do not corefer, even if they are semantically compatible entities. Since the prior probability on two mentions coreferring is so low, weakly informative features are generally not, in themselves, sufficient evidence for making coreference links. For example, knowing that two mentions refer to animate entities is not an incredibly informative signal when deciding whether those entities corefer (consider a text mentioning many people); on the other hand, knowing that one mention is animate and another is inanimate gives a strong indication that the two entities do not corefer. The script features are, generally, intended to capturing semantic information about entities and the events

[9] CEAF_e is the now-standard name for the CEAF clustering metric that uses what Luo (2005) calls function ϕ_4, the mention-level F-measure, for cluster similarity when making alignments (they also consider a number of other now-less-commonly-used notions of cluster-similarity when making alignments).

they engage in, and are therefore intended primarily to increase precision.

Results on Difficult Types of Coreference Decisions

We also give accuracy results on a number of difficult restricted decision classes, in addition to the top-level system performance. In describing the classes, we also give the short abbreviations we use to refer to the classes in tables below. System performance on these restricted classes lags behind top-level performance, as surface-level cues, as incorporated, are generally not informative enough for these hard decisions, and incorporating world knowledge may conceivably have a larger impact. The restricted decision classes considered are:

- **Nominal/proper mentions with nominal/proper antecedents, no head match ("N/P+NHM")**: the system's performance accuracy on nominal or proper mentions (that is, mentions that are not pronouns) which have at least one gold-standard nominal or proper antecedent, and, for all gold antecedents, the following condition holds:

  (a) the candidate and antecedent have distinct head nouns.

  This represents a difficult class of coreference decisions, since the majority of non-pronomial NPs in a document do not corefer, and the head-matching heuristic is very informative in such cases (Raghunathan et al., 2010). This covers, for example, the bolded NP in the following example (antecedent italicized):

  During Johnson's administration, NASA...prepared to make the

> first manned *Apollo program* flights. On January 27, 1967, the nation was stunned when the entire crew of Apollo 1 was killed in a cabin fire during a spacecraft test on the launch pad, stopping **Apollo** in its tracks.[10]

The head noun of the antecedent, *program*, does not match the head noun of its coreferent mention *Apollo*, though the latter's head is contained in the full antecedent NP.

- **Nominal/proper mentions with nominal/proper antecedents, no head containment ("N/P+NHC")**: the system's performance accuracy on nominal or proper mentions which have at least one gold-standard nominal or proper antecedent, and, for all gold antecedents, condition (a) above holds, along with the following two conditions:

  (b) the antecedent's head noun is not contained in the candidate NP; and
  (c) the candidate NP's head noun is not contained in the antecedent.

This class further restricts the surface-level string features available to inform the coreference decision. This class is represented by the bold text in the follow example (antecedent italicized):

> After the election, Johnson was quite concerned about the traditionally ineffective nature of *his new office*, and set about to assume authority not allotted to **the position**.[11]

[10] `https://en.wikipedia.org/wiki/Lyndon_B._Johnson`
[11] Ibid.

This is the most difficult class of coreference decisions analyzed. Durrett and Klein (2013) call performance on this class "the biggest weakness of [their] system."

- **Nominal/proper mentions with nominal/proper antecedents, no head match, at least one proper ("N/P+P")**: the system's performance accuracy on mentions with at least one gold-standard antecedent such that condition (a) above holds for all antecedents (that is, there are no head matches), and **either**:

  (d) the candidate is a proper mention, and at least one antecedent is a nominal/proper mention; or

  (e) at least one gold antecedent is a proper mention, and the candidate is nominal.

  This case targets the situation exemplified by *the president* in the following example:

  > Moreover, many members of the *Kennedy* White House were contemptuous of Johnson, including **the president**'s brother....[12]

Note that the baseline system's mention-synchronous framework allows us to compute accuracy straightforwardly: a given mention will have at most one predicted antecedent, and that possible-antecedent is either correct according to the gold-standard or it is incorrect.

Accuracy results on these restricted sets of coreference decisions are given

[12] Ibid.

System	**Overall**	**N/P+NHM**	**N/P+NHC**	**N/P+P**
Berkeley	77.65	15.25	6.78	22.07
B+PS	77.37	13.43	5.61	19.79
B+script	77.51	15.61	6.78	22.65
B+swap	**77.67**	15.25	6.89	21.98
B+ev	77.61	**16.15**	**7.63**	**23.41**

Table 6.2: Accuracy results (in percentages) on the restricted subsets of coreference decisions, described in the text. B+PS is baseline with proto-script features; B+script is baseline with text-level features; B+swap is baseline with swapped text-level features; B+ev is baseline with event-level script features. Results are given in percentages.

in Table 6.2. Note that, while overall accuracy is not appreciably improved by the features listed, a number of the other restricted classes see marginal improvements with additional features. The "N/P+P" class (nominal/proper mentions with nominal/proper mentions, such that at least one is proper) sees a 0.55% improvement with the text-level script features, and a 1.34% improvement with the event features. The N/P+NHC class, which is a very difficult class of decisions (the baseline system accuracy is below 7%) sees a 0.85% performance improvement with the event-level script features (corresponding to a 12.5% relative improvement). This provides evidence that event-level script features, which have a strong intuitive claim as helpful for coreference resolution, can help improve performance on classes of difficult coreference decisions.

To better understand system performance on these restricted subsets of decisions, we also examine what performance on these classes looks like "from the perspective of the classifier," so to speak, by restricting ourselves to the mention-mention links available (after pruning) to the system during inference. Table 6.3

System	**Overall**	**N/P+NHM**	**N/P+NHC**	**N/P+P**
Berkeley	87.46	43.71	41.39	46.18
B+PS	**87.49**	**54.37**	**56.06**	**54.65**
B+script	87.38	43.77	40.72	46.66
B+swap	87.45	44.05	41.03	46.08
B+ev	87.47	43.37	41.44	45.65

Table 6.3: Accuracy results, as classified with system-pruned edges. The restricted subsets of coreference decisions are classified using only the edges remaining after system pruning. B+PS is baseline with proto-script features; B+script is baseline with text-level features; B+swap is baseline with swapped text-level features; B+ev is baseline with event-level script features.

gives results for the same coreference classes as in Table 6.2, but with class membership calculated with system-pruned edges, as described in Section 6.2.1. That is, when deciding whether a particular point falls into one of the coreference classes under analysis, we look only at the mention-mention edges available to the system after edge pruning (so the denominators are typically smaller). In other words, this analysis measures the accuracy of these restricted classes from the perspective of a system that cannot see all the gold-standard links. Note, first of all, that the accuracy figures in this table are higher in absolute terms than Table 6.2—since these decision types are generally quite difficult, correct antecedents will frequently have low probability under the coreference model, and will therefore be filtered out and not considered by the system. Note, second, that the proto-script features are remarkably helpful on the restricted decision classes as calculated under the pruning, but actually harm system performance when the sets are calculated against gold. This suggests that, when these features are included, the system improperly prunes out more links to gold antecedents in these difficult classes.

6.3.4 Discussion

We have investigated a number of ways of incorporating various types of features derived from script models into a feature-based coreference system. As discussed in Section 6.3.3, these features are intended to be a weak signal capturing semantic properties of mentions, and such signals are naturally better suited for increasing precision than they are for increasing recall (since the great majority of mention-pairs represent non-coreference, a semantic signal must be very strong to overcome the high prior probability of non-coreference). Indeed, as shown in Table 6.1, precision increases on more feature sets than not, across metrics; this more often than not happens at the expense of recall, with the net result that F-scores are generally improved only marginally, if at all

As shown in Table 6.2, the event-level script features investigated improve accuracy on the class of hard decisions the baseline Berkeley system seems to most struggle with (nominal/proper mentions with new noun heads), increasing accuracy by 0.85%, from 6.78% to 7.63%. Though this is a modest increase in absolute terms, it represents a nontrivial relative improvement on a very difficult class of decisions. These difficult classes that see empirical improvement with script features do represent a minority of the dataset; though we provide evidence that script features can help with these difficult examples, an accuracy improvement of less than 1% does not have a very large top-level effect.

There are many additional possible ways of creating features appropriate for a coreference system from the models in Chapters 4 and 5, which are left to future work. One clear path forward is to integrate a neural model of events in sequence directly into a modern neural coreference system (Wiseman et al., 2016; Clark and Manning, 2016a,b), so that all latent features could be jointly learned to optimize a more meaningful coreference-related objective function.

6.4 Related Work

There have been numerous investigations into various schemes of integrating world knowledge into different coreference systems, both in the form of lexical features mined from large text corpora (as in the present chapter), and in other forms. We describe a few related efforts in this general vein.

Bean and Riloff (2004) describe a number of types of *caseframe* features, which capture syntactic interactions between candidate mentions and patterns extracted from an Information Extraction system; these features are conceptually similar to the proto-script features described above. They preprocess a corpus with an Information Extraction system and collect sets of lexicalized patterns which trigger extracted facts, which they call caseframes. They go on to identify co-occurring patterns between pairwise anaphor-antecedent links, in addition to calculating selectional preference-type statistics about which nouns participate in these patterns. They find these features to be beneficial compared to baseline systems on two different single-domain corpora.

Kehler et al. (2004) perform a number of experiments showing that predicate-argument statistics, which target selectional preference information, do not improve the performance of a number of baseline coreference systems trained and evaluated on the ACE 2002 dataset.

Ponzetto and Strube (2006) show that, on the ACE 2003 dataset, noisy automatically-labeled PropBank-style Semantic Role Labeling features (Palmer et al., 2005) provide a +1.5% improvement in MUC F1 (primarily from increased recall) over a simple baseline (that of Soon et al. (2001), an early learning-based coreference system).

Bergsma and Lin (2006) describe a coreference resolution system that uses features derived from the dependency path between two potentially coreferent mentions, that is, the sequence of nodes and dependency labels connecting two candidate mentions in a dependency parse tree. For a particular dependency path connecting two noun phrases, they calculate the percentage of occurrences of the path in a large newswire corpus which connect two likely-coreferent noun phrases (e.g. third-person singular masculine pronouns), yielding a mapping which takes a dependency path and yields the probability of its connected mentions being likely coreferent NPs; they use this to bootstrap information about the gender and number of noun phrases. This method is capable of learning selectional preferences of dependency paths, along with more general syntactic features predictive of coreference.

Rahman and Ng (2011) investigate various methods of incorporating world knowledge features into a baseline coreference system. In particular, they investigate features derived from FrameNet (Baker et al., 1998), encoding whether predicates and arguments appear in the same FrameNet frame. They also investigate indicator features encoding PropBank-style semantic arguments and predicates of mentions, which contain predicate identity information for mentions appearing as semantic roles of verbs. They find verb and FrameNet features to have modest positive effects (+0.3 to +0.9 F1 on OntoNotes) when added to the baseline Mention-Pair and Cluster-ranking models.

Irwin et al. (2011) investigate incorporating the narrative schemas described in Chambers and Jurafsky (2009) into the cluster-ranking coreference algorithm given in Rahman and Ng (2009), finding the schema features to improve system performance on B^3 F1 and decrease performance on MUC and CEAF F1.

Bansal and Klein (2012) investigate using a number of features derived from the Google n-grams corpus (Brants and Franz, 2006) as input to a mention-pair model. In particular, they investigate n-gram features derived from swapping non-pronomial candidate antecedent mentions into the position of pronomial anaphors, querying the n-gram corpus to see how many times the resulting string is attested on the web. This is, in a sense, the n-gram version of some of the RNN-based methods presented above in Section 6.2.

Peng and Roth (2016) investigate co-occurrence based sequence models mediated by automatic FrameNet parses (Baker et al., 1998). They show a performance improvement in the Illinois coref system (Peng et al., 2015a) by adding features relating candidate mention pairs to each other via invoked FrameNet frames in the text.

Levesque et al. (2011) argue that a task they call the Winograd Schema Challenge (WSC), named after the tasks' original motivation in Winograd (1972), can serve as a convincing alternative to the Turing test. An instance of the WSC task consists of a pair of very difficult coreference problems, namely, a pair of minimally-modified sentences such that the most reasonable interpretation of a pronoun changes between the two sentences. For example, the motivating example in Winograd (1972) is

(1) The city council refused the women a permit because *they* feared violence.

(2) The city council refused the women a permit because *they* advocated violence.

These examples may require many different types of background knowledge and reasoning to solve. For example, the pair

(1) The trophy would not fit in the brown suitcase because *it* was too big.

(2) The trophy would not fit in the brown suitcase because *it* was too small.

requires complex spatial reasoning to solve. A number of efforts have been made at solving WSC problems using statistical means (Rahman and Ng, 2012; Peng et al., 2015b); a corpus large enough to train a reasonably complex learned statistical system has yet to be developed.

Chapter 7

Related Work

There are a number of threads of research into modeling events in discourse in various ways. These efforts vary from modeling predicate-argument interactions in isolation (selectional preferences and generalizations thereof) to learning models of entire document or multi-document structures. The work presented in this thesis does not necessarily build directly off the work presented in this chapter; this chapter is intended to provide a broader survey of research into computational models of event and document structure.

7.1 Modeling Predicate-Argument Structures

There is a body of work investigating models of predicate-argument structures in isolation, that is, modeling compatibility of verbs with arguments, isolated from the discourse context in which they appear. There is, first, a body of work focused on building statistical models of selectional preferences, that is, which nominal arguments particular verbs “prefer” for different syntactic or semantic relations (Erk, 2007; Ritter et al., 2010; Ó Séaghdha, 2010).

There is also a body of literature on more general notions of statistical models of predicate-argument structure. Dasigi and Hovy (2014) give a neural model of verbs with nominal arguments trained to classify anomalous news events (those occurring in “weird news” sections of newswire websites) from more typical news events (that is, can we differentiate between “dog bites man” and “man bites dog”).

Grissom II et al. (2014) show that, in the context of simultaneous machine translation (incrementally constructing a translation as the input sequence is received in a stream) from a verb-final SOV language (such as German) to a verb-medial SVO language (such as English), predicting the source-side clause-terminal verb before it is observed helps to produce better target-side translations. In other words, learning to predict verbs from arguments (and other context) can help improve machine translation in certain settings.

Tilk et al. (2016) present a neural model of predicates with their nominal arguments (using PropBank-style role labels), capable of predicting predicates from arguments, or single arguments given predicates along with other arguments. Granroth-Wilding and Clark (2016) describe a feedforward neural network which composes verbs and arguments into low-dimensional vectors, evaluating on a multiple-choice version of the Narrative Cloze task.

7.2 Modeling and Using Event Sequences

Structured models of stereotypical sequences of events date back to AI research from the 1970s, in particular the seminal work of Schank and Abelson (1977). In this conception, scripts are modeled as temporally ordered sequences of symbolic structured events. These models are non-probabilistic, and the high degree of structure in the events create difficulties problems for automated script learning. Mooney and DeJong (1985) and Miikkulainen (1993) give models for automatically inducing such scripts from documents.

We first describe a number of previous investigations into models derived from various structured and semi-structured sources in Section 7.2.1. In Section 7.2.2, we survey a number of approaches to deriving script knowledge from text (note this is not a perfectly clear dichotomy, and there are a number of systems which, conceptually, could go into either of these two sections). Section 7.2.3 describes some research on different evaluations for such systems.

7.2.1 Models Derived from Structured and Semi-Structured Sources

Charniak and Goldman (1989) give toy examples of Bayes nets which can be used to recognize plans from stories, allowing them to make inferences about the presence of objects in the limited circumstances covered by the models. They propose that the general approach could could be expanded to allow inferences about states and events.

Liu and Singh (2004) describe ConceptNet, a graph automatically generated from the semi-structured English sentences in the Open Mind Common Sense corpus (Singh et al., 2002), intended for use in tasks requiring commonsense reasoning. This graph contains labeled edges between nodes, some of which relate events to each other, for example SubEventOf (e.g. “score goal” is a sub-event of “play sport”) and EffectOf (e.g. “entertainment” is an effect of “view video”).

Mueller (2004) gives a system which uses the classical logic-based discrete event calculus, using classical non-probabilistic theorem proving to process text in the MUC corpora (Grishman and Sundheim, 1996). They first use a hand-coded non-statistical classifier to identify which hand-coded script matches a document (based on either an automatic IE system's output or on human annotations of roles and events in the documents). From this, they build a reasoning problem for the script instantiation, and then apply commonsense reasoning (as encoded by their schema of logical axioms). On two held-out MUC test sets, the system is able to find satisfying logical models for 36% and 42% of the test documents, respectively, using the output of a previously published IE system for MUC. The effort reported to port the system to a new script type is three eight-hour days. This system is fundamentally non-probabilistic and requires hand-coding of situation-specific scripts.

Burchardt et al. (2005) give an in-depth theoretical treatment of how contextual relations between instantiated FrameNet frames (Baker et al., 1998) can be used to make textual inferences, walking though a manual case study, and arguing that the process could be automated using probabilistic inference rules.

Regneri et al. (2010) use Mechanical Turk to crowdsource the creation of a collection of Event Sequence Descriptions (ESDs), each of which is a human-generated sequence of events in a particular pre-determined scenario (for example, "making scrambled eggs" or "wedding"). They go on to use a Multiple Sequence Alignment algorithm to align events across different ESDs of the same scenario, allowing them to build Temporal Script Graphs, which are partial orderings of events preserving the event order of the input ESDs. In total, they collect 493 ESDs for 22 scenarios. Frermann et al. (2014) provide an improved model for the task. Wanzare et al. (2016) collect a larger corpus of 40 scenarios with 100 ESDs apiece, augmented with partial manual cross-ESD event alignment annotations. Regneri et al. (2011) give a method for grouping synonymous entity noun phrases in ESDs into equivalence classes.

Li et al. (2013) describe a method to automatically generate stories from graphs created from crowdsourced narratives, similar to the approach of Regneri et al. (2010). They first collect many crowdsourced linear textual descriptions of a situation, differing in the events described and the order in which they are stated. Second, they learn event types and typical precedence orderings, described in detail in Li et al. (2012). Third, they learn pairwise mutual exclusion constraints between events using a simple Mutual Information-based thresholding method. The system is evaluated on the task of generating stories, and system performance approaches that of untrained humans on a number of qualitative performance metrics.

Berant et al. (2014) present a method of answering questions about paragraphs in a biology textbook which pivots around a graph representation of the process described in the paragraph. In this representation, there are Trigger nodes for verbs, argument nodes for participants in triggers, and edges both between pairs of triggers (labeled with the type of relation expressed as obtaining between the triggers) and between triggers and arguments. The edges between events are labeled with ”Cause,” ”Enable,” or ”Prevent” (or the generalization of these labels to the disjunctive case), to indicate one trigger causing another, one trigger enabling another (preceding but being a necessary condition), or one trigger preventing another from obtaining, respectively. They use a simple log-linear model to generate such a representation from a novel paragraph. To answer a question, they generate a similar representation (with regular expressions in it) and try to unify the answer with the graph representing the process in the paragraph (and to try unifying potential other answers, to see if a contradiction can be reached). They achieve superior results over a number of naive (but competitive) baselines.

7.2.2 Models Derived from Text

Mooney and DeJong (1985) and Miikkulainen (1993) give models for automatically inducing script information from documents, described in Section 2.2.1. Fujiki et al. (2003) describe a simple method of learning non-statistical script information automatically from Japanese newswire. They learn pairwise relations between verbs and arguments by thresholding based on raw co-occurrence counts in a corpus of first paragraphs in news articles, relying on a thesaurus for lexical generalization.

Chklovski and Pantel (2004) describe a semi-automatically collected wide-coverage resource of pairwise relationships between verb types, including a happens-before relation type (for example, *buy* happens before *sell*), and an enablement relation type (Hobbs, 1978), which obtains when one event makes another possible (for example, *fight* enables *win*). They collect this resource by using Google to query the web for handcrafted lexico-syntactic patterns, and performing a frequency analysis on the results.

As described in Section 2.2, Chambers and Jurafsky (2008) give a method of modeling and inferring simple (verb, dependency) pair-events. Jans et al. (2012) describe a model of the same sorts of events which gives superior performance on the task of held-out event prediction; Rudinger et al. (2015b) follow this line of inquiry, concluding that the task of inferring held-out (verb, dependency) pairs from documents is best handled as a language modeling task. The work presented above in this thesis is, in a sense, an attempt to further probe the limits and utility of an approach like this.

Chambers and Jurafsky (2009) present a modification to the system of Chambers and Jurafsky (2008) that learns "narrative schemas" instead of the "narrative chains" of the earlier system. They extract chains of (verb, dependency) pairs involving a single entity, and then reason about all such chains in a document. They use two distinct perspectives when motivating methods. Mainly, they frame the work as induction of frame types, constructing objects similar to FrameNet (Baker et al., 1998) in an unsupervised manner. They also, however, evaluate on the Narrative Cloze, which evaluates systems on ability to infer document structure. There is also a body of work focusing on automatically inducing structured collections of events (Chambers, 2013; Cheung et al., 2013; Nguyen et al., 2015; Ferraro and Van Durme, 2016), typically motivated by Information Extraction tasks.

Kasch and Oates (2010) give preliminary qualitative results of a system which generates sets of topically-related predicates and arguments. They use Latent Semantic Analysis (Deerwester et al., 1990) to calculate predicates and arguments related to unigram topic queries (for example *eating* and *restaurant*), and judge relatedness of predicates and arguments to the query by maximizing Pointwise Mutual Information, inspired by (Chambers and Jurafsky, 2008).

Gerber and Chai (2010) investigate the identification of implicit nominal arguments for NomBank (Meyers, 2007) predicates, automatically identifying noun phrases which stand in implicit relations to nominal events (as opposed to the formal events considered in this thesis, which treat only verbs as predicates). They find that PMI-based (predicate, argument) pair features similar to those investigated in Chambers and Jurafsky (2008) are informative for the task.

Hajishirzi et al. (2011) give a method of translating sequences of sentences describing dynamic systems into sequences of formal events, using a domain-specific axiomatization of effect preconditions and postconditions to enforce narrative coherence of events in sequence, demonstrating its effectiveness in the RoboCup domain.

Chambers and Jurafsky (2011) give an unsupervised method for MUC-4-style template induction, for example, learning (from text alone) that bombing events have perpetrators, targets, victims, and instruments, and learning the type of language that frequently expresses these roles. They find the induced templates to be competitive for standard IE task against published algorithms which require hand-built templates.

Adel and Schütze (2014) compute millions of coreference chains across many documents and calculate two types of embeddings: first, on the Gigaword corpus, and, second, on the coreference chains of the Gigaword corpus. That is, they treat the tokens of coreferring Noun Phrases as words, and calculate word embeddings directly on that. They use these embeddings to perform antonym detection. Intuitively, antonyms and synonyms will both be distributionally similar—they appear in similar contexts—but antonyms will corefer to each other much less frequently than synonyms will. They treat the antonym/synonym decision as a binary decision problem and use the cosine similarities of embeddings in the previously described spaces as features input to an SVM. The coreference-only features perform relatively poorly in isolation, but improve in performance when added to the text-based features. The ensemble performs roughly equivalently to the text system on classifying all types of antonyms, but, when restricted to nouns, gives superior performance.

Modi and Titov (2014) present neural models of event ordering, optimizing for the binary decision of deciding if one event precedes another in text.

Le and Mikolov (2014) give a method (which they call *Paragraph Vector*, sometimes colloquially referred to as doc2vec or paragraph2vec) for learning distributed representations of arbitrary spans of text. The setup is similar to that of word2vec (Mikolov et al., 2013a); in the latter, words are mapped to dense vectors predictive of context words; in this setup, spans of text are mapped to dense vectors which, conjoined with word vectors, are predictive of context words. They demonstrate the low-dimensional embeddings are empirically useful as features for a number of downstream tasks. At inference time, calculating vectors for unseen spans is computationally expensive, as unseen spans' representations must be computed by gradient descent. This is, in a sense, the logical continuation of the approach of Mikolov et al. (2013b), who learn representations for simplex n-grams with high PMI by treating them as a single "word with spaces."

Bamman and Smith (2014) give a Bayesian method for learning a model of biographical information from biographies in Wikipedia, jointly learning event classes (consisting of distributions over terms) and distributions of ages when those events are likely to occur in the biography (using the weak supervision signals of the subject's birth year and the year of the events' occurrence).

Li and Hovy (2014) use a convolutional neural net model that, at the top layer, performs binary classification between sequences of sentences, some of which have an imposter sentence interposed, trained to identify imposters. This is used to gauge discourse coherence, defined as ability to detect permutation in sentence ordering. They try a number of ways of recursively computing a single sentence's latent input, and combine a number of these inputs in a hidden layer.

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 very similar to the systems we evaluated in Chapter 5 (which, indeed, were inspired by Kiros et al. (2015)), but they do not evaluate directly on their sequence-to-sequence system's predictive power.

Kıcıman (2015) describes a small investigation into mining individual users' Twitter streams to aggregate text which occur more often than chance after fixed query events, discovering, for example, likely outcomes mentioned after adopting an animal.

Iyyer et al. (2016) give a neural model of the dynamic trajectories of relationships between pairs of fictional characters, jointly learning relationship descriptors (similar to topics in topic models) and the manner in which realized descriptors change over the course of novels. In this way, the model describes the dynamics of how language used to describe characters interacting with each other changes over the course of fictional texts.

Ferraro and Van Durme (2016) give a model integrating various levels of event information of increasing abstraction, evaluating both on coherence of induced templates and log-likelihood of predictions of held-out events.

Peng and Roth (2016) give eight different Language-Model-style sequence models mediated by automatic FrameNet parses (Baker et al., 1998), using two sequence-construction setups and four types of Language Model on each. In the first sequence-construction setup, they construct sequences of evoked FrameNet frames and intervening discourse markers (using the list of explicit discourse connectives present in the Penn Discourse Treebank (Prasad et al., 2008)); in the second, they construct sequences of argument relations to evoked FrameNet Frames, pivoting around coreference output. That is, the first setup is conceptually similar to the more global sequences of Chapters 3 and 4 of this document, while the second setup is conceptually closer to the protagonist-level models of Chambers and Jurafsky (2008), Chambers and Jurafsky (2009), and Jans et al. (2012). In both setups, they investigate an n-gram model, the Skip-gram and CBOW models of Mikolov et al. (2013b), and the log-bilinear language model of Mnih and Hinton (2007). They show that adding, as a feature to the Illinois coref system (Peng et al., 2015a), the conditional probability of a mention's possible argument relation to a FrameNet frame in the text, conditioned on a possible antecedent's possibly-evoked FrameNet argument, provides an improvement on coreference resolution performance.

Ahrendt and Demberg (2016) investigate augmenting the co-occurrence based multi-argument event system of Chapter 3 by incorporating WordNet-mediated hypernym information about events' entity arguments. They find entity hypernym information to provide substantial improvements on the narrative cloze task across a number of different corpora.

7.2.3 Model Evaluation

A number of alternative automatic evaluation setups have been proposed for event-event co-occurrence models beyond the Narrative Cloze (which was proposed in (Chambers and Jurafsky, 2008) and used generally in this thesis).

Roemmele et al. (2011) describe the Choice of Plausible Alternatives commonsense evaluation corpus, which contains 500 development questions and 500 test questions consisting of a single sentence with two alternative causal inferences, one of which is judged by humans to be much more plausible. For example,

- Premise: The man broke his toe. *What was the cause of this?*
- Alternative 1: He got a hole in his sock.
- Alternative 2: He dropped a hammer on his foot.

- Premise: I tipped the bottle. *What happened as a result?*
- Alternative 1: The liquid in the bottle froze.
- Alternative 2: The liquid in the bottle poured out.

This corpus is not large enough to use as a training set.

Modi et al. (2016) describe the InScript corpus, which consists of 1,000 short stories comprising 200k tokens, collected via Amazon Mechanical Turk, instantiating 10 specific scenarios. These short stories are annotated with coreference information and event annotations describing which scenario-specific events and roles are realized by which tokens in stories. Modi et al. (2017) find that script-based features are empirically useful for the task of Discourse Referent prediction (determining which entity will be mentioned next, independent of the surface form realization) on this corpus.

Mostafazadeh et al. (2016) introduced a corpus of crowdsourced short stories with plausible “impostor” endings alongside the real endings; script systems can be evaluated on this corpus by their ability to discriminate the real ending from the impostor one. Roemmele et al. (2017) describe an RNN-based binary classification model to identify true last-sentences in the five-sentence stories of this dataset.

7.3 Modeling Collections of Events in Other Modalities

A recent body of work investigates the automatic induction of event structure in different modalities. Kim and Xing (2014) give a method of modeling sequences of images from ordered photo collections on the web, allowing them to perform, among other things, sequential image prediction. Huang et al. (2016) describe a new dataset of photos in temporal sequence scraped from web albums, along with crowdsourced story-like descriptions of the sequences (and methods for automatically generating the latter from the former). Bosselut et al. (2016) describe a system which learns a model of prototypical event co-occurrence from online photo albums with their natural language captions. Incorporating learned event co-occurrence structure from large-scale natural datasets of different modalities is an exciting line of future research.

Kim et al. (2016a) give a method of inducing a restaurant script from data from five modalities collected by wearable sensors on individuals in a dining setting. They evaluate on predicting states (e.g. "chatting before ordering," "drinking") from sensory input, and construct a probabilistic state-transition diagram (representing a simple probabilistic script) between the states based on the collected data.

Fast et al. (2016) describe a system which mines information about activities and their co-occurrences from a corpus of fiction for the end making of human-device-interaction more context-aware across devices. Their system is capable of making predictions about what activities smart device users are going to do next, given system predictions of users' current and recent activities. Notably, the authors elicit human judgments from Mechanical Turk about the distribution of events likely to involve various objects, and compare this distribution to the system's predicted actions, in order to measure possible systematic discrepancies between how events are represented in fiction and how humans judge them to occur in the real world.

Vondrick et al. (2016a) give a method of leveraging large corpora of unlabeled data to improve performance on the tasks of predicting labeled actions and objects in future video frames. They learn to predict future frames' latent representations; from predicted future latent frame representations, they can train a network to predict labels over future frames. Vondrick et al. (2016b) describe an adversarial model trained to directly predict the pixels of future frames in short videos.

Iyyer et al. (2017) introduce the COMICS dataset, consisting of 1.2M panels from approximately 4,000 comic books, with panel and textbox regions automatically segmented and text automatically OCR'd. They evaluate various systems on three types of cloze-style evaluation: conditioning on previous panels, they evaluate systems' ability to infer (a) the correct next panel image, without text; (b) the correct next panel text (with the panel's image); and (c) the correct correspondence between the next panel's textbox regions and the text that goes in them (to test models of character coherence). They find their various deep models to lag considerably behind human performance.

7.4 Modeling and Applying Discourse Structure

There is a body of research into models and applications of formal discourse analysis. The two most commonly-used paradigms for the automated analysis of formal discourse structure are Rhetorical Structure Theory (RST) (Mann and Thompson, 1988), represented by the RST treebank corpus (Carlson et al., 2001), and the Penn Discourse Treebank (PDTB) (Prasad et al., 2008). An RST parse gives a single tree structure over a document, with the leaf nodes representing elementary discourse units (minimally discursively meaningful spans of text), and the edges between nodes labeled with a closed set of relation types; see Mann and Thompson (1988) for more details. The PDTB gives annotations of discourse relations in text, consisting of pairs of spans of text in a document related to each other by a closed ontology of discourse relation types; a relation instance is either triggered by an explicit discourse connective (e.g. "because", "before"), or it is implicit, lacking an explicit discourse connective word. See Prasad et al. (2008) for more details. We give a brief summary of some prior work on computational models of formal discourse analysis, some of which uses one of these two frameworks, and some of which uses other sources of discourse information.

Kalchbrenner and Blunsom (2013b), propose using a Recurrent Neural Net to model sentences in discourse, which, at each timestep, takes a vector representation of a sentence calculated using a convolutional neural net. That is, they have an intra-sentential model of composition (which uses convolution to derive a single vector per sentence), and the output of this intra-sentential model is fed to an inter-sentential model, which is trained to classify speech acts ("statement," "backchannel," "opinion," and so on) in a dialogue corpus. They achieve superior results to the best prior model, which is an HMM-based model (that looks forward into the future, in addition to only conditioning on past elements, which is what their RNN-based model does). The intra-sentential composition operation, which is input a fixed-length vector for each word, iteratively performs convolutions on a per-dimension basis until arriving at a single fixed-length vector for the sentence.

Jansen et al. (2014) present evidence that discourse information is useful for question-answering when the questions are more complicated than simple factoid-type questions. They provide two ways of integrating discourse information into a reranking system for answers to questions. First, they present a shallow method which incorporates a closed list of 75 high-frequency discourse markers (e.g. "because," "after,"); second, they use automatic RST parses. They find that incorporating both types of discourse features is beneficial to a number of question-answering tasks.

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. Liu et al. (2016b) use a Convolutional Neural Network for implicit discourse relation classification.

Bhatia et al. (2015) demonstrate that automatically inferred RST parses are empirically useful for sentiment classification, despite the fact that the system they use has a fairly low accuracy score (around 60%). They use RST information in two general ways to help sentiment classification. First, they downweight the importance of words further down in the tree, under the assumption that text closer to the RST tree's root node is more discursively important. Second, they propagate information up the structure of the RST tree, calculating sentiment for the elementary discourse units at the leaf nodes, and using a recursive composition operation to calculate sentiment at non-terminal nodes.

Chapter 8

Future Work

We now describe a number of possible extensions to the work presented above.

8.1 Improvements to Models

Convolutional Models: Recently, Convolutional Neural Networks (CNNs) have been shown to be useful for many NLP tasks, proving competitive with RNN-based methods for many NLP tasks (Kalchbrenner and Blunsom, 2013b; Zhang et al., 2015; Kim et al., 2016b; Oord et al., 2016; Gehring et al., 2017). CNNs were generally popularized in the Computer Vision literature (LeCun et al., 1989; Krizhevsky et al., 2012), where networks take the form of compositions of two-dimensional convolution operations with intermediate nonlinear pooling operations; when applied to NLP problems, architectures typically use one-dimensional convolutions instead. Intuitively, the lower-level 1D text convolutions are analogous to classical n-grams, but operating over word embeddings rather than raw one-hot tokens; higher-level embeddings capture ordered co-occurrence information over (possibly overlapping) n-grams. These architectures could be fruitfully applied to discovering document-level information about paradigmatic co-occurrences of predicate-argument structures.

Encouraging Diversity of Inferences: If an event co-occurrence system's desired end task is to infer a set or sequence of implicit predicates with arguments, it may be desirable to enforce diversity of inferences; if a system's top inferences are all roughly synonymous, for example, this may not be ideal. A simple mutual information-based measure, such as the one used to learn pairwise event mutual exclusion constraints in Li et al. (2013), may be beneficial for encouraging diversity of inferences. Incorporating submodular objectives (Tschiatschek et al., 2017; Dolhansky and Bilmes, 2016) or using Determinental Point Processes (Kulesza and Taskar, 2012) may also be fruitful approaches to encourage inference diversity.

Better Bidirectional Sequence Models: In Chapter 4 we investigated one-directional sequence models, and in Chapter 5 we investigated bidirectional sequence-to-sequence models, which relied on the hierarchical nature of text sequences (raw text comprises sequences of sentences, each of which is a sequence of tokens). Bidirectional RNNs are widely used for sequence-labeling tasks in NLP, but the best way to incorporate information about the future (in addition to information about the past) in a sequence model (which iteratively predicts the next timestep's input), in settings where that is a meaningful idea, is an open question. Such models could be useful for inferring implicit events in documents.

Incorporating Additional Signals into Inferences: Additional sources of information beyond sequences of the types of events under consideration may be informative for making more high-quality inferences. One straightforward but possibly effective means of doing this is by incorporating a high-quality language model trained on a very large corpus (Venugopalan et al., 2017). One could also conceivably incorporate auxiliary information from other modalities—for example, incorporating statistical models of events as identified in a large video corpus, or a different model of the sort described in Section 7.3—into a text-based event system.

Differentiable Memory: Recently, there has been a great interest in neural models with various types of differentiable memory, which is amenable to SGD-based training (Weston et al., 2015; Sukhbaatar et al., 2015; Dyer et al., 2015; Grefenstette et al., 2015). These models are intuitively appealing for the task of inferring events from documents, which may require evidence from arbitrarily far back in a document.

Handling Proper Nouns and Rare Words: There are a number of recent neural sequence-to-sequence models which, when outputting sequences, can output either a word from the vocabulary (as in the typical sequence-to-sequence setup) or output one of the words from the input sequence verbatim (Vinyals et al., 2015a; Gu et al., 2016; Merity et al., 2017). This is useful for inferring outputs involving proper names or rare words present in the input, which may be either out-of-vocabulary or too rare to have reliably informative parameters. These models could be obviously useful for systems outputting events involving the entities in a document.

Conditioning on More Events: In Section 5.3.3, we presented evidence that, as the amount of raw text a text-prediction system is conditioned on increases, performance on the task of predicting held-out text does not necessarily increase. The extent to which this is true at the level of structured events, which are shorter and therefore less difficult to backpropagate error gradients through, was not investigated.

Expanding the Structure of Formal Events: In the regime of modeling formal syntax-mediated events, possibly for some extrinsic end-task, the verbs-with-nominal-arguments setup used in a number of the chapters of this thesis still fails to capture much important structure. A number of possible ways of expanding and improving these event structures are:

- **Nominal Events**: Many nouns introduce states or events (e.g. "carnival," "assassination"), and would be informative for a system fruitfully using event structure. A system which can reliably differentiate such nouns from decidedly non-eventful nouns ("car") may be useful to a script system.
- **Stative-Eventive Distinction**: Formal semanticists frequently differentiate between stative verbs (e.g. "know") describing conditions obtaining over an interval of time, and eventive verbs (e.g. "open") describing events taking place at a certain time. Differentiating between stative and eventive verbs may be useful for a computational script system. Note that different instances of a verb type may differ along this axis (compare "he squinted" with "he squinted constantly because of his vision")—automatically identifying stative or eventive tokens is a relatively understudied problem.
- **Adding Modals and Modifiers**: The structured events considered in this dissertation do not include many modifiers to verbs. Consider the example:

  King Frederick William I nearly executed his son for desertion.

  It is clear that the adverbial modifier *nearly* is very useful for modeling the text formally using events. Fruitfully incorporating such modifiers (in addition to modals like "might" or "must") could benefit script systems.

- **Tense and Aspect**: As the stative-eventive distinction mentioned above changes the calculus of what is described in a text as happening when, so do tense and aspect, which are very well-described in the linguistics literature. In the formal-event systems used in this dissertation, “he did eat” and “he will eat” are treated identically; script systems may benefit from incorporating such information.

Connections to High-Precision Crowdsourced Scripts: Empirical connections between the high-recall models learned from large corpora, of the sort considered in this thesis, with the high-precision situation-specific scripts derived from crowdsourced data (Regneri et al., 2010; Li et al., 2013) are currently not well-explored. These two general paradigms could be empirically useful to each other.

Incorporating Temporal Classifiers: The ordered models of predicates and arguments presented above model sequences of predicates as they occur in documents, which is not necessarily the same as modeling them in the temporal order in which they occur in the world. Systems which automatically classify temporal relationships between events, for example those trained on the TIMEBANK corpus (Pustejovsky et al., 2003), may be useful for improving and understanding event co-occurrence models.

Incorporating Formal Discourse Structure Information: There are a number of formalizations of discourse structure amenable to automatic analysis (e.g. RST (Mann and Thompson, 1988) and the PDTB (Prasad et al., 2008), described in Section 7.4). Statistical script systems and systems modeling these other formal discourse structures could be mutually beneficial.

Hierarchical RNN Models: There are multiple ways of constructing hierarchical RNN sequence models in problems with natural multilevel structure (Li et al., 2015; Sordoni et al., 2015). When modeling, e.g., a sequence of sequences, one can compose RNN models at both the lowest level and at the higher level, inputting the latent values from the former into the latter. Such approaches ease the difficulties in backpropagation through long time sequences. As event sequences are naturally modeled as sequences of sequences, such models may be usefully applied to models of events in sequence.

Structured Inference and Ensembling: The task of predicting structured events is a structured inference task. The neural methods presented above decompose the structured inference into a sequence and decompose the error into the sum of per-step errors; however, there are other ways to perform structured inference (Taskar et al., 2005; Weiss et al., 2015). These methods may be fruitful for improving prediction of structured events. Multiple different structured prediction models can be ensembled for improved performance (Henderson and Brill, 2000; Cortes et al., 2014).

Other Ways of Incorporating Scripts into Coreference Systems: In Chapter 6, we investigated a limited number of ways of constructing features for coreference systems from pre-learned event co-occurrence models. There are many more conceivable ways of doing so, for example, by directly modeling the relationship between two potential coreferents' events. Ideally, script models could be jointly trained with coreference models, perhaps in a semi-supervised setup, to enable learning discourse-level features which are informative for the end-task of coreference resolution.

8.2 New Corpora and Evaluations

Inference-Centered Question-Answering Corpora: There are a number of recent large-scale question-answering corpora that are large enough to be used to train large neural models (Rajpurkar et al., 2016; Hermann et al., 2015). In these corpora, answers to questions typically have direct textual evidence in predetermined relevant passages, and do not require reasoning about implicit events that are not explicitly stated in text. A large crowdsourced corpus of passages paired with questions that require inferences about implicit events (similar to Mostafazadeh et al. (2016) but pivoting around the task of question-answering) could progress the state of the art of answering questions about information not explicitly present in text, and would serve as a useful extrinsic evaluation for event co-occurrence models.

One possible way to construct such a corpus semi-automatically would be to transform a cloze-style evaluation, as implemented above, into a question-answering evaluation, by syntactically transforming held-out declarative sentences into queries (e.g. "Jones died" could be transformed into "who died?"). This is conceptually similar to the cloze-style evaluation corpus of Hermann et al. (2015), who turn human-generated summaries into queries; transforming held-out document sentences into queries would produce an inference-focused, rather than summary-focused, question-answering corpus. Since many such constructed queries would be unanswerable from the text, a crowdsourcing phase could be used to filter out unanswerable queries, similar to the construction of the LAMBADA corpus (Paperno et al., 2016).

Other Automatic Training and Evaluation Objectives: There is a vigorous debate in the dialog community about the appropriateness of BLEU and other automatic and unsupervised evaluation metrics for judging dialog system quality (Liu et al., 2016a). As human judgments are relatively expensive to collect and not realistic as a direct training objective, it would be beneficial to have clearly meaningful training and evaluation objectives that correlate convincingly with human judgments or some other extrinsically meaningful metric (a number of which are enumerated below).

Comparing to Human Performance on Inference: We presented human evaluations of script inference systems, wherein human annotators are asked to judge the quality of automatic inferences. We have not, however, gauged human performance on the task of making inferences directly. The human ratings of human-generated inferences would provide a meaningful performance ceiling.

Investigating Genre Effects: Different genres of text will yield different behavior with respect to co-occurring predicate-argument structures. For example, some genres of documents will tend to be more organized in temporal order than others, some genres will tend to be more focused on single individuals than others, and so on. Simonson and Davis (2015, 2016) give a number of results quantifying the relationships between narrative schemas automatically extracted from newswire and the categories and topics of the documents from which they were extracted. Further quantifying the differences between genres and the effects these differences have on NLP systems, including event co-occurrence models, may be a useful direction for future work.

Fine-Grained Analysis of Human Evaluations: We presented multiple human evaluations, wherein annotators were asked to rate the quality of inferences from text. The extent to which different properties of the inferences—grammaticality, prior probability effects, and so on—affect ratings could be elucidated with further focused crowdsourced evaluations.

Fine-Grained Analysis of Automatic Evaluations: There are presumably many different phenomena contributing to systems' behavior under the narrative cloze evaluations. For example, different systems are biased toward the unigram distribution to different extents and in different circumstances. Identifying and classifying the conditions in which systems predict common events, and those in which they predict more rare events, for example, could yield insights into the patterns learned by different script systems.

8.3 Possible Extrinsic Applications

Summarization: Models of events in sequence as they occur in documents could be fruitful for improving automatic summarization systems. Barzilay et al. (2002) give evidence that the relative chronological ordering between events is empirically useful for multi-document news summarization; presumably, more general script information could be similarly useful in similar tasks.

Zero-Anaphora Resolution: In so-called *pro-drop* languages, anaphoric pronouns may be omitted if the antecedent is clear from context (Mitkov, 2002). The task of resolving these non-realized gap pronouns, called *zero anaphora* or *null anaphora* resolution, is more challenging than anaphora resolution in a language with obligatory surface realization of pronouns (Iida et al., 2006; Chen and Ng, 2013). Event co-occurrence knowledge could be useful for the related tasks of identifying zero anaphors and resolving them.

Syntactic Parsing: Prepositional Phrase attachment ambiguity haunts syntactic parsers. Consider the following sentences:

- After taking the witness stand, she read aloud her testimony to the court.
- After taking the witness stand, she read aloud her letter to the shareholders.

In the first example, *to the court* modifies *read*, whereas in the second, *to the shareholders* almost certainly modifies *letter*. Knowing that, after taking the witness stand, people frequently read things to the court would help resolve this ambiguity. The extent to which such world knowledge can be empirically useful to syntactic parsers, and the methods of incorporating such knowledge, are open questions.

Winograd Schema Challenge: As described in Section 6.4, there are a number of small Winograd Schema Challenge evaluation corpora (Levesque et al., 2011), which are collections of hard coreference problems requiring world knowledge to resolve. Peng et al. (2015b) augment a general-purpose coreference resolution solver with features targeted to WSC problems. The augmented coreference systems we presented in Chapter 6 have not been evaluated on these corpora.

Computational Creativity: There have been a number of efforts applying similar models to those presented above to either aid humans in creative endeavors or to automatically generate text recognizable as creative in some way (McIntyre and Lapata, 2009; Roemmele, 2016; Khalifa et al., 2017; Martin et al., 2017). As models of co-occurring events improve, they will be increasingly useful to efforts in computational creativity.

Relation Extraction: A good deal of investigation has been done into extracting formal structured relations between entities from unstructured text (Culotta and Sorensen, 2004; Riedel et al., 2013); the process can in some cases be aided by mining databases of facts (Nahm and Mooney, 2000). Script inferences could, in principle, be used to improve recall of information extraction systems in a similar way: if a relation is not explicitly expressed in text, but is very probable according to an event co-occurrence model, based on the text of the document, this may be an informative feature to a relation extraction system.

Other Applications of Bidirectional Sequence-to-Sequence Models: To our knowledge, the bidirectional sequence-to-sequence framework used in Chapter 5 is not widely explored. It is, in principle, applicable to any problem with two layers of sequential semantics: if something can be modeled as a sequence of sequences, then, when decoding an individual sequence, encoded prior and posterior sequences may be useful. This could conceivably be applied to tasks such as optical character recognition error correction, imputation of missing data, or building predictive autoregressive models of cyclical time-series data.

Chapter 9

Conclusions

To summarize briefly, the main contributions of this thesis are as follows:

1. We describe a method of incorporating multiple arguments into predicate-argument structures used for a model of event co-occurrence (Chapter 3). We find that this allows for more predictive models, particularly in the sense that predicates and single arguments are better predicted by the multi-argument model.
2. We describe a method of modeling sequences of predicates with arguments using Recurrent Neural Networks (Chapter 4) which, through the lexical generalizability allowed by low-dimensional embeddings, are capable of modeling a large vocabulary of noun information about arguments. These models provide superior predictive performance of predicate-argument sequences.
3. We investigate the use of raw-text-level RNN-based encoder-decoder models for the task of modeling sequences of more structured events (Chapter 5). We find such models, which do not require language-specific parsers, to perform comparably to the direct models of sequences of predicate-argument structures, and to be better at producing text describing events.
4. We investigate the use of features from RNN-based encoder-decoder models for the task of noun coreference identification (Chapter 6). We find some of these features to be marginally beneficial for top-level precision (at the

expense of recall), and beneficial for a restricted class of very difficult coreference problems.

We have investigated a number of ways of using neural nets to infer implicit predicate-argument structures from documents, modeling them directly or implicitly in modeled raw text, and we have presented a preliminary evaluation of such models' utility for the task of coreference resolution. Hopefully, such models can be improved in the future, and will prove useful for other downstream tasks. The connections between models of events as expressed in text and models of events as they occur in the world are still largely unexplored, as are applications of such models to many extrinsic end-tasks. We look forward to continuing some of the many exciting threads of future work in the area.

References

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

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 2014)*, October 2014.

Simon Ahrendt and Vera Demberg. Improving event prediction by representing script participants. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-16)*, pages 546–551, San Diego, California, 2016.

Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources & Evaluation (LREC-98)*, pages 563–566, 1998.

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.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING–ACL 98)*, pages 86–90, 1998.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. Relgrams: 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, 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-2013)*, 2013.

David Bamman and Noah Smith. Unsupervised discovery of biographical structure from text. *Transactions of the Association for Computational Linguistics (TACL)*, 2:363–376, 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)*, pages 352–361, 2013.

Mohit Bansal and Dan Klein. Coreference semantics from web features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL-12)*, pages 389–398, Jeju Island, Korea, July 2012.

Frederic C. Bartlett. *Remembering: A Study in Experimental and Social Psychology*. Cambridge University Press, 1932.

Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55, 2002.

David Bean and Ellen Riloff. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-04)*, pages 297–304, Boston, Massachusetts, USA, 2004.

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

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, 2014.

Shane Bergsma and Dekang Lin. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 33–40, Sydney, Australia, 2006.

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*. 2009.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. 2006.

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

Antoine Bosselut, Jianfu Chen, David Warren, Hannaneh Hajishirzi, and Yejin Choi. Learning prototypical event structure from photo albums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, 2016.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*, 2015.

Thorsten Brants and Alex Franz. The Google Web 1T 5-gram Corpus Version 1.1, 2006. LDC2006T13.

William F. Brewer. Bartlett's concept of the schema and its impact on theories of knowledge representation in contemporary cognitive psychology. In *Bartlett, Culture and Cognition*. Psychology Press, 2000.

Aljoscha Burchardt, Anette Frank, and Manfred Pinkal. Building text meaning representations from contextually related frames–a case study. *Proceedings of the Sixth International Workshop on Computational Semantics (IWCS-05)*, 2005.

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*, pages 1–10, 2001.

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)*, pages 789–797, 2008.

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)*, pages 602–610, 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: Human Language Technologies (ACL-HLT-11)*, pages 976–986, 2011.

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-2013)*, 2013.

Nathanael Chambers. Behind the scenes of an evolving event cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem-17)*, 2017.

Eugene Charniak and Robert P. Goldman. Plan recognition in stories and in life. In *Proceedings of the 5th Conference on Uncertainty in Artificial Intelligence (UAI-89)*, pages 343–352, 1989.

Chen Chen and Vincent Ng. Chinese zero pronoun resolution: Some recent advances. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-2013)*, pages 1360–1365, Seattle, Washington, USA, 2013.

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: Human Language Technologies (NAACL-13)*, 2013.

Timothy Chklovski and Patrick Pantel. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 33–40, 2004.

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.

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886, 2015.

Do Kook Choe and Eugene Charniak. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-2016)*, pages 2331–2336, Austin, Texas, 2016.

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

Kevin Clark and Christopher D. Manning. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-2016)*, pages 2256–2262, Austin, Texas, 2016.

Kevin Clark and Christopher D. Manning. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, pages 643–653, Berlin, Germany, 2016.

Herbert H Clark. Bridging. In *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, pages 169–174, 1975.

Corinna Cortes, Vitaly Kuznetsov, and Mehryar Mohri. Ensemble methods for structured prediction. In *Proceedings of the 31st International Conference on Machine Learning (ICML-2014)*, pages 1134–1142, 2014.

Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 423–429, Barcelona, Spain, 2004.

Pradeep Dasigi and Eduard Hovy. Modeling newswire events using neural networks for anomaly detection. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING-14)*, pages 1414–1422, Dublin, Ireland, 2014.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources & Evaluation (LREC-06)*, volume 6, pages 449–454, 2006.

Scott C. Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

Pascal Denis and Jason Baldridge. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pages 660–669, Honolulu, Hawaii, 2008.

Brian W Dolhansky and Jeff A Bilmes. Deep submodular functions: Definitions and learning. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS-16)*, pages 3404–3412. 2016.

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.

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

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-2013)*, 2013.

Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. In *Transactions of the Association for Computational Linguistics*, volume 2, pages 477–490, 2014.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP-15)*, pages 334–343, Beijing, China, 2015.

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

Katrin Erk. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 216–223, Prague, Czech Republic, 2007.

Ethan Fast, William McGrath, Pranav Rajpurkar, and Michael S Bernstein. Augur: Mining human behaviors from fiction to power interactive systems. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 237–247, 2016.

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

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

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)*, pages 49–57, 2014.

Toshiaki Fujiki, Hidetsugu Nanba, and Manabu Okumura. Automatic acquisition of script knowledge from a text collection. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-03)*, pages 91–94, 2003.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

Matthew Gerber and Joyce Y. Chai. Beyond NomBank: A study of implicit arguments for nominal predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, 2010.

Kevin Gimpel and Noah A. Smith. Softmax-margin crfs: Training log-linear models with cost functions. In *Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-10)*, pages 733–736, Los Angeles, California, 2010.

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

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

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*, pages 1828–1836, 2015.

H. Paul Grice. Logic and conversation. In Peter Cole and Jerry Morgan, editors, *Syntax and Semantics 3: Speech Acts*, pages 41–58. Academic Press, 1975.

Ralph Grishman and Beth Sundheim. Message Understanding Conference-6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 466–471, 1996.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*, 2014.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, pages 1631–1640, Berlin, Germany, 2016.

Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir. Reasoning about robocup soccer narratives. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI-11)*, 2011.

Sanda M Harabagiu, Marius A Paşca, and Steven J Maiorano. Experiments with open-domain textual question answering. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*, pages 292–298, 2000.

Henry Head. *Studies in Neurology*. Oxford University Press, 1920.

John C. Henderson and Eric Brill. Bagging and boosting a treebank parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-00)*, pages 34–41, 2000.

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 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*, 2015.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Jerry R Hobbs. Why is discourse coherent? Technical Report 176, SRI International, 1978.

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

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A field guide to dynamical recurrent neural networks. IEEE Press*. IEEE Press, 2001.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. OntoNotes: the 90% solution. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics, Short Paper (NAACL-HLT-06 Short)*, pages 57–60, 2006.

Ting-Hao K. Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Aishwarya Agrawal, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, Larry Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. Visual storytelling. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-16)*, 2016.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 625–632, Sydney, Australia, 2006.

Joseph Irwin, Mamoru Komachi, and Yuji Matsumoto. Narrative schema as world knowledge for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-11 Shared Task)*, pages 86–92, 2011.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-16)*, pages 1534–1544, 2016.

Mohit Iyyer, Varun Manjunatha, Anupam Guha, Yogarshi Vyas, Jordan Boyd-Graber, Hal Daumé III, and Larry Davis. The amazing mysteries of the gutter: Drawing inferences between panels in comic book narratives. In *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR-17)*, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *Proceedings of the 5th International Conference on Learning Representations (ICLR-17)*, 2017.

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)*, pages 336–344, 2012.

Peter Jansen, Mihai Surdeanu, and Peter Clark. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14)*, pages 977–986, 2014.

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 (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland, June 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)*, 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, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

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

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*, pages 119–126, August 2013.

Ronald M Kaplan and Joan Bresnan. Lexical-Functional Grammar: A formal system for grammatical representation. *Formal Issues in Lexical-Functional Grammar*, pages 29–130, 1982.

Lauri Karttunen. Pronouns and variables. In *Papers from the Fifth Regional Meeting of the Chicago Linguistic Society*, 1969.

Niels Kasch and Tim Oates. Mining script-like structures from the web. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 34–42, 2010.

Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-04)*, pages 289–296, Boston, Massachusetts, USA, 2004.

Ahmed Khalifa, Gabriella AB Barros, and Julian Togelius. DeepTingle. *Proceedings of the Eighth International Conference on Computational Creativity (ICCC-17)*, 2017.

Emre Kıcıman. Towards learning a knowledge base of actions from experiential microblogs. In *AAAI Spring Symposium on Knowledge Representation and Reasoning*, 2015.

Gunhee Kim and Eric P. Xing. Reconstructing storyline graphs for image recommendation from web community photos. In *Proceedings of the 2014 27th Conference on Computer Vision and Pattern Recognition (CVPR-14)*, 2014.

Eun-Sol Kim, Kyoung-Woon On, and Byoung-Tak Zhang. DeepSchema: Automatic schema acquisition from wearable sensor data in restaurant situations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, 2016.

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.

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

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-12)*, pages 1097–1105, 2012.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286, 2012.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-2014)*, 2014.

Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4), 2013.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. The Winograd Schema Challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 2011.

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 2014)*, October 2014.

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

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. Story generation with crowdsourced plot graphs. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI-13)*, pages 598–604, 2013.

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)*, pages 2061–2069, October 2014.

Jiwei Li, Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP-15)*, pages 1106–1115, Beijing, China, 2015.

Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001.

Hugo Liu and Push Singh. ConceptNet – a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-2016)*, pages 2122–2132, Austin, Texas, 2016.

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

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 135–142, Barcelona, Spain, 2004.

Xiaoqiang Luo. On coreference resolution performance metrics. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP-05)*, pages 25–32, 2005.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the 5th International Conference on Learning Representations (ICLR-17)*, 2017.

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 *Prodeedings of the 21st International Florida Artificial Intelligence Research Society Conference (FLAIRS-2008)*, pages 159–164, 2008.

Lara J Martin, Prithviraj Ammanabrolu, William Hancock, Shruti Singh, Brent Harrison, and Mark O Riedl. Event representations for automated story generation with deep neural nets. *arXiv preprint arXiv:1706.01331*, 2017.

Sebastian Martschat and Michael Strube. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418, 2015.

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)*, pages 217–225, 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)*, pages 1562–1572, 2010.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *Proceedings of the 5th International Conference on Learning Representations (ICLR-17)*, 2017.

Adam Meyers. Annotation guidelines for NomBank - noun argument structure for PropBank. Technical report, New York University, 2007.

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

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. pages 3111–3119, 2013.

Marvin Minsky. A framework for representing knowledge. Technical Report 306, MIT-AI Laboratory, 1974.

Ruslan Mitkov. *Anaphora Resolution*. Longman, 2002.

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)*, pages 641–648, 2007.

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

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. InScript: Narrative texts annotated with script information. In *Proceedings of the Tenth*

International Conference on Language Resources and Evaluation (LREC-16), 2016.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. Modelling semantic expectation: Using script knowledge for referent prediction. *Transactions of the Association for Computational Linguistics*, 5:31–44, 2017.

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.

Nafise Sadat Moosavi and Michael Strube. Which coreference evaluation metric do you trust? A proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, pages 632–642, Berlin, Germany, 2016.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-16)*, pages 839–849, 2016.

Erik T. Mueller. Understanding script-based stories using commonsense reasoning. *Cognitive Systems Research*, 5:307–340, 2004.

Un Yong Nahm and Raymond J. Mooney. A mutually beneficial integration of data mining and information extraction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, pages 627–632, Austin, TX, 2000.

Vincent Ng. Machine learning for entity coreference resolution: A retrospective look at two decades of research. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 4877–4884, 2017.

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.

Diarmuid Ó Séaghdha. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 435–444, Uppsala, Sweden, 2010.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

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 AAAI 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):71–106, 2005.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, 2016.

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

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

Charles Sanders Peirce. *Collected Papers of Charles Sanders Peirce*. Harvard University Press, 1931–1958.

Haoruo Peng and Dan Roth. Two discourse driven language models for semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, 2016.

Haoruo Peng, Kai-Wei Chang, and Dan Roth. A joint framework for coreference resolution and mention head detection. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning (CoNLL-2015)*, pages 12–21, Beijing, China, 2015.

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 2014)*, pages 220–229, 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.

Karl Pichotta and Raymond J. Mooney. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL-16)*, Berlin, Germany, 2016.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 143–150, 2004.

Carl Pollard and Ivan A Sag. *Head-driven phrase structure grammar*. University of Chicago Press, 1994.

Simone Paolo Ponzetto and Michael Strube. Semantic role labeling for coreference resolution. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pages 143–146, 2006.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, 2012.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-14 Short)*, pages 30–35, 2014.

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.

Ellen F. Prince. Towards a taxonomy of given-new information. In P. Cole, editor, *Radical Pragmatics*, pages 223–255. Academic Press, 1981.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. The TIMEBANK corpus. In *Corpus Linguistics*, volume 2003, pages 647–656, 2003.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate 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)*, pages 492–501, Cambridge, MA, 2010.

Altaf Rahman and Vincent Ng. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 968–977, 2009.

Altaf Rahman and Vincent Ng. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT-11)*, pages 814–824, 2011.

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 and Computational Natural Language Learning (EMNLP-CoNLL-12)*, pages 777–789, 2012.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-2016)*, pages 2383–2392, Austin, Texas, 2016.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. The life and death of discourse entities: Identifying singleton mentions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-13)*, pages 627–633, Atlanta, Georgia, 2013.

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)*, Uppsala, Sweden, July 2010.

Michaela Regneri, Alexander Koller, Josef Ruppenhofer, and Manfred Pinkal. Learning script participants from unlabeled data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011 (RANLP-11)*, pages 463–470, Hissar, Bulgaria, 2011.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-13)*, pages 74–84, Atlanta, Georgia, 2013.

Alan Ritter, Mausam, and Oren Etzioni. A Latent Dirichlet Allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 424–434, Uppsala, Sweden, 2010.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, 2011.

Melissa Roemmele, Sosuke Kobayashi, Naoya Inoue, and Andrew Gordon. An RNN-based binary classifier for the story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem-17)*, pages 74–80, 2017.

Melissa Roemmele. Writing stories with help from recurrent neural networks. In *Doctoral Corsortium at the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, 2016.

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*, 2015.

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)*, 2015.

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

David E Rumelhart. Schemata: The building blocks of cognition. In *Theoretical Issues in Reading Comprehension*. Lawrence Erlbaum, 1978.

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

Roger C. Schank. *Conceptual Information Processing*. North-Holland, Oxford, 1975.

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

Dan Simonson and Anthony R. Davis. Interactions between narrative schemas and document categories. In *Proceedings of the 1st Workshop on Computing News Storylines (CnewS)*, 2015.

Dan Simonson and Anthony Davis. NASTEA: Investigating narrative schemas through annotated entities. In *Proceedings of the 2nd Workshop on Computing News Storylines (CnewS)*, 2016.

Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. Open mind common sense: Knowledge acquisition from the general public. In *On the move to meaningful internet systems 2002: CoopIS, DOA, and ODBASE*, pages 1223–1237, 2002.

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

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544, 2001.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM-15)*, pages 553–562, 2015.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*, pages 2440–2448, 2015.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-2013)*, pages 1139–1147, 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)*, pages 3104–3112, 2014.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning (ICML-05)*, pages 896–903, 2005.

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

Ottokar Tilk, Vera Demberg, Asad Sayeed, Dietrich Klakow, and Stefan Thater. Event participant modelling with neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-2016)*, 2016.

Sebastian Tschiatschek, Adish Singla, and Andreas Krause. Selecting sequences of items via submodular maximization. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

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)*, 2015.

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)*, 2015.

Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, Raymond Mooney, Trevor Darrell, and Kate Saenko. Captioning images with diverse objects. In *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR-17)*, 2017.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Conference on Message Understanding (MUC-6)*, pages 45–52, 1995.

Oriol Vinyals and Quoc Le. A neural conversational model. 2015.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*, pages 2692–2700, 2015.

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

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR-16)*, pages 98–106, 2016.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS-16)*, pages 613–621, 2016.

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

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP-15)*, pages 323–333, Beijing, China, 2015.

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

Terry Winograd. Understanding natural language. *Cognitive Psychology*, 3(1):1–191, 1972.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP-15)*, pages 1416–1426, Beijing, China, 2015.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. Learning global features for coreference resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-16)*, pages 994–1004, San Diego, California, 2016.

Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics (ACL-94)*, pages 133–138, 1994.

Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*, pages 649–657, 2015.

Vita

Karl Pichotta was born in Hoffman Estates, IL, a northwest suburb of Chicago, in 1985, and was raised in nearby Inverness, IL. He graduated from William Fremd High School in Palatine, IL in 2004, and from Stanford University in 2008 with a BS in Symbolic Systems. He enrolled in the Computer Science PhD program at UT Austin in 2010, earning an MS in CS in 2013, and spent two summers at Google during his PhD studies.

Permanent Address: `karlpichotta@gmail.com`

This dissertation was typeset with LaTeX 2$_\varepsilon$[1] by the author.

[1] LaTeX 2$_\varepsilon$ is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.