Modeling Common Sense Knowledge via Scripts

UNIVERSITÄT
DES
SAARLANDES

Ashutosh Modi

A dissertation submitted towards the degree

Doctor of Engineering

of the Faculty of Mathematics and Computer Science

of Saarland University

Saarbrücken, July 2017

Date of the colloquium:	7 July, 2017
Dean:	Prof. Dr. Frank-Olaf Schreyer
Chairman of the examination board:	Prof. Dr. Holger Hermanns
Reporters:	Dr. Ivan Titov
	Prof. Dr. Manfred Pinkal
	Prof. Dr. Alexander Koller
Scientific Assistance:	Dr. Michael Roth

To my Guru Pt. Shriram Sharma Acharya,

my Dad and my Mom

without whom this would have not been possible.

Acknowledgements

This dissertation would not have been possible without the guidance of Prof. Dr. Manfred Pinkal and Dr. Ivan Titov, from whom I have learned a lot and whose teachings are going to be useful for me throughout life.

I am grateful to my wife Nupur, who was patient enough and provided me moral and emotional support throughout my dissertation. I am also grateful to my father and my brother for motivating me. Above all I am grateful to my mother, you would always be remembered. I thank all the members of my family for being there with me.

A very special thanks to MMCI (MultiModel Cluster Initiative) and SFB-Ideal for supporting me during the research.

I am thankful to my colleagues Alexandre Klementiev, Mikhail Kozhevnikov, Fatemeh Torabi Asr, Asad Sayeed, Annemarie Friedrich, Andrea Horbach, Lilian Wanzare and Simon Ostermann. It was wonderful to have discussions with you all.

I express my gratitude towards my friends in Saarbrücken who were there with me during the tough times.

And finally, to everyone in the Computational Linguistics department who helped me and with whom I exchanged wonderful ideas.

Thank you everyone, for the support and motivation!

Abstract

It is generally believed that incorporation of common sense knowledge about the world would benefit natural language understanding systems. This dissertation considers modeling common sense knowledge associated with everyday activities. The common sense knowledge about everyday activities can be understood and modeled using the concept of scripts. Scripts are sequences of events describing stereotypical human activities (i.e. scenarios), for example, cooking pasta, baking a cake, etc. Scripts have been shown to be useful for drawing inferences from text. Consequently, scripts find applications in the area of natural language processing (NLP) and artificial intelligence (A.I.), for example, temporal reasoning, coreference resolution, story understanding and question answering systems.

To model script knowledge, three main tasks need to be addressed: event ordering, event paraphrasing and event prediction. This dissertation introduces a novel probabilistic neural network based model for learning event ordering and event paraphrasing from a crowdsourced scripts corpus and the Gigaword news corpus. The model outperforms currently prevalent graph based and count based methods. In addition to event ordering and paraphrasing, a script understanding system should have the ability to predict an upcoming event in a script. We propose an extension of the event ordering model for event prediction. Experiments on a movie summary corpus show the advantages of the event prediction model over existing count based methods.

Currently, most corpora describing narratives involve inter-mingling of scripts from many different scenarios. Such narratives containing the inter-play of different scenarios make it difficult to study the influence of script knowledge corresponding to a single scenario. In order to address this problem and analyze the contribution of script knowledge, this dissertation introduces the InScript corpus. InScript is a crowdsourced collection of stories, where each story is about only one particular script scenario. The corpus is manually annotated with script specific event types, participant types and coreference information.

Intuitively, script knowledge should influence language understanding. This dissertation contributes towards foundational work done for understanding the influence of script knowledge on language comprehension. We propose models for checking the influence of script knowledge on discourse referent prediction. From multiple experiments, we conclude that script knowledge makes a substantial contribution in predicting the upcoming discourse referents. Thus, we were able to empirically establish that script knowledge plays a significant role in language comprehension.

Kurzfassung

Im Allgemeinen wird angenommen, dass Systeme zur automatischen Sprachverarbeitung von der Benutzung von Weltwissen profitieren können. In dieser Dissertation wird Weltwissen in der Form alltäglicher Tätigkeiten modelliert. Weltwissen in dieser Form kann mit Hilfe von sogenannten Skripten verstanden und modelliert werden. Skripte sind Sequenzen von Ereignissen, die stereotypische menschliche Tätigkeiten ("Szenarien") beschreiben, z.B. das Kochen von Nudeln, das Backen eines Kuchens etc. Die Nützlichkeit von Skripten wurde vor allem bei Systemen der automatischen textuellen Inferenz gezeigt. Als Konsequenz finden Skripte Anwendung im Bereich von automatischer Verarbeitung natürlicher Sprache und in der künstlichen Intelligenz, z.B. beim zeitlichen Schließen, der Auflösung von Anaphern, dem automatischen Verstehen von Geschichten und in Systemen zur automatischen Beantwortung von Fragen.

Um Skriptwissen zu modellieren, müssen 3 zentrale Aufgaben betrachtet werden: Das zeitliche Ordnen von Ereignissen, das Paraphrasieren von Ereignissen und das Vorhersagen von Ereignissen. Diese Dissertation stellt ein neues Modell vor, das auf einem probabilistischen neuronalen Netz basiert und das die zeitliche Ordnung von Ereignissen und Paraphrasen eines Ereignisses aus einem auf Crowdsourcing basierten Skript-Corpus und aus dem Gigaword-News-Corpus lernt. Das Modell übertrifft aktuelle Graph-basierte und Kookkurrenz-basierte Modelle. Zusätzlich zum Ordnen von Ereignissen und der Paraphrasenerkennung solcher, sollte ein System, welches Skriptwissen modelliert, ein bevorstehendes Ereignis in einem Skript vorhersagen

können. Wir präsentieren eine Erweiterung des Modells zum Ordnen von Ereignissen für deren Vorhersage. Experimente auf einem Corpus von Filmzusammenfassungen zeigen die Vorteile unseres Modells zur Vorhersage von Ereignissen im Vergleich mit aktuell existierenden Methoden, die auf Vorkommenshäufigkeiten basieren.

Die Modellierung und das automatische Verstehen von narrativen Geschichten in aktuellen Korpora erfordern die Vermischung von Skripten unterschiedlicher Szenarien. Solche Narrative enthalten ein Zusammenspiel verschiedener Szenarios und erschweren es daher, den Einfluss von Skriptwissen basierend auf einem einzelnen Szenario zu erforschen. Um dieses Problem zu beheben und den Einfluss von Skriptwissen zu messen, wird in dieser Dissertation das InScript-Korpus vorgestellt. InScript ist eine auf Crowdsourcing basierte Sammlung von Geschichten, von denen jede von einem speziellen Skriptszenario handelt. Das Korpus wurde manuell mit skriptspezifischen Ereignistypen, Partizipantentypen und Koreferenzinformation annotiert.

Intuitiv sollte Skriptwissen auch das Sprachverstehen beeinflussen. Diese Dissertation leistet einen Beitrag zur grundlegenden Erforschung des Einflusses von Skriptwissen auf das Sprachverstehen. Wir stellen Modelle zur Überprüfung des Einflusses von Skriptwissen auf die Vorhersage von Diskursreferenten vor. Wir erschließen, basierend auf mehreren Experimenten, dass Skriptwissen einen substantiellen Einfluss auf die Vorhersage des textuell bevorstehenden Diskursreferenten hat. Auf Grundlage dessen sind wir also imstande, empirisch zu belegen, dass Skriptwissen eine signifikante Rolle für das Sprachverstehen spielt.

Table of contents

List of figures

List of tables

CHAPTER 1

Introduction

If you would be a real seeker after truth, it is necessary that at least once in your life you doubt, as far as possible, all things.

**

Rene Descartes

1.1 Introduction

Since time immemorial, humans have envisioned creating machines that can perform many mundane human chores. We have been successful (e.g. industrial robots, self-driving cars, etc.) to some extent but when it comes to emulating human intelligence, we still have a long way to go. With the goal to create intelligent machines[1], the field of *artificial intelligence* was introduced in 1956 by John McCarthy and colleagues at the famous Dartmouth workshop[2]. Ever since then, there have been concerted efforts to replicate human intelligence and behavior in the machines. Humans have been looking for intelligent machines that can assist them in carrying out activities on a day to day basis.

One of the important components of such an intelligent system is communication with the environment. Until now, one of the dominant ways of communicating with computers has been in the languages that computers understand, i.e. computer programs. But given the fact that humans are not born with the natural tendency to write computer programs, it will be desirable to have systems that can understand the *natural language* that humans speak/write. In contrast to computer programming, language[3] is something that humans learn early in their lives, without making too much effort. Language is a natural mode of communication for humans. Consequently, for communicating with machines effortlessly and seamlessly, it is desirable to have a natural language understanding interface for a machine.

Although language includes both the spoken component and written text, this dissertation's scope is limited to the written form of language, available in a variety

[1]The terms machines, computer systems and intelligent systems are used inter-changeably.

[2]Dartmouth workshop: http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html and http://www-formal.stanford.edu/jmc/slides/dartmouth/dartmouth/node1.html

[3]The term language is used to refer to natural language, unless the term is ambiguous, in which case it will be referred to as natural language.

of forms like documents, news articles, books, websites, blogs etc. Henceforth, in this dissertation, language processing implies processing the textual form of the language.

Developing natural language understanding systems is far from trivial. Languages come with their own complexity and inherent ambiguities. Language can have ambiguities at multiple levels. There is ambiguity at the lexical level (i.e. at the level of words), where the same word can have multiple meanings. For example, in sentence (1.1) does the word "bank" refer to "financial institution" or to "riverside"? There can be ambiguities at the level of syntax (i.e. at the level of grammar), where the same sentence can have different equally plausible syntactic structures of syntactic analysis (i.e. multiple, equally likely, syntactic parse trees). For example, in sentence (1.2), did the man see another man with the help of binoculars? Or does it indicate that the man saw another man carrying binoculars? There are ambiguities at the level of semantics (i.e. at the level of meaning), where the text is open to many possible interpretations. For example, in sentence (1.3) did Nupur leave the restaurant or was it the waitress who left?

(1.1) He went to the *bank* and saw a lot of people standing there.

(1.2) He saw a man with binoculars.

(1.3) Nupur paid the waitress and then she left the restaurant.

(1.4) John hit Bill

(1.5) Bill hit John

Ambiguities in language make communication more compact and efficient by discarding the redundant information. In a language, with a small set of words (e.g. words in a dictionary), it is possible to generate innumerable meaningful sentences, making communication compact. However, this limited vocabulary comes at the cost

of ambiguity, as a word can be overloaded and can have multiple meanings. The same set of words can be used to generate sentences having different meanings. For example, sentences (1.4) and (1.5) have the same set of words but have different meanings. Sentence (1.4) means that John acted on Bill but sentence (1.5) means the opposite, Bill acted on John.

Consequently, this compactness and efficacy of language make computational modeling of the language challenging. An important and dominant component of a language processing system deals with ambiguity resolution. One of the ways of resolving the ambiguities in a language is by taking into account the *linguistic context* of the word/phrase under consideration. Linguistic context (in this dissertation, simply referred to as context) refers to the neighboring words, sentences and other neighboring discourse related structures. For example, it becomes clear who has possession of the binoculars, if the sentence following (1.2) is "Then he zoomed in the binoculars to see the man clearly."

But many times the context is absent or does not convey enough information. In such cases, it is not possible to resolve the ambiguity on the basis of context alone. For example in the sentence (1.3), it is difficult to answer the question who left, if further context is not given. However, if we utilize *common sense knowledge* about the world, in particular, common sense knowledge about a typical restaurant scenario, then we know that, since Nupur paid the bill, she is most likely the customer and it is usually the customer (Nupur) who leaves the restaurant after paying the bill.

(1.6) I paid the bill and left.

While communicating, humans very often skip details of common daily activities (common sense knowledge), as it is taken for granted that the comprehender already knows them. For example, when you read the sentence (1.6), it implicitly implies that the speaker took out money from his wallet and paid it to the concerned person

Ashutosh started writing the article while sipping coffee. But he realized that coffee didn't have sugar. A few minutes ago, he had been to the kitchen. He had just started boiling water when Simon entered the kitchen. They both started talking about the new project that Ashutosh was pursuing. When the water had boiled, he put the coffee powder in the cup and then poured the boiling water into the cup. The coffee was ready, and then both Ashutosh and Simon came back to their office with coffee. Now, Ashutosh went back to the kitchen to put some sugar cubes in his cup.

Fig. 1.1 A small narrative with an implicit "making coffee" script.

and then left the place. As you can notice, the sentence omits this implicit knowledge about the process of paying the bill.

We distinguish common sense knowledge about everyday activities from knowledge describing facts about the world (which is readily available), for example, facts about the earth, facts about different countries, facts about different scientific phenomena or technology, news about the recent presidential candidate, etc. The focus of this dissertation is on common sense knowledge associated with prototypical activities that we perform on day to day basis, like going to a restaurant, making coffee, cooking pasta.

There are myriad sources of factual knowledge (for example, Wikipedia, books, newspapers), but there is a dearth of knowledge sources about common sense knowledge. For example, it is hard to find texts that explicitly describe how to eat food in a restaurant. Common-sense knowledge is often mentioned implicitly in the text and hard to infer computationally. For example, consider the activities involved in a visit to a restaurant: typically, you enter the restaurant, followed by looking at the menu, followed by ordering food and so on and so forth until you pay the bill and leave. This mundane and implicit knowledge is rarely mentioned in detail in texts, as it is assumed to be known to the reader.

Moreover, even if some text describes an activity, the order of actions mentioned in the text may not always correspond to the default order of actions as they would usually occur. For example, Figure 1.1 presents a short narrative which implicitly describes a "making coffee" scenario. But, as one can notice, the order of actions mentioned in the story does not exactly correspond to the order of actions that would take place while making coffee. For example, when making coffee, usually sugar is added before you start drinking the coffee.

The common sense knowledge about the sequences of prototypical activities is referred to as *script knowledge*. It has been shown that knowledge about scripts makes communication more efficient (Schank and Abelson, 1977). Script knowledge is implicitly acquired by humans and is evoked when a particular situation demands (Bower et al., 1979) it. In order to have a natural language understanding system at par with human performance, script knowledge should be included as one of the components. **This dissertation describes attempts towards acquiring and modeling script knowledge.**

Modeling script knowledge is challenging. As described before, script knowledge is implicit in the text. Also, even if a text mentions a sequence of actions describing a prototypical activity, the order of actions may not always correspond to the default order in which they would occur prototypically. Moreover, a text may not instantiate a script scenario uniquely[4]. The text may describe two or more script scenarios simultaneously, thus intermingling scenarios. For example, a text describing a restaurant scenario may also describe a paper reading scenario going on while having food in the restaurant. We discuss these issues and their solutions in more detail in Chapter 5.

In order to model script knowledge and address the challenges that come with it, we use statistical and probabilistic techniques. Recently, there have been successful

[4]A scenario refers to a specific kind of prototypical activity, for example, baking a cake, making coffee, visiting a restaurant etc. A script is an event structure describing the scenario. Refer to Chapter 3 for more details.

applications of statistical and probabilistic techniques to model language understanding systems. These techniques are data driven and infer the rules for learning the syntax or representing the meaning from the text itself. These techniques have been successfully applied to the task of machine translation, question answering, summarization, etc. (Jurafsky and Martin, 2009) In Chapter 4, we describe the statistical models proposed by us for modeling script knowledge.

1.2 Frame Semantics

The main focus of this dissertation is modeling script knowledge for natural language understanding. Before we delve deeper into different approaches for script modeling, we focus our attention on something more basic, i.e. modeling individual activities/events in a script. Typically, script events are verbalized by a sentence. One of the many possible ways of understanding the meaning of a sentence is in terms of the events/activities (and the involved participants) it describes. For example, sentence (1.7) can be understood in terms of the "cutting" activity which is initiated by the actor "Ashutosh" on the object "pizza" with the help of another object "knife". The main activity in the sentence is described by a *predicate* (typically a verb), and the people or objects participating in the activity are described by *semantic roles* for that particular predicate.

(1.7) Ashutosh cut the pizza with a knife.

(1.8) Ashutosh sliced the pizza into pieces with a knife.

For example, in the sentence (1.7), Ashutosh is the initiator of the action and is assigned the *agent* role. Pizza is the object acted upon and has the role *item* and similarly, the knife is assigned the *instrument* role. Sentence (1.8) has semantics

similar to sentence (1.7) but with different surface forms. The various predicates and the corresponding roles, expressing the idea of some activity, together constitute a *semantic frame*. As we have seen in the examples, the same concept can be expressed via different surface realizations; therefore a semantic frame has a set of predicates (called lexical units) each of which describes the main concept of the semantic frame. Similarly, semantic roles describe the participants that would be part of the activity. A semantic frame can be evoked in the text by one of the lexical units along with some of the semantic roles.

Semantic frames evoked in the text abstract out the information about main events described in the text. Semantic frames can help in understanding the text by answering questions like *who did what to whom and by what means?* The task of assigning semantic roles to phrases in sentences is referred to as *semantic role labeling* (SRL). More details about semantic frames and SRL are described in chapter 2. SRL has been shown to be useful for a variety of natural language processing applications like question answering (Shen and Lapata, 2007), summarization (Khan et al., 2015), plagiarism detection (Osman et al., 2012) etc.

This idea of understanding the text in terms of semantic frames evoked in the text was proposed by Charles J. Fillmore (Fillmore, 1976, 1968, 1977a,b, 1982, 1985; Fillmore and Baker, 2001). Semantic frames describe a type of event, relation, or entity and the participants in it. With the goal of understanding the text in terms of semantic frames, the FrameNet[5] (Baker et al., 1998) project was started in 1997 at the International Computer Science Institute in Berkeley and has been in operation since then. FrameNet is a corpus of English text annotated with semantic frames. More details about the FrameNet corpus are given in chapter 2. This dissertation proposes an unsupervised probabilistic model for inferring semantic frames from the text (Modi

[5] https://framenet.icsi.berkeley.edu/fndrupal/IntroPage

et al., 2012). This is a step towards inferring the meaning of the text, as explained earlier.

1.3 Script Knowledge

Frame semantics describes the meaning of the text in terms of activities described in the text. FrameNet captures some relations (e.g. causal, precedence) between some of the frames. FrameNet also describes script like structures called "scenario frames", which partially describe some script scenarios. However, the coverage of FrameNet is quite limited and not appropriate for modeling script knowledge. In general, frame semantics works more at the sentence level rather than at the discourse level. Intuitively, it makes sense that an event evoked in the current sentence is influenced (for example, via causal relation) by events evoked earlier in the text. For example, in the restaurant scenario mentioned previously, the ordering event is followed by the eating event. This chain of events describing a stereotypical activity constitutes a *Script*.

Script theory has been motivated by research in cognitive psychology, as described in detail in Chapter 3. Scripts are a step towards modeling procedural knowledge (Schank and Abelson, 1977). Scripts have been shown to be useful for variety of natural language processing (NLP) tasks since they provide knowledge about prototypical order of events (Bower et al., 1979; Chambers and Jurafsky, 2009, 2008; Miikkulainen, 1993; Rau et al., 1989; Schank, 1991; Schank and Cleary, 1995).

This dissertation proposes a probabilistic neural network based model for learning the prototypical order of events in a script (Modi and Titov, 2013, 2014). The proposed model achieves state of the art performance on event ordering and event paraphrase detection tasks. We also propose another neural network based probabilistic model for event prediction. The model is evaluated using a task called the *narrative cloze* (inspired from psychology) task. The narrative cloze task evaluates the ability of a

script learning model to predict a missing event in a given sequence of events. Chapter 4 describes the model and its evaluation in more details.

From the cognitive point of view, it would be interesting to know if script knowledge affects language comprehension. To address this question, this dissertation proposes different models to study the influence of script information at the level of discourse. Models are evaluated using a discourse level evaluation task. More particularly, a model is required to predict an upcoming discourse referent in the text. Predictions of different models, with a different set of features (for example, linguistic features, script features, etc.), are compared to human predictions on the same task. The resulting experiments ascertain that script knowledge does play a substantial role in language comprehension (Modi et al., 2017). A detailed description is given in Chapter 6.

1.4 Thesis Structure

This dissertation first addresses the models for frame semantics in **Chapter 2**. It proposes a non-parametric Bayesian model for inferring semantic frames from the text in an unsupervised fashion. The experiments show the efficacy of the model. As described before, frame semantics work more at sentence level rather than at discourse level. **Chapter 3** introduces the concept of scripts and motivates script theory both from cognitive and computational perspective. It describes recent work done in the area of script modeling.

Next, this dissertation proposes methods for modeling scripts in **Chapter 4**. In particular, it proposes a neural network based probabilistic model for learning prototypical event order. The model outperforms other models in event ordering evaluation experiments. This chapter also describes a model for event prediction. It is evaluated using the narrative cloze task, an important task for evaluating a model's ability to capture and generalize script knowledge.

Currently, most of the corpora describing narratives involve inter-mingling of scripts from many different scenarios. For example, a narrative text might be describing a story of a person traveling in a bus and reviewing a scientific paper. In this narrative text, two scripts are active, one corresponding to the scenario of RIDING A BUS and another one corresponding to REVIEWING A SCIENTIFIC PAPER. Such narratives, involving an inter-play of different scenarios, make it difficult to study the influence of script knowledge corresponding to a single scenario. In order to address this problem and simplify the analysis of the contribution of scenario specific knowledge, this dissertation introduces the *InScript* corpus in **Chapter 5**. InScript is a novel corpus of stories, where each story focuses on only one particular scenario. The corpus is annotated with script specific event types, participant types and coreference information.

In order to check the influence of script knowledge on language comprehension, **Chapter 6** describes models for checking the influence of script knowledge on discourse referent prediction. A battery of experiments confirms our conjecture about the role of script knowledge in language comprehension. We address a problem which explores cognitive aspects of natural language processing. Finally, **Chapter 7** concludes the dissertation and explores future directions for this line of research.

1.5 Thesis Contributions

This dissertation makes the following contributions:

- It proposes a novel unsupervised non-parametric Bayesian probabilistic model for inferring semantic frames from the text (Modi et al., 2012).
- It proposes a neural network based probabilistic model for event ordering and event paraphrasing in scripts. The model has state-of-the-art performance on script event-ordering task (Modi and Titov, 2013, 2014).

- It proposes a neural network based probabilistic model for event prediction. The model is evaluated using the narrative cloze task. Experiments show the advantages of the model over co-occurrence count based methods. (Modi, 2016).

- It introduces the InScript corpus (Modi et al., 2016). The corpus is annotated with script specific fine-grained events and participants. InScript is a novel corpus of script specific narratives. This corpus will be a useful resource for the research community working in the area of script knowledge modeling.

- This dissertation investigates the influence of script knowledge on discourse referent prediction (Modi et al., 2017). The proposed models show a substantial contribution of script knowledge in language comprehension. We hope that it will open many more avenues for research on the topic of the influence of script knowledge on language comprehension.

CHAPTER 2

Unsupervised Induction of Frame-Semantic Representations

Inside every non-Bayesian there is a Bayesian struggling to get out.

**

Dennis V. Lindley

2.1 Introduction

In Chapter 1, we gave an example which demonstrated how the meaning of a sentence can be analyzed in terms of the actions/activities it describes. This chapter describes in detail how this idea can be formalized and realized practically.

Consider, for example, sentence (2.1). This sentence talks about a buying activity where one company (Microsoft) purchases another company (Malubaa). The meaning (semantics) of the sentence can easily be captured by the predicate describing the activity (purchase) and the corresponding syntactic arguments (Microsoft and Maluuba). But there is not just one way of expressing the acquisition idea. Consider sentences (2.2) through (2.4). All of them are talking about the same activity but have different surface forms (predicates and sentence construction). A close examination indicates the commonality across all the sentences. There is a buying activity going on, which involves a buyer (Microsoft) and an entity (another company Maluuba) which is bought. Sentences also indicate the money involved and the time when it happened.

(2.1) Microsoft purchased the Canadian company Maluuba.

(2.2) A.I. company Maluuba was bought by Microsoft on January 9, 2017.

(2.3) Microsoft purchased Maluuba for $30 million.

(2.4) In a recent move, Microsoft bought the Canadian startup Maluuba.

The examples are indicative of how the semantics of the sentence can be captured in terms of semantic representations involving predicates and corresponding arguments. The roles played by different arguments are referred to as **semantic roles**. Semantic roles are an abstract concept describing the type of entity that would be involved in the

activity. For example, in sentence (2.1), two semantic roles are realized, the *buyer* role (realized by Microsoft) and the *goods* role (realized by Maluuba). The buyer role abstracts the notion of an entity that would buy in a buying activity. Similarly, the goods role abstracts the notion of an entity that would be bought in a buying activity. Sentence (2.2) has in addition a *time* role (realized by January 9, 2017) and sentence (2.3) a *money* role (realized by $30 million).

As indicated in the examples, there are multiple ways of specifying the same semantic content using different predicates and sentence constructions. But the meaning captured by all of them is similar. These many to one realizations are captured using semantic frames (Fillmore, 1985). A semantic frame is an abstract concept describing an activity and all its possible roles. A semantic frame includes all the predicates (verbal as well as nominal) which can describe the activity and the corresponding semantic roles which describe different entities that would be involved in the activity. The predicates describing the main action are referred to as *lexical units* or *frame evoking elements* and the different semantic roles are referred to as *frame elements*.

For example, sentences (2.1) through (2.4) can be described by the semantic frame "Commerce_buy". The lexical units in this semantic frame are "buy (verb), buyer (noun), client (noun), purchase [active] (noun), purchase (verb), purchaser (noun)". Each semantic frame has some core semantic roles which are essential for the activity to be described and they must be present in each realization of the frame. Other semantic roles are non-core; these roles convey additional information and may or may not be present in each realization of the frame. Non-core roles mostly correspond to modifiers and usually generalize across different frames. Core roles in the "Commerce_buy" frame are "buyer" and "goods". Some of the non-core roles are "time", "money", "place", "means", etc. Most of the non-core roles are mostly consistent across different

frames; however, some of the non-core roles (e.g. "money") can be a core role in some of the frames.

Semantic roles capture the shallow semantics of the text. These allow answering questions like "who did what to whom?" and also possibly answer questions related to additional information like "how, where, when, etc." Semantic roles have been shown to be useful for a variety of natural language processing applications, for example question answering (Shen and Lapata, 2007), summarization (Khan et al., 2015) or plagiarism detection (Osman et al., 2012). The task of assigning semantic roles to phrases in the sentences is referred to as **semantic role labeling (SRL)**. This task is important in its own right, as it helps to draw inferences from the text that would not be possible from parse trees alone.

There exists two major role annotated corpora for English: *PropBank* and *FrameNet*. PropBank (Proposition Bank) (Palmer et al., 2005) is a corpus of semantic role annotated text from the Penn TreeBank. PropBank annotates predicate-specific semantic roles. Each sense of a predicate has its own set of semantic roles for the arguments. However, some of the semantic roles are fairly general and are shared across most of the predicates. For example, Arg0 role represents proto-agent and Arg1 role represents proto-patient. In general case, there is no guarantee that same role name means the same across different predicates.

The other corpus with semantic role annotations is FrameNet[1] (Fillmore, 1976, 1968, 1977a,b, 1982, 1985; Fillmore and Baker, 2001). Unlike PropBank, where semantic roles are predicate specific, FrameNet has semantic roles that are specific to a semantic frame. As explained before, each semantic frame in FrameNet comes with its set of lexical units, core and non-core semantic roles. This thesis focuses on techniques for performing SRL on FrameNet.

[1] https://framenet.icsi.berkeley.edu/fndrupal/IntroPage

FrameNet corpus includes a dataset annotated with semantic role annotations, these annotations can be useful for training a supervised system, for learning to perform SRL on the new text. A number of approaches have been proposed in this regard. The next section motivates why supervised approaches may not always be feasible and how unsupervised techniques can be useful for SRL.

2.2 Unsupervised SRL: Motivation

Most approaches to SRL have relied on large annotated corpora (Carreras and Màrquez, 2005; Hajič et al., 2009; Surdeanu et al., 2008).By far, most of this work has focused on PropBank-style representations (Palmer et al., 2005) where roles are defined for each individual verb or even individual senses of a verb. The only exceptions are modifiers and roles $A0$ and $A1$ which correspond to proto-agent (a doer, or initiator of the action) and proto-patient (an affected entity), respectively. However, the SRL task is known to be especially hard for the FrameNet-style representations for a number of reasons, including the lack of cross-frame correspondence for most roles, fine-grain definitions of roles and frames in FrameNet, and relatively small amounts of statistically representative data (Das and Smith, 2011; Das et al., 2010; Erk and Pado, 2006; Palmer and Sporleder, 2010). Another reason for reduced interest in predicting FrameNet representations is the lack of annotated resources for most languages, with annotated corpora available or being developed only for few languages, for example, English (Ruppenhofer et al., 2006), German (Burchardt et al., 2006), Spanish (Subirats, 2009), Japanese (Ohara et al., 2004), Polish (Linde-Usiekniewicz et al., 2008), Swedish (Borin et al., 2010), Chinese (Jihong et al., 2010), Danish (Bick, 2011) and Korean (Kim et al., 2016). Different corpora across different languages vary in coverage and size and are much smaller than the English FrameNet corpus.

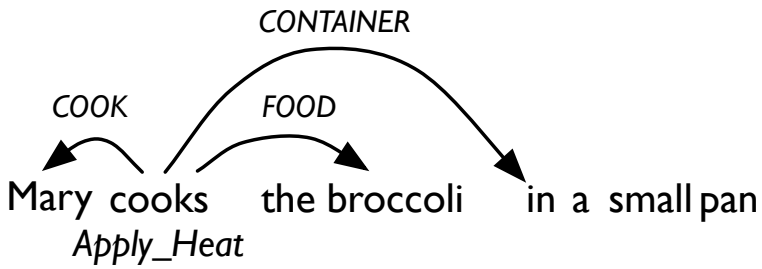

Fig. 2.1 An example of a semantic dependency graph.

Due to the scarcity of labeled data, purely unsupervised setups recently started to receive considerable attention (Grenager and Manning, 2006; Lang and Lapata, 2010, 2011a,b; Swier and Stevenson, 2004; Titov and Klementiev, 2012). However, all these approaches have focused on PropBank-style representations. FrameNet representations, though arguably more powerful, are harder to learn in the supervised setting and harder to annotate, and annotated data is available for considerably fewer languages. This is the gap which we address in this study.

More specifically, we extend an existing state-of-the-art Bayesian model for unsupervised semantic role labeling and apply it to support FrameNet-style semantics. In other words, our method jointly induces both frames and frame-specific semantic roles. We experiment only with verbal predicates and evaluate the performance of the model with respect to some natural baselines. Though the scores for frame induction are not high, we argue that this is primarily due to the very high granularity of FrameNet frames, which is hard to reproduce for unsupervised systems, as the implicit supervision signal is not capable of providing these distinctions.

2.3 Task Definition

In this work, we use dependency representations of frame semantics. Dependency representations for SRL (Johansson and Nugues, 2008) were made popular by CoNLL-

2008 and CoNLL-2009 shared tasks (Hajič et al., 2009; Surdeanu et al., 2008), but for English were limited to PropBank. Recently, English FrameNet was also released in the dependency format (Bauer et al., 2012). Instead of predicting argument spans, in dependency representation the goal is, roughly, to predict the syntactic head of the argument. The semantic dependency representation for a sentence is shown in Figure 2.1; labels on edges denote roles and labels on words denote frames. Note that in practice the structures can be more complex as, for example, arguments can evoke their own frames or the same arguments can be shared by multiple predicates, as in right node raising constructions.

The SRL task or more specifically the frame-semantic parsing task consists, at least conceptually, of four stages: (1) identification of frame-evoking elements(FEE), (2) identification of arguments, (3) frame labeling and (4) role labeling. In this work, we focus only on the frame labeling and role labeling stages, relying on the gold standard (i.e. the oracle) for FEEs and argument identification. In other words, our goal is to label (or cluster) edges and nodes in the dependency graphs (Figure 2.1). As described in Section 2.5.1, in our work, we used a dependency-parsed FrameNet corpus (Bauer et al., 2012). Since we focus in this study on verbal predicates only, the first stage would be trivial and the second stage could be handled with heuristics as in much of the previous work on unsupervised SRL (Lang and Lapata, 2011a; Titov and Klementiev, 2012).

In addition to considering only verbal predicates, we also assume that every verb belongs to a single frame. This assumption, though restrictive, may be reasonable in practice as (a) the distribution of verb instances across frames (i.e. senses) is generally highly skewed, and (b) current state-of-the-art techniques for word-sense induction hardly beat most-frequent-sense baselines in accuracy metrics (Manandhar et al., 2010). This assumption, or its minor relaxations, is relatively standard in work

on unsupervised semantic parsing tasks (Poon and Domingos, 2010, 2009; Titov and Klementiev, 2011). From the modeling perspective, there are no major obstacles to relaxing this assumption, but it would lead to a major explosion of the search space and, as a result, slow inference.

2.4 Model and Inference

We follow previous work on unsupervised semantic role labeling (Lang and Lapata, 2011a; Titov and Klementiev, 2012) and associate arguments with their frame-specific syntactic signatures, which we refer to as *argument keys*:

- Active or passive verb voice (`ACT/PASS`).
- Argument position relative to the predicate (`LEFT/RIGHT`).
- Syntactic relation to its governor.
- Preposition used for argument realization.

Semantic roles are then represented as clusters of argument keys instead of individual argument occurrences. This representation aids our models in inducing high purity clusters (of argument keys) while reducing their granularity. Thus, if an argument key k is assigned to a role r ($k \in r$), all of its occurrences are labeled r.

2.4.1 Dirichlet Process and Chinese Restaurant Process

Before delving deeper into the proposed model for frame semantic parsing, we will briefly describe some of the technical background required for understanding the model. Most of the concepts described in this section are from Teh (2007). We explain these concepts briefly; for full details refer to Teh (2007).

Our frame semantic model is based on non-parametric Bayesian statistics. An important concept in non-parametric Bayesian statistics is that of a Dirichlet process.

A Dirichlet process (DP) (Ferguson, 1973) is a stochastic process such that each draw from the process is a probability distribution. It basically defines a distribution over distributions. A Dirichlet process, $DP(\alpha, H)$, is defined by a base distribution H and a real valued concentration parameter α. Note that the base distribution can be continuous or discrete. Each distribution drawn from the DP is guaranteed to be almost surely discrete. The base distribution H is the expected value of the DP. The concentration parameter α behaves like inverse variance and controls the discretization of the distribution drawn from the DP. As $\alpha \to 0$, the drawn distributions tend to concentrate around a single value and when $\alpha \to \infty$, the drawn distributions tend to the base distribution H weakly or pointwise. In our case, a DP is used to model distributions of arguments for each role.

Formally, let Θ be the support for the base distribution H. Let $A_1, A_2, A_3, ...A_r$ be a finite measurable partition of Θ and G a distribution drawn from the DP. Then,

$$\begin{aligned} & G \sim DP(\alpha, H) \\ \Rightarrow \quad & (G(A_1), ..., G(A_r)) \sim Dir(\alpha H(A_1), ..., \alpha H(A_r)) \end{aligned} \tag{2.1}$$

where $Dir()$ is the Dirichlet distribution.

Posterior Distribution: Let $G \sim DP(\alpha, H)$ and let $\theta_1, \theta_2, ..., \theta_n$ be a sequence of draws from G. The posterior distribution of G, given observed $\theta_1, \theta_2, ..., \theta_n$, can be shown to be a distribution drawn from a DP (Teh, 2007), i.e.:

$$G \mid \theta_1, \theta_2, ..., \theta_n \sim DP(\alpha + n \ , \ \frac{\alpha}{\alpha + n} H + \frac{n}{\alpha + n} \frac{\sum_{i=1}^{n} \delta_{\theta_i}}{n})$$

The predictive distribution for a new observed value θ_{n+1} conditioned on $\theta_1, \theta_2, ..., \theta_n$ with G marginalized out can be shown to be equal to the posterior base distribution

given $\theta_1, \theta_2, ..., \theta_n$:

$$\theta_{n+1} \mid \theta_1, \theta_2, ..., \theta_n \sim \frac{1}{\alpha + n}(\alpha H + \sum_{i=1}^{n} \delta_{\theta_i}) \tag{2.2}$$

Expression (2.2) implies the clustering property of a DP. Let $\theta_1^*, \theta_2^*, ..., \theta_m^*$ be unique values among $\theta_1, \theta_2, ..., \theta_n$ and n_k be the frequency of θ_k^*. The predictive distribution can be written as:

$$\theta_{n+1} \mid \theta_1, \theta_2, ..., \theta_n \sim \frac{1}{\alpha + n}(\alpha H + \sum_{k=1}^{m} n_k \delta_{\theta_k^*}) \tag{2.3}$$

Expression (2.3) implies that θ_{n+1} takes a value θ_k^* with probability proportional to n_k, number of times θ_k^* has been observed previously. θ_{n+1} takes a new value with probability proportional to α. The above phenomenon implies the rich-gets-richer property. A new observation clusters with one of the previously existing clusters if that cluster has a large number of observations; otherwise it creates a new cluster. The clustering property of a DP induces partitioning of the set, $[n] = \{1, 2, 3, ...n\}$, such that in each cluster k, each θ_i takes the same value θ_k^*.

Chinese Restaurant Process: The distribution over partitions is called the Chinese restaurant process (CRP) (Teh, 2007). CRP derives its name from the seating arrangement of customers in a Chinese restaurant. In this setting, we have a Chinese restaurant with an infinite number of tables, each of which has infinite seats available. The first customer who comes to a restaurant sits at one of the tables. The next customer either sits at a new table or sits with the first customer. The $n+1$th customer either sits at a new table with probability proportional to α or sits at one of the occupied tables k, with probability proportional to the number of people (n_k) already sitting there. After n customers have settled in the restaurant, the distribution of customers over different tables symbolizes a partition of $[n]$ integers into different clusters. As we describe

in the next section, our frame semantic parsing model employs a CRP to model the distribution over the partition of argument keys.

2.4.2 A Model for Frame-Semantic Parsing

In the previous section, we described some of the machinery associated with non-parametric Bayesian statistics. In this section, we describe our model, which employs DPs and CRP described in the previous section. Our approach is similar to the models of Titov and Klementiev (2011, 2012). Please see Section 2.6 for a discussion of the differences.

Our model encodes three assumptions about frames and semantic roles. First, we assume that the distribution of lexical units (verbal predicates) is sparse for each semantic frame. Second, we enforce the selectional restriction assumption: we assume that the distribution over potential argument fillers is sparse for every role, implying that 'peaky' distributions of arguments for each role r are preferred to flat distributions. Third, each role normally appears at most once per predicate occurrence. Our inference will search for a frame and role clustering which meets the above requirements to the maximal extent.

Our model associates three distributions with each frame. The first one (ϕ) models the selection of lexical units, the second (θ) governs the selection of argument fillers for each semantic role and the third (ψ) models (and penalizes) duplicate occurrence of roles. Each frame occurrence is generated independently given these distributions. Let us describe the model by first defining how the set of model parameters and an argument key clustering are drawn, and then explaining the generation of individual frame instances. The generative story is formally presented in Figure 2.2.

For each frame, we begin by drawing a distribution of its lexical units from a DP prior $DP(\gamma, H^{(P)})$ with a small concentration parameter γ, and a base distribution

$H^{(P)}$, pre-computed as normalized counts of all verbs in our dataset. Next, we generate a partition of argument keys B_f from CRP(α) with each subset $r \in B_f$ representing a single frame specific semantic role. The crucial part of the model is the set of selectional preference parameters $\theta_{f,r}$, the distributions of arguments x for each role r of frame f. We represent arguments by lemmas of their syntactic heads.[2] In order to encode the assumption about sparseness of the distributions $\theta_{f,r}$, we draw them from the DP prior $DP(\beta, H^{(A)})$ with a small concentration parameter β; the base probability distribution $H^{(A)}$ is just the normalized frequencies of arguments in the corpus. Finally, the geometric distribution $\psi_{f,r}$ is used to model the number of times a role r appears with a given frame occurrence. The decision whether to generate at least one role r is drawn from the uniform Bernoulli distribution. If 0 is drawn then the semantic role is not realized for the given occurrence; otherwise the number of additional roles r is drawn from the geometric distribution $Geom(\psi_{f,r})$. The Beta priors over ψ indicate the preference towards generating at most one argument for each role.

Now, when parameters and argument key clusterings are chosen, we can summarize the remainder of the generative story as follows. We begin by independently drawing occurrences for each frame. For each frame occurrence, we first draw its lexical unit. Then for each role, we independently decide on the number of role occurrences. Then we generate each of the arguments (see **GenArgument** in Figure 2) by generating an argument key $k_{f,r}$ uniformly from the set of argument keys assigned to the cluster r, and finally choosing its filler $x_{f,r}$, where the filler is either a lemma or the syntactic head of the argument.

[2]For prepositional phrases, we take as head the head noun of the object noun phrase as it encodes crucial lexical information. However, the preposition is not ignored, but rather encoded in the corresponding argument key.

Parameters:

for each frame $f = 1, 2, \ldots$:
  $\phi_f \sim DP(\gamma, H^{(P)})$ [distrib of lexical units]
  $B_f \sim CRP(\alpha)$ [partition of arg keys]
  for each role $r \in B_f$:
    $\theta_{f,r} \sim DP(\beta, H^{(A)})$ [distrib of arg fillers] $\psi_{f,r} \sim Beta(\eta_0, \eta_1)$ [geom distr for dup roles]

Data Generation:

for each frame $f = 1, 2, \ldots$:
  for each occurrence of frame f:
    $p \sim \phi_f$ [draw a lexical unit]
    for every role $r \in B_f$:
      if $[n \sim Unif(0, 1)] = 1$: [role appears at least once]
        GenArgument(f, r) [draw one arg]
        while $[n \sim \psi_{f,r}] = 1$: [continue generation]
          GenArgument(f, r) [draw more args]

GenArgument(f, r):
  $k_{f,r} \sim Unif(1, \ldots, |r|)$ [draw arg key]
  $x_{f,r} \sim \theta_{f,r}$ [draw arg filler]

Fig. 2.2 Generative story for the frame-semantic parsing model.

2.4.3 Inference

We use a simple approximate inference algorithm based on greedy search for the maximum a-posteriori clustering of lexical units and argument keys. We begin by assigning each verbal predicate to its own frame, and then iteratively choose a pair of frames and merge them. Note that each merge involves inducing a new set of roles, i.e. a re-clustering of argument keys, for the new merged frame. We use the search procedure proposed in Titov and Klementiev (2012), in order to cluster argument keys for each frame.

Our search procedure chooses a pair of frames to merge based on the largest incremental change to the objective due to the merge. Computing the change involves re-clustering of argument keys, so considering all pairs of initial frames containing single verbal predicates is computationally expensive. Instead, we prune the space of possible

pairs of verbs using a simple but effective pre-processing step. Each verb is associated with a vector of normalized aggregate corpus counts of syntactic dependents of the verb (ignoring the type of dependency relation). Cosine similarity of these vectors is then used to prune the pairs of verbs so that only verbs which are distributionally similar enough are considered for a merge. Finally, the search terminates when no additional merges result in a positive change to the objective.

2.5 Experimental Evaluation

2.5.1 Data

We used the dependency representation of the FrameNet corpus (Bauer et al., 2012). The corpus is automatically annotated with syntactic dependency trees produced by the Stanford parser. The corpus is manually annotated with frame and role information. The data consists of 158,048 annotated sentences with 3,474 unique verbal lemmas organized in 722 gold frames.

2.5.2 Evaluation Metrics

We cannot use supervised metrics to evaluate our models, since we do not have an alignment between gold labels and clusters induced in the unsupervised setup. Instead, we use the standard purity (PU) and collocation (CO) metrics as well as their harmonic mean (F1) to measure the quality of the resulting clusters. Purity measures the degree to which each cluster contains arguments (verbs) sharing the same gold role (gold frame) and collocation evaluates the degree to which arguments (verbs) with the same gold roles (gold frame) are assigned to a single cluster; see Lang and Lapata (2010). As in previous work, for role induction, the scores are first computed for individual predicates

and then averaged with the weights proportional to the total number occurrences of roles for each predicate.

2.5.3 Model Parameters

The model parameters were tuned coarsely by visual inspection: $\alpha = 1.e\text{-}5$, $\beta = 1.e\text{-}4$, $\gamma = 1$, $\eta_0 = 100$, $\eta_1 = 1.e\text{-}10$. Only a single model was evaluated quantitatively to avoid overfitting to the evaluation set.

2.5.4 Qualitative Evaluation

Our model induced 128 multi-verb frames from the dataset. Out of 78,039 predicate occurrences in the data, these correspond to 18,963 verb occurrences (or, approximately, 25%). Some examples of the induced multi-verb frames are shown in Table 2.1 on page 29. As we can observe from the table, our model clusters semantically related verbs into a single frame, even though they may not correspond to the same gold frame in FrameNet. Consider, for example, the frame *(ratify::sign::accede)*: the verbs are semantically related and hence they should go into a single frame, as they all denote a similar action.

Another result worth noting is that the model often clusters antonyms together as they are often used in similar context. For example, consider the frame *(cool::heat::warm)*: the verbs *cool*, *heat*, and *warm* all denote a change in temperature. This agrees well with annotation in FrameNet. Similarly, we cluster *sell* and *purchase* together. This contrasts with FrameNet annotation, as FrameNet treats them not as antonyms but as different views on the same situation and according to their guidelines, different frames are assigned to different views.

Often frames in FrameNet correspond to more fine-grained meanings of the verbs, as we can see in the example for *(plait::braid::dye)*. The three describe a similar

activity involving hair but FrameNet gives them a finer distinction. Arguably, implicit supervision signal present in the unlabeled data is not sufficient to provide such fine-grained distinctions.

The model does not distinguish verb senses, i.e. it always assigns a single frame to each verb, so there is an upper bound on our clustering performance.

2.5.5 Quantitative Evaluation

Now we turn to quantitative evaluation of both frame and role induction.

Frame Labeling. In this section, we evaluate how well the induced frames correspond to the gold standard annotation. Because of the lack of relevant previous work, we use only a trivial baseline which places each verb in a separate cluster (*NoClustering*). The results are summarized in Table 2.2.

As we can see from the results, our model achieves a small, but probably significant, improvement in the F1 score. Though the scores are fairly low, note that, as discussed in Section 2.5.4, the model is severely penalized even for inducing semantically plausible frames such as the frame *(plait::braid::dye)*.

Role Labeling. In this section, we evaluate how well the induced roles correspond to the gold standard annotation. We use two baselines: one is the syntactic baseline *SyntF*, which simply clusters arguments according to the dependency relation to their head, as described in Lang and Lapata (2010), and the other one is a version of our model which does not attempt to cluster verbs and only induces roles (*NoFrameInduction*). Note that the *NoFrameInduction* baseline is equivalent to the *factored* model of Titov and Klementiev (2012). The results are summarized in Table 2.3.

First, observe that both our full model and its simplified version *NoFrameInduction* significantly outperform the syntactic baseline. It is important to note that the syntactic baseline is not trivial to beat in the unsupervised setting (Lang and Lapata, 2010).

Induced frames	**FrameNet frames corresponding to the verbs**
(rush::dash::tiptoe)	**rush**: [Self_motion](150) [Fluidic_motion](19) **dash**: [Self_motion](100) **tiptoe**: [Self_motion](114)
(ratify::sign::accede)	**ratify**: [Ratification](41) **sign**: [Sign_agreement](81) [Hiring](18) [Text_Creation](1) **accede**: [Sign_Agreement](31)
(crane::lean::bustle)	**crane**: [Body_movement](26) **lean**: [Change_posture](70) [Placing](22) [Posture](12) **bustle**: [Self_motion](55)
(cool::heat::warm)	**cool**: [Cause_temperature_change](27) **heat**: [Cause_temperature_change](52) **warm**: [Cause_temperature_change](41) [Inchoative_change_of_temperature](16)
(want::fib::dare)	**want**: [Desiring](105) [Possession](44) **fib**: [Prevarication](9) **dare**: [Daring](21)
(encourage::intimidate::confuse)	**encourage**: [Stimulus_focus](49) **intimidate**: [Stimulus_focus](26) **confuse**: [Stimulus_focus](45)
(happen::transpire::teach)	**happen**: [Event](38) [Coincidence](21) [Eventive_affecting](1) **transpire**: [Event](15) **teach**: [Education_teaching](7)
(do::understand::hope)	**do**: [Intentionally_affect](6) [Intentionally_act](56) **understand**: [Grasp](74) [Awareness](57) [Categorization](15) **hope**: [Desiring](77)
(frighten::vary::reassure)	**frighten**: [Emotion_directed](44) **vary**: [Diversity](24) **reassure**: [Stimulus_focus](35)
(plait::braid::dye)	**plait**: [Hair_configuration](11) [Grooming](12) **braid**: [Hair_configuration](7) [Clothing_parts](6) [Rope_manipulation](4) **dye**: [Processing_materials](18)
(sell::purchase)	**sell**: [Commerce_sell](107) **purchase**: [Commerce_buy](93)
(glisten::sparkle::gleam)	**glisten**: [Location_of_light](52) [Light_movement](1) **sparkle**: [Location_of_light](23) [Light_movement](3) **gleam**: [Location_of_light](77) [Light_movement](4)
(forestall::shush)	**forestall**: [Thwarting](12) **shush**: [Silencing](6)

Table 2.1 Examples of the induced multi-verb frames. The left column shows the induced verb clusters and the right column lists the gold frames corresponding to each verb and the number in the parentheses are their occurrence counts.

	PU	CO	**F1**
Our approach	77.9	31.4	**44.7**
NoClustering	80.8	29.0	**42.7**

Table 2.2 Frame labeling performance.

	PU	CO	**F1**
Our approach	78.9	71.0	**74.8**
NoFrameInduction	79.2	70.7	**74.7**
SyntF	69.9	73.3	**71.6**

Table 2.3 Role labeling performance.

Though there is a minor improvement from inducing frames first, it is small and may not be significant.[3]

Another observation is that the absolute scores of all the systems, including the baselines, are significantly below the results reported in Titov and Klementiev (2012) on the CoNLL-08 version of PropBank in a comparable setting (auto parses, gold argument identification): 73.9% and 77.9% F1 for *SyntF* and *NoFrameInduction*, respectively. We believe that the main reason for this discrepancy is the difference in the syntactic representations. The CoNLL-08 dependencies include function tags (e.g., *TMP*, *LOC*), and therefore modifiers do not need to be predicted, whereas the Stanford syntactic dependencies do not provide this information and the model needs to induce it.

It is clear from these results, and also from the previous observation that only 25% of verb occurrences belong to multi-verb clusters, that the model does not induce sufficiently rich clustering of verbs. Arguably, this is largely due to the relatively small size of FrameNet, as it may not provide enough evidence for clustering. Given that our method is reasonably efficient, a single experiment was taking around 8 hours on a

[3]There is no well-established methodology for testing statistical significance when comparing two clustering methods.

single CPU, and the procedure is highly parallelizable, the next step would be to use a much larger and statistically representative corpus to induce the representations.

Additional visual inspection suggests that the data is quite noisy primarily due to mistakes in parsing. The large proportion of mistakes can probably be explained by the domain shift: the parser is trained on the WSJ newswire data and tested on more general BNC texts.

2.6 Related Work

Aside from the original model of Titov and Klementiev (2012), the most related previous method is the Bayesian method of Titov and Klementiev (2011). In that work, along with predicate-argument structure, they also induce clusterings of dependency tree fragments (not necessarily verbs). However, their approach uses a different model for argument generation, a different inference procedure, and it has only been applied and evaluated on biomedical data. The same shallow semantic parsing task has also been considered in the work of Poon and Domingos (2010, 2009), but using a Markov logic network (MLN) model and, again, only on the biomedical domain. Another closely related vein of research is on semi-supervised frame-semantic parsing (Das and Smith, 2011; Fürstenau and Lapata, 2009).

Recently, in follow-up work, Materna (2013) proposed an unsupervised approach for inducing semantic frames from a large unannotated corpus. They refer to such induced frames as "LDA-frames". They propose a generative non-parametric Bayesian model for frame induction. In their setup, a frame is a tuple of semantic roles, connected with grammatical relations e.g. subject, object, modifier, etc. A semantic role is represented as a probability distribution over all its realizations in the corpus. Each lexical unit has a distribution over all semantic frames. Parameters for the model (e.g. number of frames, roles) is estimated using CRP and DP. The generative process

proposed by them is outlined as follows: For each a lexical unit, a frame distribution is sampled from a Dirichlet distribution prior. For each realization of lexical unit, a frame is generated by drawing from a multinomial distribution over the sampled frame distribution. Then, for each slot of the frame a grammatical relation is generated by drawing from a multinomial distribution over mappings between a semantic role and a slot for a frame. In their model, the inference is performed using collapsed Gibbs sampling.

In the spirit of non-parametric Bayesian unsupervised methods for learning semantic frames, Kawahara et al. (2014) propose inducing verb (as well for each sense of the verb) specific semantic frames. They induce the frames over a large corpus i.e. Gigaword. The frames induced by them are similar to PropBank style SRL. In their setting, a frame has several syntactic slots and each of these slots has a set of words (with frequencies) which can fill the slot. In order to induce verb specific frames from a corpus, Kawahara et al. (2014) first parse the corpus with a dependency parser, extracting predicate-argument structure for each verb. Then, they merge predicate-argument structures having the same meaning, to get initial frames. Finally, they cluster the initial frames using a Chinese restaurant process (CRP) to get final semantic frames. The general ideas proposed by Materna (2013) and Kawahara et al. (2014) for unsupervised frame induction are quite similar to the approach proposed in our unsupervised frame semantic model.

2.7 Conclusions

This chapter described one of the first attempts to consider the task of unsupervised frame-semantic parsing. Though the quantitative results are mixed, we showed that meaningful semantic frames are induced. In order to improve the model results in the future, it would be interesting to consider the use of much larger corpora. It would

also be interesting to relax the assumption that frames are evoked only by verbal predicates and consider a more general set-up. In the next chapter, we go one level up and describe the concept of scripts which consider a sequence of events in a particular scenario.

CHAPTER 3

Scripts

An investment in knowledge pays the best interest.

**

Benjamin Franklin

3.1 Introduction

In Chapter 1 of the thesis, we introduced the concept of scripts. This chapter gives a detailed description of script knowledge and a brief background about work done in the area of script processing.

Scripts are defined as sequences of actions describing stereotypical human activities, for example, cooking pasta, making coffee, etc. (Schank and Abelson, 1977). As stated in the definition, scripts capture knowledge about common daily activities and consequently incorporate common sense knowledge about the world. A typical instantiation of a script, illustrating BAKING A CAKE, is shown in Figure 3.1. It shows typical activities that would take place when someone bakes a cake. Related to the concept of the script is the concept of a **scenario**. A scenario refers to a specific kind of prototypical activity, for example baking a cake, making coffee, visiting a restaurant, etc. A script is an event structure describing the scenario.

At the linguistic level, a script may be realized in a variety of ways. In a text instantiating a script, the lexical realization of each action/activity in the script is referred to as an **event description** (ED). An event description consists of a verbal predicate and the associated noun phrases (NP). For example, Figure 3.2 shows the event description "Nupur added butter to the bowl", where "added" (or simply taking the lemma "add") is the predicate, and "Nupur", "butter" and "bowl" are the associated NPs. The predicate in an event description is referred to as an *event verb*. An event verb in an event description is associated with a script specific **event type**. NPs in an event description are instances of different semantic roles, but in the larger context of scripts they have a different global function, and we refer to these as *participants*. Participants are specific to a script. Each of the participants is associated with a script specific **participant type**.

1. Take out box of cake mix from shelf
2. Gather together cake ingredients
3. Get mixing bowl
4. Get mixing tool or spoon or fork
5. Add ingredients to bowl
6. Stir together and mix
7. Use fork to breakup clumps
8. Preheat oven
9. Spray pan with non stick or grease
10. Pour cake mix into pan
11. Put pan into oven
12. Set timer on oven
13. Bake cake
14. Remove cake pan when timer goes off
15. Stick tooth pick into cake to see if done
16. Let cake pan cool then remove cake

Fig. 3.1 Event sequence for the BAKING A CAKE scenario.

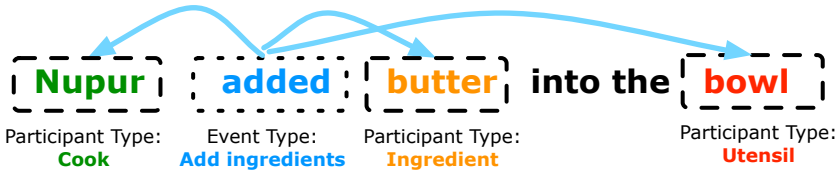

Fig. 3.2 An event in the BAKING A CAKE scenario.

1. Take out box of cake mix from shelf
2. Gather together cake ingredients
3. Get mixing bowl
4. Get mixing tool or spoon or fork
5. Add ingredients to bowl
6. Stir together and mix
7. Use fork to breakup clumps
8. Preheat oven
9. Spray pan with non stick or grease
10. Pour cake mix into pan
11. Put pan into oven
12. Set timer on oven
13. Bake cake
14. Remove cake pan when timer goes off
15. Stick tooth pick into cake to see if done
16. Let cake pan cool then remove cake

1. Get a cake mix
2. Mix in the extra ingredients
3. Prepare the cake pan
4. Preheat the oven
5. Put the mix in the pans
6. Put the cake batter in the oven
7. Take it out of the oven

1. Purchase cake mix
2. Preheat oven
3. Grease pan
4. Open mix and add ingredients
5. Mix well
6. Pour into prepared pan
7. Bake cake for required time
8. Remove cake from oven and cool
9. Turn cake out onto cake plate
10. Apply icing or glaze

Fig. 3.3 Different event sequence descriptions (ESDs) for the BAKING A CAKE scenario (Wanzare et al., 2016).

For example, in Figure 3.2, the event verb add has the event type "add ingredients", the participant butter has the participant type "ingredients", the participant Nupur has the participant type "cook" and the participant bowl has the participant type "utensil". In an event description, an event verb is an instantiation of an event type and similarly, a participant is an instance of a participant type. As mentioned before, this dissertation introduces the InScript corpus in Chapter 5, which describes and illustrates event types and participant types for different scenarios. In this thesis, sometimes, an event description is also referred to simply as an event, in which case it refers to both the event verb and participants. In most of the cases, the context makes it clear what is being referred to when the term "event" is used. When there is a possibility for confusion, we make it explicit.

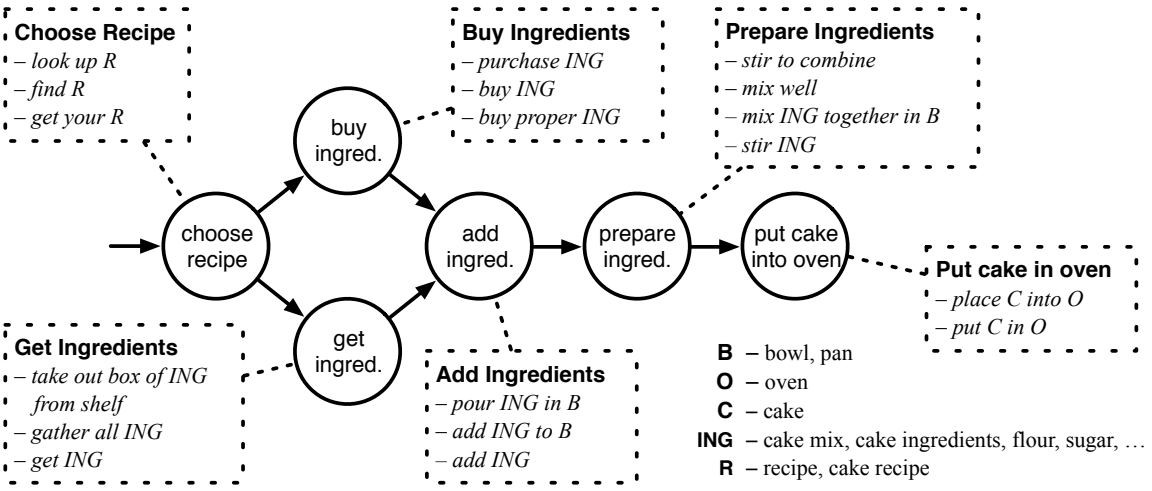

Fig. 3.4 Script for the BAKING A CAKE scenario represented as a directed acyclic graph (Wanzare et al., 2016).

At the lexical level, when describing a scenario, one may be more verbose about an event and describe it in terms of two or more sub-events. One may verbalize an event in different ways. For example, an event describing the activity of "adding ingredients" can be done in multiple ways, for example, "pour ingredients to a bowl", "add ingredients to the mixer" or "add eggs, butter, baking soda, and flour". Each instantiation of a script describing the corresponding prototypical activity is referred to as an **event sequence description** (ESD). An example of some of the possible ESDs for the BAKING A CAKE scenario are shown in Figure 3.3.

In addition to linguistic variations which are possible across different ESDs, as evident from Figure 3.3, there are multiple ways of executing a BAKING A CAKE scenario. Some of the events are optional and can be skipped. For example, a cake "icing or glazing" might not be done in some cases. Some of the events may be executed in different order. For example, one may preheat the oven first and then mix the ingredients, or the other way round.

Learning script structure from ESDs is challenging due to various possibilities for variations when describing a scenario. Moreover, sometimes in a narrative text (e.g.

Figure 1.1 in Chapter 1) describing a scenario, the order of events mentioned in the text may not be the order in which they would be executed in a script.

Scripts essentially have a non-linear structure. Figure 3.4 shows a possible script structure for the BAKING A CAKE scenario. In order to induce script structure from ESDs, two main tasks need to be addressed. One of the tasks is about learning to identify event descriptions that are paraphrases of each other. Another important task is about learning the structure of a script by learning the prototypical order of events in a script. In Chapter 4, we describe our approach for modeling script knowledge, by developing a neural network based probabilistic model for event ordering and event paraphrasing tasks.

3.2 Scripts: Motivation

As described in the previous section, scripts capture a prototypical order of events. When instantiated in a narrative text, script knowledge affects the discourse structure of the text in non-trivial ways. The events evoked in the beginning of the text may affect the lexical formulation of events later in the text. Script information may affect the lexical formulation of the noun phrases in the text. For example, when describing a restaurant scenario, addressing a person for the first time, one may simply refer to the person as "the waiter", instead of more verbose description like "a person who takes an order". Global nature of scripts makes them important for understanding both the linguistic and cognitive aspects of language processing.

For example, from the linguistic point of view, script knowledge plays a part in discourse phenomena. As demonstrated in this dissertation (see Chapter 6) script knowledge contributes to predicting an upcoming discourse referent in a narrative text. From the application point of view, it shows that script information can be useful for coreference resolution systems.

Nupur went to LeMéridian. She ordered a sirloin steak with French fries. She tipped well.

Fig. 3.5 A small narrative describing a visit to a restaurant.

As explained in detail in Section 3.3, the notion of scripts has its origins in cognitive psychology. Script knowledge is an important component of human memory and cognition. This makes it attractive to study the role of script knowledge in human cognition, with the help of computational models.

Script knowledge has been applied in a variety of NLP applications. Ordering information associated with a script has been utilized in the area of information extraction (Rau et al., 1989). In their experiments, Bower et al. (1979) have shown script knowledge to be useful for summarization. Script knowledge has been applied in automated storytelling systems (Schank, 1991) and in the development of intelligent tutoring systems for educational purposes (Schank and Cleary, 1995).

Apart from linguistic and cognitive aspects, scripts find wider applications in the area of artificial intelligence (A.I.) Since scripts capture common sense knowledge about the world, scripts can be useful for a variety of AI applications. For example, script knowledge can be useful for robot planning systems, guiding a robot in taking the next possible action. Some of the scripts (corresponding to certain scenarios) also capture procedural knowledge about the world and can be applied in systems implementing personal digital assistants and dialog systems. Scripts can be useful for drawing inferences from the text (Miikkulainen, 1993), which can be useful for NLP applications, for example, question answering systems (Bower et al., 1979).

As an example, consider the small narrative described in Figure 3.5. The narrative skips many details that would typically be part of a restaurant script, for example, the narrative omits mention of a waiter, but still a reader can easily guess that the

order was placed with a waiter. It also skips many events that would occur in a typical restaurant script; for example, it does not mention about the sitting action in the restaurant, or the eating event. In spite of skipping so many details, a reader can easily infer them due to common sense knowledge about scripts. After reading the story, if a reader is asked the question "Did Nupur eat food at LeMéridian?" As mentioned, in the story where Nupur ordered food, it should not be difficult to infer that most likely, she received her order without much delay, she ate the food, and she enjoyed the food and the restaurant ambiance. Being happy with her overall experience, she gave a good tip to the waiter. The above example demonstrates that script knowledge can be crucial for a natural language understanding system.

3.3 Scripts: Cognitive Perspective

The script theory has underpinnings in cognitive psychology. Scripts are derived from a broader concept in cognitive psychology referred to as a *schema* (plural schemata). Schemata are mental frameworks or concepts used to organize and understand the world (Rumelhart, 1978). A schema consists of a general template for the knowledge shared by all instances and a number of slots that can take on different values for a specific instance. For example, the concept of a dog comes with its own schema (for example, wags tail, barks at strangers, chews bones, has four legs, etc.) Similarly, a restaurant scenario comes with its own schema in the form of a sequence of events that take place when you are in a restaurant (enter the restaurant, order food, eat, pay the bill and leave).

A script is a schema for stereotypical activities, with a specific order of events in a particular scenario. Script knowledge guides the expectations of a person in a given scenario. For example, when you plan to go to a restaurant, you already know that you are going to sit on a chair and eat at a table, and you are going to place an order with

a waiter. These expectations affect our language comprehension abilities and influence the sentence production, as the speaker is able to anticipate upcoming participants. For example, when we are writing about a restaurant, we do not write each and every detail, as these are assumed to be known to the comprehender.

Schemas and consequently script knowledge is related to the concept of *episodic memory* (Tulving, 1985). Episodic memory refers to the part of our memory associated with experiences and specific happenings in time in the serial form.[1] Episodic memory is the memory of autobiographical events (times, places, associated emotions, and other contextual who, what, when, where, why knowledge) that can be explicitly stated.[2] Research in cognitive psychology has indicated that events stored in episodic memory can be activated by some particular external event (Baars and Gage, 2010; Terry, 2015). For example, when a person visits a restaurant for the first time, the whole sequence of actions and associated participants (waiter, menu, the person himself etc) are registered in the episodic memory of the person. Next time, when the person visits another restaurant, this activates the old restaurant script (with its actions and participants) stored in the episodic memory. Episodic memory activates the relevant actions and participants for the scenario being experienced by the person.

Cognitive aspects of script theory have been extensively researched in the late 1970's and 1980's (see, for example, Abelson (1981); Ahn et al. (1987); Bellezza and Bower (1982); Den Uyl and Van Oostendorp (1980); Dyer et al. (1987); Dyer (1982); Galambos (1983); Galambos and Black (1982); Galambos and Rips (1982); Graesser et al. (1979, 1980); Kolodner (1984); Mooney (1990); Nelson and Gruendel (1981); Sharkey and Mitchell (1985); Sharkey and Sharkey (1987); Thorndyke and Hayes-Roth (1979)).

[1] http://www.human-memory.net/types_episodic.html
[2] https://en.wikipedia.org/wiki/Episodic_memory

A prominent and seminal work for the understanding role of script knowledge in human cognition was done by Bower et al. (1979) at Stanford University. They investigated cognitive and psychological implications of the Schank and Abelson (1977) script theory. The authors (Bower et al., 1979) performed a battery of experiments to test various aspects of script theory.

In general, human subjects participating in these experiments mostly agreed on the types of activity sequences and participants belonging to a script. In one set of experiments, the authors examined how script knowledge is organized in the human mind. The authors tested the non-linear nature of a script via psychological experiments involving human subjects. From the experiment, the authors concluded: "The script is not a linear chain of actions at one level but rather a hierarchically organized tree of events with several levels of subordinate actions" (Bower et al., 1979). Furthermore, the authors postulated the activity hierarchy as an event tree, with the main event at the root and series of sub-events as children. Such event hierarchies can be useful for answering "why", "how" and "when" questions. These event hierarchies can also be useful for summarizing texts.

In another set of experiments, Bower et al. (1979) investigated script event recall among the subjects. In particular, the authors presented script-centric text to human subjects with a few event descriptions removed.[3] We saw an example of such a text in Figure 3.5, where the eating event is not mentioned. Later, when subjects were asked to recall the text, they were able to recall the stated script actions more often than unstated script actions, which in turn were more than other actions. However, recall of unstated script actions increased when more instances of the same script were presented. In a related experiment, subjects were able to correctly order sequences of

[3]This is similar to the narrative cloze task mentioned in Chapter 1 and described in detail in Chapter 4.

events presented to them in a random order. This is indicative of implicit script event ordering knowledge in memory.

In another interesting experiment, Bower et al. (1979) verified the *local spread hypothesis*. The local spread hypothesis states that a script-centric sentence in the text should be read and comprehended more quickly if it is preceded in the text by a statement referring to an action that just precedes it in the actual underlying script. The authors measured the reading times of subjects who were asked to read script related texts. Subjects, in general, read the statements about events following each other in a script faster than the statements which were far apart in terms of script events.

Other researchers have tried to explain the script theory in terms of its role in the human memory system; for example, Schank (1982) introduced script theory as a dynamic model of memory. In their research on studying causality in scripts, Galambos and Black (1985) found that position and importance of an event in a script affected how quickly subjects answered questions related to a script.

In the direction of understanding cognitive aspects of the script theory, this thesis proposes a new line of research in studying the influence of the script knowledge on language comprehension (Modi et al., 2017). Chapter 6 describes in detail the battery of experiments conducted to check the contribution of script knowledge to the task of predicting upcoming discourse referents in a narrative text.

3.4 Scripts: Computational Perspective

From the computational point of view, scripts are related to the concept of frames in artificial intelligence (Minsky, 1975). The concept of frames is a variation of the schema concept described before. Frames are data structures used in artificial intelligence for representing knowledge about prototypical situations. A frame comes with information

about its application and possible frames that could follow the current frame. A frame has a hierarchical structure and can be seen as a network of nodes and relations. Top levels of the frame are fixed (being the truth about the supposed situation) and lower levels of the frame have terminals (slots) that are filled with specific instances. Frames are linked to each other and interact with each other based on actions.

Another concept related to script learning, modeling common sense knowledge about the world, is *case based reasoning* (CBR) (Kolodner, 2014). Case based reasoning proposes methods for computer reasoning based on the world knowledge stored in the memory. It goes in the direction of procedural knowledge. We will not go into details of case based reasoning, as it is beyond the scope of this dissertation.

There have been attempts to capture common sense knowledge about the world by manually constructing ontologies and knowledge bases. One such project has been *Cyc*[4] (Matuszek et al., 2006). The Cyc project attempts to build a comprehensive source of everyday common sense knowledge by incorporating all of the information about the world in an ontology and a knowledge base (KB). In order to reason over facts in the KB, Cyc has an inference engine based on predicate calculus. Cyc suffers from a number of limitations and has been described as "one of the most controversial endeavors of the artificial intelligence history"[5] (Bertino et al., 2001). Cyc is an overly complicated system due to its ambition to incorporate all the knowledge of the world and consequently, it is difficult to add facts to it manually. There are large gaps in knowledge related to ordinary objects. Since the Cyc system is complicated and has limited documentation, it is difficult to work with the system. The Cyc system lacks the ability to evolve on its own and needs an unending amount of data to produce viable results (Domingos, 2015). The Cyc system is a huge collection of facts but does

[4] http://opencyc.org/
[5] https://en.wikipedia.org/wiki/Cyc

not explicitly model the temporal relations between events as they would typically occur in a common daily activity.

Computational aspects of script theory were first addressed by Schank and Abelson (1977) for explaining how the knowledge about stereotypical activities is used in language processing (Schank, 1982). In the incipient stages of its conception, script theory was primarily used for developing computer systems for story understanding and story generation (Cullingford, 1977; Dejong, 1979; Schank and Abelson, 1977).

Initial approaches to modeling scripts for story understanding used symbolic approaches, i.e. rule based (mostly handwritten) approaches that attempt to model high level processes in human cognition and language comprehension. There are significant disadvantages to symbolic approaches; rules are hard-coded and are limited to a particular domain and data. Symbolic approaches lack the flexibility to adapt according to the statistical properties of the data (Miikkulainen, 1993). Since the rules are designed by humans, symbolic approaches are limited to the intuition and experience of the creator.

The decade 1980-90 saw concentrated efforts towards the use of *sub-symbolic* approaches for a variety of NLP applications (see e.g. Dale et al. (2000); Dorffner et al. (1992); Dyer (1995); Elman (1991); Hinton et al. (1986); Reilly and Sharkey (1992); Rumlehart et al. (1986)). Sub-symbolic approaches are also referred to as *connectionist* approaches or *artificial neural network* (or simply *neural network*) or more recently, *deep learning* approaches. Sub-symbolic approaches are based on the concept of distributed representations (Hinton, 1984, 1986). The key idea about distributed representations is that, instead of assigning a fixed symbol to a concept, it is represented as a pattern of activity distributed over several basic units. These methods draw inspiration from mechanisms that explain memory storage capability of the human brain. Sub-symbolic approaches offer significant advantages over symbolic

approaches. In particular, distributed representations are real-valued, compact and holographic.[6] Since these representations are not localized, the same set of computing units can be used for representing several concepts. Representation for similar concepts is similar. Unlike symbolic approaches, these representations are not hand-engineered but are directly learned from data and capture semantic properties of the concepts. From the cognitive point of view as well, sub-symbolic approaches appear to be plausible models of human cognition.

Sub-symbolic approaches have also been applied to the area of script learning (e.g. Lee et al. (1989); Lee (1991); Lee et al. (1992); Miikkulainen (1991, 1993, 1995); Miikkulainen and Dyer (1989); Rumlehart et al. (1986)). In particular, one of the first end to end connectionist systems modeling script knowledge is DISCERN (DIstributed SCript processing and Episodic memoRy Network) introduced by Miikkulainen (1993). The DISCERN system reads narratives (in the form of stories), identifies the corresponding script and identifies the script events in the narrative. It identifies the participants corresponding to the script, and the script participants not mentioned in the story can be inferred. It can answer questions related to the story and also generate expanded paraphrases of the original story.

The DISCERN system processes a story word by word sequentially. The architecture of the system is shown in Figure 3.6. The "lexicon" module stores distributed representations for different words and concepts. The "sentence parser" module processes an input sentence word by word and forms an internal representation of the sentence. The "story parser" module composes internal representations of the sentences to get an internal representation of the story. This representation is stored in the episodic memory module. The "story generator" module generates sentence paraphrases one at a time from the internal representations in the episodic memory. The "episodic

[6]Holographic representation refers to the fact that a partial part of the representation can be used to reconstruct the whole.

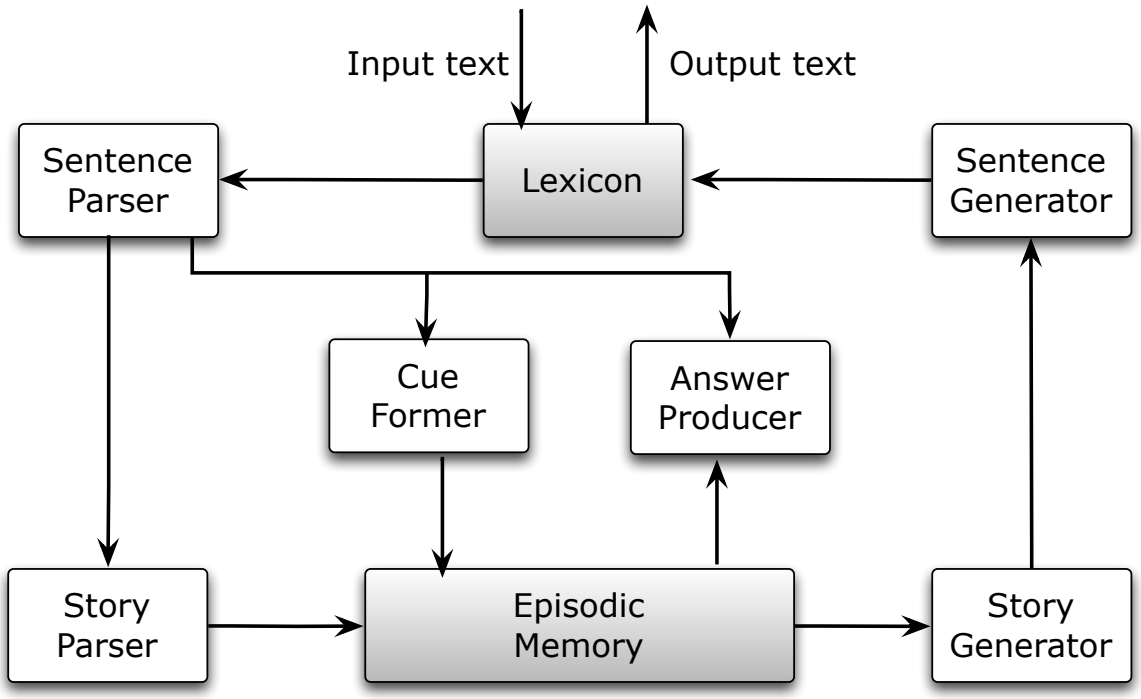

Fig. 3.6 Architecture of the DISCERN system (Miikkulainen, 1993)

memory" module is a hierarchical self-organized map (SOM) network. Representation storage in the episodic memory refers to the pattern of weight activities in the SOM network. The "sentence generator" module generates corresponding words for each of the sentences. The "cue former" and "answer producer" modules are used for retrieving information and answering questions related to the script story. The cue former module produces a cue pattern which is used to retrieve the correct story representation. The answer module is used to generate answer representations for questions asked about the story. The answer representations are input to the sentence generator module, which in turn generates an answer word by word.

All the modules are trained separately and subsequently trained jointly as well, in order to fine tune the modules. We will not go into details of the system and each of its modules; interested readers are pointed to Miikkulainen (1993) for details. Although the DISCERN system shows some degree of script understanding and inferencing

capabilities, it is restricted to some narrow scenarios only. It is not entirely clear how well the system generalizes to new scenarios. Perhaps it would require changes to the module and system architecture.

Similar to the DISCERN system, Lee et al. (1992) proposed a neural-network based DYNASTY (DYNAmic STory understanding sYstem) system for processing script oriented stories. They demonstrated the system for a few artificially created short stories with limited linguistic variation. The above-described connectionist systems have been demonstrated in limited scenarios. These systems were created in the early 1990's and also suffered from the lack of sufficient computational power at that time. Nevertheless, these systems show the potential and advantages of such approaches for script processing and understanding.

3.5 Recent Work on Scripts

As described before, some of the early systems in the 1980's used small hand crafted narrative texts for modeling scripts. In contrast, recent times have seen the emergence of large annotated corpora for training natural language understanding systems. On the technique side as well, there has been growth in statistical techniques for understanding and modeling natural language.

There have been renewed interests in modeling script knowledge with the help of available corpora and statistical techniques. Accordingly, Chambers and Jurafsky (2009, 2008) were first to propose count based methods for learning narrative chains and narrative schemas. Narrative chains, like scripts, are sequences of events, but with a common protagonist as one of the participants in each event. A protagonist refers to an entity like a person or an object, who/which is the focus of the narrative. Narrative chains are not scripts in the conventional sense, but may be called *super-scripts* as they involve a sequence of event descriptions from different scripts, with a common

protagonist across all events. For example, a narrative chain corresponding to a president would involve event descriptions (from different scripts) like attend meetings, formulate policies, give speeches, etc. We go into details of the approach of Chambers and Jurafsky (2008) and other related approaches in Chapter 4.

In count-based methods used for script learning, the key idea is to learn co-occurrence counts of events in a large corpus. The co-occurrence counts along with a suitable measure (for example, pointwise mutual information (PMI)) are used to learn the most likely sequence of event descriptions. A number of count based methods have been proposed for script learning, for example Jans et al. (2012); Manshadi et al. (2008); Pichotta and Mooney (2014); Rudinger et al. (2015a). More details about count-based methods and their disadvantages are discussed in detail in Chapter 4.

Regneri et al. (2010) proposed a different approach. They proposed an unsupervised algorithm for learning script structure from crowd-sourced data. They induce a temporal script graph (TSG) for each scenario. Formally, a TSG for a scenario s is a directed graph $G = (V_s, E_s)$, where V_s are nodes of the graph representing events in the scenario and E_s are the directed edges representing the temporal order between events; for example, edge (e_i, e_j) represents that typically, event e_i happens before e_j in the scenario s. A TSG represents the script structure induced from different ESDs for a scenario. An example of a TSG for an EATING IN A FAST FOOD RESTAURANT scenario is shown in Figure 3.7. The algorithm proposed by Regneri et al. (2010) takes a set of ESDs for a scenario as input and outputs a TSG, where each node is a collection of events, which are paraphrases of each other, and directed edges represent the temporal order in the script.

Regneri et al. (2010) collected ESD for 22 scenarios via an Amazon Mechanical Turk (AMT) experiment. In the crowd-sourcing experiment, they asked the participants to describe a scenario in bullet point style, with one event description per point. As

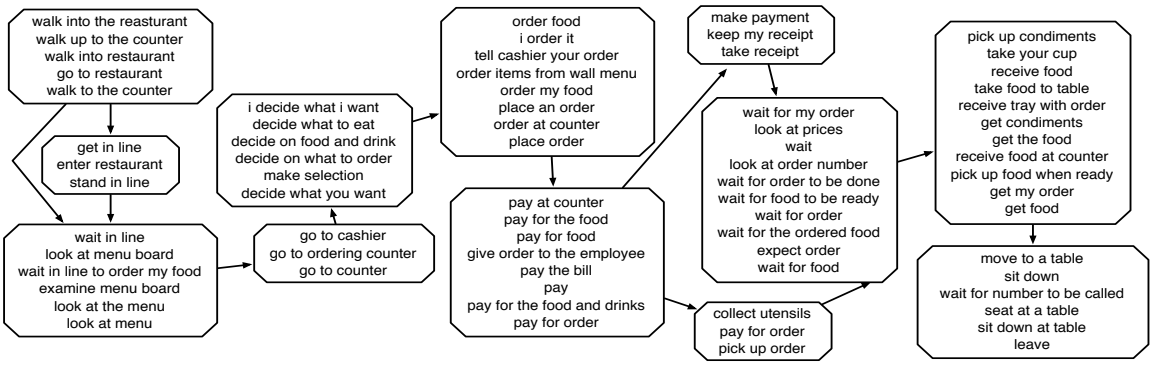

Fig. 3.7 An example of a temporal script graph (TSG) induced from ESDs for EATING IN A FAST FOOD RESTAURANT (Regneri et al., 2010)

explained in Section 3.1, similar events may be verbalized differently across ESDs. Also, the order of events may differ across ESDs, with some events skipped in some ESDs. In order to induce a TSG from crowd-sourced ESDs, Regneri et al. (2010) apply a two step approach. Firstly, they identify similar event descriptions using the multiple sequence alignment (MSA) algorithm. Next, the identified event paraphrases along with certain constraints are used to induce the temporal graph structure for the scenario.

MSA identifies event paraphrases by optimizing a cost function based on semantic similarity between event descriptions. For the similarity measure, Regneri et al. (2010) use a weighted sum of the predicate, subject and object similarity, where each of the component similarities is based on WordNet senses. As the output, MSA gives an initial noisy graph. Next, this graph is automatically post-processed taking into account certain semantic and structural constraints, to obtain a temporal script graph for the scenario.

Frermann et al. (2014) have proposed an unsupervised non-parametric Bayesian model for inducing scripts. They propose a generative model for joint learning of event ordering and event paraphrasing over the crowdsourced data of Regneri et al. (2010).

The model proposed by them clusters similar descriptions into event types and learns a scenario specific temporal ordering over the event types.

In the setup of Frermann et al. (2014), a set of possible event types across all scripts is maintained. The generative story in their approach is as follows: For each ESD, a set of script relevant event types is decided by independently drawing from a binomial distribution. An ordering over event types is drawn from a generalized Mallows model (GMM) (Mallows, 1957). A GMM specifies distribution over permutations and penalizes an ordering which deviates from the canonical ordering. For each realized event type in the generated ordering, a predicate is generated by drawing from a language model. Similarly, a participant type for the event type is selected by drawing from the binomial distribution. For each participant type, a participant word is generated by another language model. Due to limited data, the model is augmented by adding prior knowledge. This is done by including information about the semantically similarity between words (via WordNet) in the priors for the language model. Since the posterior distribution for the model is intractable, the inference is done via collapsed Gibbs sampling (Griffiths and Steyvers, 2004). Unlike Regneri et al. (2010), the model proposed by Frermann et al. (2014) learns event ordering and event paraphrasing jointly.

Regneri et al. (2010) and Frermann et al. (2014) evaluated their model for event-paraphrasing and event-ordering tasks. We will not go into details of these tasks as they are described in detail in Chapter 4, where we compare our probabilistic model with Regneri et al. (2010)'s model.

Most of the A.I. research on modeling common sense knowledge or every day activities has focused in isolation on either Minsky's frames or Fillmore's frames. However, recently, Ferraro and Van Durme (2016) have proposed a model that unifies Minsky's frames and Fillmore's frames into one common framework. They posit their

model on the hierarchical nature of frames as originally proposed by Minsky (1975). In their framework a document is represented at different hierarchical levels, starting from the lowest level, with surface syntactic frames (verb noun structures), followed by surface semantic frames (these are akin to Fillmore's frames). Thematic frames (akin to Minsky's thematic frames, indicative of topic, activities, and setting) are at the next level. At the topmost level are the narrative frames (indicative of plot forms, or stories). A narrative frame invokes different thematic frames. While surface syntactic and semantic frames are localized to a sentence, thematic and narrative frames may span over the whole of the document.

Ferraro and Van Durme (2016) propose a generative model based on Bayesian techniques. In their generative framework, a document is generated by invoking a narrative frame. A narrative frame is a mixture of thematic frames and consequently invokes different thematic frames. Each thematic frame itself is a mixture of semantic frames and each semantic frame invokes the syntactic verb-argument structure. We will not delve into details of the probabilistic formulation and inference mechanism for the model; we refer the reader to Ferraro and Van Durme (2016) for details.

Ferraro and Van Durme (2016) evaluate their model in terms of log-likelihood and topic coherence. They experimentally demonstrate the efficacy of their model in inferring narrative frames and thematic frames. The framework proposed by Ferraro and Van Durme (2016) looks promising but it does not encode the temporal ordering information associated with frames. They claim that their framework can be extended to include ordering information, but it is not entirely clear how exactly one would adapt their framework for modeling script knowledge.

Recently, a number of neural network based approaches have also been proposed that address the shortcomings of count based and graph based methods. For example, in a follow-up work to this dissertation, Pichotta and Mooney (2016a) have proposed

RNN-LSTM (Recurrent Neural Network-Long Short Term Memory) based architecture for modeling an event sequence in a script. Similar to Chambers and Jurafsky (2008) (C&J), they too model super-scripts rather than conventional scripts as described before. In their work, Pichotta and Mooney (2016a) extract event sequences by parsing English Wikipedia. They define an event as a 5-tuple (v, e_s, e_o, e_p, p), where v is the verb lemma, and e_s is the nominal argument for the subject. Similarly, e_o and e_p are nominal arguments for objects and prepositional relations, respectively, and p is the prepositional relation. They consider 4 variants of the basic RNN-LSTM model architecture. Pichotta and Mooney (2016a) evaluated their model on the narrative cloze task. As explained in Chapter 1, the narrative cloze task tests a model's ability to predict a missing event in a sequence of events. They evaluated their model against count-based models using different variants of Recall-At-K (R@K) and accuracy metrics. The Recall-At-K metric evaluates the percentage of missing events that fall under top-K predictions of the model. The proposed LSTM model outperforms count-based methods on all the metrics. Pichotta and Mooney (2016a) also evaluated the response of different models via crowdsourcing. In this experiment, they asked human workers to evaluate the response of a model on a 5-point scale. Human evaluations, too, confirm the superiority of LSTM-based models over count-based baseline models.

In another interesting work, Pichotta and Mooney (2016b) compare the previously proposed event (based on predicate-argument structure) sequence script model with a sentence level language model. Their system works on raw text without any explicit syntactic annotations. Their aim is to study the contribution due to the event structure in the previously proposed models, where events are defined as a verb-arguments tuple. They propose the RNN-LSTM encoder-decoder architecture based model for learning sentence level language models. The model is trained on raw sentences in the text. The input to the model, at the encoder side, is the current sentence, and generated as

an output at the decoder is the subsequent sentence. Events are extracted from the generated sentence. Pichotta and Mooney (2016b) reported that their system, based on extracting events from generated sentences, has performance comparable to models based on predicting structured events.

Other neural network based approaches for script learning have also been proposed (e.g. (Granroth-Wilding and Clark, 2015; Rudinger et al., 2015b)). We will discuss these in detail in Chapter 4. We also describe our neural model for learning script knowledge in Chapter 4.

3.6 Conclusion

This chapter discussed in detail script structure and motivations for research on scripts. It traced back the history of script theory research and its origins in cognitive psychology. It also briefly described the computational efforts in the area of script learning. In the subsequent chapter, we will delve deeper into computational approaches for modeling script knowledge. We describe in detail the proposed neural network model for learning script event sequence order. Experiments show the advantages of using a neural network based model for script processing. In Chapter 6 of the thesis, we will describe our approach for studying the cognitive aspects of scripts. In particular, we describe our experiments to study the effect of script knowledge on language comprehension.

CHAPTER 4

Neural Models of Script Knowledge

All models are wrong, but some are useful.

**

George E. P. Box

4.1 Introduction

In the previous chapter, we introduced the concept of scripts. We explained the role of scripts in a natural language understanding system. We also described that early work on scripts focused on structured representations of this knowledge (Schank and Abelson, 1977)) and manual construction of script knowledge bases. However, these approaches do not scale to complex domains (Gordon, 2001; Mueller, 1998).

In this chapter, we describe an alternate approach. We propose a neural network based probabilistic model for learning event ordering in scripts in Section 4.2. As described in Section 4.3.2, the confidence scores associated with ordering of events can be used for event paraphrasing as well. Experiments show the advantages of the model over existing count based and graph based approaches. We address another important task associated with script learning, namely event prediction. We extend our neural network based ordering model for predicting a missing event in an event sequence. We show the advantages of this approach in Section 4.7.4.

4.2 Event Ordering Script Model

Recently, automatic induction of script knowledge from text has started to attract attention: these methods exploit either natural texts (Chambers and Jurafsky, 2009, 2008) or crowdsourced data (Regneri et al., 2010), and, consequently, do not require expensive expert annotation. Given a text corpus, they extract structured representations (i.e. graphs), for example, chains (Chambers and Jurafsky, 2008) or more general directed acyclic graphs (Regneri et al., 2010). These graphs are scenario-specific; nodes in them correspond to events (and associated sets of potential event mentions) and arcs encode the temporal precedence relation. These graphs can then be used to inform NLP applications (e.g., question answering) by providing information on whether one

event is likely to precede or succeed another. Note that these graphs encode common sense knowledge about the prototypical ordering of events rather than temporal order of events as described in a given text.

Though representing the script knowledge as graphs is attractive from the human interpretability perspective, it may not be optimal from the application point of view. More specifically, these representations (1) require a model designer to choose an appropriate granularity of event mentions (e.g., whether nodes in the graph should be associated only with verbs, or also their arguments); (2) do not provide a mechanism for deciding which scenario applies in a given discourse context and (3) often do not associate confidence levels with information encoded in the graph (e.g., the precedence relation in Regneri et al. (2010)).

Instead of constructing a graph and using it to provide information (e.g., prototypical event ordering) to NLP applications, we advocate for constructing a statistical model which is capable to "answer" at least some of the questions these graphs can be used to answer, but doing this without explicitly representing the knowledge as a graph. In our method, the distributed representations (i.e. vectors of real numbers) of event realizations are computed based on distributed representations of predicates and their arguments, and then the event representations are used in a ranker to predict the prototypical ordering of events. Both the parameters of the compositional process for computing the event representation and the ranking component of the model are estimated from texts (either relying on unambiguous discourse clues or natural ordering in the text). In this way, we build on recent research on compositional distributional semantics (Baroni and Zamparelli, 2011; Socher et al., 2012), though our approach specifically focuses on embedding predicate-argument structures rather than arbitrary phrases, and learning these representations to be especially informative for prototypical event ordering.

In order to get an intuition on why the embedding approach may be attractive, consider a situation where a prototypical ordering of events *the bus disembarked passengers* and *the bus drove away* needs to be predicted. An approach based on the frequency of predicate pairs (Chambers and Jurafsky, 2008) (henceforth CJ08), is unlikely to make a right prediction, as driving usually precedes disembarking. Similarly, an approach which treats the whole predicate-argument structure as an atomic unit (Regneri et al., 2010) will probably fail as well, as such a sparse model is unlikely to be effectively learnable even from large amounts of unlabeled data. However, our embedding method would be expected to capture relevant features of the verb frames, namely the transitive use for the predicate *disembark* and the effect of the particle *away*, and these features will then be used by the ranking component to make the correct prediction.

In previous work on learning inference rules (Berant et al., 2011), it has been shown that enforcing transitivity constraints on the inference rules results in significantly improved performance. The same is likely to be true for the event ordering task, as scripts have largely linear structure, and observing that $a \prec b$ and $b \prec c$ is likely to imply $a \prec c$. Interestingly, in our approach, we learn the model which satisfies transitivity constraints, without the need for any explicit global optimization on a graph. This results in a significant boost in performance when using embeddings of just predicates (i.e. ignoring arguments) with respect to using frequencies of ordered verb pairs, as in CJ08 (76% vs. 61% on the natural data).

Our model is solely focusing on the ordering task and admittedly does not represent all the information encoded by a script graph structure. For example, it cannot be directly used to predict a missing event given a set of events (the narrative cloze task (Chambers and Jurafsky, 2009)). Nevertheless, we believe that the framework (a probabilistic model using event embeddings as its component) can be extended

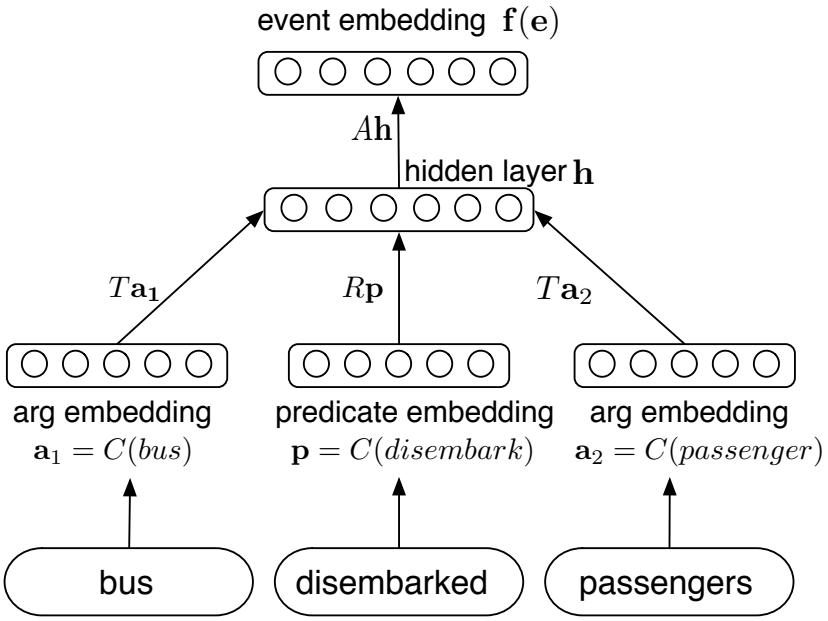

Fig. 4.1 Computation of an event representation for a predicate with two arguments (*the bus disembarked passengers*), an arbitrary number of arguments is supported by our approach.

to represent other aspects of script knowledge by modifying the learning objective, as described in the later part of the chapter for the narrative cloze task. In later sections, we show how our model can be used to predict if two event mentions are likely paraphrases of the same event.

The approach is evaluated in two set-ups. First, we consider the crowdsourced dataset of Regneri et al. (2010) and demonstrate that using our model results in the 13.5% absolute improvement in $F1$ on event ordering with respect to their graph induction method (84.1% vs. 70.6%). Secondly, we derive an event ordering dataset from the Gigaword corpus, where we also show that the embedding method beats the frequency-based baseline (i.e. reimplementation of the scoring component of CJ08) by 22.8% in accuracy (83.5% vs. 60.7%).

In this section, we describe the model we use for computing event representations as well as the ranking component of our model.

4.2.1 Event Representation

Learning and exploiting distributed word representations (i.e. vectors of real values, also known as *embeddings*) have been shown to be beneficial in many NLP applications (Bengio et al., 2001; Collobert et al., 2011; Turian et al., 2010). These representations encode semantic and syntactic properties of a word, and are normally learned in the language modeling setting (i.e. learned to be predictive of local word context), though they can also be specialized by learning in the context of other NLP applications such as PoS tagging or semantic role labeling (Collobert et al., 2011). More recently, the area of distributional compositional semantics has started to emerge (Baroni and Zamparelli, 2011; Socher et al., 2012); they focus on inducing representations of phrases by learning a compositional model. Such a model would compute a representation of a phrase by starting with embeddings of individual words in the phrase, often this composition process is recursive and guided by some form of syntactic structure.

In our work, we use a simple compositional model for representing the semantics of a verb frame e (i.e. the predicate and its arguments). We will refer to such verb frames as events. The model is shown in Figure 4.6. Each word c_i in the vocabulary is mapped to a real vector based on the corresponding lemma (the embedding function C). The hidden layer is computed by summing linearly transformed predicate and argument[1] embeddings and passing it through the logistic sigmoid function. We use different transformation matrices for arguments and predicates, T and R, respectively. The event representation $\boldsymbol{f}(e)$ is then obtained by applying another linear transform (matrix A) followed by another application of the sigmoid function. Another point to note in here is that, as in previous work on script induction, we use lemmas for predicates and specifically filter out any tense markers, as our goal is to induce common

[1]Only syntactic heads of arguments are used in this work. If an argument is *a coffee maker*, we will use only the word *maker*.

sense knowledge about an event rather than properties predictive of temporal order in a specific discourse context.

These event representations are learned in the context of event ranking: the transformation parameters, as well as representations of words, are forced to be predictive of the temporal order of events. In our experiments, we also consider initialization of predicate and arguments with the SENNA word embeddings (Collobert et al., 2011).

4.2.2 Learning to Order

The task of learning the stereotyped order of events naturally corresponds to the standard ranking setting. We assume that we are provided with sequences of events, and our goal is to capture this order. We discuss how we obtain this learning material in the next section. We learn a linear ranker (characterized by a vector $\boldsymbol{w}$) which takes an event representation and returns a ranking score. Events are then ordered according to the score to yield the model prediction. Note that during the learning stage we estimate not only $\boldsymbol{w}$ but also the event representation parameters, i.e. matrices T, R and A, and the word embedding C. Note that by casting the event ordering task as a global ranking problem we ensure that the model implicitly exploits transitivity of the relation, the property which is crucial for successful learning from a finite amount of data, as we argued in the introduction and will confirm in our experiments.

At training time, we assume that each training example k is a list of events $e_1^{(k)}, \ldots, e_{n^{(k)}}^{(k)}$ provided in the stereotypical order (i.e. $e_i^{(k)} \prec e_j^{(k)}$ if $i < j$); $n^{(k)}$ is the length of the list k. We minimize the L_2-regularized ranking hinge loss:

$$\sum_k \sum_{i<j\leq n^{(k)}} \max(0, 1 - \mathbf{w}^T \boldsymbol{f}(e_i^{(k)}; \Theta) + \mathbf{w}^T \boldsymbol{f}(e_j^{(k)}; \Theta)) + \alpha(\|\mathbf{w}\|_2 + \|\Theta\|_2) \qquad (4.1)$$

where $\boldsymbol{f}(e;\Theta)$ is the embedding computed for event e and Θ are all embedding parameters corresponding to elements of the matrices $\{R, C, T, A\}$. We use stochastic gradient descent, gradients w.r.t. Θ are computed using back propagation.

4.3 Event Ordering Experiments

We evaluate our approach in two different set-ups. First, we induce the model from the crowdsourced data specifically collected for script induction by Regneri et al. (2010); secondly, we consider an arguably more challenging set-up of learning the model from news data (Gigaword (Parker et al., 2011)); in the latter case we use a learning scenario inspired by Chambers and Jurafsky (2008).[2]

4.3.1 Learning from Crowdsourced Data

Data and task

Regneri et al. (2010) collected descriptions (called *event sequence descriptions, ESDs*) of various types of human activities (e.g., going to a restaurant, ironing clothes) using crowdsourcing (Amazon Mechanical Turk); this dataset was also complemented by descriptions provided in the OMICS corpus (Gupta and Kochenderfer, 2004). The datasets are fairly small, containing 30 ESDs per activity type on average (we will refer to different activities as *scenarios*), but in principle, the collection can easily be extended given the low cost of crowdsourcing. The ESDs list events forming the scenario and are written in a bullet-point style. The annotators were asked to follow the prototypical event order in writing. As an example, consider an ESD for the scenario *prepare coffee* :

[2]Details about downloading the data and models are at:
http://www.coli.uni-saarland.de/projects/smile/docs/nmReadme.txt

{go to coffee maker} → *{fill water in coffee maker}* → *{place the filter in holder}* → *{place coffee in filter}* → *{place holder in coffee maker}* → *{turn on coffee maker}*

Regneri et al. also automatically extracted predicates and heads of arguments for each event, as needed for their MSA system and our compositional model.

Though individual ESDs may seem simple, the learning task is challenging because of the limited amount of training data, variability in the used vocabulary, optionality of events (e.g., going to the coffee machine may not be mentioned in an ESD), different granularity of events and variability in the ordering (e.g., coffee may be put in the filter before placing it in the coffee maker). Unlike our work, Regneri et al. (2010) rely on WordNet to provide the extra signal when using the multiple sequence alignment (MSA) algorithm. As in their work, each description was preprocessed to extract a predicate and heads of argument noun phrases to be used in the model.

The methods are evaluated on human annotated scenario-specific tests: the goal is to classify event pairs as appearing in a stereotypical order or not (Regneri et al., 2010).[3]

The model was estimated as explained in Section 4.2.2 with the order of events in ESDs treated as gold standard. We used 4 held-out scenarios to choose model parameters, no scenario-specific tuning was performed and the 10 test scripts were not used to perform model selection. The selected model used the dimensionality of 10 for an event and word embeddings. The initial learning rate and the regularization parameter were set to 0.005 and 1.0, respectively, and both parameters were reduced by a factor of 1.2 every epoch the error function went up. We used 2000 epochs of stochastic gradient descent. Dropout (Hinton et al., 2012) with a rate of 20% was used for the hidden layers in all our experiments. When testing, we predicted that the event

[3]The event pairs are not coming from the same ESDs, making the task harder, as the events may not be in any temporal relation.

pair (e_1,e_2) is in the stereotypical order ($e_1 \prec e_2$) if the ranking score for e_1 exceeded the ranking score for e_2.

Results and discussion

We evaluated our event embedding model (*EE*) against baseline systems (*BL*, *MSA*, and *BS*). MSA is the system of Regneri et al. (2010). BS is a hierarchical Bayesian model by Frermann et al. (2014). BL chooses the order of events based on the preferred order of the corresponding verbs in the training set: (e_1, e_2) is predicted to be in the stereotypical order if the number of times the corresponding verbs v_1 and v_2 appear in this order in the training ESDs exceeds the number of times they appear in the opposite order (not necessary at adjacent positions); a coin is tossed to break ties (or if v_1 and v_2 are the same verb). This frequency counting method was previously used in CJ08.[4]

We also compare to the version of our model which uses only verbs (EE_{verbs}). Note that EE_{verbs} is conceptually very similar to BL, as it essentially induces an ordering over verbs. However, this ordering can benefit from the implicit transitivity assumption used in EE_{verbs} (and EE), as we discussed in the introduction. The results are presented in Table 4.1 on page 67.

The first observation is that the full model improves substantially over the baseline and the previous method (MSA) in F1 (13.5% improvement over MSA and 6.5% improvement over BS). Note also that this improvement is consistent across scenarios: EE outperforms MSA and BS on 9 scenarios out of 10 and 8 out of 10 scenarios in the case of BS. Unlike MSA and BS, no external knowledge (i.e. WordNet) was exploited in our method.

[4]They scored permutations of several events by summing the logarithmed differences of the frequencies of ordered verb pairs. However, when applied to event pairs, their approach would yield exactly the same prediction rule as BL.

We also observe a substantial improvement in all metrics from using transitivity, as seen by comparing the results of BL and EE_{verb} (11% improvement in F1). This simple approach already substantially outperforms the pipelined MSA system. These results seem to support our hypothesis in the introduction that inducing graph representations from scripts may not be an optimal strategy from the practical perspective.

We performed additional experiments using the SENNA embeddings (Collobert et al., 2011). Instead of randomly initializing arguments and predicate embeddings (vectors), we initialized them with pre-trained SENNA embeddings. We have not observed any significant boost in performance from using the initialization (average F1 of 84.0% for EE). We attribute the lack of significant improvement to the following three factors. First of all, the SENNA embeddings tend to place antonyms / opposites near each other (e.g., come and go, or end and start). However, 'opposite' predicates appear in very different positions in scripts. Additionally, the SENNA embeddings have a dimensionality of 50, which appears to be too high for small crowd-sourced datasets, as it forces us to use larger matrices T and R. Moreover, the SENNA embeddings are estimated from Wikipedia, and the activities in our crowdsourced domain are perhaps underrepresented there.

	Precision (%)					**Recall (%)**					**F1 (%)**				
	BL	EE_{verb}	MSA	BS	EE	BL	EE_{verb}	MSA	BS	EE	BL	EE_{verb}	MSA	BS	EE
Bus	70.1	81.9	80.0	76.0	85.1	71.3	75.8	80.0	76.0	91.9	70.7	78.8	80.0	76.0	88.4
Coffee	70.1	73.7	70.0	68.0	69.5	72.6	75.1	78.0	57.0	71.0	71.3	74.4	74.0	62.0	70.2
Fastfood	69.9	81.0	53.0	97.0	90.0	65.1	79.1	81.0	65.0	87.9	67.4	80.0	64.0	78.0	88.9
Return	74.0	94.1	48.0	87.0	92.4	68.6	91.4	75.0	72.0	89.7	71.0	92.8	58.0	79.0	91.0
Iron	73.4	80.1	78.0	87.0	86.9	67.3	69.8	72.0	69.0	80.2	70.2	69.8	75.0	77.0	83.4
Microw.	72.6	79.2	47.0	91.0	82.9	63.4	62.8	83.0	74.0	90.3	67.7	70.0	60.0	82.0	86.4
Eggs	72.7	71.4	67.0	77.0	80.7	68.0	67.7	64.0	59.0	76.9	70.3	69.5	66.0	67.0	78.7
Shower	62.2	76.2	48.0	85.0	80.0	62.5	80.0	82.0	84.0	84.3	62.3	78.1	61.0	85.0	82.1
Phone	67.6	87.8	83.0	92.0	87.5	62.8	87.9	86.0	87.0	89.0	65.1	87.8	84.0	89.0	88.2
Vending	66.4	87.3	84.0	90.0	84.2	60.6	87.6	85.0	74.0	81.9	63.3	84.9	84.0	81.0	88.2
Average	69.9	81.3	65.8	85.0	**83.9**	66.2	77.2	78.6	71.7	**84.3**	68.0	79.1	70.6	77.6	**84.1**

Table 4.1 Results on the crowdsourced data for the verb-frequency baseline (BL), the verb-only embedding model (EE_{verb}), Regneri et al. (2010) (MSA), Frermann et al. (2014)(BS) and the full model (EE).

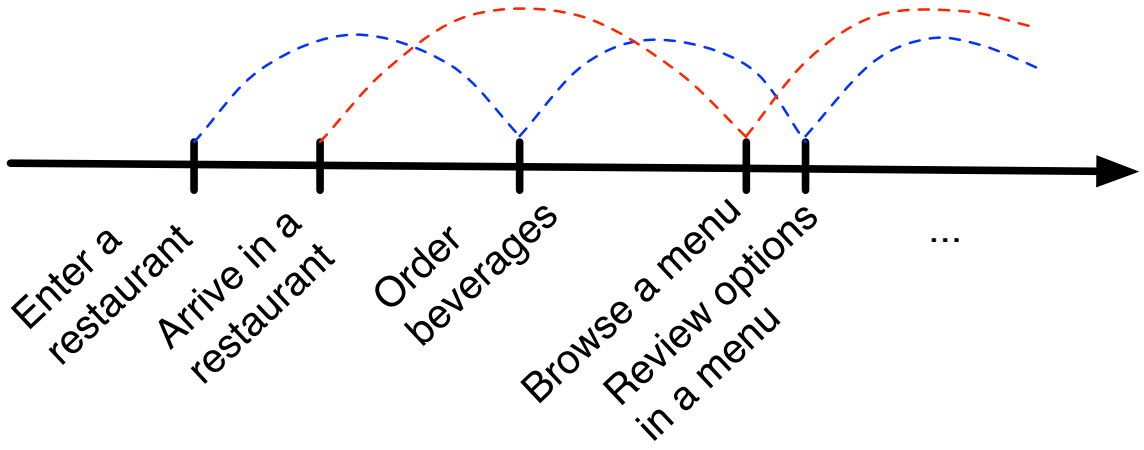

Fig. 4.2 Events on the timeline; dotted arcs link events from the same ESD.

4.3.2 Event Paraphrasing

Regneri et al. (2010) additionally measure paraphrasing performance of the MSA system by comparing it to human annotation they obtained: a system needs to predict if a pair of event mentions are paraphrases or not. The dataset contains 527 event pairs for the 10 test scenarios. Each pair consists of events from the same scenario. The dataset is fairly balanced, containing from 47 to 60 examples per scenario.

This task does not directly map to any statistical inference problem with our model. Instead, we use an approach inspired by the interval algebra of Allen (1983).

Our ranking model maps event mentions to positions on a timeline (see Figure 4.2). However, it would be more natural to assume that events are intervals rather than points. In principle, these intervals can be overlapping, to encode a rich set of temporal relations (see (Allen, 1983)). However, we make a simplifying assumption that the intervals do not overlap and every real number belongs to an interval. In other words, our goal is to induce a segmentation of the line: event mentions corresponding to the same interval are then regarded as paraphrases.

One natural constraint on this segmentation is the following: if two event mentions are from the same training ESD, they cannot be assigned to the same interval (as events in ESD are not supposed to be paraphrases). In Figure 4.2 arcs link event mentions from the same ESD. We look for a segmentation which produces the minimal number of segments and satisfies the above constraint for event mentions appearing in training data.

Though inducing intervals given a set of temporal constraints is known to be NP-hard in general (see, e.g., (Golumbic and Shamir, 1993)), for our constraints a simple greedy algorithm finds an optimal solution. We trace the line from the left maintaining a set of event mentions in the current unfinished interval and create a boundary when the constraint is violated; we repeat the process until we have processed all mentions. In Figure 4.2, we would create the first boundary between *arrive in a restaurant* and *order beverages*: *order beverages* and *enter a restaurant* are from the same ESD and continuing the interval would violate the constraint. It is not hard to see that this results in an optimal segmentation. First, the segmentation satisfies the constraint by construction. Secondly, the number of segments is minimal as the arcs which caused boundary creation are non-overlapping; each of these arcs needs to be cut and our algorithm cuts each arc exactly once.

This algorithm prefers to introduce a boundary as late as possible. For example, it would introduce a boundary between *browse a menu* and *review options in a menu* even though the corresponding points are very close on the line. We modify the algorithm by moving the boundaries left as long as this move does not result in new constraint violations and increases margins at boundaries. In our example, the boundary would be moved to be between *order beverages* and *browse a menu*, as desired.

The resulting performance is reported in Table 4.2. We report results of our method, as well as results for MSA, BS and a simple all-paraphrase baseline which predict

Scenario	**F1 (%)**			
	APBL	MSA	BS	EE
Take bus	53.7	74.0	47.0	63.5
Make coffee	42.1	65.0	52.0	63.5
Order fastfood	37.0	59.0	80.0	62.6
Return food back	64.8	71.0	67.0	81.1
Iron clothes	43.3	67.0	60.0	56.7
Microwave cooking	43.2	75.0	82.0	57.8
Scrambled eggs	57.6	69.0	76.0	53.0
Take shower	42.1	78.0	67.0	55.7
Answer telephone	71.0	89.0	81.0	79.4
Vending machine	56.1	69.0	77.0	69.3
Average	51.1	**71.6**	68.9	64.5

Table 4.2 Paraphrasing results on the crowdsourced data for Regneri et al. (2010) (MSA), Frermann et al. (2014)(BS) and the all-paraphrase baseline (APBL) and using intervals induced from our model (EE).

that all mention pairs in a test set are paraphrases (APBL)[5] We can see that interval induction technique results in a lower F1 than that of MSA or BS. This might be partially due to not using external knowledge (WordNet) in our method.

We performed extra analyses on the development scenario *doorbell*. The analyses revealed that the interval induction approach is not very robust to noise: removing a single noisy ESD results in a dramatic change in the interval structure induced and in a significant increase of F1. Consequently, soft versions of the constraint would be beneficial. Alternatively, event embeddings (i.e. continuous vectors) can be clustered directly. We leave this investigation for future work.

4.3.3 Learning from Natural Text

In the second set of experiments, we consider a more challenging problem, inducing knowledge about the stereotyped ordering of events from natural texts. In this work,

[5]The results for the random baseline are lower: F1 of 40.6% on average.

we are largely inspired by the scenario of CJ08. The overall strategy is the following: we process the Gigaword corpus with a high precision rule-based temporal classifier relying on explicit clues (e.g., "then", "after") to get ordered pairs of events and then we train our model on these pairs (note that clues used by the classifier are removed from the examples, so the model has to rely on verbs and their arguments). Conceptually, the difference between our approach and CJ08 is in using a different temporal classifier, not enforcing that event pairs have the same protagonist, and learning an event embedding model instead of scoring event sequences based on verb-pair frequencies.

We also evaluate our system on examples extracted using the same temporal classifier (but validated manually), which allows us to use much larger test set, and, consequently, provide more detailed and reliable error analysis.

Data and task

The Gigaword corpus consists of news data from different news agencies and newspapers. For testing and development, we took the AFP (Agence France-Presse) section, as it appeared most different from the rest when comparing sets of extracted event pairs (other sections correspond mostly to US agencies). The AFP section was not used for training. This selection strategy was chosen to create a negative bias for our model which is more expressive than the baseline methods and, consequently, better at memorizing examples.

As a rule-based temporal classifier, we used high precision "happens-before" rules from the VerbOcean system (Chklovski and Pantel, 2004). Consider "*to* ⟨*verb-x*⟩ *and then* ⟨*verb-y*⟩" as one example of such rules. We used predicted collapsed Stanford dependencies (de Marneffe et al., 2006) to extract arguments of the verbs, and used only a subset of dependents of a verb.[6] This preprocessing ensured that (1) clues which

[6]The list of dependencies not considered: *aux*, *auxpass*, *attr*, *appos*, *cc*, *conj*, *complm*, *cop*, *dep*, *det*, *punct*, *mwe*.

	Accuracy (%)
BL	60.7
CJ08	60.1
EE_{verb}	75.9
EE	**83.5**

Table 4.3 Results on the Gigaword data for the verb-frequency baseline (BL), the verb-only embedding model (EE_{verb}), the full model (EE) and CJ08 rules.

form part of a pattern are not observable by our model both at train and test time; (2) there is no systematic difference between both events (e.g., for collapsed dependencies, the noun subject is attached to both verbs even if the verbs are conjoined); and (3) no information about the order of events in the text is available to the models. Applying these rules resulted in 22,446 event pairs for training, and we split an additional 1,015 pairs from the AFP section into 812 for final testing and 203 for development. We manually validated a random 50 examples and all 50 of them followed the correct temporal order, so we chose not to hand correct the test set.

We largely followed the same training and evaluation regime as for the crowdsourced data. We set the regularization parameter and the learning rate to 0.01 and $5.e-4$ respectively. The model was trained for 600 epochs. The embedding sizes were 30 and 50 dimensions for words and events, respectively.

Results and discussion

In our experiments, as before, we use BL as a baseline and EE_{verb} as a verb only simplified version of our approach. We used another baseline consisting of the verb pair ordering counts provided by Chambers and Jurafsky (2008).[7] We refer this baseline as

[7]These verb pair frequency counts are available at www.usna.edu/Users/cs/nchamber/data/schemas/acl09/verb-pair-orders.gz

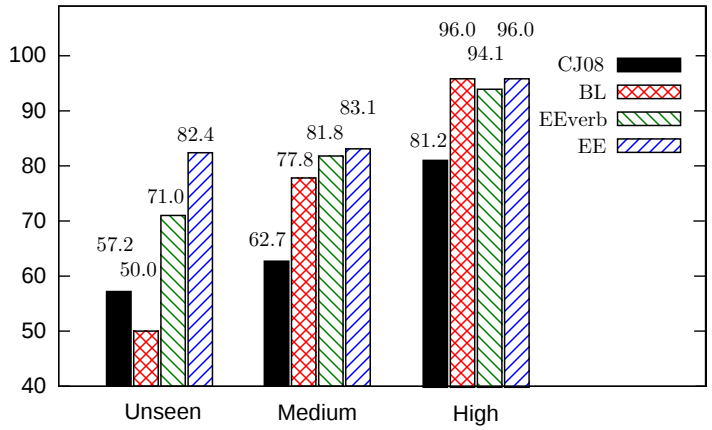

Fig. 4.3 Results for different frequency bands: unseen, medium frequency (between 1 and 10) and high frequency (> 10) verb pairs.

CJ08. Note also that BL can be regarded as a reimplementation of CJ08 but with a different temporal classifier. We report results in Table 4.3.

The observations are largely the same as before: (1) the full model substantially outperforms all other approaches (p-level < 0.001 with the permutation test); (2) enforcing transitivity is very helpful (75.9 % for EE_{verb} vs. 60.1% for BL). Surprisingly CJ08 rules produce as good results as BL, suggesting that maybe our learning set-ups are not that different.

However, an interesting question is in which situations using a more expressive model, EE, is beneficial. If these accuracy gains have to do with memorizing the data, it may not generalize well to other domains or datasets. In order to test this hypothesis, we divided the test examples into three frequency bands according to the frequency of the corresponding verb pairs in the training set (total, in both orders). There are 513, 249 and 50 event pairs in the bands corresponding to unseen pairs of verbs, frequency ≤ 10 and frequency > 10, respectively. These counts emphasize that correct predictions on unseen pairs are crucial and these are exactly where BL would be equivalent to a random guess. Also, this suggests, even before looking into the

results, that memorization is irrelevant. The results for BL, CJ08, EE_{verb} and EE are shown in Figure 4.3.

One observation is that most gains for EE and EE_{verb} are due to an improvement on unseen pairs. This is fairly natural, as both transitivity and information about arguments are the only sources of information available. In this context it is important to note that some of the verbs are *light*, in the sense that they have little semantic content of their own (e.g., *take*, *get*) and the event semantics can only be derived from analyzing their arguments (e.g., *take an exam* vs. *take a detour*). On the high frequency verb pairs, all systems perform equally well, except for CJ08, as it was estimated from somewhat different data.

In order to understand how transitivity works, we considered a few unseen predicate pairs where the EE_{verb} model was correctly predicting their order. For many of these pairs, there were no inference chains of length 2 (e.g., chain of length 2 was found for the pair *accept* $\prec$ *carry*: *accept* $\prec$ *get* and *get* $\prec$ *carry* but not many other pairs). This observation suggests that our model captures some non-trivial transitivity rules.

4.4 Count Based Event Prediction

In the previous sections, we described approaches to script event ordering and event paraphrasing. Now, we focus our attention on event prediction models. We describe the previous work on event prediction and then later, we describe our model (an extension of our event ordering model) for event prediction.

Much of the previous work on event prediction has focused on count-based techniques using, for example, either the generative framework (Frermann et al., 2014) or relying on information-theoretic measures such as pointwise mutual information (PMI) (Chambers and Jurafsky, 2008). Some of these techniques treat predicate-argument structures as an atomic whole (e.g., Pichotta and Mooney (2014)); in other words,

their probability estimates are based on co-occurrences of entire (predicate, arguments) tuples. Clearly, such methods fail to adequately take into account the compositional nature of expressions used to refer to events and suffer from data sparsity.

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

4.4.1 Background

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

using a narrative cloze test as explained in Section 4.5. In the cloze test, an event is held-out from an event sequence and the task is to predict the missing event.

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

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

Nupur cooked spaghetti for dinner. Later, Nupur ate dinner with her husband. After dinner, Nupur took the dog for a walk. After 30 minutes, Nupur came home. After a while, Nupur slept on the bed.

Fig. 4.4 A small narrative text

Sparsity issues can result in failure of these models. This can be exemplified as follows. Suppose the text shown in Figure 4.4 is observed during model training.

The event sequence (script) corresponding to the above story is:

```
cook(nupur,spaghetti,dinner)→ eat(nupur,dinner,husband)→
take(nupur,dog,walk)→come(nupur,home)→sleep(nupur,bed)
```

Suppose during testing the following event sequence is observed :

```
prepared(jenny,pasta,dinner)→eat(jenny,dinner,boyfriend)→
take(jenny,pet,walk)→ ?  →sleep(jenny, couch)
```

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

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

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

4.5 Event Ordering Tasks Definition

One of the standard tasks used for evaluating event prediction in scripts is narrative cloze (Chambers and Jurafsky, 2008; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015a). The origins of narrative cloze lie in psychology, where it was used to assess a child's ability to fill in a missing word in a sentence (Taylor, 1953). In our setting the cloze task is described as follows: given a sequence of events with an event held-out from the sequence, guess the removed event. For example, given the following sequence, predict the event that should be at the position marked by ?:

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

The narrative cloze task evaluates models for the exact correctness of the prediction. It penalizes predictions even if they are semantically plausible. It would be more realistic to evaluate script models on a task that gives credit for predicting semantically plausible alternatives as well. We propose the *adversarial narrative cloze* task. In this task, the model is presented with two event sequences: one is the correct event

sequence and another is the same sequence but with one event replaced by a random event. The task is to guess which of the two is the correct event sequence. For example, given two sequences below, the model should be able to distinguish the correct event sequence from the incorrect one.

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

Interestingly, Manshadi et al. (2008) also propose a similar task for evaluating event based language model and they refer to it as event ordering task. As explained in Section 4.7, we evaluate our model on both the tasks: narrative cloze and adversarial narrative cloze.

4.6 Event Prediction Script Model

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

As in previous work, (Chambers and Jurafsky, 2008; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015a), each event in a sequence of events has a

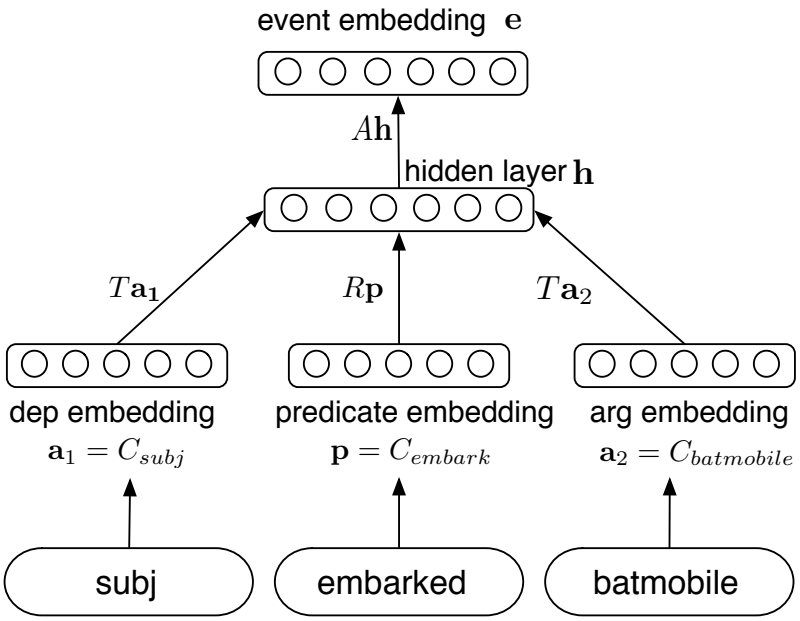

Fig. 4.5 Computation of an event representation for a predicate with dependency and an argument (*subj (batman) embarked batmobile*), an arbitrary number of arguments is supported by our approach.

common entity (the protagonist) as one of the arguments. We represent an event as a tuple $v(d, a^{(1)}, a^{(2)})$ where v is the verbal predicate, d is the position (subj, obj or prep) of the protagonist, and $a^{(1)}$ and $a^{(2)}$ are the other dependent arguments of the verb. We marked an absent argument as 'NULL'.

4.6.1 Event Representation

For event representation, one could use a sophisticated compositional model based on recursive neural networks (Socher et al., 2012); we take a simpler approach and choose a feedforward network based compositional model as it is easier to train and more robust to the choice of hyper-parameters. Our event representations model is inspired from the previously described, event ordering model (Modi and Titov, 2014).

The event model is a simple compositional model representing an event. For reference, the model is shown again in Figure 4.5. Given an event, $e = (v, d, a_1, a_2)$,

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

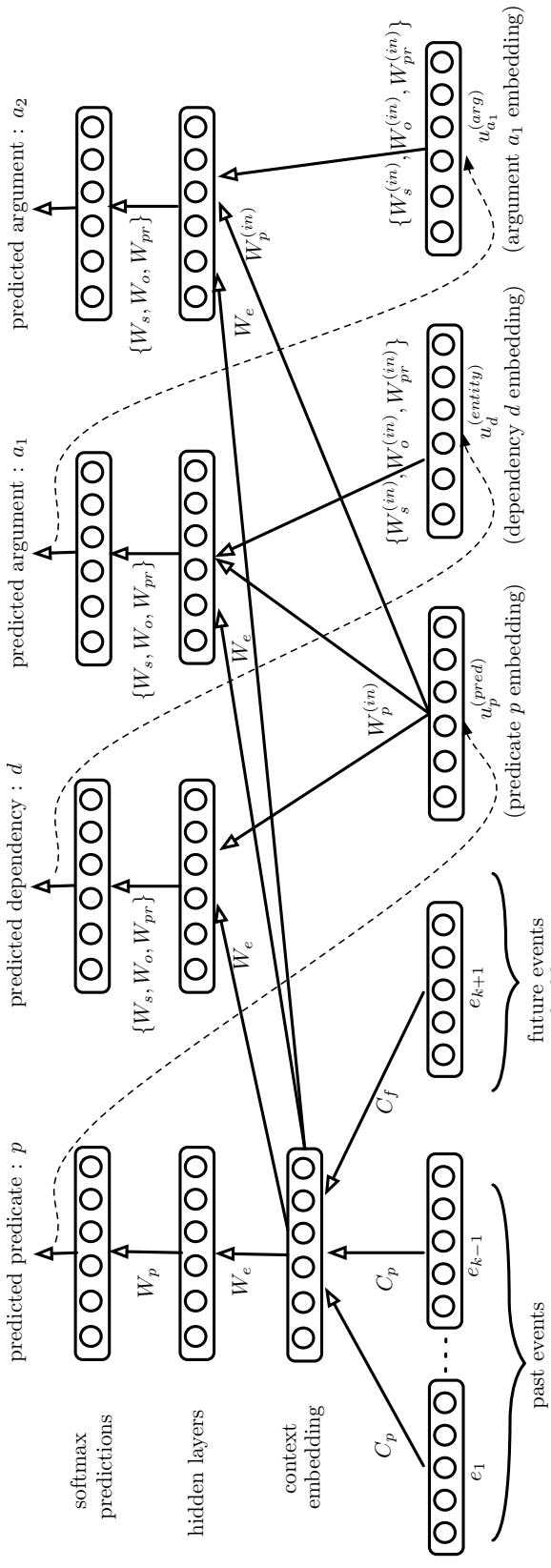

Fig. 4.6 Model for learning event sequences. Here, we are given sequence of events $e_1, e_2, ..., e_{k-1}, e_k, e_{k+1}$. Event e_k is removed from the sequence and it is predicted incrementally.

4.6.2 Event Sequence Model

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

We propose a neural network based probabilistic model for event sequences, for learning event sequences as well as the event representations. The model is shown in Figure 4.6 on the preceding page. The model is trained by predicting a held out event in an event sequence. During training, a window (size = 5 = 3*2 + 1) is moved over all events in each event sequence corresponding to each entity. The event in the window's center is the event to be predicted, and events on the left and right of the window are the context events. As explained earlier, the missing event is predicted incrementally, beginning with a predicate, followed by the protagonist position, followed by other participants in the event.

In order to get an intuition on how our model predicts an event, consider the following event sequence in a script, with a held-out event: $(e_1 \rightarrow e_2 \cdots \rightarrow e_{k-1} \rightarrow ? \rightarrow e_{k+1} \rightarrow \ldots e_n)$. We would like to predict the removed event, say e_k. The event model is used to obtain event representations for each event in the context. These event representations are then composed into a context representation by summing the representation for each of the event in the context. We sum the representations as this formulation works well in practice. The desired event e_k is predicted incrementally, beginning with the predicate p for e_k. The context embedding is used to predict the verbal predicate via a hidden layer followed by a multiclass logistic regression (softmax) classification. Next, the protagonist position d (subject, object etc) is predicted. For predicting d, the context embedding and the predicate embedding (corresponding to the predicate predicted in the previous step) are linearly combined to be given as input to a hidden layer. This is followed by regular softmax prediction. Similarly,

arguments are predicted. For each of the arguments, predicate embedding and the previous prediction (position or argument) are linearly combined with the context embedding. If at each prediction stage we used gold predicate/position/argument embedding for linearly combining with context embedding, our model would not be robust to wrong predictions during testing. Using the embeddings corresponding to the predicted unit would make the model robust against noise and would help the model to partially recover from wrong predictions during testing.

$$\Theta^* = argmin_\Theta - J(\Theta) \tag{4.2}$$

$$\begin{aligned} J(\Theta) &= \prod_{i=1}^{N} \underbrace{p(e_i \mid e_1, ..., e_{i-1}, e_{i+1}, e_k, \Theta)}_{prob.\ of\ an\ event\ given\ context} \\ &= \prod_{i=1}^{N} p(e_i \mid \underbrace{\mathbf{e}}_{\substack{\text{context} \\ \text{events}}}, \Theta) \end{aligned} \tag{4.3}$$

$$\begin{aligned} p(e_i \mid \mathbf{e}, \Theta) &= p(v_i, d_i, a_i^{(1)}, a_i^{(2)} \mid \mathbf{e}, \Theta) \\ &= \underbrace{p(v_i \mid \mathbf{e}, \Theta)}_{verb\ prob.} * \underbrace{p(d_i \mid v_i, \mathbf{e}, \Theta)}_{dependency\ prob.} * \\ &\quad \underbrace{p(a_i^{(1)} \mid v_i, d_i, \mathbf{e}, \Theta)}_{first\ arg\ prob.} * \\ &\quad \underbrace{p(a_i^{(2)} \mid v_i, a_i^{(1)}, \mathbf{e}, \Theta)}_{second\ arg\ prob.} \end{aligned} \tag{4.4}$$

$$p(v_i \mid \mathbf{e}, \Theta) = \frac{exp(u_{v_i}^T(W_p \tanh(W_e E)) + b_{v_i})}{\sum_k exp(u_k^T(W_p \tanh(W_e E)) + b_k)} \tag{4.5}$$

We train the model by minimizing the negative likelihood function for the event prediction. Formally, we minimize the objective function $-J(\Theta)$ as shown in Equations 4.2 and 4.3. As shown in Equation 4.4, we factorize the event distribution into constituents, making appropriate independence assumptions as explained earlier. Each factor is a multiclass logistic regression (softmax) function. Equation 4.5 illustrates

the probability distribution for the predicate given the context. Here, u_{v_i} is the word embedding for the predicate v_i, E is the context embedding and b_{v_i} is the bias.

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

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

4.7 Event Prediction Model Evaluation

4.7.1 Data

There is no standard dataset for evaluating event prediction in scripts. We experimented with a movie summary corpus[8] (Bamman et al., 2014). The corpus was created by extracting 42,306 movie summaries from a November 2012 dump of Wikipedia[9]. Each document in the corpus concisely describes a movie plot along with descriptions of various characters involved in the plot. The average length of a document in the corpus is 176 words. But more popular movies have much more elaborate descriptions going up to a length of 1,000 words. The corpus has been processed by the Stanford Core

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

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

Table 4.4 Data statistics

NLP pipeline (Manning et al., 2014). The texts in the corpus were tokenized and annotated with POS tags, dependencies, NER and coreference (coref) information. Since each of the documents in the corpus is about a movie, the scripts in this corpus involve interesting interactions between different entities (actors/objects). In order to study and explore rich script structure, we selected this corpus for our experiments. Nevertheless, our model is domain agnostic; the experiments performed on this corpus are generalizable to any other corpus as well.

As mentioned in Section 4.4.1, we extract scripts corresponding to each of the entities using the dependency annotations and coref information. The corpus documents are divided randomly into three parts: train (~70%), development (~10%) and test (~20%). Data statistics about the dataset are given in Table 4.4. As a preprocessing step, low frequency (< 100) predicates and arguments are mapped to a special UNK (unknown) symbol. Similarly, arguments consisting of only digits are mapped to a NUMB (number) symbol. There are 703 unique predicate lemmas and $46,644$ unique argument lemmas in the train set. The average length of a script is 10 events. During testing, predicate and arguments not observed during training are mapped to the same UNK symbol.

4.7.2 Baselines Systems

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

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

MultiProtagonist (M-Pr) is the model proposed by Pichotta and Mooney (2014) and described as a *joint* model. The model calculates the conditional probability of an event given another context event ($P(e_2 \mid e_1\ )$ by counting the co-occurrence counts of the events in the corpus. The model predicts the missing event given the context events by maximizing the sum of log conditional probabilities of an event w.r.t each of the context events, i.e. $e^* = \text{argmax}_e \sum_{i=1}^{k-1} log\ P(e \mid e_i) + \sum_{i=k+1}^{K} log\ P(e_i \mid e)$

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

4.7.3 Evaluation Metrics

We evaluated models for the narrative cloze task with three metrics: **Recall@50**, **Accuracy** and **Event Perplexity**. Recall@50 is the standard metric used for evaluating script models (Jans et al., 2012; Pichotta and Mooney, 2014). The idea here is to evaluate top 50 predictions of a script model on a test script with a held-out event. The metric is calculated as the fraction of the predictions containing the gold held-out event. Its value lies in the range 0 (worst) and 1(best). Accuracy is a new metric introduced by Pichotta and Mooney (2014). This metric evaluates the event prediction, taking into account prediction of each constituent. Specifically, it is defined as the

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

Table 4.5 Model evaluation on test set for narrative cloze task against the baselines

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

Table 4.6 Model evaluation on predicate only event test set for narrative cloze task against the baselines

average of the accuracy of the predicate, the dependency, the first argument and the second argument predictions. This is a more robust metric as it does not treat an event as an atomic unit. This is in contrast to Recall@50, which penalizes semantically correct guesses and awards only events which have exactly the same surface form.

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

4.7.4 Narrative Cloze Evaluation

As described before, previous systems for script event prediction have used the narrative cloze task for evaluation. We also evaluated our model for the same, and the results are shown in Tables 4.5 and 4.6. We evaluated with two versions of the cloze task. In

the first version, events are the predicate argument tuple as defined before. The second version evaluates on predicates only, i.e. an event is not a tuple but only a predicate. Our model, NNSM, outperforms both the unigram and M-Pr models on both the versions of the task with all the metrics. This further strengthens our hypothesis of having distributed representation for events rather than atomic representations. Unigram, although a simple model, is competitive with the M-Pr model.

We evaluated using another simplistic baseline model, most frequent event. This baseline model predicts by sampling from the top-5 most frequent predicates/arguments in the full event narrative cloze task. Surprisingly, the accuracy reported by this simple baseline is 45.04%, which is slightly more (though not statistically significant) than our best performing NNSM model and much more than the M-Pr baseline. This simple looking baseline is hard to beat by count-based methods.

We believe that the narrative cloze evaluation with the accuracy metric may not be the best way to assess script models. As demonstrated by our experiments, any simple, uninformed model could perform impressively well compared to a well-informed model. None of the previous methods have been evaluated against a most frequent event baseline.

In order to assess if the model is capturing the script structure, it might make more sense to evaluate using event perplexity, as shown in Tables 4.5 and 4.6. Additionally, we propose using another evaluation methodology, namely: adversarial narrative cloze. The next section describes our experiments using these methodologies.

4.7.5 Adversarial Narrative Cloze Evaluation

Similar to narrative cloze, the adversarial narrative cloze task was evaluated on 29,943 test set scripts. In each of the event sequences, an event was replaced by a random event. The results for the adversarial narrative cloze task are shown in Tables 4.7 and

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

Table 4.7 Model evaluation on test set for adversarial narrative cloze task against the baselines

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

Table 4.8 Model evaluation on predicate only test set for adversarial narrative cloze task against the baselines

4.8. As evident from the results, the unigram model is as good as random. In this task as well, our model outperforms the count based M-Pr model by 2.3% and 2.9% for the full and pred model, respectively.

4.8 Related Work

As mentioned in Chapter 3, in the past few years a number of count based systems for script learning have been proposed for learning script knowledge in an unsupervised fashion. (Chambers and Jurafsky, 2008; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015a). Recently, Regneri et al. (2010) and Wanzare et al. (2016) have used crowd-sourcing methods for acquiring script knowledge. In the process, they created script databases which are used to develop automatic script learning systems. McIntyre and Lapata (2009) developed a system for generating stories, they learned an object based script model on fairy tales.

Other than neural network based script models described in Chapter 3, Orr et al. (2014) proposed a hidden markov model (HMM) approach to learning scripts. The model clusters event descriptions into different event types and then learns an HMM over the sequence of event types. Again, this model treats an event as an atomic unit and the inference algorithm may not generalize well as the number of event types increases.

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

4.9 Conclusion

In this chapter, we developed statistical models for representing common sense knowledge about prototypical event orderings and event prediction.

Our event ordering model induces distributed representations of events by composing predicate and argument representations. These representations capture properties relevant to predicting stereotyped orderings of events. We learn these representations and the ordering component from unannotated data. As shown in the experiments, our model outperforms the existing graph based and co-occurrence count based methods.

We also proposed a probabilistic compositional model for event predictions in scripts. Experiments show the advantages of the model over conventional count-based approaches. This further reinforces our hypothesis of having richer compositional representations for events. Current tasks to evaluate script models are crude, in the sense that they penalize semantically plausible events. In the future, we propose to create a standard dataset of event sequence pairs (correct sequence vs incorrect sequence). The replaced event in the incorrect sequence should not be a random event, but rather a semantically close but incorrect event. Models evaluated on this dataset would be a better indicator of the script learning capability of the model.

CHAPTER 5

InScript: Narrative Texts Annotated with Script Information

I never guess. It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.

**

Sir Arthur Conan Doyle

5.1 Introduction

In the previous chapter, we have explained the concept of scripts and described computational methods for learning script knowledge. We motivated script knowledge from a cognitive perspective in Chapter 3. We would like to study the effect of script knowledge on language comprehension. Crowdsourced script corpora, for example, the OMICS and SMILE corpora (Singh et al. (2002), Regneri et al. (2010), Regneri (2013)), discussed in the previous chapter, describe events in a script in telegram style. Not being narrative texts, these have minimalistic descriptions and consequently, have limited linguistic variation. These telegram-style descriptions may not exhibit a wide range of discourse phenomena that contribute towards language comprehension. In order to study the contribution of script knowledge to language understanding, one needs to look into narrative texts instantiating script scenarios.

However, a systematic study of the influence of script knowledge in texts is far from trivial. Typically, text documents (e.g. narrative texts) describing various scenarios instantiate many different scripts, making it difficult to study the contribution of the script corresponding to one particular scenario.

Previously, narrative texts from newspapers and blogs have been used for modeling scripts knowledge (Chambers and Jurafsky, 2008; Rudinger et al., 2015a). But these texts instantiate only rudimentary script structure corresponding to different scenarios. As explained in Chapters 3 and 4, these narrative texts instantiate super scripts. These narrative texts are protagonist oriented and describe all the activities that a protagonist performs. These activities may not specifically correspond to a single script scenario. For example, it is difficult to find narratives instantiating an EATING IN A RESTAURANT scenario in isolation. Usually, most of the available narratives instantiating an EATING IN A RESTAURANT scenario, also talk about other activities

> I was sitting on my couch when I decided that I hadn't taken a bath in a while so I stood up and walked to the bathroom where I turned on the faucet in the sink and began filling the bath with hot water. While the tub was filling with hot water I put some bubble bath into the stream of hot water coming out of the faucet so that the tub filled with not only hot water[...]

Fig. 5.1 An excerpt from a story on the TAKING A BATH script.

(corresponding to some other scenario) happening within the eating scenario performed by the protagonist. For example, there might be a story about a person reviewing a research paper (corresponding to a REVIEWING A SCIENTIFIC PAPER scenario), while having food in the restaurant (e.g. see the corpus created by Rudinger et al. (2015a)). One of the possible reasons for scarcity of such scenario focused texts is lack of interest of the writers in describing something which is quite common, mundane and lacks unusual events.

This chapter presents the InScript[1] corpus (Narrative Texts **In**stantiating **Script** structure) (Modi et al., 2016). It is a corpus of simple narrative texts in the form of stories, wherein each story is centered around a specific scenario. The stories have been collected via Amazon Mechanical Turk (M-Turk)[2]. In this experiment, workers were asked to write down a concrete experience about a bus ride, a grocery shopping event etc. We concentrated on 10 scenarios and collected 100 stories per scenario, giving a total of 1,000 stories with about 200,000 words. Relevant verbs and noun phrases in all stories are annotated with *event types* and *participant types*, respectively. Additionally, the texts have been annotated with coreference information in order to facilitate the study of the interdependence between script structure and coreference.

The InScript corpus is a unique resource that provides a basis for studying various aspects of the role of script knowledge in language processing by humans. The

[1]The corpus can be downloaded at: http://www.sfb1102.uni-saarland.de/?page_id=2582
[2] https://www.mturk.com

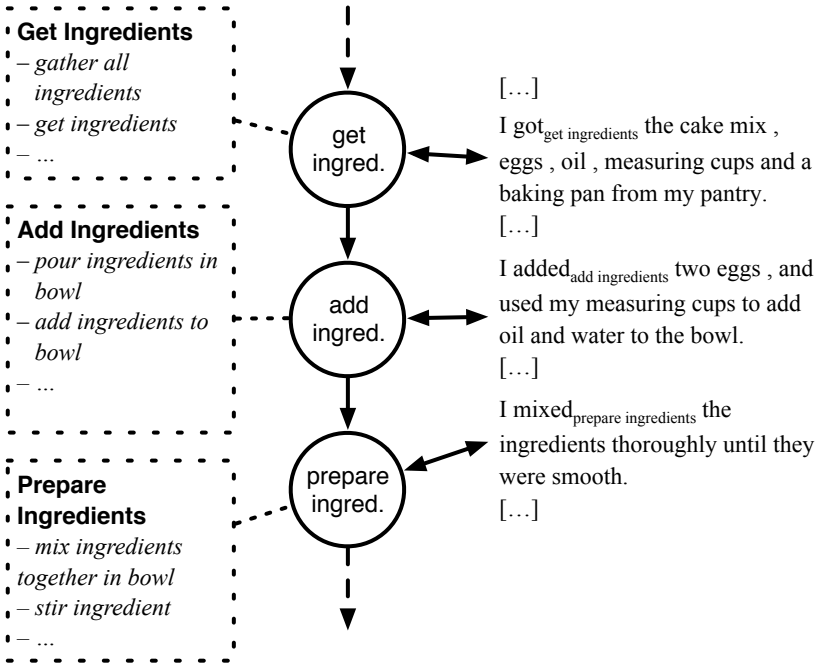

Fig. 5.2 Connecting DeScript and InScript: an example from the BAKING A CAKE scenario (InScript participant annotation is omitted for better readability).

acquisition of this corpus is part of a research effort that aims at using script knowledge to model the surprisal and information density in written text (see Chapter 6).

Figure 5.1 shows the first few sentences of a story from the InScript corpus, describing the scenario TAKING A BATH. This narrative illustrates how script knowledge guides the expectation of the reader and plays an important role in text understanding. For example, once the TAKING A BATH scenario is evoked by the noun phrase (NP) "a bath", the reader can effortlessly interpret the definite NP "the faucet" as an implicitly present standard participant of the TAKING A BATH script. Although in this story, "entering the bathroom", "turning on the water" and "filling the tub" are explicitly mentioned, a reader could nevertheless have inferred the "turning on the water" event, even if it was not explicitly mentioned in the text.

Besides InScript, there is another related corpus of generic descriptions of script activities called DeScript (for **De**scribing **Script** Structure, Wanzare et al. (2016)). The DeScript contains a range of short and textually simple phrases (referred to as

event descriptions; see Section 3.1) that describe script events in the style of OMICS or SMILE (Singh et al. (2002), Regneri et al. (2010)). The DeScript corpus contains a collection of event sequence descriptions (ESD, see Section 3.1) for different scenarios. We describe in detail the DeScript corpus and how it differs from the InScript corpus in Section 5.4.3. Figure 5.2 shows an excerpt of a script in the BAKING A CAKE scenario. The figure shows event descriptions for 3 different events in the DeScript corpus (left) and fragments of a story in the InScript corpus (right) that instantiate the same event type.

5.2 Data Collection

5.2.1 Collection via Amazon M-Turk

We selected 10 scenarios from different available scenario lists (e.g. Regneri et al. (2010) , Raisig et al. (2009), and the OMICS corpus Singh et al. (2002)), including scripts of different complexity (TAKING A BATH vs. FLYING IN AN AIRPLANE) and specificity (RIDING A PUBLIC BUS vs. REPAIRING A FLAT BICYCLE TIRE). For the full scenario list see Table 5.1.

Texts were collected via the Amazon Mechanical Turk platform, which provides an opportunity to present an online task to humans (a.k.a. *workers*). In order to gauge the effect of different M-Turk instructions on our task, we first conducted pilot experiments with different variants of instructions explaining the task. The first set of instructions asked workers to describe (in the form of a story) their latest experience in a particular scenario. The second set of instructions posed the task as a memory test. The instructions asked the workers to describe a scenario by recalling the scenario from their memory. The workers were specifically asked to recall all the mundane and trivial details that they performed in the scenario. The last set of instructions asked

Scenario Name	#Stories	Avg. Sentences Per Story	Avg. Word Type Per Story	Avg. Word Count Per Story	Avg. Word Type Overlap
Riding in a public bus (Bus)	92	12.3 (4.1)	97.4 (23.3)	215.1 (69.7)	35.7 (7.5)
Baking a cake (Cake)	97	13.6 (4.7)	102.7 (23.7)	235.5 (78.5)	39.5 (8.1)
Taking a bath (Bath)	94	11.5 (2.6)	91.9 (13.1)	***197.5 (34.5)***	37.9 (6.3)
Going grocery shopping (Grocery)	95	13.1 (3.7)	102.9 (19.9)	228.3 (58.8)	38.6 (7.8)
Flying in an airplane (Flight)	86	**14.1 (5.6)**	**113.6 (30.9)**	**251.2 (99.1)**	**40.9 (10.3)**
Getting a haircut (Haircut)	88	13.3 (4.0)	100.6 (19.3)	227.2 (63.4)	39.0 (7.9)
Borrowing a book from the library (Library)	93	11.2 (2.5)	***88.0 (14.1)***	200.7 (43.5)	34.9 (5.5)
Going on a train (Train)	87	12.3 (3.4)	96.3 (19.2)	210.3 (57.0)	35.3 (6.9)
Repairing a flat bicycle tire (Bicycle)	87	11.4 (3.6)	88.9 (15.0)	203.0 (53.3)	33.8 (5.2)
Planting a tree (Tree)	91	***11.0 (3.6)***	93.3 (19.2)	201.5 (60.3)	***34.0 (6.6)***
Average	*91*	*12.4*	*97.6*	*216.9*	*37.0*

Table 5.1 Corpus statistics for different scenarios (standard deviation given in parentheses). The maximum per column is highlighted in **boldface**, the minimum in ***boldface italics***.

the workers to describe a scenario in the form of a story as if explaining it to a child. In each of the instructions above we set a minimum word count of 150 words. During the pilot experiment, the set corresponding to the explaining-to-a-child instructions was selected. As during the pilot experiment, it resulted in comparably simple and explicit scenario-related stories. In total 190 workers participated. Since Mturk provides an option for restricting the region and the native language of the workers participating in an experiment, we restricted the workers to those living in the USA who were native speakers of English. We paid USD $0.50 per story to each worker. On average, the workers took 9.37 minutes per story with a maximum duration of 17.38 minutes.

5.2.2 Data Statistics

Statistics for the corpus are given in Table 5.1. On average, each story has a length of 12 sentences and 217 words, belonging to 98 word types on average. Stories are coherent and concentrate mainly on the corresponding scenario. Neglecting auxiliaries, modals and copulas, on average each story has 32 verbs, out of which 58% denote events related to the respective scenario. As can be seen in Table 5.1, there is some variation in stories across scenarios: The FLYING IN AN AIRPLANE scenario, for example, is most complex in terms of the number of sentences, tokens and word types that are used. This is probably due to the inherent complexity of the scenario: Taking a flight, for example, is more complicated and takes more steps than taking a bath. The average count of sentences, tokens and types is also very high for the BAKING A CAKE scenario. Stories from the scenario often resemble cake recipes, which usually contain very detailed steps, so people tend to give more detailed descriptions in the stories.

For both FLYING IN AN AIRPLANE and BAKING A CAKE, the standard deviation is higher in comparison to other scenarios. This indicates that different workers described the scenario with a varying degree of detail and can also be seen as an indicator for the complexity of both scenarios. In general, different people tend to describe situations subjectively, with a varying degree of detail.

In contrast, texts from the TAKING A BATH and PLANTING A TREE scenarios contain a relatively smaller number of sentences and fewer word types and tokens. Both planting a tree and taking a bath are simpler activities, which results in generally less complex texts.

The average pairwise word type overlap can be seen as a measure of lexical variety among stories: If it is high, the stories resemble each other more. We can see that stories in the FLYING IN AN AIRPLANE and BAKING A CAKE scenarios have the highest values here, indicating that most workers used a similar vocabulary in their stories.

In general, the response quality was good. We had to discard 9% of the stories as these lacked the quality we were expecting. Among the discarded stories, some of the stories had gibberish written in them, as some of the workers had cheated. Some of the workers misunderstood the instructions and wrote about some unusual events happening in the scenario, with the main focus of the story on the unusual event. Such stories had to be discarded as well. In total, we selected 910 stories for annotation.

5.3 Annotation

This section deals with the annotation of the data. We first describe the final annotation schema. Then, we describe the iterative process of corpus annotation and the refinement of the schema. This refinement was necessary due to the complexity of the annotation.

5.3.1 Annotation Schema

For each of the scenarios, we designed a specific annotation template. A *script template* consists of scenario-specific event and participant labels. An example of a template for the TAKING A BATH scenario is shown in Table 5.2. All NP heads in the corpus were annotated with a participant label; all verbs were annotated with an event label. For both participants and events, we also offered the label UNCLEAR if the annotator could not assign another label. We additionally annotated coreference chains between NPs. Thus, the process resulted in three layers of annotation: event types, participant types and coreference annotation. These are described in detail below.

Event Type

As a first layer, we annotated event types. There are two kinds of event type labels, scenario-specific event type labels and general labels. The general labels are used across all scenarios and mark general features, for example whether an event belongs to the

Event Types	Participant Types
ScrEv_take_clean_clothes	
ScrEv_prepare_bath	ScrPart_Bath
ScrEv_enter_bathroom	ScrPart_bath_means
ScrEv_turn_water_on	ScrPatr_bather
ScrEv_check_temp	ScrPart_bathroom
(temperature)	ScrPart_bathtub
ScrEv_close_drain	ScrPart_body_part
ScrEv_wait	ScrPart_clothes
ScrEv_turn_water_off	ScrPart_drain
ScrEv_put_bubble_bath_scent	ScrPart_hair
ScrEv_undress	ScrPart_hamper
ScrEv_sink_water	ScrPart_in-bath_entertainment
ScrEv_relax	(candles, music, books)
ScrEv_apply_soap	ScrPart_plug
ScrEv_wash	ScrPart_shower
ScrEv_open_drain	(as bath equipment)
ScrEv_get_out_bath	ScrPart_tap (knob)
ScrEv_get_towel	ScrPart_temperature
ScrEv_dry	ScrPart_towel
ScrEv_put_after_shower	ScrPart_washing_tools
textscScrEv_get_dressed	(washcloth, soap)
ScrEv_leave	ScrPart_water
ScrEv_air_bathroom	

Table 5.2 Bath scenario template (labels added in the second phase of annotation are marked in bold).

scenario at all. For the scenario-specific labels, we designed a unique template for every scenario, with a list of script-relevant event types that were used as labels. Such labels include for example ScrEv_close_drain in taking a bath as in Example (5.1) (see Figure 5.2 for a complete list for the taking a bath scenario):

(5.1) I start by **closing**$_{\text{ScrEv\_close\_drain}}$ the drain at the bottom of the tub.

The general labels that were used in addition to the script-specific labels in every scenario are listed below:

ScrEv_other. An event that belongs to the scenario, but its event type occurs too infrequently (for details, see below, Section 5.3.4). We used the label "other" because event classification would become too fine-grained otherwise. For example, see (5.2):

(5.2) After I am dried I put my new clothes on and **clean up**$_{\text{ScrEv\_other}}$ the bathroom.

RelNScrEv. Related non-script event. An event that *can* plausibly happen during the execution of the script and is related to it, but that is not part of the script. For example, see (5.3):

(5.3) After finding on what I wanted to wear, I went into the bathroom and **shut**$_{\text{RelNScrEv}}$ the door.

UnrelEv. An event that is unrelated to the script. For example, see (5.4):

(5.4) I sank into the bubbles and **took** $_{\text{UnrelEv}}$ a deep breath.

Additionally, the annotators were asked to annotate verbs and phrases that evoke the script without referring to a specific script event with the label EVOKING, as shown in Example (5.5).

(5.5) Today I **took a bath** $_{\text{Evoking}}$ in my new apartment.

Participant Type

As in the case of the event type labels, there are two kinds of participant labels: general labels and scenario-specific labels. The latter are part of the scenario-specific templates, e.g. SCRPART_DRAIN in the TAKING A BATH scenario, as can be seen in Example (5.6).

(5.6) I start by closing the **drain** $_{\text{ScrPart\_drain}}$ at the bottom of the tub.

The general labels that are used across all scenarios mark noun phrases with scenario-independent features. There are the following general labels:

ScrPart_other. A participant that belongs to the scenario, but its participant type occurs only infrequently. Example:

(5.7) I find my **bath mat**$_{\text{ScrPart\_other}}$ and lay it on the floor to keep the floor dry.

NPart. Non-participant. A referential NP that does not belong to the scenario. Example:

(5.8) I washed myself carefully because I did not want to spill water onto the **floor**$_{\text{NPart}}$.

SuppVComp. A support verb complement. For further discussion of this label, see Section 5.3.5. Example:

(5.9) I sank into the bubbles and took a deep **breath**$_{\text{SuppVComp}}$.

Head_of_Partitive. The head of a partitive or a partitive-like construction. For a further discussion of this label see Section 5.3.5. Example:

(5.10) I grabbed a **bar**$_{\text{Head\_of\_Partitive}}$ of soap and lathered my body.

No_label. A non-referential noun phrase that cannot be labeled with another label. Example:

(5.11) I sat for a **moment**$_{\text{No\_label}}$, relaxing, allowing the warm water to soothe my skin.

All NPs labeled with one of the labels SUPPVCOMP, HEAD_OF_PARTITIVE or NO_LABEL are considered to be non-referential. NO_LABEL is used mainly in four cases in our data: non-referential time expressions (*in a **while**, a million **times** better*), idioms (*no **matter** what*), the non-referential "it" (***it** felt amazing*, ***it** is better*) and other abstracta (*a **lot** better*, *a little **bit***).

In the first annotation phase, annotators were asked to mark verbs and noun phrases that have an event or participant type, that is *not* listed in the template, as MISSSCREV/MISSSCRPART (missing script event or participant, respectively). These annotations were used as a basis for extending the templates (see Section 5.3.4) and replaced later by either newly introduced labels or SCREV_OTHER and SCRPART_OTHER labels. An example annotation of event types and participant types in the TAKING A BATH scenario is shown in Figure 5.3 on page 106.

Coreference Annotations

All noun phrases were annotated with coreference information indicating which entities denote the same discourse referent. The annotation was done by linking heads of NPs (see Example (5.12), where the links are indicated by coindexing). As a rule, we assume that each element of a coreference chain is marked with the same participant type label.

(5.12) **I** COREF1 washed **my** COREF1 entire **body** COREF2, starting with **my** COREF1 **face** COREF3 and ending with the **toes** COREF4. **I** COREF1 always wash **my** COREF1 **toes** COREF4 very thoroughly ...

The assignment of an entity to a referent is not always trivial, as is shown in Example (5.13). There are some cases in which two discourse referents are grouped in a plural NP. In the example, *those things* refers to the group made up

of *shampoo*, *soap* and *sponge*. In this case, we asked annotators to introduce a new coreference label, the name of which indicates which referents are grouped together (COREF_GROUP_WASHING_TOOLS). All NPs are then connected to the group phrase, resulting in an additional coreference chain.

(5.13) **I**$_{\text{COREF1}}$ made sure that **I**$_{\text{COREF1}}$ have **my** $_{\text{COREF1}}$ **shampoo**$_{\text{COREF2 + COREF\_GROUP\_WASHING\_TOOLS}}$, **soap** $_{\text{COREF3 + COREF\_GROUP\_WASHING\_TOOLS}}$ and **sponge** $_{\text{COREF4 + COREF\_GROUP\_WASHING\_TOOLS}}$ ready to get in. Once **I** $_{\text{COREF1}}$ have those **things** $_{\text{COREF\_GROUP\_WASHING\_TOOLS}}$ **I** $_{\text{COREF1}}$ sink into the bath..... **I** $_{\text{COREF1}}$ applied some **soap** $_{\text{COREF3}}$ on **my** $_{\text{COREF1}}$ body and used the **sponge** $_{\text{COREF4}}$ to scrub a bit. ... **I** $_{\text{COREF1}}$ rinsed the **shampoo** $_{\text{COREF2}}$.

Example (5.13) thus contains the following coreference chains:

(5.14) **COREF1:** I → I → my → I → I → I → my → I
COREF2: shampoo → shampoo
COREF3: soap → soap
COREF4: sponge → sponge
COREF_GROUP_WASHING_ TOOLS: shampoo → soap → sponge → things

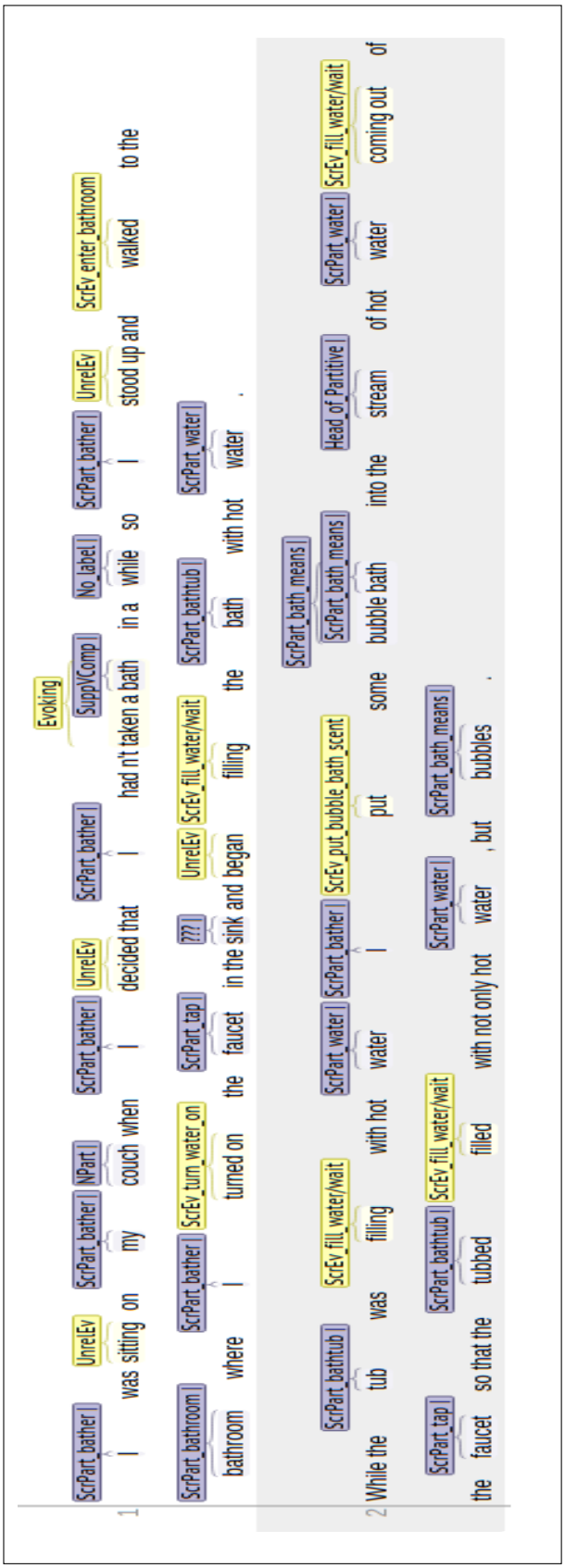

Fig. 5.3 Sample event and participant annotation for the TAKING A BATH script.

5.3.2 Development of the Schema

The templates were carefully designed in an iterated process. For each scenario, we came up with a preliminary version of the template based on the inspection of some of the stories. For a subset of the scenarios, preliminary templates developed at our department for a psycholinguistic experiment on script knowledge were used as a starting point. Subsequently, we manually annotated 5 randomly selected texts for each of the scenarios based on the preliminary template. Necessary extensions and changes in the templates were discussed and agreed upon. Most of the cases of disagreement were related to the granularity of the event and participant types. We agreed on the *script-specific functional equivalence* as a guiding principle. For example, reading a book, listening to music and having a conversation are subsumed under the same event label in the FLIGHT scenario, because they have the common function of in-flight entertainment in the scenario. In contrast, we assumed different labels for the cake tin and other utensils (bowls etc.), since they have different functions in the BAKING A CAKE scenario and accordingly occur with different script events.

Note that scripts and templates as such are not meant to describe an activity as exhaustively as possible and to mention all steps that are logically necessary. Instead, scripts describe cognitively prominent events in an activity. An example can be found in the FLIGHT scenario. While more than a third of the workers mentioned the event of fastening the seat belts in the plane (BUCKLE_SEAT_BELT), no person wrote about *undoing* their seat belts again, although in reality both events appear equally often. Consequently, we added an event type label for buckling up, but no label for undoing the seat belts.

5.3.3 First Annotation Phase

We used the WebAnno annotation tool (Yimam et al., 2013) for the corpus. The stories from each scenario were distributed among four different annotators. In a calibration phase, annotators were presented with some sample texts for test annotations; the results were discussed with the author. Throughout the whole annotation phase, annotators could discuss any emerging issues with the authors. All annotations were done by undergraduate students of computational linguistics. The annotation was rather time-consuming due to the complexity of the task, and thus we decided for single annotation mode. To assess annotation quality, a small sample of texts was annotated by all four annotators and their inter-annotator agreement was measured (see Section 5.4.1). It was found to be sufficiently high.

Annotation of the corpus together with some pre- and post-processing of the data required about 500 hours of work. All stories were annotated with event and participant types (a total of 12,188 and 43,946 instances, respectively). On average there were 7 coreference chains per story with an average length of 6 tokens.

5.3.4 Modification of the Schema

After the first annotation round, we extended and changed the templates based on the results. As mentioned before, we used MissScrEv and MissScrPart labels to mark verbs and noun phrases instantiating events and participants for which no appropriate labels were available in the templates. Based on the instances with these labels (a total of 941 and 1717 instances, respectively), we extended the guidelines to cover the sufficiently frequent cases.

In order to include new labels for event and participant types, we tried to estimate the number of instances that would fall under a certain label. We added new labels according to the following conditions:

- For the participant annotations, we added new labels for types that we expected to appear at least 10 times in total in at least 5 different stories (i.e. in approximately 5% of the stories).
- For the event annotations, we chose those new labels for event types that would appear in at least 5 different stories.

In order to avoid too fine a granularity of the templates, all other instances of MissScrEv and MissScrPart were re-labeled with ScrEv_other and ScrPart_other. We also relabeled participants and events from the first annotation phase with ScrEv_other and ScrPart_other, if they did not meet the frequency requirements. The event label air_bathroom (the event of letting fresh air into the room after the bath), for example, was only used once in the stories, so we relabeled that instance to ScrEv_other.

Additionally, we looked at the DeScript corpus (Wanzare et al., 2016), which contains manually clustered event paraphrase sets for the 10 scenarios that are also covered by InScript (see Section 5.4.3). Every such set contains event descriptions that define a certain event type. We extended our templates with additional labels for these events, if they were not yet part of the template. This harmonized the InScript annotation with DeScript gold event paraphrase clusters.

5.3.5 Special Cases

During annotation we came across certain linguistic constructions that had to be dealt on case by case basis. We next describe these special cases:

Noun-Noun Compounds. Noun-noun compounds were annotated twice with the same label (whole span plus the head noun), as indicated by Example (5.15). This redundant double annotation is motivated by potential processing requirements.

(5.15) I get my **[wash [cloth** $_{\text{ScrPart\_washing\_tools}}$**]]** $_{\text{ScrPart\_washing\_tools}}$ and put it under the water.

Support Verb Complements. A special treatment was given to support verb constructions such as *take time*, *get home* or *take a seat* in Example (5.16). The semantics of the verb itself is highly underspecified in such constructions; the event type is largely dependent on the object NP. As shown in Example (5.16), we annotate the head verb with the event type described by the whole construction and label its object with SuppVComp (support verb complement), indicating that it does not have a proper reference.

(5.16) I step into the tub and **take**$_{\text{ScrEv\_sink\_water}}$ a **seat**$_{\text{SuppVComp}}$.

Head of Partitive. We used the Head_of_Partitive label for the heads in partitive constructions, assuming that the only referential part of the construction is the complement.

(5.17) Our seats were at the **back** $_{\text{Head\_of\_Partitive}}$ of the train $_{\text{ScrPart\_train}}$.

(5.18) In the library you can always find a **couple** $_{\text{Head\_of\_Partitive}}$ of interesting books $_{\text{ScrPart\_book}}$.

This is not completely correct, since different partitive heads vary in their degree of concreteness (cf. Examples (5.17) and (5.18)), but we did not see a way to make the distinction sufficiently transparent to the annotators.

Scenario	Events	Participants
BATH	20	18
BICYCLE	16	16
BUS	17	17
CAKE	19	17
FLIGHT	29	26
GROCERY	19	18
HAIRCUT	26	24
LIBRARY	17	18
TRAIN	15	20
TREE	14	15
Average	*19.2*	*18.9*

Fig. 5.4 The number of participants and events in the templates.

Mixed Participant Types. Group denoting NPs sometimes refer to groups whose members are instances of different participant types. In Example (5.19), the first-person plural pronoun refers to the group consisting of the passenger (I) and a non-participant (*my friend*). To avoid a proliferation of event type labels, we labeled these cases with UNCLEAR.

(5.19) **I** SCRPART_PASSENGER wanted to visit **my** SCRPART_PASSENGER **friend** NPART in New York. ... **We** UNCLEAR met at the train station.

We made an exception for the GETTING A HAIRCUT scenario, where the mixed participant group consisting of the hairdresser and the customer occurs very often, as in Example (5.19). Here, we introduced the additional ad-hoc participant label SCR_PART_HAIRDRESSER_CUSTOMER.

(5.20) While **Susan** SCRPART_HAIRDRESSER is cutting **my** SCRPART_CUSTOMER hair **we** SCR_PART_HAIRDRESSER_CUSTOMER usually talk a bit.

	Average Fleiss' Kappa			
	All Labels		Script Labels	
Scenario	Events	Participants	Events	Participants
Bus	0.68	0.74	0.76	0.74
Cake	0.61	0.76	0.64	0.75
Flight	0.65	0.70	0.62	0.69
Grocery	0.64	0.80	0.73	0.80
Haircut	0.64	0.84	0.67	0.86
Tree	0.59	0.76	0.63	0.76
Average	*0.64*	*0.77*	*0.68*	*0.77*

Fig. 5.5 Inter-annotator agreement statistics: Average Fleiss' Kappa.

5.4 Data Analysis

5.4.1 Inter-Annotator Agreement

In order to calculate inter-annotator agreement, a total of 30 stories from 6 scenarios were randomly chosen for parallel annotation by all 4 annotators after the first annotation phase[3]. We checked the agreement on these data using Fleiss' kappa (Fleiss, 1971). The results are shown in Figure 5.5 and indicate moderate to substantial agreement (Landis and Koch, 1977). Interestingly, if we calculated the kappa only on the subset of cases that were annotated with script-specific event and participant labels by all annotators, results were better than those of the evaluation on all labeled instances (including also unrelated and related non-script events). This indicates one of the challenges of the annotation task: In many cases it is difficult to decide whether a particular event should be considered a central script event, or an event loosely related or unrelated to the script.

For coreference chain annotation, we calculated the percentage of pairs which were annotated by at least 3 annotators (qualified majority vote) compared to the set of

[3]We did not test for inter-annotator agreement after the second phase, since we did not expect the agreement to change drastically due to the only slight changes in the annotation schema.

Scenario	%Coreference Agreement
Bus	88.9
Cake	94.7
Flight	93.6
Grocery	93.4
Haircut	94.3
Tree	78.3
Average	*90.5*

Fig. 5.6 Inter-annotator agreement statistics: Coreference agreement.

	avg	min	max
event annotations in a story	15.9	1	52
event types in a story	10.1	1	23
participant annotations in a story	52.3	16	164
participant types in a story	10.9	2	25
coref chains	7.3	0	23
tokens per chain	6	2	52

Fig. 5.7 Annotation statistics over all scenarios.

those pairs annotated by at least one person (see Figure 5.6). We take the result of 90.5% between annotators to be a good agreement.

5.4.2 Annotated Corpus Statistics

Figure 5.4 gives an overview of the number of event and participant types provided in the templates. Taking a flight and getting a haircut stand out with a large number of both event and participant types, which is due to the inherent complexity of the scenarios. In contrast, planting a tree and going on a train contain the fewest labels. There are 19 event and participant types on average.

Figure 5.7 presents overview statistics about the usage of event labels, participant labels and coreference chain annotations. As can be seen, there are usually many more mentions of participants than events. For coreference chains, there are some chains that are really long (which also results in a large scenario-wise standard deviation). Usually, these chains describe the protagonist.

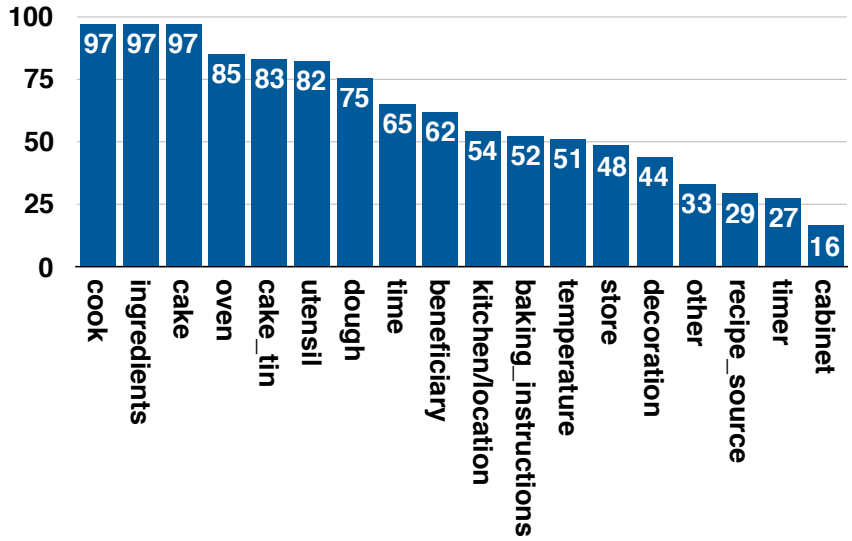

Fig. 5.8 The number of stories in the BAKING A CAKE scenario that contain a certain participant label.

We also found again that the FLYING IN AN AIRPLANE scenario stands out in terms of participant mentions, event mentions and average number of coreference chains.

Figure 5.8 shows for every participant label in the BAKING A CAKE scenario the number of stories in which they occurred. This indicates how relevant a participant is for the script. As can be seen, a small number of participants are highly prominent: COOK, INGREDIENTS and CAKE are mentioned in every story. The fact that the protagonist appears most often consistently holds for all other scenarios, where the acting person appears in every story, and is mentioned most frequently.

Figure 5.9 shows the distribution of participant/event type labels over all appearances over all scenarios on average. The groups stand for the most frequently appearing label, the top 2 to 5 labels in terms of frequency and the top 6 to 10. SCREV_OTHER and SCRPART_OTHER are shown separately. As can be seen, the most frequently used participant label (the protagonist) makes up about 40% of overall participant instances. The four labels that follow the protagonist in terms of frequency together appear in

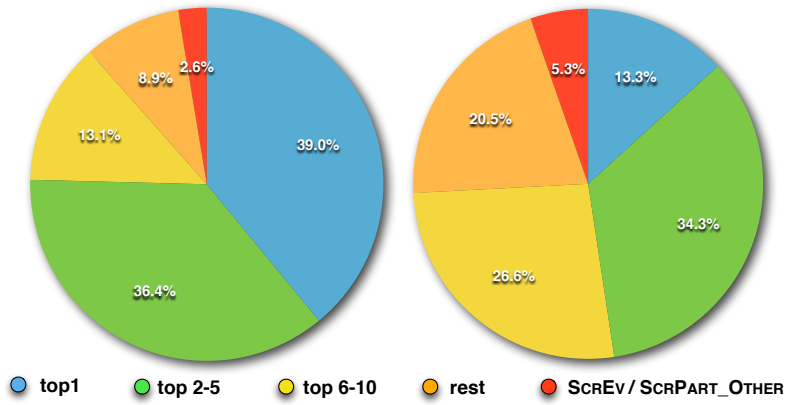

Fig. 5.9 Distribution of participants (left) and events (right) for the 1, the top 2-5, top 6-10 most frequently appearing events/participants, ScrEv/ScrPart_Other and the rest.

37% of the cases. More than 2 out of 3 participants in total belong to one of only 5 labels.

In contrast, the distribution for events is more balanced. 14% of all event instances have the most prominent event type. ScrEv_other and ScrPart_other both appear as labels in at most 5% of all event and participant instantiations: The specific event and participant type labels in our templates cover by far most of the instances.

In Figure 5.10, we grouped participants similarly into the first, the top 2-5 and top 6-10 most frequently appearing participant types. The figure shows for each of these groups the average frequency per story, and in the rightmost column the overall average. The results correspond to the findings from the last paragraph.

5.4.3 Comparison to the DeScript Corpus

DeScript covers 40 scenarios, and also contains the 10 scenarios from InScript. This corpus contains texts that describe scripts on an abstract and generic level, while InScript contains instantiations of scripts in narrative texts. Script events in DeScript are described in a very simple, telegram-style language (see Figure 5.2).

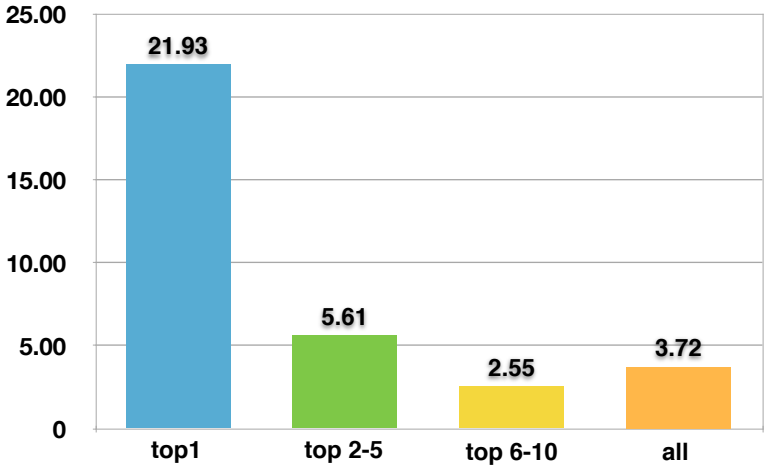

Fig. 5.10 Average number of participant mentions for a story, for the first, the top 2-5, and top 6-10 most frequently appearing events/participants, and the overall average.

The InScript corpus exhibits much more lexical variation than DeScript. Many approaches use the *type-token ratio* to measure this variance. However, this measure is known to be sensitive to text length (see e.g. Tweedie and Baayen (1998)), which would result in very small values for InScript and relatively large ones for DeScript, given the large average difference of text lengths between the corpora. Instead, we decided to use the *Measure of Textual Lexical Diversity (MTLD)* (McCarthy and Jarvis (2010), McCarthy (2005)), which is well known in corpus linguistics. This metric measures the average number of tokens in a text that are needed to retain a type-token ratio above a certain threshold. If the *MTLD* for a text is high, many tokens are needed to lower the type-token ratio under the threshold, so the text is lexically diverse. In contrast, a low *MTLD* indicates that only a few words are needed to make the type-token ratio drop, so the lexical diversity is smaller. We use the threshold of 0.71, which is proposed by the authors as a well-proven value.

Figure 5.11 compares the lexical diversity of both resources. As can be seen, the InScript corpus with its narrative texts is generally much more diverse than the DeScript corpus with its short event descriptions, across all scenarios. For both resources, the

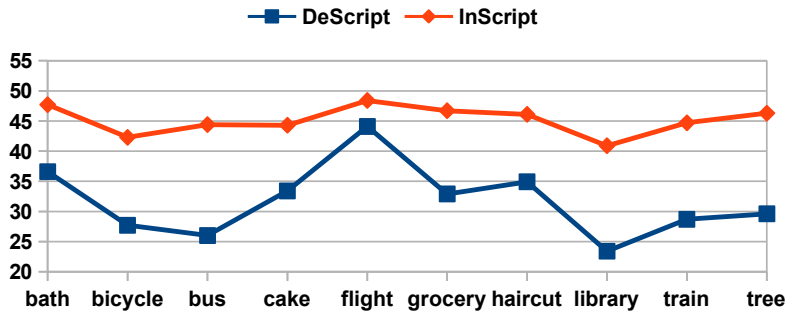

Fig. 5.11 MTLD values for DeScript and InScript, per scenario.

FLYING IN AN AIRPLANE scenario is most diverse (as was also indicated above by the mean word type overlap). However, the difference in the variation of lexical variance of scenarios is larger for DeScript than for InScript. Thus, the properties of a scenario apparently influence the lexical variance of the event descriptions more than the variance of the narrative texts.

We used entropy (Shannon, 1948) over lemmas to measure the variance of lexical realizations for events. We excluded events for which there were less than 10 occurrences in DeScript or InScript. Since there is only an event annotation for 50 ESDs per scenario in DeScript, we randomly sampled 50 texts from InScript for computing the entropy to make the numbers more comparable.

Figure 5.12 shows as an example the entropy values for the event types in the GOING ON A TRAIN scenario. As can be seen in the graph, the entropy for InScript is in general higher than for DeScript. In the stories, a wider variety of verbs is used to describe events. There are also large differences between events: While WAIT has a really low entropy, SPEND_TIME_TRAIN has an extremely high entropy value. This event type covers many different activities such as reading, sleeping etc.

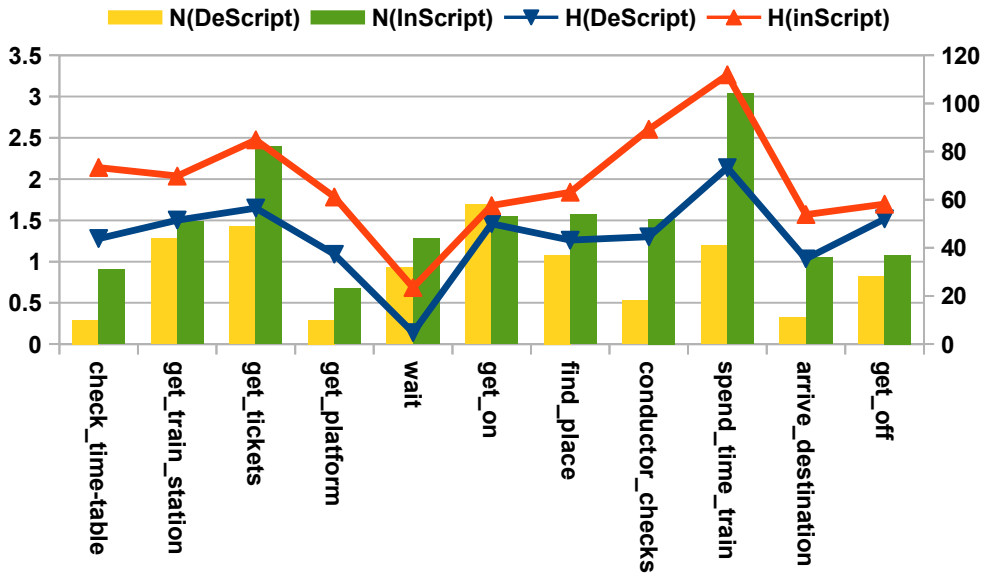

Fig. 5.12 Entropy over verb lemmas for events (left y-axis, H(x)) in the GOING ON A TRAIN SCENARIO. Bars in the background indicate the absolute number of occurrence of instances (right y-axis, N(x)).

5.5 Conclusion

In this chapter we described the InScript corpus of 1,000 narrative texts annotated with script structure and coreference information. We described the annotation process, various difficulties encountered during annotation and different remedies that were taken to overcome these. We compared the InScript corpus with another script related corpus, i.e. DeScript. It would be interesting to explore methods for text-to-script mapping, i.e. for the alignment of text segments with script states. We consider InScript and DeScript together as a resource for studying this alignment. The InScript corpus shows rich lexical variation and is a unique resource for the study of the role of script knowledge in natural language processing, as described in the next chapter.

CHAPTER 6

Modeling Semantic Expectation: Using Script Knowledge for Referent Prediction

When you have eliminated the impossible, whatever remains, however improbable, must be the truth.

**

Sir Arthur Conan Doyle

6.1 Introduction

Being able to anticipate upcoming content is a core property of human language processing (Kuperberg and Jaeger, 2016; Kutas et al., 2011) that has received a lot of attention in the psycholinguistic literature in recent years. Expectations about upcoming words help humans comprehend language in noisy settings and deal with ungrammatical input. In this chapter, we use a computational model to address the question of how different layers of knowledge (linguistic knowledge as well as common-sense knowledge) influence human anticipation.

Here we focus our attention on semantic predictions of *discourse referents* for upcoming noun phrases. This task is particularly interesting because it allows us to separate the semantic task of anticipating an intended referent and the processing of the actual surface form. For example, in the context of *I ordered a medium sirloin steak with fries. Later, the waiter brought ...*, there is a strong expectation of a specific discourse referent, i.e., the referent introduced by the object NP of the preceding sentence, while the possible referring expression could be either *the steak I had ordered*, *the steak*, *our food*, or *it*. Existing models of human prediction are usually formulated using the information-theoretic concept of *surprisal*. In recent work, however, surprisal is usually not computed for DRs, which represent the relevant semantic unit, but for the surface form of the referring expressions, even though there is an increasing amount of literature suggesting that human expectations at different levels of representation have separable effects on prediction and, as a consequence, that the modeling of only one level (the linguistic surface form) is insufficient (Kuperberg, 2016; Kuperberg and Jaeger, 2016; Zarcone et al., 2016). The present model addresses this shortcoming by explicitly modeling and representing common-sense knowledge and conceptually

separating the semantic (discourse referent) and the surface level (referring expression) expectations.

Our discourse referent prediction task is related to the NLP task of coreference resolution, but it substantially differs from that task in the following ways: 1) we use only the incrementally available left context, while coreference resolution uses the full text; 2) coreference resolution tries to identify the DR for a given target NP in context, while we look at the expectations of DRs based only on the context before the target NP is seen.

The distinction between referent prediction and prediction of referring expressions also allows us to study a closely related question in natural language generation: the choice of a type of referring expression based on the predictability of the DR that is intended by the speaker. This part of our work is inspired by a referent guessing experiment by Tily and Piantadosi (2009), who showed that highly predictable referents were more likely to be realized with a pronoun than unpredictable referents, which were more likely to be realized using a full NP. The effect they observe is consistent with a Gricean point of view, or the principle of uniform information density (see Section 6.5.1). However, Tily and Piantadosi do not provide a computational model for estimating referent predictability. Also, they do not include selectional preference or common-sense knowledge effects in their analysis.

We believe that script knowledge represents a good starting point for modeling conversational anticipation. This type of common-sense knowledge includes temporal structure which is particularly relevant for anticipation in continuous language processing. Furthermore, our approach can build on progress that has been made in recent years in methods for acquiring large-scale script knowledge; see Section 6.1.1. Our hypothesis is that script knowledge may be a significant factor in human anticipation

of discourse referents. Explicitly modeling this knowledge will thus allow us to produce more human-like predictions.

Script knowledge enables our model to generate anticipations about discourse referents that have already been mentioned in the text, as well as anticipations about textually new discourse referents which have been activated due to script knowledge. By modeling event sequences and event participants, our model captures many more long-range dependencies than normal language models are able to. As an example, consider the following two alternative text passages:

(6.1) `We got seated, and had to wait for 20 minutes.`
`Then, the waiter brought the ____`

(6.2) `We ordered, and had to wait for 20 minutes.`
`Then, the waiter brought the ____`

Preferred candidate referents for the object position of *the waiter brought the ...* are instances of the *food*, *menu*, or *bill* participant types. In the context of the alternative preceding sentences, there is a strong expectation of instances of a *menu* and a *food* participant, respectively.

This chapter represents foundational research investigating human language processing. However, it also has the potential for application in assistant technology and embodied agents. The goal is to achieve human-level language comprehension in realistic settings, and in particular to achieve robustness in the face of errors or noise. Explicitly modeling expectations that are driven by common-sense knowledge is an important step in this direction.

In order to be able to investigate the influence of script knowledge on discourse referent expectations, we use the InScript corpus, which contains frequent reference to script knowledge, and provides annotations for coreference information, script events and participants. In Section 6.3, we present a large-scale experiment for empirically

assessing human expectations on upcoming referents, which allows us to quantify at what points in a text humans have very clear anticipations vs. when they do not. Our goal is to model human expectations, even if they turn out to be incorrect in a specific instance. The experiment was conducted via Mechanical Turk and follows the methodology of Tily and Piantadosi (2009). In Section 6.4, we describe our computational model that represents script knowledge. The model is trained on the gold standard annotations of the corpus, because we assume that human comprehenders usually will have an analysis of the preceding discourse which closely corresponds to the gold standard. We compare the prediction accuracy of this model to human predictions, as well as to two baseline models, in Section 6.4.3. One of them uses only structural linguistic features for predicting referents; the other uses general script-independent selectional preference features. In Section 6.5, we test whether surprisal (as estimated from human guesses vs. computational models) can predict the type of referring expression used in the original texts in the corpus (pronoun vs. full referring expression). This experiment also has wider implications with respect to the ongoing discussion of whether the referring expression choice is dependent on predictability, as predicted by the uniform information density hypothesis.

6.1.1 Scripts

As explained in detail in Chapter 3, scripts represent knowledge about typical event sequences (Schank and Abelson, 1977), for example the sequence of events happening when eating at a restaurant.

Modeling anticipated events and participants is motivated by evidence showing that event representations in humans contain information not only about the current event, but also about previous and future states, that is, humans generate anticipations about event sequences during normal language comprehension (Schütz-Bosbach and

Prinz, 2007). Script knowledge representations have been shown to be useful in NLP applications for ambiguity resolution during reference resolution (Rahman and Ng, 2012).

6.2 Data: The InScript Corpus

In Chapter 5, we explained the need for a corpus on the lines of InScript. Ordinary texts, including narratives, encode script structure in a way that is too complex and too implicit at the same time to enable a systematic study of script-based expectation. They contain interleaved references to many different scripts, and they usually refer to single scripts in a point-wise fashion only, relying on the ability of the reader to infer the full event chain using their background knowledge.

We use the InScript corpus (Modi et al., 2016) to study the predictive effect of script knowledge. InScript focuses on stories that are centered around a specific scenario and that explicitly mention mundane details. The InScript corpus is labeled with event-type, participant-type, and coreference information. Full verbs are labeled with event-type information, and heads of all noun phrases with participant types, using scenario-specific lists of event types (such as *enter bathroom, close drain* and *fill water* for the "taking a bath" scenario) and participant types (such as *bather, water* and *bathtub*). Thus, stories in InScript generally realize longer event chains associated with a single script, which makes them particularly appropriate to our purpose.

We use gold-standard event- and participant-type annotation to study the influence of script knowledge on the expectation of discourse referents. In addition, InScript provides coreference annotation, which makes it possible to keep track of the mentioned discourse referents at each point in the story. We use this information in the computational model of DR prediction and in the DR guessing experiment described in the next section. An example of an annotated InScript story is shown in Figure 6.1.

(I)$^{(1)}_{\text{P\_bather}}$ [**decided**]$_{\text{E\_wash}}$ to take a (bath)$^{(2)}_{\text{P\_bath}}$ yesterday afternoon after working out . Once (I)$^{(1)}_{\text{P\_bather}}$ got back home , (I)$^{(1)}_{\text{P\_bather}}$ [**walked**]$_{\text{E\_enter\_bathroom}}$ to (my)$^{(1)}_{\text{P\_bather}}$ (bathroom)$^{(3)}_{\text{P\_bathroom}}$ and first quickly scrubbed the (bathroom tub)$^{(4)}_{\text{P\_bathtub}}$ by [**turning on**]$_{\text{E\_turn\_water\_on}}$ the (water)$^{(5)}_{\text{P\_water}}$ and rinsing (it)$^{(4)}_{\text{P\_bathtub}}$ clean with a rag . After (I)$^{(1)}_{\text{P\_bather}}$ finished , (I)$^{(1)}_{\text{P\_bather}}$ [**plugged**]$_{\text{E\_close\_drain}}$ the (tub)$^{(4)}_{\text{P\_bathtub}}$ and began [**filling**]$_{\text{E\_fill\_water}}$ (it)$^{(4)}_{\text{P\_bathtub}}$ with warm (water)$^{(5)}_{\text{P\_water}}$ set at about 98 (degrees)$^{(6)}_{\text{P\_temperature}}$.

Fig. 6.1 An excerpt from a story in the InScript corpus. The referring expressions are in parentheses, and the corresponding discourse referent label is given by the superscript. Referring expressions of the same discourse referent have the same color and superscript number. Script-relevant events are in square brackets and colored in orange. Event type is indicated by the corresponding subscript.

6.3 Referent Cloze Task

We use the InScript corpus to develop computational models for the prediction of discourse referents (DRs) and to evaluate their prediction accuracy. This can be done by testing how often our models manage to reproduce the original discourse referent (cf. also the "narrative cloze" task by Chambers and Jurafsky (2008), which tests whether a verb together with a role can be correctly guessed by a model). However, we do not only want to predict the "correct" DRs in a text, but also to model human expectation of DRs in context. To empirically assess human expectation, we created an additional database of crowdsourced human predictions of discourse referents in context using Amazon Mechanical Turk. The design of our experiment closely resembles the guessing game of Tily and Piantadosi (2009) but extends it in a substantial way.

Workers had to read stories of the InScript corpus[1] and guess upcoming participants: for each target NP, workers were shown the story up to this NP excluding the NP itself, and they were asked to guess the next person or object most likely to be referred to. In case they decided in favor of a discourse referent already mentioned, they had to choose

[1]The corpus is available at : http://www.sfb1102.uni-saarland.de/?page_id=2582

(I)$^{(1)}$ decided to take a (bath)$^{(2)}$ yesterday afternoon after working out . Once (I)$^{(1)}$ got back home , (I)$^{(1)}$ walked to (my)$^{(1)}$ (bathroom)$^{(3)}$ and first quickly scrubbed the (bathroom tub)$^{(4)}$ by turning on the (water)$^{(5)}$ and rinsing (it)$^{(4)}$ clean with a rag . After (I)$^{(1)}$ finished , (I)$^{(1)}$ plugged **XXXXXX**

Fig. 6.2 An illustration of the Mechanical Turk experiment for the referent cloze task. Workers are supposed to guess the upcoming referent (indicated by **XXXXXX** above). They can either choose from the previously activated referents, or they can write something new.

among the available discourse referents by clicking an NP in the preceding text, i.e., some noun with a specific, coreference-indicating color; see Figure 6.2. Otherwise, they would click the "New" button, and would in turn be asked to give a short description of the new person or object they expected to be mentioned. The percentage of guesses that agree with the actually referred to entity was taken as a basis for estimating the surprisal.

The experiment was done for all stories of the test set: 182 stories (20%) of the InScript corpus, evenly taken from all scenarios. Since our focus is on the effect of script knowledge, we only considered those NPs as targets that are direct dependents of script-related events. Guessing started from the third sentence only in order to ensure that a minimum of context information was available. To keep the complexity of the context manageable, we restricted guessing to a maximum of 30 targets and skipped the rest of the story (this applied to 12% of the stories). We collected 20 guesses per NP for 3346 noun phrase instances, which amounts to a total of around 67K guesses. Workers selected a context NP in 68% of cases and "New" in 32% of cases.

Our leading hypothesis is that script knowledge substantially influences human expectation of discourse referents. The guessing experiment provides a basis to estimate human expectation of already mentioned DRs (the number of clicks on the respective NPs in text). However, we expect that script knowledge has a particularly strong

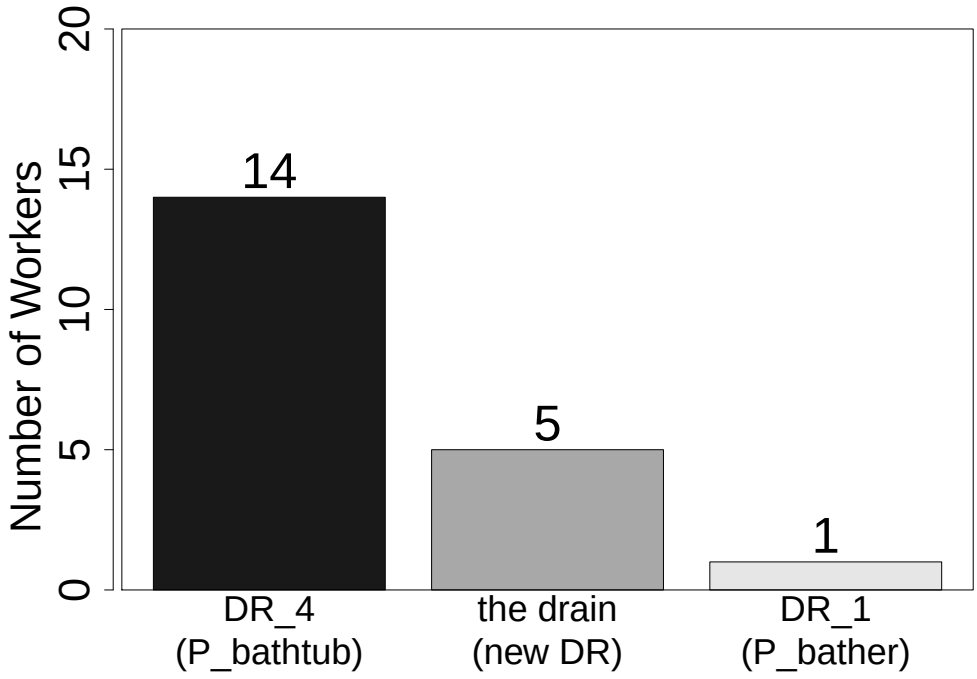

Fig. 6.3 Response of workers corresponding to the story in Fig. 6.2. Workers guessed two already activated discourse referents (DR) DR_4 and DR_1. Some of the workers also chose the "new" option and wrote different lexical variants of "bathtub drain", a new DR corresponding to the participant type "the drain".

influence in the case of first mentions. Once a script is evoked in a text, we assume that the full script structure, including all participants, is activated and available to the reader.

Tily and Piantadosi (2009) are interested in second mentions only and therefore do not make use of the worker-generated noun phrases classified as "New". To study the effect of activated but not explicitly mentioned participants, we carried out a subsequent annotation step on the worker-generated noun phrases classified as "New". We presented annotators with these noun phrases in their contexts (with co-referring NPs marked by color, as in the M-Turk experiment) and, in addition, displayed all participant types of the relevant script (i.e., the script associated with the text in the InScript corpus). Annotators did not see the "correct" target NP. We asked annotators

to either (1) select the participant type instantiated by the NP (if any), (2) label the NP as unrelated to the script or (3) link the NP to an overt antecedent in the text, in the case that the NP is actually a second mention that had been erroneously labeled as new by the worker. Option (1) provides a basis for a fine-grained estimation of first-mention DRs. Option (3), which we added when we noticed the considerable number of overlooked antecedents, serves as correction of the results of the M-Turk experiment. Out of the 22K annotated "New" cases, 39% were identified as second mentions, 55% were linked to a participant type, and 6% were classified as really novel.

6.4 Referent Prediction Model

In this section, we describe the model we use to predict upcoming discourse referents (DRs).

6.4.1 Model

Our model should not only assign probabilities to DRs already explicitly introduced in the preceding text fragment (e.g., "bath" or "bathroom" for the cloze task in Figure 6.2) but also reserve some probability mass for 'new' DRs, i.e., DRs activated via the script context or completely novel ones not belonging to the script. In principle, different variants of the activation mechanism must be distinguished. For many participant types, a single participant belonging to a specific semantic class is expected (referred to with *the bathtub* or *the soap*). In contrast, the "towel" participant type may activate a set of objects, elements of which then can be referred to with *a towel* or *another towel*. The "bath means" participant type may even activate a group of DRs belonging to different semantic classes (e.g., *bubble bath* and *salts*). Since it is not feasible to enumerate all potential participants, for 'new' DRs we only predict their participant

type ("bath means" in our example). In other words, the number of categories in our model is equal to the number of previously introduced DRs plus the number of participant types of the script plus 1, reserved for a new DR not corresponding to any script participant (e.g., *cellphone*). In what follows, we slightly abuse the terminology and refer to all these categories as discourse referents.

Unlike standard co-reference models, which predict co-reference chains relying on the entire document, our model is incremental; that is, when predicting a discourse referent $d^{(t)}$ at a given position t, it can look only in the history $h^{(t)}$ (i.e., the preceding part of the document), excluding the referring expression (RE) for the predicted DR. We also assume that past REs are correctly resolved and assigned to correct participant types (PTs). Typical NLP applications use automatic coreference resolution systems, but since we want to model human behavior, this might be inappropriate, since an automated system would underestimate human performance. This may be a strong assumption, but for reasons explained above, we use gold standard past REs.

We use the following log-linear model ("softmax regression"):

$$p(d^{(t)} = d|h^{(t)}) = \frac{\exp(\mathbf{w}^T\mathbf{f}(d, h^{(t)}))}{\sum_{d'}\exp(\mathbf{w}^T\mathbf{f}(d', h^{(t)}))} \tag{6.1}$$

where $\mathbf{f}$ is the feature function we will discuss in the following subsection, $\mathbf{w}$ are model parameters, and the summation in the denominator is over the set of categories described above.

Some of the features included in $\mathbf{f}$ are a function of the predicate syntactically governing the unobservable target RE (corresponding to the DR being predicted). However, in our incremental setting, the predicate is not available in the history $h^{(t)}$ for subject NPs. In this case, we use an additional probabilistic model, which estimates the probability of the predicate v given the context $h^{(t)}$, and marginalize out its predictions:

Feature	Type
Recency	Shallow Linguistic
Frequency	Shallow Linguistic
Grammatical function	Shallow Linguistic
Previous subject	Shallow Linguistic
Previous object	Shallow Linguistic
Previous RE type	Shallow Linguistic
Selectional preferences	Linguistic
Participant type fit	Script
Predicate schemas	Script

Table 6.1 Summary of feature types

$$p(d^{(t)}=d|h^{(t)})=\sum_{v}p(v|h^{(t)})\frac{\exp(\mathbf{w}^T\mathbf{f}(d,h^{(t)},v))}{\sum_{d'}\exp(\mathbf{w}^T\mathbf{f}(d',h^{(t)},v))} \tag{6.2}$$

The predicate probabilities $p(v|h^{(t)})$ are computed based on the sequence of preceding predicates (i.e., ignoring any other words) using the recurrent neural network language model estimated on our training set.[2] The expression $\mathbf{f}(d,h^{(t)},v)$ denotes the feature function computed for the referent d, given the history composed of $h^{(t)}$ and the predicate v.

6.4.2 Features

Our features encode properties of a DR as well as characterize its compatibility with the context. We face two challenges when designing our features. First, although the sizes of our datasets are respectable from the script annotation perspective, they are too small to learn a richly parameterized model. For many of our features, we address

[2]We used RNNLM toolkit (Mikolov et al., 2010, 2011) with default settings.

this challenge by using external word embeddings[3] and associate parameters with some simple similarity measures computed using these embeddings. Consequently, there are only a few dozen parameters which need to be estimated from scenario-specific data. Second, in order to test our hypothesis that script information is beneficial for the DR prediction task, we need to disentangle the influence of script information from general linguistic knowledge. We address this by carefully splitting the features apart, even if it prevents us from modeling some interplay between the sources of information. We will describe both classes of features below; see also a summary in Table 6.1.

Shallow Linguistic Features

These features are based on Tily and Piantadosi (2009). In addition, we consider a selectional preference feature.

Recency feature. This feature captures the distance $l_t(d)$ between the position t and the last occurrence of the candidate DR d. As a distance measure, we use the number of sentences from the last mention and exponentiate this number to make the dependence more extreme; only very recent DRs will receive a noticeable weight: $\exp(-l_t(d))$. This feature is set to 0 for new DRs.

Frequency. The frequency feature indicates the number of times the candidate discourse referent d has been mentioned so far. We do not perform any bucketing.

Grammatical function. This feature encodes the dependency relation assigned to the head word of the last mention of the DR or a special `none` label if the DR is new.

Previous subject indicator. This binary feature indicates whether the candidate DR d is coreferential with the subject of the previous verbal predicate.

Previous object indicator. The same but for the object position.

[3] We use 300-dimensional word embeddings estimated on Wikipedia with the skip-gram model of Mikolov et al. (2013): https://code.google.com/p/word2vec/

Previous RE type. This three-valued feature indicates whether the previous mention of the candidate DR d is a pronoun, a non-pronominal noun phrase, or has never been observed before.

Selectional Preferences Feature

The selectional preference feature captures how well the candidate DR d fits a given syntactic position r of a given verbal predicate v. It is computed as the cosine similarity $\text{sim}_{\cos}(\mathbf{x}_d^T, \mathbf{x}_{v,r})$ of a vector-space representation of the DR $\mathbf{x}_d$ and a structured vector-space representation of the predicate $\mathbf{x}_{v,r}$. The similarities are calculated using a Distributional Memory approach similar to that of Baroni and Lenci (2010). Their structured vector space representation has been shown to work well on tasks that evaluate correlation with human thematic fit estimates (Baroni and Lenci, 2010; Baroni et al., 2014; Sayeed et al., 2016) and is thus suited to our task.

The representation $\mathbf{x}_d$ is computed as an average of head word representations of all the previous mentions of DR d, where the word vectors are obtained from the TypeDM model of Baroni and Lenci (2010). This is a count-based, third-order co-occurrence tensor whose indices are a word w_0, a second word w_1, and a complex syntactic relation r, which is used as a stand-in for a semantic link. The values for each (w_0, r, w_1) cell of the tensor are the local mutual information (LMI) estimates obtained from a dependency-parsed combination of large corpora (ukWaC, BNC, and Wikipedia).

Our procedure has some differences from that of Baroni and Lenci. For example, for estimating the fit of an alternative new DR (in other words, $\mathbf{x}_d$ based on no previous mentions), we use an average over head words of all REs in the training set, a "null referent." $\mathbf{x}_{v,r}$ is calculated as the average of the top 20 (by LMI) r-fillers for v in TypeDM; in other words, the prototypical instrument of *rub* may be represented by summing vectors like *towel, soap, eraser, coin...* If the predicate has not yet been

(I)$^{(1)}$ decided to take a (bath)$^{(2)}$ yesterday afternoon after working out . (I)$^{(1)}$ was getting ready to go out and needed to get cleaned before (I)$^{(1)}$ went so (I)$^{(1)}$ decided to take a (bath)$^{(2)}$. (I)$^{(1)}$ filled the (bathtub)$^{(3)}$ with warm (water)$^{(4)}$ and added some (bubble bath)$^{(5)}$. (I)$^{(1)}$ got undressed and stepped into the (water)$^{(4)}$. (I)$^{(1)}$ grabbed the (soap)$^{(5)}$ and rubbed it on (my)$^{(1)}$ (body)$^{(7)}$ and rinsed XXXXXX

Fig. 6.4 An example of the referent cloze task. Similar to the Mechanical Turk experiment (Figure 6.2), our referent prediction model is asked to guess the upcoming DR.

encountered (as for subject positions), scores for all scenario-relevant verbs are emitted for marginalization.

Script Features

In this section, we describe features which rely on script information. Our goal will be to show that such common-sense information is beneficial in performing DR prediction. We consider only two script features.

Participant type fit

This feature characterizes how well the participant type (PT) of the candidate DR d fits a specific syntactic role r of the governing predicate v; it can be regarded as a generalization of the selectional preference feature to participant types and also its specialization to the considered scenario. Given the candidate DR d, its participant type p, and the syntactic relation r, we collect all the predicates in the training set which have the participant type p in the position r. The embedding of the DR $\mathbf{x}_{p,r}$ is given by the average embedding of these predicates. The feature is computed as the dot product of $\mathbf{x}_{p,r}$ and the word embedding of the predicate v.

Predicate schemas

The following feature captures a specific aspect of knowledge about prototypical sequences of events. This knowledge is called *predicate schemas* in the recent co-

reference modeling work of Peng et al. (2015). In predicate schemas, the goal is to model pairs of events such that if a DR d participated in the first event (in a specific role), it is likely to participate in the second event (again, in a specific role). For example, in the restaurant scenario, if one observes a phrase *John ordered*, one is likely to see *John waited* somewhere later in the document. Specific arguments are not that important (where it is *John* or some other DR); what is important is that the argument is reused across the predicates. This would correspond to the rule *X-subject-of-order* $\rightarrow$ *X-subject-of-eat*.[4] Unlike the previous work, our dataset is small, so we cannot induce these rules directly as there will be very few rules, and the model would not generalize to new data well enough. Instead, we again encode this intuition using similarities in the real-valued embedding space.

Recall that our goal is to compute a feature $\varphi(d, h^{(t)})$ indicating how likely a potential DR d is to follow, given the history $h^{(t)}$. For example, imagine that the model is asked to predict the DR marked by XXXXXX in Figure 6.4. Predicate-schema rules can only yield previously introduced DRs, so the score $\varphi(d, h^{(t)}) = 0$ for any new DR d. Let us use "soap" as an example of a previously introduced DR and see how the feature is computed. In order to choose which inference rules can be applied to yield "soap", we can inspect Figure 6.4. There are only two preceding predicates which have DR "soap" as their object (*rubbed* and *grabbed*), resulting in two potential rules *X-object-of-grabbed* $\rightarrow$ *X-object-of-rinsed* and *X-object-of-rubbed* $\rightarrow$ *X-object-of-rinsed*. We define the score $\varphi(d, h^{(t)})$ as the average of the rule scores. More formally, we can write

$$\varphi(d, h^{(t)}) = \frac{1}{|N(d, h^{(t)})|} \sum_{(u,v,r) \in N(d,h^{(t)})} \psi(u, v, r), \tag{6.3}$$

where $\psi(u, v, r)$ is the score for a rule *X-r-of-u* $\rightarrow$ *X-r-of-v*, $N(d, h^{(t)})$ is the set of

[4] In this work, we limit ourselves to rules where the syntactic function is the same on both sides of the rule. In other words, we can, in principle, encode the pattern *X pushed Y* $\rightarrow$ *X apologized* but not the pattern *X pushed Y* $\rightarrow$ *Y cried*.

applicable rules, and $|N(d, h^{(t)})|$ denotes its cardinality.[5] We define $\varphi(d, h^{(t)})$ as 0, when the set of applicable rules is empty (i.e. $|N(d, h^{(t)})| = 0$).

The scoring function $\psi(u, v, r)$ is a linear function of a joint embedding $\mathbf{x}_{u,v}$ of verbs u and v:

$$\psi(u, v, r) = \alpha_r^T \mathbf{x}_{u,v}. \tag{6.4}$$

The two remaining questions are (1) how to define the joint embeddings $\mathbf{x}_{u,v}$, and (2) how to estimate the parameter vector α_r. The joint embedding of two predicates, $\mathbf{x}_{u,v}$, can, in principle, be any composition function of embeddings of u and v, for example their sum or component-wise product. Inspired by Bordes et al. (2013), we use the difference between the word embeddings:

$$\psi(u, v, r) = \alpha_r^T (\mathbf{x}_u - \mathbf{x}_v), \tag{6.5}$$

where $\mathbf{x}_u$ and $\mathbf{x}_v$ are external embeddings of the corresponding verbs. Encoding the succession relation as translation in the embedding space has one desirable property: the scoring function will be largely agnostic to the morphological form of the predicates. For example, the difference between the embeddings of *rinsed* and *rubbed* is very similar to that of *rinse* and *rub* (Botha and Blunsom, 2014), so the corresponding rules will receive similar scores. Now, we can rewrite the equation (6.3) as

$$\varphi(d, h^{(t)}) = \alpha_{r(h^{(t)})}^T \frac{\sum_{(u,v,r) \in N(d,h^{(t)})} (\mathbf{x}_u - \mathbf{x}_v)}{|N(d, h^{(t)})|} \tag{6.6}$$

where $r(h^{(t)})$ denotes the syntactic function corresponding to the DR being predicted (object in our example).

[5] In all our experiments, rather than considering all potential predicates in the history to instantiate rules, we take into account only 2 preceding verbs. In other words, u and v can be interleaved by at most one verb and $|N(d, h^{(t)})|$ is in $\{0, 1, 2\}$.

As for the parameter vector α_r, there are again a number of potential ways how it can be estimated. For example, one can train a discriminative classifier to estimate the parameters. However, we opted for a simpler approach—we set it equal to the empirical estimate of the expected feature vector $x_{u,v}$ on the training set:[6]

$$\alpha_r = \frac{1}{D_r} \sum_{l,t} \delta_r(r(h^{(l,t)})) \sum_{(u,v,r') \in N(d^{(l,t)}, h^{(l,t)})} (\mathbf{x}_u - \mathbf{x}_v), \tag{6.7}$$

where l refers to a document in the training set, t is (as before) a position in the document, and $h^{(l,t)}$ and $d^{(l,t)}$ are the history and the correct DR for this position, respectively. The term $\delta_r(r')$ is the Kronecker delta, which equals 1 if $r = r'$ and 0, otherwise. D_r is the total number of rules for the syntactic function r in the training set:

$$D_r = \sum_{l,t} \delta_r(r(h^{(l,t)})) \times |N(d^{(l,t)}, h^{(l,t)})|.$$

Let us illustrate the computation with an example. Imagine that our training set consists of the document in Figure 1, and the trained model is used to predict the upcoming DR in our referent cloze example (Figure 6.4). The training document includes the pair *X-object-of-scrubbed* → *X-object-of-rinsing*, so the corresponding term ($\mathbf{x}_{scrubbed}$ - $\mathbf{x}_{rinsing}$) participates in the summation (6.7) for α_{obj}. As we rely on external embeddings, which encode semantic similarities between lexical items, the dot product of this term and ($\mathbf{x}_{rubbed}$ - $\mathbf{x}_{rinsed}$) will be high.[7] Consequently, $\varphi(d, h^{(t)})$ is expected to be positive for d = "soap", thus predicting "soap" as the likely forthcoming DR. Unfortunately, there are other terms ($\mathbf{x_u} - \mathbf{x_v}$) both in expression (6.7) for α_{obj} and in

[6] This essentially corresponds to using the Naive Bayes model with the simplistic assumption that the score differences are normally distributed with spherical covariance matrices.

[7] The score would have been even higher, should the predicate be in the morphological form *rinsing* rather than *rinsed*. However, embeddings of *rinsing* and *rinsed* would still be sufficiently close to each other for our argument to hold.

Model Name	**Feature Types**	**Features**
Base	Shallow Linguistic Features	Recency, Frequency, Grammatical function, Previous subject, Previous object
Linguistic	Shallow Linguistic Features + Linguistic Feature	Recency, Frequency, Grammatical function, Previous subject, Previous object + Selectional Preferences
Script	Shallow Linguistic Features + Linguistic Feature + Script Features	Recency, Frequency, Grammatical function, Previous subject, Previous object + Selectional Preferences + Participant type fit, Predicate schemas

Table 6.2 Summary of model features

expression (6.6) for $\varphi(d, h^{(t)})$. These terms may be irrelevant to the current prediction, as *X-object-of-plugged* $\rightarrow$ *X-object-of-filling* from Figure 1, and may not even encode any valid regularities, as *X-object-of-got* $\rightarrow$ *X-object-of-scrubbed* (again from Figure 1). This may suggest that our feature will be too contaminated with noise to be informative for making predictions. However, recall that independent random vectors in high dimensions are almost orthogonal, and, assuming they are bounded, their dot products are close to zero. Consequently, the products of the relevant ("non-random") terms, in our example ($\mathbf{x}_{scrubbed}$ - $\mathbf{x}_{rinsing}$) and ($\mathbf{x}_{rubbed}$ - $\mathbf{x}_{rinsed}$), are likely to overcome the ("random") noise. As we will see in the ablation studies, the predicate-schema feature is indeed predictive of a DR and contributes to the performance of the full model.

6.4.3 Experiments

We would like to test whether our model can produce accurate predictions and whether the model's guesses correlate well with human predictions for the referent cloze task.

In order to be able to evaluate the effect of script knowledge on referent predictability, we compare three models: our full *Script model* uses all of the features introduced in section 4.2; the *Linguistic model* relies only on the 'linguistic features' and not the script-specific ones; and the *Base model* includes all the shallow linguistic features. The Base model differs from the Linguistic model in that it does not model selectional preferences. Table 6.2 summarizes features used in different models.

The data set was randomly divided into training (70%), development (10%, 91 stories from 10 scenarios), and test (20%, 182 stories from 10 scenarios) sets. The feature weights were learned using L-BFGS (Byrd et al., 1995) to optimize the log-likelihood.

Evaluation against original referents. We calculated the percentage of correct DR predictions. See Table 6.3 for the averages across 10 scenarios. We can see that the task appears hard for humans: their average performance reaches only 73% accuracy. As expected, the Base model is the weakest system (with accuracy of 31%). Modeling selectional preferences yields an extra 18% in accuracy (Linguistic model). The key finding is that incorporation of script knowledge increases the accuracy by a further 13%, although it is still far behind human performance (62% vs. 73%). Besides accuracy, we use perplexity, which we computed not only for all our models but also for human predictions. This was possible as each task was solved by multiple humans. We used unsmoothed normalized guess frequencies as the probabilities. As we can see from Table 6.3, the perplexity scores are consistent with the accuracies: the script model again outperforms other methods, and, as expected, all the models are weaker than humans.

Scenario	Human Model		Script Model		Linguistic Model		Base Model	
	Accuracy	Perplexity	Accuracy	Perplexity	Accuracy	Perplexity	Accuracy	Perplexity
Grocery shopping	74.80	2.13	68.17	3.16	53.85	6.54	32.89	24.48
Repairing a flat bicycle tyre	78.34	2.72	62.09	3.89	51.26	6.38	29.24	19.08
Riding a public bus	72.19	2.28	64.57	3.67	52.65	6.34	32.78	23.39
Getting a haircut	71.06	2.45	58.82	3.79	42.82	7.11	28.70	15.40
Planting a tree	71.86	2.46	59.32	4.25	47.80	7.31	28.14	24.28
Borrowing book from library	77.49	1.93	64.07	3.55	43.29	8.40	33.33	20.26
Taking bath	81.29	1.84	67.42	3.14	61.29	4.33	43.23	16.33
Going on a train	70.79	2.39	58.73	4.20	47.62	7.68	30.16	35.11
Baking a cake	76.43	2.16	61.79	5.11	46.40	9.16	24.07	23.67
Flying in an airplane	62.04	3.08	61.31	4.01	48.18	7.27	30.90	30.18
Average	73.63	2.34	62.63	3.88	49.52	7.05	31.34	23.22

Table 6.3 Accuracies (in %) and perplexities for different models and scenarios. The Script model substantially outperforms Linguistic and Base models (with $p < 0.001$, significance tested with McNemar's test (Everitt, 1992)). As expected, the human prediction model outperforms the Script model (with $p < 0.001$, significance tested by McNemar's test).

Model	Accuracy	Perplexity
Linguistic Model	49.52	7.05
Linguistic Model + Predicate Schemas	55.44	5.88
Linguistic Model + Participant type fit	58.88	4.29
Full Script Model (both features)	62.63	3.88

Table 6.4 Accuracies from ablation experiments.

As we used two sets of script features, capturing different aspects of script knowledge, we performed extra ablation studies (Table 6.4). The experiments confirm that both feature sets were beneficial.

Evaluation against human expectations. In the previous subsection, we demonstrated that the incorporation of selectional preferences and, perhaps more interestingly, the integration of automatically acquired script knowledge lead to improved accuracy in predicting discourse referents. Now we turn to another question raised in the introduction: does incorporation of this knowledge make our predictions more human-like? In other words, are we able to accurately estimate human expectations? This includes not only being sufficiently accurate but also making the same kind of incorrect predictions.

In this evaluation, we therefore use human guesses collected during the referent cloze task as our target. We then calculate the relative accuracy of each computational model. As can be seen in Figure 6.5, the Script model, at approx. 53% accuracy, is a lot more accurate in predicting human guesses than the Linguistic model and the Base model. We can also observe that the margin between the Script model and the Linguistic model is a lot larger in this evaluation than between the Base model and the Linguistic model. This indicates that the model which has access to script knowledge is much more similar to human prediction behavior in terms of top guesses than the script-agnostic models.

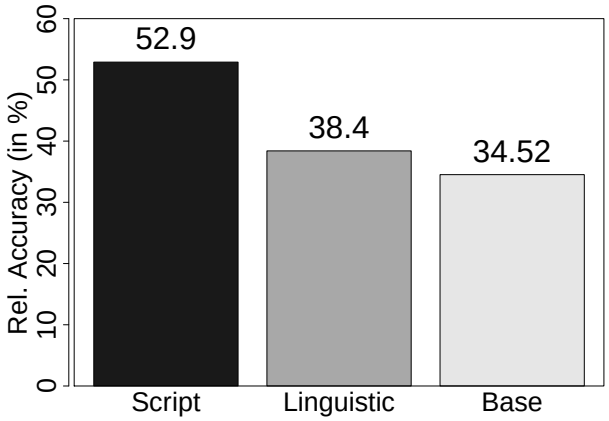

Fig. 6.5 Average relative accuracies of different models w.r.t human predictions.

Now we would like to assess if our predictions are similar as distributions rather than only yielding similar top predictions. In order to compare the distributions, we use the Jensen-Shannon divergence (JSD), a symmetrized version of the Kullback-Leibler divergence.

Intuitively, JSD measures the distance between two probability distributions. A smaller JSD value is indicative of more similar distributions. Figure 6.6 shows that the probability distributions resulting from the Script model are more similar to human predictions than those of the Linguistic and Base models.

In these experiments, we have shown that script knowledge improves predictions of upcoming referents and that the script model is the best among our models in approximating human referent predictions.

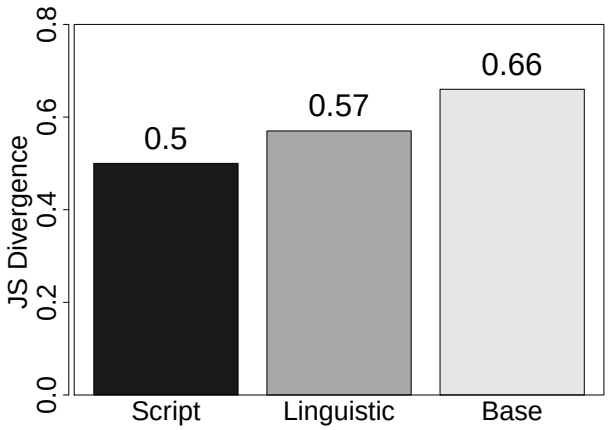

Fig. 6.6 Average Jensen-Shannon divergence between human predictions and models.

6.5 Referring Expression Type Prediction Model (RE Model)

Using the referent prediction models, we next attempt to replicate Tily and Piantadosi's findings that the choice of the type of referring expression (pronoun or full NP) depends in part on the predictability of the referent.

6.5.1 Uniform Information Density Hypothesis

The uniform information density (UID) hypothesis suggests that speakers tend to convey information at a uniform rate (Jaeger, 2010). Applied to the choice of referring expression type, it would predict that a highly predictable referent should be encoded using a short code (here a pronoun), while an unpredictable referent should be encoded using a longer form (here a full NP). Information density is measured using the information-theoretic measure of the *surprisal* S of a message m_i:

$$S(m_i) = -\log P(m_i \mid context)$$

UID has been very successful in explaining a variety of linguistic phenomena; see Jaeger et al. (2016). There is, however, controversy about whether UID affects

pronominalization. Tily and Piantadosi (2009) report evidence that writers are more likely to refer using a pronoun or proper name when the referent is easy to guess and use a full NP when readers have less certainty about the upcoming referent; see also Arnold (2001). But other experiments (using highly controlled stimuli) have failed to find an effect of predictability on pronominalization (Fukumura and Van Gompel, 2010; Rohde and Kehler, 2014; Stevenson et al., 1994). The present study hence contributes to the debate on whether UID affects referring expression choice.

6.5.2 A Model of Referring Expression Choice

Our goal is to determine whether referent predictability (quantified in terms of surprisal) is correlated with the type of referring expression used in the text. Here we focus on the distinction between pronouns and full noun phrases. Our data also contains a small percentage (ca. 1%) of proper names (like "John"). Due to this small class size and earlier findings that proper nouns behave much like pronouns (Tily and Piantadosi, 2009), we combined pronouns and proper names into a single class of short encodings.

For the referring expression type prediction task, we estimate the surprisal of the referent from each of our computational models from Section 6.4 as well as the human cloze task. The surprisal of an upcoming discourse referent $d^{(t)}$ based on the previous context $h^{(t)}$ is thereby estimated as:

$$S(d^{(t)}) = -\log\ p(d^{(t)} \mid h^{(t)}) \tag{6.8}$$

In order to determine whether referent predictability has an effect on referring expression type *over and above* other factors that are known to affect the choice of referring expression, we train a logistic regression model with referring expression type as a response variable and discourse referent predictability as well as a large set of other linguistic factors (based on Tily and Piantadosi, 2009) as explanatory variables. The

model is defined as follows:

$$p(n^{(t)} = n|d^{(t)}, h^{(t)}) = \frac{\exp(\mathbf{v}^T\mathbf{g}(n, d^t, h^{(t)}))}{\sum_{n'} \exp(\mathbf{v}^T\mathbf{g}(n', d^t, h^{(t)}))}, \tag{6.9}$$

where $d^{(t)}$ and $h^{(t)}$ are defined as before, $\mathbf{g}$ is the feature function, and $\mathbf{v}$ is the vector of model parameters. The summation in the denominator is over NP types (full NP vs. pronoun/proper noun).

6.5.3 RE Model Experiments

We ran four different logistic regression models. These models all contained exactly the same set of linguistic predictors but differed in the estimates used for referent type surprisal and residual entropy. One logistic regression model used surprisal estimates based on the human referent cloze task, while the three other models used estimates based on the three computational models (Base, Linguistic and Script). For our experiment, we are interested in the choice of referring expression type for those occurrences of references where a "real choice" is possible. We therefore exclude for our analysis reported below all first mentions as well as all first and second person pronouns (because there is no optionality in how to refer to first or second person). This subset contains 1345 data points.

6.5.4 Results

The results of all four logistic regression models are shown in Table 6.5. We first take a look at the results for the linguistic features. While there is a bit of variability in terms of the exact coefficient estimates between the models (this is simply due to small correlations between these predictors and the predictors for surprisal), the effect of all of these features is largely consistent across models. For instance, the positive

coefficients for the recency feature mean that when a previous mention happened very recently, the referring expression is more likely to be a pronoun (and not a full NP).

The coefficients for the surprisal estimates of the different models are, however, not significantly different from zero. Model comparison shows that they do not improve model fit. We also used the estimated models to predict referring expression type on new data and again found that surprisal estimates from the models did not improve prediction accuracy. This effect even holds for our human cloze data. Hence, it cannot be interpreted as a problem with the models—even human predictability estimates are, for this dataset, not predictive of referring expression type.

We also calculated regression models for the full dataset including first and second person pronouns as well as first mentions (3346 data points). The results for the full dataset are fully consistent with the findings shown in Table 6.5: there was no significant effect of surprisal on referring expression type.

This result contrasts with the findings by Tily and Piantadosi (2009),who reported a significant effect of surprisal on RE type for their data. In order to replicate their settings as closely as possible, we also included residualEntropy as a predictor in our model (see last predictor in Table 6.5); however, this did not change the results.

	Estimate				Std. Error				Pr(>| z |)			
	Human	Script	Linguistic	Base	Human	Script	Linguistic	Base	Human	Script	Linguistic	Base
(Intercept)	-3.4	-3.418	-3.245	-3.061	0.244	0.279	0.321	0.791	<2e-16 ***	<2e-16 ***	<2e-16 ***	0.00011 ***
recency	1.322	1.322	1.324	1.322	0.095	0.095	0.096	0.097	<2e-16 ***	<2e-16 ***	<2e-16 ***	<2e-16 ***
frequency	0.097	0.103	0.112	0.114	0.098	0.097	0.098	0.102	0.317	0.289	0.251	0.262
pastObj	0.407	0.396	0.423	0.395	0.293	0.294	0.295	0.3	0.165	0.178	0.151	0.189
pastSubj	-0.967	-0.973	-0.909	-0.926	0.559	0.564	0.562	0.565	0.0838 .	0.0846 .	0.106	0.101
pastExpPronoun	1.603	1.619	1.616	1.602	0.21	0.207	0.208	0.245	2.19e-14 ***	5.48e-15 ***	7.59e-15 ***	6.11e-11 ***
depTypeSubj	2.939	2.942	2.656	2.417	0.299	0.347	0.429	1.113	<2e-16 ***	<2e-16 ***	5.68e-10 ***	0.02994 *
depTypeObj	1.199	1.227	0.977	0.705	0.248	0.306	0.389	1.109	1.35e-06 ***	6.05e-05 ***	0.0119 *	0.525
surprisal	-0.04	-0.006	0.002	-0.131	0.099	0.097	0.117	0.387	0.684	0.951	0.988	0.735
residualEntropy	-0.009	0.023	-0.141	-0.128	0.088	0.128	0.168	0.258	0.916	0.859	0.401	0.619

Table 6.5 Coefficients obtained from regression analysis for different models. Two NP types considered: full NP and Pronoun/ProperNoun, with base class full NP. Significance: '***' < 0.001, '**' < 0.01, '*' < 0.05, and '.' < 0.1.

6.6 Conclusion

Our study on incrementally predicting discourse referents showed that script knowledge is a highly important factor in determining human discourse expectations. Crucially, the computational modeling approach allowed us to tease apart the different factors that affect human prediction, as we cannot manipulate this in humans directly (by asking them to "switch off" their common-sense knowledge).

By modeling common-sense knowledge in terms of event sequences and event participants, our model captures many more long-range dependencies than normal language models. The script knowledge is automatically induced by our model from crowdsourced scenario-specific text collections.

In a second study, we set out to test the hypothesis that uniform information density affects referring expression type. This question is highly controversial in the literature: while Tily and Piantadosi (2009) find a significant effect of surprisal on referring expression type in a corpus study very similar to ours, other studies that use a more tightly controlled experimental approach have not found an effect of predictability on RE type (Fukumura and Van Gompel, 2010; Rohde and Kehler, 2014; Stevenson et al., 1994). The present study, while replicating exactly the setting of T&P in terms of features and analysis, did not find support for a UID effect on RE type. The difference in results between T&P 2009 and our results could be due to the different corpora and text sorts that were used; specifically, we would expect that larger predictability effects might be observable at script boundaries, rather than within a script, as is the case in our stories.

A next step in moving our participant prediction model towards NLP applications would be to replicate our modeling results on automatic text-to-script mapping instead of gold-standard data as done here (in order to approximate a human level of processing).

Furthermore, we aim to move to more complex text types that include reference to several scripts.

CHAPTER 7

Conclusion and Future Directions

If I have seen further it is by standing on the shoulders of giants.

**

Sir Isaac Newton

7.1 Thesis Summary

This dissertation advocated the idea of including common sense knowledge in natural language understanding systems. As shown by various examples in Chapter 1, common sense knowledge would be beneficial for systems modeling natural language. In this dissertation, we modeled common sense knowledge about everyday activities via scripts.

Script have underpinnings in cognitive psychology as described in Chapter 3. Computational modeling of scripts can be traced back to the late 1970's when Schank and Abelson (1977) formulated the script theory and applied it within a story understanding system. Recent work on scripts has applied probabilistic and statistical techniques for modeling scripts.

In order to model script knowledge, three main tasks need to be addressed: event ordering, event paraphrasing, and event prediction. In Chapter 4, we proposed a neural network based probabilistic model for event ordering. The model learns dense vector based representations for events. These representations capture semantic properties of the events and achieve better generalization than conventional count based models. The proposed model outperforms current graph based and count based approaches for script modeling. To address the event prediction task, we proposed an extension of the event ordering model. The model addresses the shortcomings of the current count based methods namely, sparsity and poor atomic event representation. We also pointed out the problems with the current accuracy metric for narrative cloze task. We proposed another alternative metric, i.e. event perplexity. Additionally, we proposed an alternative task (adversarial narrative cloze) for evaluating script models.

In Chapter 5, we explained the limitations of previously available corpora used for modeling scripts. Previously, corpora used for modeling script knowledge have consisted of narratives instantiating multiple scenarios. This makes it difficult to

study the contribution of script knowledge in isolation. We addressed this challenge by creating a new corpus for scripts: InScript. The newly introduced corpus Inscript is a unique resource for script research. It is a collection of 910 stories, where each story instantiates a single scenario, e.g. baking a cake, getting a haircut, etc. The stories are manually annotated with script specific event types and participant types. Additionally, the stories are annotated with coreference chains. InScript would serve as a useful resource for pursuing further research on scripts and common sense knowledge in general.

As a final step in the dissertation, in Chapter 6, we presented our attempts to study the influence of script knowledge on language comprehension. We experimented with the task of predicting an upcoming discourse referent (DR) in a narrative. To get an upper bound on the performance, we ran the DR prediction experiment with humans. We experimented with different models, some with script related features and some without script related features. We tested our hypothesis with the help of the Inscript corpus. We ran a battery of experiments to study the contribution of different features. In all the experiments, across all the metrics, we observed a significant contribution of script knowledge in predicting the upcoming discourse referent. Our experiments concluded that script knowledge makes a significant contribution towards language comprehension. This is a foundational result and encourages further research on cognitive and computational aspects of scripts.

7.2 Future Directions

In this dissertation, we explored some of the foundational questions concerning script knowledge. We believe this is just the beginning and a lot more needs to be explored. We highlight some of the possible directions that would be interesting from a research

point of view. Note that the ideas presented next are speculative and do not go into details but only outline some new directions which could be explored in the future.

7.2.1 Multi-Script Modeling

The script models explored in this dissertation modeled each scenario in isolation. This rarely happens in the typically occurring natural texts in newspapers, blog stories, etc. Typically, most of the natural texts instantiate several scenarios. For example, a text may describe a story of a person traveling in a bus. While doing so the person may be reviewing a research paper and discussing it with his co-passenger. In this story, events from two script scenarios are instantiated: Riding in a bus and Review a research paper.

One of the ways of modeling such a mixture of scenarios could be via latent variable models. Such natural texts can be modeled using a mixture of scenarios, where each scenario is a latent variable. We do not explicitly propose any model but outline the possible generative process. Text can be generated as follows: a scenario would be sampled from a distribution over scenarios. The event would be generated from the sampled scenario (via some scenario specific event distribution). With the help of the sampled event, one could sample relevant verbal predicate and corresponding participants. Since event types instantiated in the text do not occur in isolation, in the generative model described above, one would have to model dependencies among the events and scenarios, for example with techniques used for modeling dependencies in topic models (or more generally, latent variable models) (Inouye et al., 2014).

7.2.2 Script Modeling via Reinforcement Learning

An ESD for the script could be generated by incrementally predicting events one by one. This is akin to the situation wherein an agent is traversing through the space of

script events. By the space of script events, we refer to a discretized space where each point/state is an event. The agent is at one particular point (event) in the space and takes an action based on current and previous states. The action results in the agent being in another state/event. Every action has a reward associated with it.

Looking from the above described perspective, scripts can be modeled in the framework of reinforcement learning (Kaelbling et al., 1996; Sutton and Barto, 1998) or more generally in the framework of deep reinforcement learning (Li, 2017). In the reinforcement learning setup, an agent would be exploring all the possible states (events in our case) in order to achieve the final goal of completing the scenario. The sequence of actions taken by the agent is the policy for the agent. The policy corresponds to an ESD for the script. The action taken at each state (event) would depend on the previous state and reward for the next state. Our neural event ordering model proposed in Chapter 4 has state of the art performance. Is it possible to approximate the action value function via the neural event ordering model? It seems it could be possible to learn an optimal policy (sequence of events) for a scenario by training via the conventional reinforcement learning setup combined with the neural approximate action-value function. Research in the area of dialogue generation using deep reinforcement learning (Li et al., 2016) may be helpful in pursuing the above-described idea.

7.2.3 Scripts for Coreference Resolution

In Chapter 6, we empirically established that script knowledge makes a significant contribution towards predicting an upcoming discourse referent. An obvious next step is to include script knowledge in coreference resolution systems. This seems to be a natural step in this direction but it comes with its own challenges. Most of the coreference resolution systems are modeled on natural texts, which involve a mixture of scenarios. One would need some technique which would identify the possible scenarios

invoked in the text and add appropriate script specific knowledge about participants for coreference resolution. Recently, there has been work showing the benefits of including semantic knowledge in an existing coreference resolution system (Peng and Roth, 2016). We believe that script knowledge should make a significant contribution to conference resolution.

7.2.4 Scripts and SRL

We have already described the similarities between script participant types and semantic roles. Since scripts work at the level of discourse, is it possible to employ script knowledge for semantic role labeling (SRL)? Intuitively, semantic roles invoked in the current sentence should be influenced by the events and participants in the previous sentences. SRL systems with script knowledge face similar challenges as outlined in Section 7.2.3. Recently, there have been attempts to incorporate discourse-level information in SRL systems (Roth and Lapata, 2015); the idea proposed here, goes in this direction.

7.2.5 Inference via Scripts

In Section 3.2 on page 39, we outlined an example on how script knowledge could be useful for drawing inferences. In order to test the contribution of script knowledge to language understanding and drawing inferences, it might be interesting to create a question answering task based on using the InScript corpus. This is similar to currently prevalent approaches for testing machines' ability to answer questions based on observed text and common sense reasoning, e.g. Facebook's bAbI[1] project (Bordes et al., 2015), SQuAD[2] (Stanford Question Answering Dataset) (Rajpurkar et al.,

[1] https://research.fb.com/downloads/babi/
[2] https://rajpurkar.github.io/SQuAD-explorer/

2016), the Winograd Schema Challenge[3] (Levesque et al., 2011), etc. The proposed question answering dataset would be different from existing ones as it would specifically contain questions which would require drawing inferences based on script knowledge. There could be different categories of questions that could be included in the dataset, for example causal questions, reasoning questions, or questions about identifying participants performing an action.

7.2.6 Multimodal Script Modeling

In this dissertation, we explored one modality for modeling script knowledge, i.e. text. But script knowledge is not restricted to just text. Videos are also a good source of script information. It would be interesting to complement the textual information with information from videos. Previously, attempts (e.g. Regneri et al. (2013); Rohrbach et al. (2012, 2013)) have been made in this direction, but they have been limited to a single domain, i.e. cooking. As explained in the thesis, annotating script oriented text is challenging. Is it possible to complement this with information from videos? With great advances made in the area of deep learning and image/video processing (Karpathy et al., 2014; Szegedy et al., 2015), today we have fairly accurate activity recognition systems (Chen, 2010; Ji et al., 2013; Le et al., 2011; Taylor et al., 2010) available at our disposal. It would be interesting to have models that jointly learn event structure from actions in the videos and event descriptions in the texts.

[3] http://commonsensereasoning.org/winograd.html

References

Robert P. Abelson. Psychological Status of the Script Concept. *American Psychologist*, 36(7):715, 1981.

Woo-Kyoung Ahn, Raymond J. Mooney, William F. Brewer, and Gerald F. DeJong. Schema Acquisition from One Example: Psychological Evidence for Explanation-Based Learning. Technical report, DTIC Document, 1987.

James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

Jennifer E. Arnold. The Effect of Thematic Roles on Pronoun Use and Frequency of Reference Continuation. *Discourse Processes*, 31(2):137–162, 2001.

Bernard J. Baars and Nicole M. Gage. *Cognition, Brain, and Consciousness: Introduction to Cognitive Neuroscience*. Academic Press, 2010.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet Project. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics (ACL-COLING'98)*, pages 86–90, Montreal, Canada, 1998.

David Bamman, Brendan O'Connor, and Noah A. Smith. Learning Latent Personas of Film Characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, page 352, 2014.

Marco Baroni and Alessandro Lenci. Distributional Memory: A General Framework for Corpus-based Semantics. *Computational Linguistics*, 36(4):673–721, 2010.

Marco Baroni and Robert Zamparelli. Nouns Are Vectors, Adjectives Are Matrices: Representing Adjective-noun Constructions in Semantic Space. In *Proceedings of EMNLP*, 2011.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't Count, Predict! A Systematic Comparison of Context-counting Vs. Context-predicting Semantic Vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Daniel Bauer, Hagen Fürstenau, and Owen Rambow. The Dependency-parsed FrameNet Corpus. In *International conference on Language Resources and Evaluation (LREC)*, Istanbul, Turkey, 2012.

Francis S. Bellezza and Gordon H. Bower. Remembering Script-based Text. *Poetics*, 11(1):1–23, 1982.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A Neural Probabilistic Language Model. In *Advances in Neural Information Processing Systems, NIPS*, 2001.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. Global Learning of Typed Entailment Rules. In *Proceedings of ACL*, 2011.

Elisa Bertino, Barbara Catania, and Gian Piero Zarri. *Intelligent Database Systems*. Addison-Wesley, 2001. ISBN 0-201-87736-8.

Eckhard Bick. A FrameNet for Danish. *Nealt Proceedings Series*, 11:34, 2011.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems, NIPS*, 2013.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale Simple Question Answering with Memory Networks. *arXiv preprint arXiv:1506.02075*, 2015.

Lars Borin, Dana Dannélls, Markus Forsberg, Maria Toporowska Gronostaj, and Dimitrios Kokkinakis. The past Meets the Present in Swedish FrameNet++. In *14th EURALEX International Congress*, pages 269–281, 2010.

Jan A. Botha and Phil Blunsom. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of ICML*, 2014.

Gordon H. Bower, John B. Black, and Terrence J. Turner. Scripts in Memory for Text. *Cognitive Psychology*, 11(2):177–220, 1979.

A. Burchardt, K. Erk, A. Frank, A. Kowalski, S. Pado, and M. Pinkal. The SALSA Corpus: A German Corpus Resource for Lexical Semantics. In *LREC*, 2006.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

Xavier Carreras and Lluís Màrquez. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *CoNLL*, 2005.

Nathanael Chambers and Dan Jurafsky. Unsupervised Learning of Narrative Schemas and Their Participants. In *Proceedings of ACL*, 2009.

Nathanael Chambers and Daniel Jurafsky. Unsupervised Learning of Narrative Event Chains. In *Proceedings of ACL*, 2008.

Bo Chen. *Deep Learning of Invariant Spatio-temporal Features from Video*. PhD thesis, University of British Columbia, 2010.

Timothy Chklovski and Patrick Pantel. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of EMNLP*, 2004.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

Richard Edward Cullingford. *Script Application: Computer Understanding of Newspaper Stories.* PhD thesis, Yale University, New Haven, CT, USA, 1977.

Robert Dale, Hermann Moisl, and Harold Somers. A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text, 2000.

D. Das and N.A. Smith. Semi-supervised Frame-semantic Parsing for Unknown Predicates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1435–1444. Association for Computational Linguistics, 2011.

D. Das, N. Schneider, D. Chen, and N.A. Smith. Probabilistic Frame-semantic Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 948–956. Association for Computational Linguistics, 2010.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of LREC*, 2006.

Gerald Francis Dejong, II. *Skimming Stories in Real Time: An Experiment in Integrated Understanding.* PhD thesis, Yale University, New Haven, CT, USA, 1979. AAI7925621.

Martijn Den Uyl and Herre Van Oostendorp. The Use of Scripts in Text Comprehension. *Poetics*, 9(1-3):275–294, 1980.

Pedro Domingos. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World.* Basic Books, 2015.

Georg Dorffner, R. Reilly St, and N. Sharkey. A Step toward Sub-symbolic Language Models without Linguistic Representations. *Connectionist Approaches to Natural Language Processing*, 1, 1992.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12 (Jul):2121–2159, 2011.

Michael G. Dyer. Connectionist Natural Language Processing: A Status Report. In *Computational Architectures Integrating Neural and Symbolic Processes*, pages 389–429. Springer, 1995.

Michael G. Dyer, Richard E. Cullingford, and Sergio Alvarado. Scripts. *Encyclopedia of Artificial Intelligence*, 2:980–994, 1987.

Michael George Dyer. In-Depth Understanding. A Computer Model of Integrated Processing for Narrative Comprehension. Technical report, DTIC Document, 1982.

Jeffrey L. Elman. Distributed Representations, Simple Recurrent Networks, and Grammatical Structure. *Machine learning*, 7(2-3):195–225, 1991.

K. Erk and S. Pado. SHALMANESER–a Toolchain for Shallow Semantic Parsing. In *Proceedings of LREC*, volume 6. Citeseer, 2006.

Brian S. Everitt. *The Analysis of Contingency Tables*. CRC Press, 1992.

Thomas S. Ferguson. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2):209–230, 1973.

Francis Ferraro and Benjamin Van Durme. A Unified Bayesian Model of Scripts, Frames and Language. In *AAAI*, pages 2601–2607, 2016.

Charles Fillmore. Frame Semantics and the Nature of Language. *Annals of the New York Academy of Sciences*, 280(1):20–32, 1976.

Charles J. Fillmore. The Case for Case. In Bach E. and Harms R.T., editors, *Universals in Linguistic Theory*, pages 1–88. Holt, Rinehart, and Winston, New York, 1968.

Charles J. Fillmore. *The Need for a Frame Semantics in Linguistics*. Scriptor, 1977a.

Charles J. Fillmore. *Scenes-and-Frames Semantics*. Fundamental Studies in Computer Science. North Holland Publishing, 1977b.

Charles J. Fillmore. *Frame Semantics*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea, 1982.

Charles J. Fillmore. Frames and the Semantics of Understanding. *Quaderni di Semantica*, 6(2):222–254, 1985.

Charles J. Fillmore and Collin F. Baker. Frame Semantics for Text Understanding. In *Proceedings of WordNet and Other Lexical Resources Workshop*, Pittsburgh, June 2001. NAACL, NAACL.

Joseph L. Fleiss. Measuring Nominal Scale Agreement among Many Raters. *Psychological bulletin*, 76(5):378, 1971.

Lea Frermann, Ivan Titov, and Manfred Pinkal. A Hierarchical Bayesian Model for Unsupervised Induction of Script Knowledge. In *EACL*, 2014.

Kumiko Fukumura and Roger P.G. Van Gompel. Choosing Anaphoric Expressions: Do People Take into Account Likelihood of Reference? *Journal of Memory and Language*, 62(1):52–66, 2010.

Hagen Fürstenau and Mirella Lapata. Graph Alignment for Semi-Supervised Semantic Role Labeling. In *EMNLP*, 2009.

James A. Galambos. Normative Studies of Six Characteristics of Our Knowledge of Common Activities. *Behavior Research Methods & Instrumentation*, 15(3):327–340, 1983.

James A. Galambos and John B. Black. Getting and Using Context: Functional Constraints on the Organization of Knowledge. In *Proceedings of the Fourth Conference of the Cognitive Science Society*, 1982.

James A. Galambos and John B. Black. Using Knowledge of Activities to Understand and Answer Questions. *The Psychology of Questions*, pages 157–189, 1985.

James A. Galambos and Lance J Rips. Memory for Routines. *Journal of Verbal Learning and Verbal Behavior*, 21(3):260–281, 1982.

Martin Charles Golumbic and Ron Shamir. Complexity and Algorithms for Reasoning About Time: A Graph-theoretic Approach. *Journal of ACM*, 40(5):1108–1133, 1993.

Andrew Gordon. Browsing Image Collections with Representations of Common-sense Activities. *JAIST*, 52(11), 2001.

Arthur C. Graesser, Sallie E. Gordon, and John D. Sawyer. Recognition Memory for Typical and Atypical Actions in Scripted Activities: Tests of a Script Pointer+ Tag Hypothesis. *Journal of Verbal Learning and Verbal Behavior*, 18(3):319–332, 1979.

Arthur C Graesser, Stanley B Woll, Daniel J Kowalski, and Donald A Smith. Memory for Typical and Atypical Actions in Scripted Activities. *Journal of Experimental Psychology: Human Learning and Memory*, 6(5):503, 1980.

Mark Granroth-Wilding and Stephen Clark. What Happens Next? Event Prediction Using a Compositional Neural Network Model. In *Proceedings of AAAI*, 2015.

Trond Grenager and Christoph D. Manning. Unsupervised Discovery of a Statistical Verb Lexicon. In *EMNLP*, 2006.

Thomas L Griffiths and Mark Steyvers. Finding Scientific Topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235, 2004.

Rakesh Gupta and Mykel J. Kochenderfer. Common Sense Data Acquisition for Indoor Mobile Robots. In *Proceedings of AAAI*, 2004.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009), June 4-5*, 2009.

Geoffrey E. Hinton. Distributed Representations. Technical Report CMU-CS-84-157, Carnegie Mellon University, 1984.

Geoffrey E. Hinton. Learning Distributed Representations of Concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, volume 1, page 12. Amherst, MA, 1986.

Geoffrey E. Hinton, James L. Mc Clelland, and David E. Rumelhart. Distributed Representations, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations, 1986.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. *arXiv: CoRR, abs/1207.0580*, 2012.

David I. Inouye, Pradeep Ravikumar, and Inderjit S. Dhillon. Admixture of Poisson MRFs: A Topic Model with Word Dependencies. In *ICML*, pages 683–691, 2014.

T. Florian Jaeger. Redundancy and Reduction: Speakers Manage Syntactic Information Density. *Cognitive Psychology*, 61(1):23–62, 2010.

T. Florian Jaeger, Esteban Buz, E.M. Fernandez, and H.S. Cairns. Signal Reduction and Linguistic Encoding. *Handbook of Psycholinguistics. Wiley-Blackwell*, 2016.

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, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-19-0. URL http://dl.acm.org/citation.cfm?id=2380816.2380858.

Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

Li Jihong, Wang Ruibo, Wang Weilin, and Li Guochen. Automatic Labeling of Semantic Roles on Chinese FrameNet. *Journal of Software*, 21(4):597–611, 2010.

Richard Johansson and Pierre Nugues. Dependency-based Semantic Role Labeling of PropBank. In *EMNLP*, 2008.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2^{nd} Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009. ISBN 0131873210.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale Video Classification with Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

Daisuke Kawahara, Daniel Peterson, Octavian Popescu, Martha Palmer, and Fondazione Bruno Kessler. Inducing Example-based Semantic Frames from a Massive Amount of Verb Uses. In *EACL*, pages 58–67, 2014.

Atif Khan, Naomie Salim, and Yogan Jaya Kumar. A Framework for Multi-document Abstractive Summarization Based on Semantic Role Labelling. *Applied Soft Computing*, 30:737–747, 2015.

Jeong-uk Kim, Younggyun Hahm, and Key-Sun Choi. Korean FrameNet Expansion Based on Projection of Japanese FrameNet. *CoLing*, 2016.

Janet Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 2014.

Janet L. Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory : A Computer Model*, volume 16. Hillsdale, NJ, Erlbaum, 1984.

Gina R Kuperberg. Separate Streams Or Probabilistic Inference? What the N400 Can Tell Us about the Comprehension of Events. *Language, Cognition and Neuroscience*, 31(5):602–616, 2016.

Gina R. Kuperberg and T. Florian Jaeger. What Do We Mean by Prediction in Language Comprehension? *Language, Cognition and Neuroscience*, 31(1):32–59, 2016.

Marta Kutas, Katherine A. DeLong, and Nathaniel J. Smith. A Look around at What Lies Ahead: Prediction and Predictability in Language Processing. *Predictions in the Brain: Using Our Past to Generate a Future*, pages 190–207, 2011.

J. Richard Landis and Gary G. Koch. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):pp. 159–174, 1977. ISSN 0006341X. URL http://www.jstor.org/stable/2529310.

Joel Lang and Mirella Lapata. Unsupervised Induction of Semantic Roles. In *ACL*, 2010.

Joel Lang and Mirella Lapata. Unsupervised Semantic Role Induction via Split-Merge Clustering. In *ACL*, 2011a.

Joel Lang and Mirella Lapata. Unsupervised Semantic Role Induction with Graph Partitioning. In *EMNLP*, 2011b.

Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. Learning Hierarchical Invariant Spatio-temporal Features for Action Recognition with Independent Subspace Analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3361–3368. IEEE, 2011.

G. Lee, M. Flowers, and M.G. Dyer. A Symbolic/connectionist Script Applier Mechanism. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 714–721, 1989.

Geunbae Lee. *Distributed Semantic Representations for Goal/plan Analysis of Narratives in a Connectionist Architecture*. PhD thesis, University of California at Los Angeles, Los Angeles, CA, USA, 1991. UMI Order No. GAX91-22694.

Geunbae Lee, Margot Flowers, and Michael G. Dyer. Learning Distributed Representations of Conceptual Knowledge and Their Application to Script-based Story Processing. In *Connectionist Natural Language Processing*, pages 215–247. Springer, 1992.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. The Winograd Schema Challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47, 2011.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep Reinforcement Learning for Dialogue Generation. *arXiv preprint arXiv:1606.01541*, 2016.

Yuxi Li. Deep Reinforcement Learning: An Overview. *CoRR*, abs/1701.07274, 2017. URL http://arxiv.org/abs/1701.07274.

Jadwiga Linde-Usiekniewicz, Magdalena Derwojedowa, and Magdalena Zawisławska. Verbal Aspect and the Frame Elements in the FrameNet for Polish, 2008.

Colin L. Mallows. Non-null Ranking Models. I. *Biometrika*, 44(1/2):114–130, 1957.

Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer S. Pradhan. SemEval-2010 Task 14: Word Sense Induction and Disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, 2010.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

M. Manshadi, R. Swanson, and A.S. Gordon. Learning a Probabilistic Model of Event Sequences from Internet Weblog Stories. In *Proceedings of the 21st FLAIRS Conference*, 2008.

Jirí Materna. Parameter Estimation for LDA-Frames. In *HLT-NAACL*, pages 482–486, 2013.

Cynthia Matuszek, John Cabral, Michael J. Witbrock, and John DeOliveira. An Introduction to the Syntax and Content of Cyc. In *AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pages 44–49, 2006.

Philip M. McCarthy. *An Assessment of the Range and Usefulness of Lexical Diversity Measures and the Potential of the Measure of Textual, Lexical Diversity (MTLD).* PhD thesis, The University of Memphis, 2005.

Philip M. McCarthy and Scott Jarvis. MTLD, Vocd-D, and HD-D: A Validation Study of Sophisticated Approaches to Lexical Diversity Assessment. *Behavior Research Methods*, 2010.

Neil McIntyre and Mirella Lapata. Learning to Tell Tales: A Data-driven Approach to Story Generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, pages 217–225. Association for Computational Linguistics, 2009.

Risto Miikkulainen. A Neural Network Model of Script Processing and Memory. *Proceedings of International Workshop on Fundamental Res. for the Next Generation of Natural Language Processing, Kyoto, Japan*, 1991.

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

Risto Miikkulainen. Script-based Inference and Memory Retrieval in Subsymbolic Story Processing. *Applied Intelligence*, 5(2):137–163, 1995.

Risto Miikkulainen and Michael George Dyer. A Modular Neural Network Architecture for Sequential Paraphrasing of Script-based Stories. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 49–56. IEEE, 1989.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent Neural Network Based Language Model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky. RNNLM-Recurrent Neural Network Language Modeling Toolkit. In *Proc. of the 2011 ASRU Workshop*, pages 196–201, 2011.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems, NIPS*, pages 3111–3119, 2013.

Marvin Minsky. *A Framework for Representing Knowledge.* New York: McGraw-Hill, 1975.

Ashutosh Modi. Event Embeddings for Semantic Script Modeling. *CoNLL-2016*, 2016.

Ashutosh Modi and Ivan Titov. Learning Semantic Script Knowledge with Event Embeddings. *arXiv preprint arXiv:1312.5198*, 2013.

Ashutosh Modi and Ivan Titov. Inducing Neural Models of Script Knowledge. *CoNLL-2014*, page 49, 2014.

Ashutosh Modi, Ivan Titov, and Alexandre Klementiev. Unsupervised Induction of Frame-Semantic Representations. In *Proceedings of the NAACL-HLT Workshop on Inducing Linguistic Structure*, Montreal, Canada, June 2012.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. InScript: Narrative Texts Annotated with Script Information. *LREC-2016*, 2016.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. Modeling Semantic Expectations: Using Script Knowledge for Referent Prediction. *Transactions of ACL*, 5:31–44, 2017.

Raymond J Mooney. Learning Plan Schemata from Observation: Explanation-Based Learning for Plan Recognition. *Cognitive Science*, 14(4):483–509, 1990.

Erik T. Mueller. *Natural Language Processing with Thought Treasure.* Signiform, 1998.

Katherine Nelson and Janice Gruendel. Generalized Event Representations: Basic Building Blocks of Cognitive Development. *Advances in Developmental Psychology*, 1981.

K.H. Ohara, S. Fujii, T. Ohori, R. Suzuki, H. Saito, and S. Ishizaki. The Japanese FrameNet Project: An Introduction. In *Proceedings of LREC-04 Satellite Workshop "Building Lexical Resources from Semantically Annotated Corpora" (LREC 2004)*, pages 9–11, 2004.

John Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, and Thomas G Dietterich. Learning Scripts as Hidden Markov Models. In *AAAI*, pages 1565–1571, 2014.

Ahmed Hamza Osman, Naomie Salim, Mohammed Salem Binwahlan, Ssennoga Twaha, Yogan Jaya Kumar, and Albaraa Abuobieda. Plagiarism Detection Scheme Based on Semantic Role Labeling. In *Information Retrieval & Knowledge Management (CAMP), 2012 International Conference on*, pages 30–33. IEEE, 2012.

Alexis Palmer and Caroline Sporleder. Evaluating FrameNet-style Semantic Parsing: The Role of Coverage Gaps in FrameNet. In *COLING*, 2010.

M. Palmer, D. Gildea, and P. Kingsbury. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106, 2005.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English Gigaword Fifth Edition. *Linguistic Data Consortium, LDC2011T07*, 2011.

Haoruo Peng and Dan Roth. Two Discourse Driven Language Models for Semantics. *arXiv preprint arXiv:1606.05679*, 2016.

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–Human Language Technologies, Denver, USA*, 2015.

Karl Pichotta and Raymond J. Mooney. Statistical Script Learning with Multi-argument Events. *EACL 2014*, page 220, 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)*, Phoenix, Arizona, 2016a.

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, 2016b.

H. Poon and P. Domingos. Unsupervised Ontology Induction from Text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 296–305. Association for Computational Linguistics, 2010.

Hoifung Poon and Pedro Domingos. Unsupervised Semantic Parsing. In *EMNLP*, 2009.

Altaf Rahman and Vincent Ng. Resolving Complex Cases of Definite Pronouns: The Winograd Schema Challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789. Association for Computational Linguistics, 2012.

Susanne Raisig, Tinka Welke, Herbert Hagendorf, and Elke van der Meer. Insights Into Knowledge Representation: The Influence of Amodal and Perceptual Variables on Event Knowledge Retrieval From Memory. *Cognitive Science*, 33(7):1252–1266, 2009. ISSN 1551-6709. doi: 10.1111/j.1551-6709.2009.01044.x. URL http://dx.doi.org/10.1111/j.1551-6709.2009.01044.x.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQUAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv preprint arXiv:1606.05250*, 2016.

Lisa F. Rau, Paul S. Jacobs, and Uri Zernik. Information Extraction and Text Summarization Using Linguistic Knowledge Acquisition. *Information Processing & Management*, 25(4):419–428, 1989.

Michaela Regneri. *Event Structures in Knowledge, Pictures and Text.* PhD thesis, Universität des Saarlandes, 2013.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. Learning Script Knowledge with Web Experiments. In *Proceedings of ACL*, Uppsala, Sweden, 2010.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. Grounding Action Descriptions in Videos. *Transactions of the Association for Computational Linguistics*, 1:25–36, 2013.

Ronan G. Reilly and Noel E. Sharkey. *Connectionist Approaches to Natural Language Processing.* Lawrence Erlbaum Associates, Inc, 1992.

Hannah Rohde and Andrew Kehler. Grammatical and Information-structural Influences on Pronoun Production. *Language, Cognition and Neuroscience*, 29(8):912–927, 2014.

Marcus Rohrbach, Michaela Regneri, Mykhaylo Andriluka, Sikandar Amin, Manfred Pinkal, and Bernt Schiele. Script Data for Attribute-based Recognition of Composite Activities. In *European Conference on Computer Vision*, pages 144–157. Springer, 2012.

Marcus Rohrbach, Wei Qiu, Ivan Titov, Stefan Thater, Manfred Pinkal, and Bernt Schiele. Translating Video Content to Natural Language Descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 433–440, 2013.

Michael Roth and Mirella Lapata. Context-aware Frame-Semantic Role Labeling. *Transactions of the Association for Computational Linguistics*, 3:449–460, 2015.

Rachel Rudinger, Vera Demberg, Ashutosh Modi, Benjamin Van Durme, and Manfred Pinkal. Learning to Predict Script Events from Domain-specific Text. *Lexical and Computational Semantics (* SEM 2015)*, page 205, 2015a.

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

David E. Rumelhart. *Schemata: The Building Blocks of Cognition.* Center for Human Information Processing, University of California, San Diego, 1978.

D. E. Rumlehart, P. Smolensky, J. L. McClelland, and G. E. Hinton. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 2. In David E. Rumelhart, James L. McClelland, and Corporate PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 2*, chapter Schemata and Sequential Thought Processes in PDP Models, pages 7–57. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-13218-4.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. FrameNet II: Extended Theory and Practice. Available at http://framenet.icsi.berkeley.edu/index.php?option=com_wrapper&Itemid=126, 2006.

Asad Sayeed, Clayton Greenberg, and Vera Demberg. Thematic Fit Evaluation: An Aspect of Selectional Preferences. In *Proceedings of the First Workshop on Evaluating Vector Space Representations for NLP (RepEval2016) at ACL 2016*, Berlin, Germany, August 2016. Assocation for Computational Linguistics.

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

Roger C. Schank. *Dynamic Memory: A Theory of Learning in People and Computers.* Cambridge: Cambridge University Press, 1982.

Roger C. Schank. Tell Me a Story: A New Look at Real and Artificial Intelligence. *Simon & Schuster*, 1991.

Roger C. Schank and Chip Cleary. *Engines for Education.* Lawrence Erlbaum Associates, Inc, 1995.

Simone Schütz-Bosbach and Wolfgang Prinz. Prospective Coding in Event Representation. *Cognitive Processing*, 8(2):93–102, 2007.

Claude E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Noel E. Sharkey and Don C. Mitchell. Word Recognition in a Functional Context: The Use of Scripts in Reading. *Journal of Memory and Language*, 24(2):253–270, 1985.

Noel E. Sharkey and Amanda J.C. Sharkey. What Is the Point of Integration? The Loci of Knowledge-based Facilitation in Sentence Processing. *Journal of Memory and Language*, 26(3):255–276, 1987.

Dan Shen and Mirella Lapata. Using Semantic Roles to Improve Question Answering. In *EMNLP-CoNLL*, pages 12–21, 2007.

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. Springer, 2002.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Proceedings of EMNLP*, 2012.

Rosemary J. Stevenson, Rosalind A. Crawley, and David Kleinman. Thematic Roles, Focus and the Representation of Events. *Language and Cognitive Processes*, 9(4): 519–548, 1994.

Carlos Subirats. Spanish FrameNet: A Frame-semantic Analysis of the Spanish Lexicon. In Hans Boas, editor, *Multilingual FrameNets in Computational Lexicography: Methods and Applications*, pages 135–162. Mouton de Gruyter, Berlin/New York, 2009.

Mihai Surdeanu, Adam Meyers Richard Johansson, Lluís Màrquez, and Joakim Nivre. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *CoNLL 2008: Shared Task*, 2008.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.

Richard Swier and Suzanne Stevenson. Unsupervised Semantic Role Labelling. In *EMNLP*, 2004.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

Graham W. Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional Learning of Spatio-temporal Features. In *European conference on computer vision*, pages 140–153. Springer, 2010.

Wilson L. Taylor. "Cloze Procedure": a New Tool for Measuring Readability. *Journalism Quarterly*, 30:415–433, 1953.

Yee Whye Teh. Dirichlet Process. *Encyclopedia of Machine Learning*, 2007.

W Scott Terry. *Learning and Memory: Basic Principles, Processes, and Procedures.* Psychology Press, 2015.

Perry W. Thorndyke and Barbara Hayes-Roth. The Use of Schemata in the Acquisition and Transfer of Knowledge. *Cognitive Psychology*, 11(1):82–106, 1979.

Harry Tily and Steven Piantadosi. Refer Efficiently: Use Less Informative Expressions for More Predictable Meanings. In *Proceedings of the Workshop on the Production of Referring Expressions: Bridging the Gap Between Computational and Empirical Approaches to Reference*, 2009.

Ivan Titov and Alexandre Klementiev. A Bayesian Model for Unsupervised Semantic Parsing. In *ACL*, 2011.

Ivan Titov and Alexandre Klementiev. A Bayesian Approach to Semantic Role Induction. In *Proc. EACL*, Avignon, France, 2012.

Endel Tulving. *Elements of Episodic Memory.* New York: Oxford University Press, 1985.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word Representations: A Simple and General Method for Semi-supervised Learning. In *Proceedings of ACL*, 2010.

Fiona J. Tweedie and R. Harald Baayen. How Variable May a Constant Be? Measures of Lexical Richness in Perspective. *Computers and the Humanities*, 32(5):323–352, 1998.

Lilian Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. DeScript: A Crowdsourced Database for the Acquisition of High-quality Script Knowledge. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 16), Portorož, Slovenia*, 2016.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations. In *ACL (Conference System Demonstrations)*, pages 1–6, 2013.

Alessandra Zarcone, Marten van Schijndel, Jorrig Vogels, and Vera Demberg. Salience and Attention in Surprisal-based Accounts of Language Processing. *Frontiers in Psychology*, 7:844, 2016.