Unsupervised Extraction and Prediction of Narrative Chains

Unüberwachtes Extrahieren und Vorhersagen von Narrativen Ketten
Master-Thesis von Uli Fahrer
Tag der Einreichung: 22.08.2016

1. Gutachten: Prof. Dr. Chris Biemann
2. Gutachten: Steffen Remus, MSc

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Unsupervised Extraction and Prediction of Narrative Chains
Unüberwachtes Extrahieren und Vorhersagen von Narrativen Ketten

Vorgelegte Master-Thesis von Uli Fahrer

1. Gutachten: Prof. Dr. Chris Biemann
2. Gutachten: Steffen Remus, MSc

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 22. August 2016

(Uli Fahrer)

Abstract

A major goal of research in natural language processing is the semantic understanding of natural language text. This task is particularly challenging since it requires a deep understanding of the causal relationships between events. Humans implicitly use common-sense knowledge about abstract roles and stereotypical sequences of events for story understanding. This knowledge is organized in common scenarios, called *scripts*, such as *going to school* or *riding a bus*. Hence, story understanding systems have historically depended on hand-written knowledge structures capturing common-sense knowledge. In recent years, much work on learning script knowledge automatically from corpora has emerged.

This thesis proposes a number of further extensions to this work. In particular, several script models tackling the problem of script induction by learning narrative chains from text collections are introduced. These narrative chains describe typical sequences of events related to the actions of a single protagonist. A script model might for example encode the information that the events *going to the cash-desk* and *paying for the goods* are very likely to occur together.

In this context, various event representations aiming to encode the most important narrative document information such as *what happened* are introduced. It is further demonstrated in a user study how these events can be exploited to support users in obtaining a broad and fast overview of the important information of a document.

The script induction systems are finally evaluated on whether they are able to infer held-out events from documents (the *narrative cloze test*). The best performing system is based on a language model and utilizes a novel inference algorithm that considers the importance of individual events in a sequence. The model attains improvements of up to 9 percent over prior methods on the narrative cloze test.

Zusammenfassung

Eines der Hauptziele der Forschung zur natürlichen Sprachverarbeitung ist das semantische Verstehen der natürlichen Sprache in Texten. Diese Aufgabe ist besonders anspruchsvoll, da sie ein tieferes Verständnis für die kausalen Zusammenhänge zwischen Ereignissen voraussetzt. Menschen benutzen unterbewusst *Common-Sense Wissen* wie soziale Rollen und sterotypische Abfolgen von Ereignissen, um Geschichten zu verstehen. Dieses Wissen ist in wiederkehrende Schemata gruppiert, auch *Skripte* genannt, wie beispielsweise *zur Schule gehen* oder *mit dem Bus fahren*. Daher basierten frühere *Story-Understanding-Systeme* auf handgeschriebenen Wissensstrukturen, welche Common-Sense Wissen abbildeten. In den letzten Jahren sind verschiedene Arbeiten über das automatisierte Lernen von Skript-Wissen erschienen.

In dieser Thesis werden eine Reihe von Erweiterungen dieser Arbeiten vorgeschlagen. Insbesondere werden verschiedene Skript-Modelle vorgestellt, welche durch das Lernen von narrativen Ketten aus Textsammlungen automatisch Skripte induzieren. Diese narrativen Ketten beschreiben typische Abfolgen von Ereignissen über die Aktivitäten eines Protagonisten. Ein Skript-Modell kann beispielsweise lernen, dass die Ereignisse *an die Kasse gehen* und *für die Ware bezahlen* sehr wahrscheinlich gemeinsam auftreten.

In diesem Zusammenhang werden verschiedene Darstellungen für Ereignisse vorgestellt, welche das Ziel haben, die wichtigsten narrativen Elemente eines Dokumentes zu erfassen. In einer Benutzerstudie wird weiter gezeigt, wie diese Darstellungen genutzt werden können, um einen umfassenden und schnellen Überblick über die wichtigsten Informationen eines Dokumentes zu geben.

Die Skript-Induktionssysteme werden schließlich evaluiert, indem getestet wird, ob diese in der Lage sind ein Ereignis vorherzusagen, das aus einem Dokument entfernt wurde (der *narrative cloze test*). Das beste Ergebnis erzielt ein System basierend auf einem Sprachmodell, welches einen neuartigen Vorhersagealgorithmus benutzt, der die Bedeutung einzelner Ereignisse in einer Abfolge von Ereignissen berücksichtigt. Das Modell erreicht eine Verbesserung von bis zu 9 Prozent gegenüber bisheriger Verfahren im narrative cloze test.

Acknowledgements

I would like to thank my thesis supervisor Prof. Dr. Chris Biemann for his guidance and inputs throughout this process. He always supported me whenever I had questions about my research. Finally, I want to thank my family and friends for their support, particularly Julia Kadur for all of her love and encouragement during my studies at Technische Universität Darmstadt.

Contents

List of Abbreviations

NLP natural language processing

PMI pointwise mutual information

POS part-of-speech

UI user-interface

NER Named Entity Recognition

HMM hidden Markov model

MLE maximum likelihood estimate

CRF conditional random field

AI artificial intelligence

API application programming interface

SVM support vector machine

CBOW continuous bag of words

LSTM long short-term memory neural network

List of Figures

List of Tables

1 Foundations

1.1 Introduction and Motivation

Humans are great in organizing general knowledge in form of common sequences of events. This common-sense knowledge is acquired throughout lifetime and is implicitly used to understand the world around. It comprises everyday life events and their causal and temporal relations [Schank and Abelson, 1977]. This concept also includes certain roles and events associated with them as shown in the following example:

(1) John and his family visited the restaurant nearby. After having lunch, the children fell against a vase while playing. However, the owner was not mad at them since he did not like the vase.

When reading this example, humans know that the *vase* broke although it is not explicitly stated in the story. Humans can further infer that *John and his family* are the customer in the narrative and that the *owner* refers to the owner of the restaurant. This implicit used common-sense knowledge also captures that *visiting the restaurant* precedes *having lunch*.

In early years of *artificial intelligence (AI)*, the encoding of such event chains was very popular. For instance, Minsky [1974] proposed *knowledge frames* and Rumelhart [1975] proposed *schemas*. Schank and Abelson [1977] introduced *scripts*, a knowledge representation that describes typical sequence of events in a particular context. The most prominent example is the *restaurant script*. This script consists of stereotypical and temporally ordered events for eating in a restaurant e.g. *finding a seat, reading the menu, ordering food and drinks from the waiter, eating the food, paying for the food*.

Scripts were a central theme to research in the 1970s for tasks such as question answering, story understanding, summarization and coreference resolution. For example, Cullingford [1978] showed that script knowledge improves common-sense reasoning for text understanding and McTear [1987] showed applications for script-like knowledge in anaphora resolution.

Following Schank and Abelson [1977], script formalisms typically use a quite complex notion of events to model the interactions between actors of a particular scenario. This kind of information is difficult to represent in a machine-readable way, because machine learning algorithms typically focus on shallower representations. Therefore, the representation of common-sense knowledge needs to be formalized and simplified in a way that is understandable for machines. This formalization is a major challenge in natural language processing.

The aforementioned approaches for organizing common-sense knowledge were based on hand-written knowledge. It turns out that the acquisition of such knowledge is a time-consuming process. It also reveals that people learn much more scripts throughout lifetime than researchers can write down. Thus, manually-written script knowledge bases clearly do not scale.

With the increasing development of the Internet over recent years, large collections of textual data are available. These could be exploited to learn common-sense knowledge automatically. This enables to develop systems, which function in a completely unsupervised way without expert annotators.

This work presents and explores several script systems that learn script-like knowledge from text collections automatically. A script system captures the events and their relations involved in everyday scenarios, such as *dining in a restaurant* or *riding a bus*. Thereby, it is able to infer events that have been removed from an input text by reasoning and reacting towards the situation the system encounters. For instance, given the event *eat food*, it should predict the *pay for the food* event according to the restaurant scenario. The script models presented here utilize classical language models [Manning and Schütze, 1999, p. 71], but also apply recent word embedding language modeling techniques [Mikolov et al., 2013].

The major part of this thesis concentrates on the question of how machines can learn common-sense knowledge from corpora. However, as already emphasized, the event representation is at least as important as the actual learning algorithm. The way of how the knowledge is encoded plays an essential factor for successful script learning. Moreover, the combination of a script model and an event representation should allow to generalize over the different encoded situations. For instance, the *check reservation* event that is associated with the *waiter* does not necessarily need to occur in the restaurant scenario.

While this research direction focuses on *how machines can learn humans' common-sense*, the work presented here further examines whether the same underlying concepts can support humans in different tasks such as to aid in reading texts. For example, information about protagonists and their associated events extracted from a document could be exploited for reducing information overload to provide humans a broad overview of that document. Hence, these concepts facilitate the extraction of information about key elements of the document without reading the whole text. Based on this idea, a text-reading tool is described that visualizes narrative information of a document.

In particular, the thesis tackles the following research questions that will guide through the work:

(1) How can script knowledge automatically be learned from corpora?

(2) How should a script model be designed to allow flexible inference of events?

(3) How can events be represented in order to improve the performance of script models?

(4) Do events extracted from a document give a broad and fast overview of the important information on that document?

This thesis is structured as follows. In the remainder of this chapter, some theoretical foundations will be covered that are used throughout this work, while giving potential applications of common-sense knowledge in Section 1.4. Chapter 2 presents a brief but essential background on automatic script induction and then further introduces the state-of-the-art by presenting different approaches that tackle the problem of learning script-like knowledge from corpora automatically. Chapter 3 outlines the event extraction methodology, proposes an event extraction framework and motivates different event representations. In Chapter 4, a web-based platform for visualizing narrative events is described and evaluated in terms of its utility for giving a broad and fast overview of a document. The various script models explored in this thesis are described in Chapter 5 and Chapter 6 evaluates the performance of these models in comparison to an informed baseline. Additionally, a qualitative analysis discusses the common types of errors made by the systems. Finally, the work is concluded in Chapter 7 and ends with an outlook on possible future research topics and further development of the proposed script induction models.

1.2 Terminology

This section introduces recurring concepts and terms used in this thesis. If not stated otherwise, these concepts come from Chambers and Jurafsky [2008]. The following story serves for illustration purposes:

> *Andrea was **looking** for a new pet. She **was considering adopting** a dog. After visiting the local dog shelter, she **decided to rescue** a puppy. After the paperwork was finalized, Andrea **brought** the dog home. Andrea **introduced** the dog to the family.*
>
> Source: Mostafazadeh et al. [2016]

The example above contains several *narrative events*, which describe actions performed by the protagonists of the story. WordNet[1] [Fellbaum, 1998] describes a protagonist as "the principal character in a work of fiction". According to this definition, the main protagonists can be identified as *Andrea* and the *dog*, whereas all coreferent mentions[2] of *Andrea* and the *dog* are straight and dashed underlined, respectively.

Section 3 gives a further specification of the broad term "narrative event". For the time being, a narrative event e is defined as a tuple (v, d), where v is the verb that has the protagonist a as its typed dependency d, such that $d \in \{\text{subj, obj, prep}\}$[3]. Following this definition, the narrative events for the second sentence can be extracted as `(adopting,subj)` for *Andrea* and `(adopting,obj)` for the *dog*. Note that the same verb may participate in multiple events as it can have several arguments.

On this basis, a *narrative chain* is introduced as a partially ordered set of narrative events that share a common protagonist. Thus, a narrative chain consists of a set of narrative events L and a binary relation $\geq (e_i, e_j)$ that is true "if event e_i occurs strictly before e_j" [Chambers and Jurafsky, 2008]. Accordingly, the following narrative chain for *Andrea* can be defined as:

$L = \{$`(looking,subj)`,`(adopting,subj)`,`(rescue,subj)`,`(brought,subj)`,`(introduced,subj)`$\}$

`(looking,subj)` $\geq$ `(adopting,subj)` $\geq$ `(rescue,subj)` $\geq$ `(brought,subj)` $\geq$ `(introduced,subj)`

Chambers and Jurafsky [2008] were the first to introduce these concepts, which tackle the problem of script induction by learning narrative chains from text collections. The assumption that events with shared arguments are connected by a similar narrative context builds the base for their entity model. For example, the verbs *rescue* and *adopting* share the same protagonist and are therefore considered as related. In this context, Chambers and Jurafsky formulated the following narrative coherence assumption:

> *Verbs sharing coreferring arguments are semantically connected by virtue of narrative discourse structure.*
>
> Source: Chambers and Jurafsky [2008]

This assumption can be compared to the distributional hypothesis, which is the basis for the concept of *distributional learning*. Harris [1954] formulated the distributional hypothesis as follows: "words that occur in the same contexts tend to have similar meanings".

[1] WordNet project page: `https://wordnet.princeton.edu/` (accessed July 2016).
[2] Two mentions are said to corefer, if they refer to the same entity.
[3] Typed dependencies describe grammatical relationships in a sentence. For example, *Mary* stands in subject relation to *had* in the sentence *Mary had a little lamb*.

Chambers and Jurafsky [2008] stated that in contrast to distributional learning, narrative learning reveals additional information about the participant. For instance, distributional learning might indicate that the verb *push* relates to the verb *fall*. However, narrative learning also provides the information that the object of *push* is the subject of *fall*.

Following Chambers' and Jurafsky's work, the script induction systems proposed in Section 5 are based on the learning of narrative relations between events. This task also includes the extraction of narrative events from document collections and the identification of coreferent mentions to build narrative chains as further discussed in Section 3.

1.3 Resources of Common-Sense Knowledge

The following section introduces various models for representing common-sense knowledge. Some of the resources are long-running projects, others are suspended but are worth mentioning due to their contribution to the research community.

Knowledge Frames

The idea to use *frames* in artificial intelligence as a structured representation for conceptualizing common-sense knowledge is attributed to Minsky [1974]. According to Minsky, a frame is a data structure for representing a stereotyped situation like *being in a certain kind of living room*, or *going to a child's birthday party*. He also showed the relevance of frames for tasks related to language understanding like the understanding of storytelling. The concept of frames can be seen as a mental model that stores knowledge about objects or events in memory as a unit. When a new situation requires common-sense reasoning, the appropriate frame is selected from the memory.

A frame is a structured data collection, which consists of *slots* and *slot values*. Slots can be of any size and contain one or more nested fields, called *facets*. Facets may have a name and an arbitrary number of values. In addition to descriptive information, slots can contain pointer information used as references to other frames. The general concept is flexible and allows inheritance and inferencing. Hence, frames are often linked to indicate *has-a* or *is-a* relationships. Figure 1.1 illustrates the general frame structure.

```
(<frame name>
  (<slot 1> (<facet 1><value 1> ... <value k1>)
            (<facet 2><value 2> ... <value k2>)
                   ...
                   ...
            (<facet n><value n> ... <value kn>
                         ...
                         ...
  (<slot 2> (<factet 1><value 1> ... <value k1)
```

Figure 1.1: A general frame structure (Source: Akerkar [2005]).

TABLE 1

THEORETICAL RESTAURANT SCRIPT (ADAPTED FROM SCHANK & ABELSON, 1977)

Name: *Restaurant*

Props: Tables, Menu, Food, Bill, Money, Tip

Roles: Customer, Waiter, Cook, Cashier, Owner

Entry Conditions: Customer hungry; Customer has money

Results: Customer has less money; Owner has more money; Customer is not hungry

Scene 1: *Entering*

Customer enters restaurant
Customer looks for table
Customer decides where to sit
Customer goes to table
Customer sits down

Scene 2: *Ordering*

Customer picks up menu
Customer looks at menu
Customer decides on food
Customer signals waitress
Waitress comes to table
Customer orders food
Waitress goes to cook
Waitress gives food order to cook
Cook prepares food

Scene 3: *Eating*

Cook gives food to waitress
Waitress brings food to customer
Customer eats food

Scene 4: *Exiting*

Waitress writes bill
Waitress goes over to customer
Waitress gives bill to customer
Customer gives tip to waitress
Customer goes to cashier
Customer gives money to cashier
Customer leaves restaurant

Figure 1.2: Illustration of the *restaurant script* formalization (Source: Bower et al. [1979]).

Scripts

The idea of scripts came in the 1970s from Schank and Abelson [1977]. A script is a knowledge structure that describes a stereotyped sequence of events in a particular context. Scripts are closely related to frames but contain additional information about the sequence of events and the goal of the involved protagonists. Thus, this representation is less general than frames. According to Schank and Abelson, a script has the following components:

- The **scenario** describes the underlying type of the situation. For instance, *riding a bus*, *going to a restaurant* or *robbing a bank*.
- **Roles** are the participants involved in the events.
- **Props** is short for property and the term refers to the objects that the participants use to accomplish the actions.
- In order to instantiate a script, certain **entry conditions** must be satisfied.
- The **results** describe conditions that will be true when the script is exited.
- The plot of a script is grouped into several **scenes**. Each scene describes a particular situation and is further divided into events. An event represents an atomic action associated with one or more participants of the script scenario. Precondition and postcondition describe the causal relationships and are defined for each event accordingly.

Figure 1.2 shows the most prominent script that describes events, which occur in the individual scenes corresponding to the situation of *dining in a restaurant*. The preconditions for *going to a restaurant* are that *the customer is hungry* and *is able to pay for the food*. The involved protagonists are the *customer*, the *owner* and other *personnel staff*. The props include *tables*, *a menu*, *food*, *a bill*, and *money*. The final results are that the customer *is no longer hungry*, but *has less money*.

The illustration has been simplified in order to highlight the high-level concepts. For example, each event in the restaurant script results in conditions, which trigger the next event.

FrameNet

The notion of frames has a wide range and occurs in different research disciplines. Fillmore's theory brings Minsky's ideas about frames into connection with linguistics [Fillmore, 1976]. His frame semantic theory describes complex semantic relations related to concepts. The basic idea refers to the assumption that humans can better understand the meaning of a single word with additional contextual knowledge related to that word.

A semantic frame represents a set of concepts associated with an event and involves various participants, props, and other conceptual roles. A common example for a frame is the *commercial event frame* [Fillmore, 1976]. This frame describes the relationship between *a buyer*, *a seller*, *goods*, and *money* related to the situation of *commercial transfer*. Different words evoke and establish frames. This is motivated by the fact that several lexical items can refer to the same event type. In the previous example, the word *pay* or *charge* evokes the frame from the perspective of the buyer, whereas *sell* evokes it from the perspective of the seller.

A prominent example that captures script-like structures for a particular type of situation along with participants and props is FrameNet [Baker et al., 1998]. The *FrameNet* project[4] is a realization

[4] FrameNet project page: `https://framenet.icsi.berkeley.edu/fndrupal/` (accessed July 2016).

of Fillmore's frame semantics as an online lexical resource. If offers a broad set of frames that range from simple to complex scenarios constructed through expert annotators. Each frame consists of semantic roles, called *frame elements*, and *lexical units* that model the words evoking a frame. Frames additionally include relationships to other frames at various levels of generality, called *frame-to-frame relations*. For example, *selling* and *paying* are subtypes of *giving* as shown in Figure 1.3. Although FrameNet covers script information in general, script scenarios are quite rare and not explicitly marked. In the current version (1.5, as of August 2016), FrameNet consists of 1019 frames, 11.829 lexical units, 8.884 unique roles labels and 1.507 frame-to-frame relations.

However, frame-to-frame relations only allow the building of sequences of events to a certain extent. For example, the commercial transfer frame has no frame-to-frame relation that describes the negotiation between both parties, though it is considered as a typical event in common-sense. Moreover, the creation of such a corpus is extremely expensive and requires effort over many years.

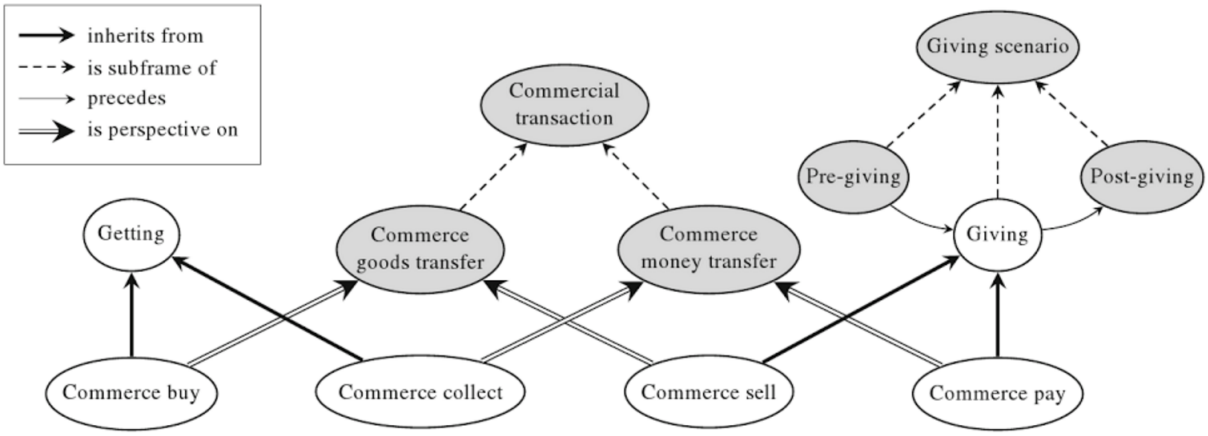


Figure 1.3: Illustration of the frame-to-frame relations corresponding to the commercial transfer frame (Source: Gamerschlag et al. [2013]).

1.4 Application in Natural Language Processing

Script knowledge has a wide range of applications in modern language understanding systems. Systems that operate on the document level would benefit the most from knowledge about entities, events and their causal relation. In contrast, systems that work on the sentence or word level have only limited context. Due to the limited context, such applications would not benefit from information on higher level concepts and their relations. The following presents a few showcases for applications that could profit from script knowledge.

Question Answering

A *question answering system* is designed to answer textual questions posted by humans in a natural language [Manning and Schütze, 1999, p. 377]. Knowledge-based question answering systems use a huge structured database containing an enormous amount of information. These

systems transform the meaning of the question into a semantic representation, which is then used to query the database.

Most of these systems focus on factoid questions (e.g. what, when, which, who, etc.) that can be answered with a simple fact. Consider the following examples. Each of these examples can be answered with a short text that corresponds to a name or a location:

(1) Who shot Mr. Burns?

(2) Where is Mount Everest?

(3) What is Peter Parker's middle name?

For the examples above, the questions can be reformulated to statements that can be looked up with simple patterns in the knowledge base. Assuming that the knowledge base is large enough, it is very likely that the database contains the answers to such questions.

While these type of questions do not require script knowledge, more complicated questions would require flexible inference based on entities and their actions in events as well as the causal relations between them. For example, causal questions such as *why* or *how* require world knowledge and common sense reasoning. The answer to such questions contains further elaborations related to specific events or actors and the system requires therefore deeper understanding of the text.

Coreference Resolution

Winograd [1972] proposed a schema that makes the implicit use of common-sense knowledge apparent. Their schemas consist of one sentence that requires anaphora resolution to one of two involving actors. A mention *A* is an anaphoric antecedent of mention *B* if and only if it is required for comprehending the meaning of *B*. When one term in the Winograd schema is changed, the correct actor for the anaphora changes. The following pair of sentences illustrate this kind of schema:

(1) The city council refused the demonstrators a permit because they **advocated** violence.

(2) The city council refused the demonstrators a permit because they **feared** violence.

Source: Winograd [1972]

In the first sentence, the mention *they* refers to the *demonstrators*, whereas the same mention refers to the *city council* in the second example. While the answer is immediately obvious to humans, it proves difficult for current automatic language understanding systems. The resolution of this ambiguity requires knowledge about the relation of *city councils* and *demonstrators* to *violence*. Script knowledge could help to solve this problem through its representation of actors and their roles in events. A script model will ideally encode the fact that it is more likely that the city council members engage in a *fear violence* event than an *advocated violence* event. Such a system could be incorporated into a coreference resolution system[5] to enable this sort of inferences.

Levesque [2011] proposed a collection of similar sentences as an evaluation metric for artificial intelligence and an improvement on the Turing test.

[5] Coreferring mentions could represent an anaphoric relation, but do not necessarily have to. However, the outlined benefits also apply to the problem of coreference resolution.

Summarization

The task of *automatic summarization* in natural language processing describes the process of reducing the content of a text document to its core information [Mani, 1999].

An essential part of this task is to identify sentences that describe the story's main events. Script knowledge can assist summarization systems in this task and help to organize the summary. It provides important events that are expected to occur in common situations. For example, for a scenario covering a political demonstration one would expect to find some of the events shown in Figure 1.4.

DeJong [1982] used this idea for an automatic summarization system called *FRUMP*. The system covers various scenarios like *public demonstration* or *car accidents* and is focused on the summarization of newspaper stories. However, the approach is not applicable for stories that require common-sense knowledge like *dining in a restaurant* or *riding a bus* since events that are associated with these type of scenarios are rather not explicitly mentioned in newspaper stories.

The demonstrators arrive at the demonstration location.
The demonstrators march.
Police arrive on the scene.
The demonstrators communicate with the target of the demonstration.
The demonstrators attack the target of the demonstration.
The demonstrators attack the police.
The police attack the demonstrators.
The police arrest the demonstrators.

Figure 1.4: The example is part of the sketchy script $DEMONSTRATION (Source: DeJong [1982]).

1.5 Contributions

The main contributions of this work are:

- An unsupervised narrative event and chain extraction framework that is designed to extract events in different variants.
- A web-based platform that supports reading by extracting and visualizing narrative events from text.
- An unsupervised script induction system that attains improvements over prior methods on the narrative cloze test.
- A qualitative evaluation of the proposed script induction systems on a publicly available dataset.

2 Background and Related Work

This chapter reviews the related literature of the two research directions of this thesis. Section 2.1 gives a short history of automatic script induction and presents the state-of-the-art. Section 2.2 discusses related work in the field of visualizing narrative structures that aims at supporting humans in exploring collections of text.

2.1 Script Models

First attempts in story understanding have already been made back in the 1970s. This task is extremely challenging and has a long running history. Schank and Abelson [1977] identified that common-sense knowledge such as common occurrences and relationships between them is implicitly used to understand stories. The term common-sense knowledge in the field of artificial intelligence research refers to the collection of facts and background information that a human is expected to know. While humans acquire this knowledge just by interacting with the environment, it is hard to add this ability to machines in a way that allows flexible inference. This raises the question of how to represent and provide common-sense knowledge to machines.

One way of aggregating common-sense knowledge are *scripts*, a "structure that describes appropriate sequences of events in a particular context" [Schank and Abelson, 1977]. Scripts are stereotypical sequences of causally connected events, such as *dining in a restaurant*. They also include roles that different actors can play and are hand-written from the point of view of a protagonist. Various other knowledge structures have been proposed aiming to capture common-sense knowledge as well [Rumelhart, 1975; Minsky, 1974; Winograd, 1972].

However, all of these approaches are non-probabilistic and rely on complicated hand-coded information. The acquisition of scripts is a time-consuming task and requires expert knowledge in order to annotate events, their relation and participant roles. Although hand-structured knowledge contains little noise, it is less flexible and will have a low recall. A story may contain the events exactly as it is defined in the script, but any variation on the structure is difficult to handle.

Therefore, researchers have been trying to learn scripts from natural language corpora automatically. The work on unsupervised learning of event sequences from text began with Chambers and Jurafsky [2008]. They first proposed narrative chains as a partially ordered set of narrative events that share a common protagonist. Chambers and Jurafsky learned co-occurrence statistics from narrative chains between simple events consisting of a verb and its participant represented as a typed dependency (see Section 1.2). This co-occurrence statistic $C(e_1, e_2)$ describes the number of times the pair (e_1, e_2) and (e_2, e_1) has been observed across all narrative chains extracted from all documents. For instance, `(eat,obj)` and `(drink,obj)` is expected to have a low co-occurrence count, because things that are eaten are not typically drunk[6].

In order to infer new verb-dependency pair events that have happened at some point in a sequence, Chambers and Jurafsky maximize over the *pointwise mutual information (PMI)* [Church and Hanks, 1989] given the events in the sequence. Formally, the next most likely narrative event in a sequence of events $c_1, ..., c_n$ that involves an entity is inferred by maximizing

[6] The example is taken from Pichotta and Mooney [2016].

$\text{argmax}_{e \in V}\left(\sum_{i=0}^{n} pmi(c_i, e)\right)$, where V are the events in the training corpus and pmi is the pointwise mutual information as described in Church and Hanks [1989].

In Chambers and Jurafsky [2009], they extend the narrative chain model and propose *event schemas*, a representation more similar to semantic frames [Fillmore, 1976]. In contrast to their previous work, the focus here is on learning structured collections of events. In addition, Chambers and Jurafsky use all entities of a document when inferring new events rather than just a single entity. As a consequence, they can only infer untyped events instead of verb-dependency pair events. Results show that this approach improves the quality of the induced untyped narrative chains. Numerous others focus on schema induction rather than event inference [Chambers, 2013; Cheung et al., 2013; Balasubramanian et al., 2013; Nguyen et al., 2015]. However, this work focuses on the original work of Chambers and Jurafsky [2008] and the field of event inference instead of learning abstract event schema representations.

Previous attempts to acquire script knowledge from corpora automatically can be divided into two principal areas of research: (1) *open-domain script acquisition* and (2) *closed-domain script acquisition*.

Pichotta and Mooney [2016], Rudinger et al. [2015b], Jans et al. [2012] and Chambers and Jurafsky [2008] focused on open-domain script acquisition. They extracted narrative chains from large corpora such as Wikipedia or the Gigaword corpus [Graff et al.] to train their statistical models. Thereby, a large number of scripts is learned. However, there is no guarantee of a specific set of scripts such as the *restaurant script* being learned.

The problem of *implicit knowledge* is a more serious drawback of this approach i.e. newspaper text does not state stereotypical common-sense knowledge explicitly. In addition, such articles contain knowledge that deviates from everyday life events. The *man bites dog* aphorism is a good example to illustrate the problem. This anecdotal states: "When a dog bites a man, that is not news, because it happens so often. But if a man bites a dog, that is news." and is attributed to John B. Bogart of New York Sun. Given such an article, a script model would learn the fact that humans bite dogs, even if it is more likely that dogs bite humans.

Rudinger et al. [2015a] argue that for many specialized applications, however, knowledge of a few relevant scripts may be more useful than knowledge of many irrelevant scripts. With this scenario in mind, they learn the *restaurant script* by applying narrative chain learning methods to a specialized domain-specific corpus of dinner narratives[7]. Based on this approach, other work that focuses on closed-script acquisition has been published [Ahrendt and Demberg, 2016]. According to Rudinger et al. [2015a] this thesis is also directed towards closed-script acquisition and therefore uses domain-specific corpora for training.

A variety of expansions and improvements of Chambers and Jurafsky [2008] have been proposed:

Jans et al. [2012] explored several strategies to collect the model's statistics. Their results show that a language-model-like approach performs better than using word association measures like the pointwise mutual information metric. Furthermore, they found that skip-grams [Guthrie et al., 2006] outperform vanilla bigrams, while 2-skip-gram and 1-skip-gram perform similarly. Unlike Chambers and Jurafsky [2008], Jans et al. [2012] include the relative ordering between events in a document to their model. Section 5 gives more details about this bigram model and discusses the differences in comparison with the script model proposed by Chambers and Jurafsky [2008].

[7] Website with stories about restaurant dining disasters: `http://www.dinnersfromhell.com` (accessed July 2016).

This work further extends the bigram model mentioned above to reflect the individual importance of each event in a sequence. Similar to Jans et al. [2012], the script models proposed here also take the ordering between events in a document into account and do not rely on a pure bag of events model. Finally, the original bigram model will be compared to the modified version in order to show the benefit of such a modification.

Rudinger et al. [2015b] contributed a log-bilinear discriminative language model [Mnih and Hinton, 2007] and also showed improved results in modeling narrative chains of verb-dependency pair events. Overall, their log-bilinear language model reaches 36% recall in top 10 ranking compared to 30% with the bigram model.

Pichotta and Mooney [2014] extended the verb-dependency pair event model to support multi-argument events such as `ask(Mary,Bob,question)` for the sentence *Mary asks Bob a question*. This representation not only includes the verb and its dependency, but also considers the arguments. However, gathering raw co-occurrence statistics from these events would only count the actions performed by the involved entity mentions, resulting in poor generalization. Thus, Pichotta and Mooney [2014] also model the interactions between all distinct entities x, y and z in a script. For example, if one participant asks the other (e.g. `ask(x,y,z)`), the other is likely to respond (e.g. `answer(y,•,•)`)[8]. Their model achieves slightly higher performance on predicting simple verb-dependency pair events than the one that models co-occurring pair events directly.

This work adapts the multi-argument representation for modeling event sequences, but does not model the interactions between entities explicitly. Instead, several other strategies are explored that help to generalize over the training data.

Recently, the *long short-term memory neural network (LSTM)* [Hochreiter and Schmidhuber, 1997] has been applied successful to a number of difficult natural language problems such as machine translation [Sutskever et al., 2014]. There has been also a number of recent work that approach the problem of script induction with neural models. Pichotta and Mooney [2016] use a recurrent neural network model with long short-term memory and show that their model outperforms previous bigram models in predicting verbs with their arguments.

Granroth-Wilding and Clark [2016] present a feedforward neural network model for script induction. This model predicts whether two events are likely to appear in the same narrative chain by learning a vector representation of verbs and argument nouns and a composition function that builds a dense vector representation of the events. Their neural model achieves a substantial improvement over the bigram model and the word association measure based model originally introduced by Chambers and Jurafsky [2008]. According to Granroth-Wilding and Clark [2016], one possible reason for its success is its ability to capture non-linear interactions between verbs and arguments. This allows for example that the events *play golf* and *play dead* lie in different regions of the vector space.

As the learning of vector representations gives a more robust model, this thesis also implements vector space based models and compares them to the traditional language-model-based approaches.

All of these algorithms above require evaluation metrics to determine successful learning of narrative knowledge. Chambers and Jurafsky [2008] proposed the *narrative cloze test*, in which an event is held out from chains of events and the model is tested on whether it can fill in the left-out event. This evaluation metric is inspired by the idea that people can fill in gaps in stories using their common-sense knowledge. Thus, a script model that claims to demonstrate narrative knowledge

[8] The filler (•) indicates that no entity stands in that dependency relation with the verb.

should be able to recover a held-out event from a partial event chain. This task has already been used for various script induction models and is therefore used as a comparative measure in this work [Chambers, 2013; Pichotta and Mooney, 2016; Rudinger et al., 2015b].

2.2 Visualization of Narrative Structures

The visualization of information extracted from unstructured text has become a very popular topic in recent years [Jänicke et al., 2016; Keim et al., 2006]. It functions not only as an instrument to present the result of an analysis, but also as an independent analysis instrument. The combination of natural language processing and information visualization techniques enables new ways to explore data and reveal hidden connections and correlations that were not visible before. This kind of fusion is not only scientifically rewarding, but also has great benefit in practical applications.

Yimam et al. [2016] have recently shown the added value in investigative journalism. They provide journalists with a data analysis tool[9] that combines latest results from natural language processing and information visualization. The platform enables journalists to process large collections of newly gained text documents in order to find interesting pieces of information.

There are also NLP-based systems that aim to aid humans in reading text by using latest visualization techniques. The following two systems visualize narrative structures and offer several exploration mechanisms similar to the tool proposed in this thesis.

Reiter et al. [2014] described and implemented a web-based tool for the exploration and visualization of narratives in an entity-driven way. They visualize the participants of a discourse and their event-based relations using entity-centric graphs. While these graphs show entities jointly participating in single events, they do not provide context information about the individual events. Although the application offers an interface that allows searching for events and event sequences, it lacks the ability to give a global overview of the narrative information of a document.

John et al. [2016] presented a web-based application that combines natural language processing (NLP) methods with visualization techniques to support character analysis in novels. They extract named entities such as characters and places and offer several views for exploring these entities and their relationships. While the text view supports basic search mechanisms, entity highlighting and a chapter outline, it does not present prominent information of the selected chapter. However, such a feature could aid researchers in literary studies since it reduces information overload.

The approach described and implemented in this work enables both, entity-driven exploration of the underlying document and the acquisition of a broad overview by visualizing events extracted from that document in a structured outline. In contrast to the discussed systems, the system proposed here only works on document level.

[9] Project page: `http://newsleak.io` (accessed June 2016).

3 Event Extraction and Representation

Based on the idea of learning relationships between everyday life events from narrative chains, this chapter tackles the subproblem of extracting narrative events from text. The main part of this chapter deals with an extraction framework for narrative chains, which was developed as part of this work.

Section 3.1 places the broad term *event* into the context of narrative learning and motivates the serious need for a flexible extraction framework for narrative events. Section 3.2 gives a qualitative analysis of two state-of-the-art information extraction systems that seeks to answer whether these approaches are suitable for the extraction of narrative chains and then describes the event extraction methodology in the remainder of the section.

3.1 Definition of an Event

The TimeML[10] annotation schema provides a definition for an event:

> *TimeML considers events a cover term for* ***situations that happen or occur****. [...] We also consider as events those predicates describing states or circumstances in which* ***something obtains or holds true****.* Source: Pustejovsky et al. [2003]

TimeML is a specification language for events and temporal expressions in natural language and was originally developed to improve the performance of question answering systems. According to the definition above, the phrase *meet him* would be annotated as an event since it captures a situation that occurs or happens. Likewise, the phrase *is angry* is considered as an event, because it describes an event of state.

However, in the research community for the field of automatic script induction, there is no common understanding of what should be considered as an event. Chambers and Jurafsky [2008] represent an event as a pair of a verb and a dependency between this verb and its entity argument (subj, obj). Pichotta and Mooney [2014] model events with a multi-argument representation (v, s, o, p), where v is the lemma of the verb, s, o and p its corresponding subject, object and prepositional object argument, respectively. Granroth-Wilding and Clark [2016] also consider predicative adjectives[11] where an entity is an argument to the verb *be*, *seem* or *become*. For instance, the copula *is* links the subject *Elizabeth* to the predicative adjective *hungry* in the sentence *Elizabeth is hungry*. In this case, Granroth-Wilding and Clark extract the corresponding narrative event as `be(Elizabeth,hungry)` in which the predicative adjective *hungry* describes a situation that holds for a certain amount of time. This approach most closely resembles the event definition above, because it incorporates narrative state information to the event representation.

It becomes apparent that the extraction of narrative events from documents has to be a flexible process in terms of information representation. This raises a serious need for an automatic event extraction framework that is capable to support various event representations. This includes

[10] TimeML project page: `http://www.timeml.org/` (accessed June 2016).

[11] A predicative adjective is an adjective that follows a linking verb (copula) and complements the subject of the sentence by describing it. Any form of *be, become* and *seem* is always a linking verb.

the generation of simple verb-dependency pair events, but also complex multi-argument representations. The ultimate goal is to have a framework that assembles individual components like prepositional phrases, direct objects and even predicative adjectives to complete event representations. The separation of the identification of such fragments from the actual representation allows numerous possibilities to model events. Thereby, it is possible to explore different event variants without requiring expert knowledge about open information extraction.

3.2 Event Extraction Methodology

This subsection introduces *Eventos*, an *unsupervised open information extraction system* that is designed to extract narrative events from unstructured text. It is highly customizable and supports both, verb-dependency pair events and multi-argument event representations. Its design allows to assemble different event representations without expert knowledge. Furthermore, the information representation can be adapted to utilize the system for other applications. The utility of such a system for other applications is assessed in a user study in Chapter 4.

Eventos is publicly available in open-source[12]. To date, no code has been published for generating narrative chains since back Chambers and Jurafsky released their work[13]. The release of Eventos should enable other researchers to catch up with the current state-of-the-art and encourage others to make their work publicly available.

Open information extraction system comparison

The term *information extraction* describes the task of automatically extracting structured information from unstructured or semi-structured documents [Andersen et al., 1992]. An open information extraction system processes sentences and creates structured extractions that represent relations in text. For example, the extraction `(Angela,was born in,Danzig)` corresponds to the relation *was born in* in the sentence *Angela was born in Danzig*.

Two recent and prominent state-of-the-art information extraction systems are *Stanford OpenIE* [Angeli et al., 2015] and *OpenIE 4*. The latter is the successor to Ollie [Mausam et al., 2012], which was developed by the AI group of the University of Washington. The following discussion raises a few problems with these systems when applied to the extraction of narrative events[14].

Both systems create synthetic clauses with artificial verbs that do not occur in the sentence, so called *noun-mediated extractions*. They apply dependency and surface patterns like *appositions* and *possessives* to segment noun phrases into additional extractions. For example, the sentence *I visited Germany, a beautiful country* creates the open information triples `(I,visited,Germany)` and `(Germany,is,a beautiful country)`. The latter is extracted by applying a pattern that matches the apposition *a beautiful country* in the sentence. The matched parts together with the supplementary created predicate *be* then form the noun-mediated extraction. However, such extractions are not considered as events, because they usually contain no narrative information.

[12] The project page is available at `http://uli-fahrer.de/thesis/` (accessed August 2016).

[13] Code available at `https://github.com/nchambers/schemas` (accessed June 2016).

[14] For the tests, the latest available version for both system were taken. That is, Washington's OpenIE in version 4.1.x downloaded from their project page and Stanford OpenIE compiled from their code repository.
- OpenIE project page: `http://knowitall.github.io/openie/` (accessed June 2016).
- Stanford OpenIE repository: `https://github.com/stanfordnlp/CoreNLP/`
  Commit ID `4fd28dc4848616e568a2dd6eeb09b9769d1e3f4e` (accessed June 2016).

More importantly, the task of extracting narrative chains requires separate events for each protagonist mentioned in the document. Hence, the system is expected to produce independent events for *Tom* and for *Jerry* given the sentence *Tom and Jerry are fighting*. Stanford's system is designed to extract only complete triples and since there is no second argument available for the example, the system yields no result. A possible interpretation of the sentence would be the fact that *Tom* and *Jerry* fight with each other. Thus, the extraction `(Tom,fight,Jerry)` represents a valid open information triple in this case. However, the system is not able to derive such a triple. In comparison, OpenIE 4 extracts the proposition `(Tom and Jerry,are fighting,•)`. This result reveals a drawback of Washington's OpenIE 4. Their system is not able to process coordinated conjunctions like *and* or *or* in order to create multiple extractions for conjoined actions. In contrast, the Stanford system is theoretically able to process coordinated conjunctions, if the sentence contains enough fragments to assemble a triple.

Furthermore, only Washington's OpenIE 4 is able to process simple relative clauses. Consider the following sentences that are composed of such an additional and independent subordinate clause. For the examples, the relative clause is underlined and the associated relative pronoun is highlighted in bold.

(1) I told you about the woman **who** lives next door.

(2) The boy **who** lost his watch was careless.

(3) The hamburgers **that** I made were delicious.

In the first sentence, the relative pronoun *who* is the subject of the subordinate clause, but references the *woman* in the main clause. The pronoun needs to be resolved in order to generate an independent extraction for the relative clause. OpenIE 4 implements special rules to handle such cases and generates `(I,told,you,about the woman)` and `(the woman,lives,next door)` as extractions. The Stanford system in contrast only yields the extraction `(I,told,you)` and ignores the relative clause.

In the second example, the relative clause occurs within the sentence, but the relative pronoun is still the subject of the subordinate clause. For this example, OpenIE 4 yields the extractions `(The boy,lost,his watch)` and `(The boy who lost his watch,was,careless)`. Although these are valid extractions, they are too over-specified for predicting narrative events. The system always tries to extract the arguments in accordance with the *longest match rule*. Similar observations can be made for the sentence *Thomas Mueller from the FC Bayern club plays soccer*. The result will contain *Thomas Mueller from the FC Bayern club* as first argument. Stanford OpenIE yields no results for the second sentence at all.

The third sentence is different from the previous examples. Here, the relative pronoun acts as object of the relative clause. This sort of relative clauses is called *non-defining relative clauses* and OpenIE has no full support for this kind of sentences. For the given sentence the system returns `(The hamburgers I made,were,delicious)` and `(I,made,•)`. While the first extraction is correct, the second extraction misses the word *hamburgers* referenced by the relative pronoun *that* as additional argument.

It has been shown that both systems lack essential features and are therefore not suitable for the extraction of narrative events. Eventos in contrast is designed with the purpose of serving as an extraction framework for narrative chains. Although it is developed for this purpose, it can still be used as general information extraction system. The framework is rule-based and requires no

additional training. It operates on dependency parse annotations and utilizes a novel processing concept.

This concept differs from traditional extraction approaches in that it separates the identification of the syntactic constituents within a sentence from the actual event representation. This allows to identify the head of the verb phrase as an event and delegate the decision of adding the dependents to a post-processing step. Figure 3.1 illustrates the architecture of Eventos. It consists of two higher-level parts: (1) a traditional *NLP pipeline* and (2) the *event generation*. The NLP pipeline annotates unstructured text with linguistic annotations and assembles the result in a *RichDocument*. The event generation takes the RichDocument as input and produces narrative events as a result. Such a pipeline design has proven to be successful and is also employed in several industrial applications and frameworks [Ferrucci and Lally, 2004; Cunningham et al., 2002]. In addition, the whole framework can be embedded in an environment for big data processing like Apache Spark[15] [Zaharia et al., 2010] to scale up to large document collections.

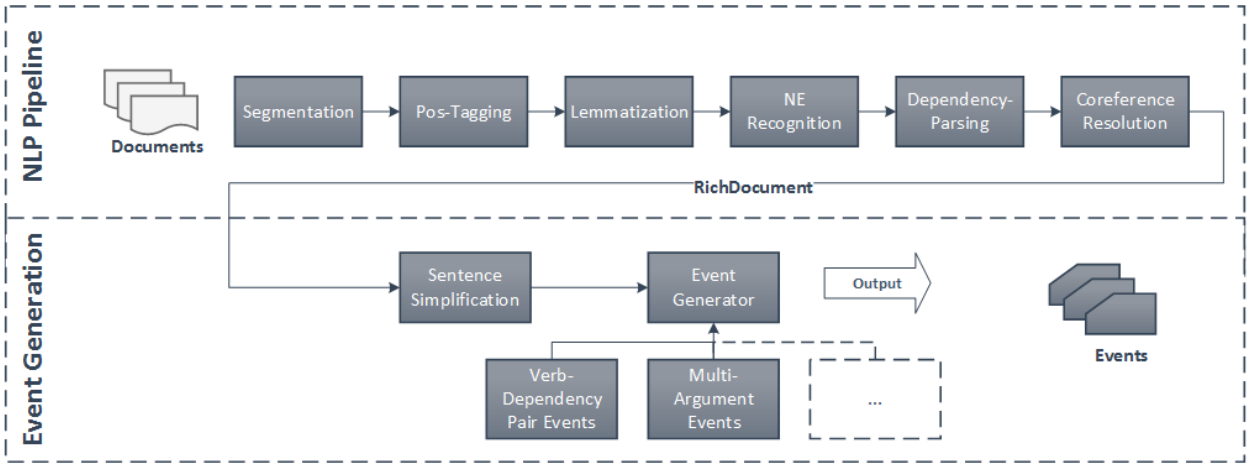


Figure 3.1: Architecture of the Eventos framework.

[15] Apache Spark project page: `http://spark.apache.org/` (accessed June 2016).

3.2.1 Preprocessing

The NLP pipeline consists of several coherent processing units. Each unit performs a different analysis in language understanding and consumes the enhanced output of the previous unit. The individual components can be replaced as long as a RichDocument with the required annotations is provided. The following briefly outlines each component and its usage in the pipeline.

Segmentation

Segmentation in general describes the process of dividing text into meaningful units like words or sentences. Different kinds of text segmentation are typically applied for different tasks in language understanding, such as *paragraph segmentation*, *sentence segmentation*, *word segmentation* and *topic segmentation*.

Sentence segmentation is the problem of recognizing sentence boundaries in plain text. Since sentences usually end with punctuation, the task thus becomes the identification of ambiguous use of punctuation in the input text [Grefenstette and Tapanainen, 1994]. For example, abbreviations like *Dr.* or *i.e.* usually do not indicate sentence boundaries, whereas the question mark or exclamation mark are almost unambiguous examples. Once these usages are resolved, the rest of the separators are non-ambiguous and can be used to delimit the plain text in sentences. This process is important, since most linguistic analyzers require sentences as input units to provide meaningful results.

Word segmentation, also called *tokenization*, is the problem of dividing an input text in word-tokens. A word-token usually corresponds to an inflected form of a word. The following exemplifies the process of tokenization[16]:

> **Input:** John likes Mary and Mary likes John.
>
> **Output:** [“John”, “likes”, “Mary”, “and”, “Mary”, “likes”, “John”]

Tokens are also often referred to as words. However, the term *word* would be ambiguous for the type and token distinction i.e. multiple occurrences of the same word in a sentence are distinct tokens of a single type. The segmentation unit in the pipeline includes both, sentence segmentation and word segmentation for English. These annotations are created with the Stanford *PTBTokenizer* [Manning et al., 2014] that is implemented as a deterministic finite automaton [McCulloch and Pitts, 1988]. All subsequent components require sentence and word annotations.

Pos-Tagging

Pos-Tagging is the process of classifying words into their *part-of-speech (POS)*. Parts of speech are also known as word classes or lexical categories. Those categories have generally similar grammatical properties. For instance, words that belong to the same part of speech show similar usage within the grammatical structure of a sentence. A *part-of-speech tagger* processes a sequence of words and attaches part-of-speech tags to each word automatically.

The collection of part-of-speech tags used is called *tag set*. In practice, various tag sets are used. They differ in terms of granularity and can be grouped into fine-grained and coarse-grained tag sets

[16] The example is taken from the *NLP for the Web* course at TU Darmstadt. Course page: `https://www.lt.informatik.tu-darmstadt.de/de/teaching/lectures-and-classes/winter-term-1516/natural-language-processing-and-the-web/` (accessed June 2016).

such as the *universal tag set* proposed by Petrov et al. [2012]. A prominent fine-grained example is the tag set used in the Penn Treebank Project [Marcus et al., 1994] that comprises 36 different parts of speech.

Figure 3.2 shows a sentence tagged with the part-of-speech labels from the Penn Treebank tag set. This tag set distinguishes between tags for verbs with respect to their form such as tense and case. For example, the tag *VBZ* indicates a 3rd person verb in singular present, whereas *VBG* is an indicator for the gerund form. A similar distinction is made for nouns and pronouns. The words *dog* and *sausage* are classified as singular common nouns (NN) and *my* is labeled as possessive pronoun (*PRP$*). The complete list of tags is available online[17].

The part-of-speech tagged data is required in subsequent processing steps like dependency parsing and is an essential information for the event generation since the extraction patterns rely on it. The pipeline of Eventos uses the maximum-entropy based Pos-tagger (log-linear model) proposed in Toutanova et al. [2003] that achieves state-of-the-art performance on the Penn Treebank Wall Street Journal.

My	dog	also	likes	eating	sausage	.
PRP$	NN	RB	VBZ	VBG	NN	SYM

Figure 3.2: Part-of-speech tagged sentence.

Dependency Parsing

A dependency parser analyses the grammatical structure of a sentence and derives a directed graph between words of the sentence representing dependency relationships between the words. These dependency relations are part of the current dependency grammar theory that is represented by *head-dependent relations* (directed arcs), *functional categories* (arc labels) and *structural categories* like part-of-speech tags.

Figure 3.3 shows a sample dependency parse for the sentence *John loves Mary*. The arc from the node *John* to the node *loves* shows that *loves* modifies *John*. The arc label *nsubj* further describes the functional category. The root of the sentence is identified as the word that has no governor. Within a sentence, there is only one root node.

The dependency parser is one of the most important components in the pipeline. Parses are used to identify individual parts of the sentence required for creating the event representations. The framework uses the transition-based parser described in Chen and Manning [2014]. This parser is based on a neural network and supports English and Chinese.

[17] Penn Treebank labels: `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html` (accessed June 2016).

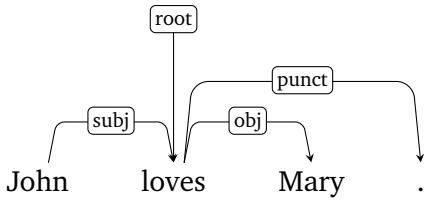


Figure 3.3: Simple dependency parse.

Lemmatization

The goal of *lemmatization* is to reduce the inflected form of a word to a common base form, called *lemma*. This is especially useful for tasks that involve searching i.e. a search engine should be able to return documents containing the words *ate* or *eat*, given the search query *eating*.

To disambiguate ambiguous cases, lemmatization is usually combined with Pos-tagging. Consider for example the noun *dove*, which is a homonym[18] for the past tense form of the verb *to dive*. The combination of Pos-tagging and lemmatization allows to normalize the word *dove* to its proper form, such as *dove* for the noun or *dive* for the verb.

The lemmatizer in Eventos uses the *MorphaAnnotator* from the CoreNLP suit [Manning et al., 2014] that also annotates morphological features such as number and gender. This component maps different inflected verbs to the same base form and is therefore essential to reduce sparsity for the event representation. For example, *go swimming* and *goes swimming* should be mapped to the same event. Additional features such as number and gender are further required for subsequent processing steps like coreference resolution.

Named Entity Recognition

The task of Named Entity Recognition (NER) is to identify and classify atomic elements in documents into predefined categories such as *persons*, *organizations* and *locations*. Current state-of-the-art systems[19] achieve nearly human performance.

In Eventos, the Stanford Named Entity Recognizer [Finkel et al., 2005] is employed. This recognizer uses a conditional random field (CRF) classifier, a probabilistic framework introduced first by Lafferty et al. [2001]. CRFs are a type of graphical model and have been successfully applied to several NLP tasks [Sha and Pereira, 2003; Settles, 2005]. Similar to *hidden Markov model (HMM)*, the algorithm finds the best tagging for an input sequence. However, in contrast to the HMM, CRFs define and maximize conditional probabilities and normalize over the whole label sequence. This allows to use much more features.

For the pipeline, a four class model (location, person, organization and miscellaneous) trained on the CoNLL 2003 named entity data[20] is used. Along with the morphological annotations produced by the lemmatizer, the coreference resolution system uses named entity types as additional feature.

[18] Homonyms is a group of words that share the same spelling and the same pronunciation, but have different meanings. This is a rather restrictive definition that considers homonyms as homographs and homophones.

[19] MUC-07 proceedings: `http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_toc.html#named` (accessed June 2016).

[20] CoNLL 2003 shared task page: `http://www.cnts.ua.ac.be/conll2003/ner/` (accessed June 2016).

Coreference Resolution

Coreference resolution seeks to cluster nominal mentions in a document, which refer to the same entity. A possible clustering of coreference resolution might be: {{*server*, *waiter*, *he*}, {customer, Frank, him, he}, ...}, where each cluster represents an equivalence class. This component requires part-of-speech tags to identify pronouns and also uses features like grammatical information and named entity types to cluster coreferent mentions.

The coreference resolution system used in Eventos implements both, pronominal and nominal coreference resolution [Clark and Manning, 2015]. Next to the dependency parser, the coreference system is the key component for generating narrative chains since it allows to group events that share a common protagonist. For example, all verbs of a document that have one of {*server*, *waiter*, *he*} as argument, will be part of the same narrative chain.

3.2.2 Event Generation

The process of event generation is divided into two components (see Figure 3.1). The first component (*Sentence Simplifier*) creates an abstract representation consisting of relevant parts of the sentence. The second component (*Event Generator*) transforms this intermediate representation into narrative events according to a predefined but exchangeable event template.

Sentence Simplification: Clause and Constituent Identification

Based on the idea of ClausIE [Del Corro and Gemulla, 2013], sentences are split into smaller, but still consistent and coherent units, called *clauses*. A clause is a basic unit of a sentence and consists of a set of constituents, such as *subject*, *verb*, *object*, *complement* or *adverbial*. Each clause contains at least a subject and a verb.

In general, a clause is a simple sentence like *Frank likes hamburgers*. In this case, the clause contains a subject (S), a verb (V) and a direct object (Dobj) and describes one event corresponding to the protagonist *Frank*. However, a sentence can be composed of more than one clause. For instance, the sentence $[\![\textit{Frank likes hamburgers}]\!]_{C1}$ *but* $[\![\textit{Mia cooked vegetables}]\!]_{C2}$ is composed of two independent clauses $C1$ and $C2$ joined via the word *but*. The event generator is expected to create two different narrative events, each for every protagonist. The task of sentence simplification includes therefore the recognition of such composed clauses.

The goal of this phase is to extract the headwords for all constituents of the new clause. If desired, additional dependents can be added in a subsequent processing step. For example, the sentence *The waitress carries hot soup* should create the clause `(S: waitress; V: carries; dObj: soup)`, where *soup* is the headword of the constituent *hot soup* that functions as the direct object in the sentence.

Clauses are generated from subject dependencies like *nsubj*[21], extracted from the dependency graph for a given sentence. This approach is called *verb-mediated extractions* and means that every subject dependency yields a new clause. The subject relation already identifies the subject and the verb as its governor of the clause. All other constituents of the clause are either dependents of this

[21] The dependency parser annotates parses with universal dependencies: `http://universaldependencies.github.io/docs/` (accessed August 2016).

verb or the subject. Objects and complements are connected via *dobj*, *iobj*, *xcomp*, *ccomp* and *cop*, while *nmod*, *advcl* or *advmod* connect adverbials. A set of dependency and surface patterns is used to identify these parts as well.

The following exemplifies two rule subsets, which tackle common problems that are relevant for open information extraction systems. The concepts behind these problems are especially important for the extraction of narrative chains and are not fully supported by state-of-the-art systems as shown in the previous comparison.

Coordinated conjunction processing

As already mentioned, a sentence can be composed of two or more clauses. These clauses are called *conjoints* and are usually joined via coordinated conjunctions also known as coordinators such as *and* or *or*. For instance, the example in Figure 3.4 shows the conjunction *and* in a subject argument. As it is the interest to create separate events for both entities, independent clauses for each entity need to be generated. The given sentence should therefore create the following two clauses:

(1) `Clause(S: Sam; V: prefer; Dobj: apples)`

(2) `Clause(S: Fry; V: prefer; Dobj: apples)`

Different dependency parsers use different styles of dependency representation [Ruppert et al., 2015; Chen and Manning, 2014]. *Basic dependencies* as presented in Figure 3.4a are a surface-oriented representation, where each word in the sentence is the dependent of exactly one other word. The representation is strictly syntactic and broadly used in applications like machine translation, where the overall structure is more important than the individual relation between content words. However, the task of extracting narrative events recognizes the dependency structure as a semantic representation. From this point of view, basic dependencies follow the structure of the sentence too closely and therefore miss direct dependencies between individual words. For example, the word *Fry* stands in subject relation with the verb *prefer*, but there is no direct connection between them. Given those dependencies, the system would only identify one clause with *Sam* as subject, *prefer* as verb and *apples* as direct object.

In contrast, the *collapsed dependencies* as shown in Figure 3.4b are a representation that is more semantic. Here, dependencies such as prepositions or conjuncts are collapsed to direct dependencies between content words. For instance, the coordinated conjunction dependency in the example will be collapsed into a single relation. As a result, the relations `cc(Sam-1, and-2)` and `conj(Sam-1, Fry-3)` change to the collapsed dependency `conj:and(Sam-1, Fry-3)`[22].

Given dependencies in the collapsed representation, another mechanism called *dependency propagation* can be used on top to further enhance the dependencies. This mechanism propagates the collapsed conjunctions to other dependencies involving the conjuncts. For instance, one additional dependency can be added to the parse in the example i.e. the subject relation of the first conjunct *Sam* should be propagated to the second conjunct *Fry*. Figure 3.4c illustrates the result of the propagation.

The collapsed and propagated representation is useful for simplifying patterns in the clause extraction. Thereby, extractions are less prone to errors due to simpler and much more manageable

[22] Inline dependency representation:
dependency_label(govenorGloss-govenorIndex, dependentGloss-dependentIndex).

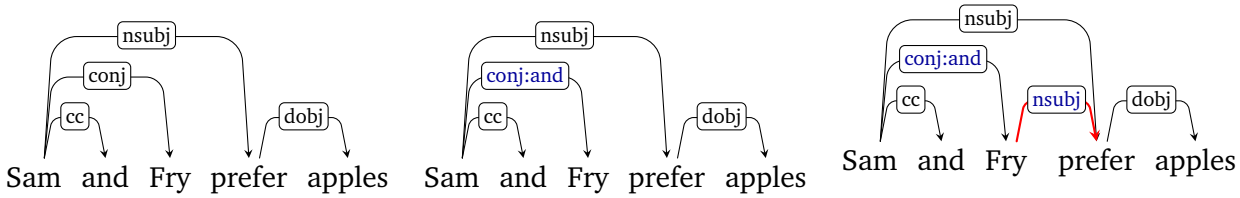
rules. It also solves the problem of obtaining multiple clauses for conjunctions in both, verb and subject arguments as illustrated below.

(1) ⟦Tim and Frank⟧$_{Subject\_Arg}$ like swimming.

(2) Tim likes ⟦swimming and dancing.⟧$_{Verb\_Arg}$

The first sentence exemplifies the use of a conjunction in a subject argument similar to the example in Figure 3.4. The second example shows the usage of a conjunction in a verb argument, where the same entity is associated with two actions. Likewise, the system is expected to generate two independent clauses in this case. However, in contrast to the first example, the two clauses correspond to the same protagonist. To return to the previous example in Figure 3.4c, the system generates two independent clauses using the collapsed and propagated dependencies. One clause for the original subject relation `nsubj(Sam-1, prefer-4)` and another clause for the propagated dependency `nsubj(Fry-3, prefer-4)`.

Collapsed dependencies and propagation mechanisms have been successfully implemented in several dependency parsers [Ruppert et al., 2015; Chen and Manning, 2014]. Eventos uses the Stanford dependency parser [Chen and Manning, 2014] as a basis that produces typed dependencies in the collapsed and propagated representation. Find further details about the parser in Section 3.2.1.



(a) Basic Dependencies (b) Collapsed Dependencies (c) Collapsed and Propagated Dependencies

Figure 3.4: Illustration of different styles of dependency representations.

Relative clause processing

As opposed to the other two tested systems, Eventos implements additional rules to process relative clauses. Those were added to increase the informativeness of extractions e.g. by replacing relative pronouns (e.g. *who, which*, etc.) with its antecedents. English differentiates between two types of relative clauses (1) *defining relative clauses* and (2) *non-defining relative clauses*. The system supports both cases.

A defining relative clause is a subordinate clause that modifies a noun phrase and adds essential information to it. This type of clause follows the pattern *relative pronoun as subject + verb* and can occur after the subject or the object of the main clause. Without the relative clause, the sentence is still grammatically correct, but its meaning would have changed. As a subject of the subordinate clause, the relative pronoun can never be omitted. Consider the following two examples, where the relative clause is underlined and the associated relative pronoun is marked in bold:

(1) The boy **who** lost his watch was careless.

(2) She has a son **who** is a doctor.

In the first sentence, the relative pronoun is the subject of the subordinate clause and references the subject of the main clause. For that reason, the relative pronoun *who* becomes the subject argument of the second clause. For the two subject dependencies, the system would therefore extract the clauses as `Clause(S: boy; V: be; C: careless)` and `Clause(S: who; V: lost; C: watch)`. However, after this transformation, no evidence is left to which entity *who* refers to. Furthermore, the coreference resolution system is not able to resolve the relative pronoun, because it is only capable to cluster personal pronouns and nominal mentions. Hence, the event generated from the second clause cannot be assigned to the narrative chain corresponding to the *boy*.

To solve this problem and to increase the informativeness of the extraction, the pronoun *who* is resolved to the entity mention *boy*. This is achieved with a surface pattern that matches the relative clause dependency relation and extracts the relative pronoun together with its associated representative mention. Although the relative pronoun follows the object and not the subject of the sentence in the second example, the same rule can be applied.

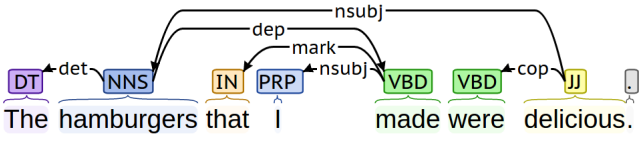
In contrast, a non-defining relative clause adds extra information, which is not necessary for understanding the statement of a sentence. In this case, the relative pronoun functions as an object of the subordinate clause. In comparison to the defining-relative clause, the relative pronoun can also be omitted as shown in the following examples:

(1) The hamburgers **that** I made were delicious.

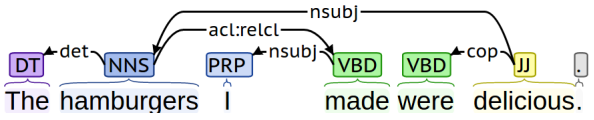
(2) The hamburgers I made were delicious.

Although the relative pronoun is missing in the second sentence, the representative mention *hamburger* functions as object of the relative clause. This observation is used to extract the same clauses for both cases. The framework creates the clauses in both examples accordingly as `Clause(S: I; V: made; Dobj: hamburgers)` and `Clause(S: hamburgers; V: be; C: delicious)`.

Figure 3.5a and Figure 3.5b additionally show the corresponding dependency parses for both situations.



(a)



(b)

Figure 3.5: Non-defining relative clause in which the relative pronoun *that* functions as the object of the subordinate clause and follows after the subject of the main clause. The images are created with the web visualizer at `http://nlp.stanford.edu:8080/corenlp/process` (accessed August 2016).

The generation of open information facts is a flexible process as different applications require different representations. This also applies to event representations for generating narrative chains. Recent work has successfully shown the value of different forms of event representation for representing common-sense knowledge in machines [Ahrendt and Demberg, 2016; Pichotta and Mooney, 2016, 2014]. Some approaches depend on triple such as `(Thomas,plays,football in Munich)`, whereas others are based on n-ary extractions like `(Thomas,plays,football,in,Munich)` as described by Pichotta and Mooney [2016] or Granroth-Wilding and Clark [2016].

Similarly, the granularity and form of extractions varies. One could consider to represent the protagonist through the whole nominal phrase or just by its headword. For instance, the subject in *Thomas Mueller from FC Bayern plays soccer in Munich* can be represented as *Thomas* or more specialized as *Thomas Mueller from FC Bayern*. The same holds for the relational part of the extraction that can be represented as *plays* or *plays in*. The latter also considers the verb particle as a fragment of the narrative event. A potential variation might be also the incorporation of negated expressions or conditionals into the event representation. This emphasizes the separation of information gathering that tackles the question of *What information is expressed?* and its actual representation in a two-step approach.

Several event generators were implemented for experiments, not only to existing proposals from recent work, but also new representations not used so far. Each event generator utilizes the intermediate clause representation of the sentence simplification unit and generates narrative events enhanced with coreference information. Narrative chains can then be build by grouping together all events that share the same protagonist i.e. the same coreference key in one of its arguments.

The following presents and motivates the different event representation used in the experiments. Each representation is illustrated with examples and the section concludes with a comparison between all proposed representations.

Verb-dependency pair events

The verb-dependency pair event representation is an adoption of the approach presented by Chambers and Jurafsky [2008]. This representation models a narrative event as a pair consisting of the verb lemma and the grammatical dependency relation between the verb and the protagonist. For their experiments, Chambers and Jurafsky considered subject and direct object dependency relations. Here, the representation has been extended to model not only subjects and direct objects, but also indirect objects. Formally, a narrative event $e = (v, d)$, is a verb lemma v that has some protagonist as dependency d, where d is in {subj, dobj, iobj}.

For example, the sentence *Sandy ordered a large pizza and she ate it all alone* generates two narrative chains corresponding to the protagonists *Sandy* and *pizza*. The first chain about *Sandy* consists of the two pair events, modeled as `(order,subj)` and `(eat,subj)`. The second chain is associated with *pizza* and also contains two events that are represented as `(order,dobj)` and `(eat,dobj)`.

Multi-argument events

The representation so far only considers the verb and its syntactic relation like `(arrest,dobj)`. The given event indicates that somebody or something is arrested, because the protagonist stands in an object relation to the verb. In this case the verb contains the most important information. However, the argument often changes the meaning of an event e.g. *perform play* vs. *perform surgery*[23]. In other cases, the verb carries almost no meaningful information as in `(go,subj)`. In that sense *going to the beach* is the same as *going to heaven*. This raises the need of having richer semantic representations for narrative events.

As one of the first, Pichotta and Mooney [2014] proposed a script model that employs events with multi-arguments. They define a multi-argument event as a relational atom (v, e_s, e_o, e_p), where v is the verb lemma and e_s, e_o and e_p are possibly-null entities, which stand in subject, direct object and prepositional relation to v, respectively. Multi-argument events can have arbitrary number of arguments with different grammatical relations. For instance, a multi-argument event could be modeled with predicative adjectives rather than with prepositional relations. Though, the representation needs to capture the underlying story of a document and describe the most important narrative information.

Similar to Pichotta and Mooney [2014], multi-argument events are represented as 4-tuples. However, instead of prepositional phrases, indirect objects are added to the representation. Thus, a multi-argument is described as $v{:}d(e_{subj}, e_{dobj}, e_{iobj})$, where v is the verb lemma and e_{subj}, e_{dobj} and e_{iobj} are possibly-null entities that stand in subject, direct object and indirect object relation to v, respectively. The value of d specifies the relation of the protagonist e_d for each event in a narrative chain[24]. The following example illustrates the representation:

(1) `take:dobj(PRP,train,•)`, `schedule:subj(train,•,•)`, `leave:subj(train,•,•)`, `exit:dobj(PRP,train,•)`

This chain describes the common scenario for *traveling with the public transport* from the point of view of the *train*: After boarding the train, it leaves the platform according to a scheduled time. Once arrived at the destination the passengers leave the train.

The filler (•) indicates that no entity stands in that dependency relation with the verb. To reduce sparsity, the place holder *PRP* replaces personal pronouns such as *he* or *we*. This generalization is an optional post-processing step.

Multi-argument events with supersenses

Script models trained with narrative events need not simply learn the knowledge that is encoded, but rather have to generalize over the training data to apply their knowledge to new situations. This is even more important for multi-argument events, since such richer semantic representations are more specific. For example, the multi-argument events `drive:dobj(busman,vehicle,•)` and `drive:dobj(driver,bus,•)` describe the same situation though it is expressed differently due to the ambiguous nature of natural language.

[23] The example is taken from Granroth-Wilding and Clark [2016].

[24] This event notation is similar to the formalization used by Pichotta and Mooney [2014], but explicitly illustrates the relation d of the protagonist. Although Pichotta and Mooney omit this information in their formalization, they use it in their script model. The enhanced formalization should emphasize that the relation d is used for narrative learning.

GROUP	place	*EVENT*	experience	*NATURAL OBJ*	flower	*PLANT*	tree
PERSON	people	*MOTIVE*	reason	*RELATION*	portion	*TIME*	day
ARTIFACT	car	*POSSESSION*	price	*SUBSTANCE*	oil	*STATE*	pain
COGNITION	way	*ATTRIBUTE*	quality	*FEELING*	discomfort	*SHAPE*	square
FOOD	food	*QUANTITY*	amount	*PROCESS*	process	*OTHER*	stuff
ACT	service	*ANIMAL*	dog	*PHENOMENON*	result		
LOCATION	area	*BODY*	hair	*COMMUNICATION*	review		

Table 3.1: Different noun sense categories taken from WordNet. The categories represent top-level hypernyms in the taxonomy (Source: Schneider and Smith [2015]).

To abstract from the information that is encoded, the *supersense tagger* described in Schneider and Smith [2015] is applied to the arguments of the event representation. Supersenses offer coarse-grained semantic labels for lexical expression and are broadly applicable such as in question answering or machine translation. These supersenses are labels based on WordNet's lexicographer files [Fellbaum, 1998] and represent top-level hypernyms in the taxonomy. WordNet contains 15 supersenses for verbs and 26 supersenses for nouns. The different noun categories along with the most frequent lexical item in the *STREUSLE corpus*[25] are listed in Table 3.1.

The task of *supersense tagging* can be considered as a form of coarse word sense disambiguation. The goal is to assign each noun and verb its appropriate sense according to the given list of supersenses. The supersense tagger employed here is based on a sequence tagging model that uses *a first-order structured perceptron* [Collins, 2002]. For the event representation only noun arguments are replaced with their supersense. The following example shows the differences to the event representation using raw arguments:

(1) **Multi-argument representation:**
`get:subj(PRP,snack,•)`, `watch:subj(PRP,scenery,•)`, `enjoy:subj(PRP,•,•)`, `see:subj(PRP,place,•)`, `arrive:subj(PRP,hour,•)`, `pick:dobj(friend,PRP,•)`

(2) **Multi-argument representation with supersenses:**
`get:subj(•,FOOD,•)`, `watch:subj(•,ARTIFACT,•)`, `enjoy:subj(•,•,•)`, `see:subj(•,GROUP,•)`, `arrive:subj(•,TIME,•)`, `pick:dobj(PERSON,•,•)`

In addition, where either an argument is not filled with a dependency relation or there is no supersense available, a place holder (•) is inserted for that argument. The generation of such events also addresses the problem of assigning the same supersense to different mentions, which reference the same entity. For instance, the coreferent entities *bus* and *it* in the narrative chain `drive:subj(bus,•,•)`, `stop:subj(it,•,•)` should be tagged with the same supersense.

There already exist a few approaches that utilize supersenses in the context of frames and events. For example, Rusu et al. [2014] cluster events by applying supersenses to obtain a generalized event representation and Coppola et al. [2009] learn domain-specific frames by matching the frame elements to the associated supersenses. However, to date, no work has integrated supersenses into event representation for narrative learning.

[25] Corpus annotated with multiword expressions and supersenses for nouns and verbs [Schneider and Smith, 2015].

Multi-argument events with participant labels

Recently, Ahrendt and Demberg [2016] have proposed an event representation that captures script-relevant entities in narrative chains. This approach is similar to the supersense tagging, but assigns participant roles to entity arguments for a given scenario. Like in the supersense representation, all other entities are mapped onto a single *other* representation.

Following Ahrendt and Demberg, the max-hypernym heuristic [Kampmann et al., 2015] is used to label the arguments with participant roles. This heuristic combines information from WordNet and information about coreferent protagonist mentions to automatically categorize these mentions in terms of their role within the script.

The participant labeler assigns one role label for each coreference chain c. In order to achieve this, it calculates the similarity between all synsets that are associated with one of the words in c and all the synsets that correspond to a participant label. The similarity calculation uses NLTK's[26] build-in path similarity. The max-hypernym heuristic assigns then the highest obtained similarity score for this label. The algorithm ignores all words that do not represent noun mentions. For instance, it ignores personal pronouns like *he*, because this mention refers to the synset associated with *Helium*.

The example in Figure 3.6 exemplifies the algorithm. Each of the words on the left side of the figure corresponds to one synset obtained from WordNet for the participant label *bus staff*. The right side of the figure shows all distinct protagonist mentions for one coreference chain. The edges represent the different similarities obtained for each pair of synsets. The maximum of all similarity values will be assigned as score for the coreference chain and the label *bus staff*.

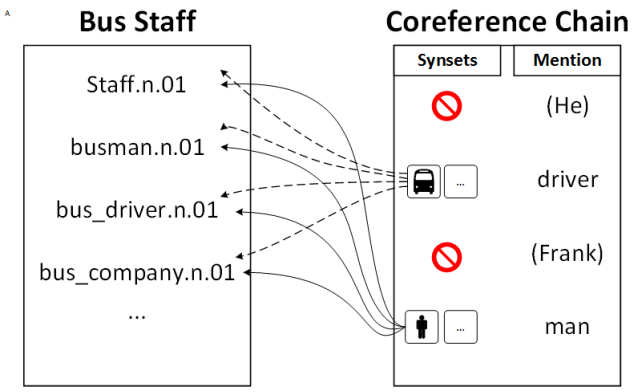


Figure 3.6: Similarity between a set of participant label synsets and a coreference chain.

[26] Project page: `http://www.nltk.org/` (accessed June 2016).

The following example illustrates the difference between all three event representations using the story below. The sequence of events highlighted in bold corresponds to the main protagonist referred to *bus*. The narrative chain describes the common scenario of *riding a bus*.

> *[...] I decided to* ***take*** *the bus. [...] At the bus stop, I waited only two minutes before the bus* ***arrived****. [...] The trip into town took fifteen minutes because the bus* ***had*** *other stops to make in the area before returning to the station . [...] The driver* ***slowed*** *down , and* ***maneuvered*** *the bus into its numbered space . [...] I got off the bus and went to the theater.* Source: Modi et al. [2016]

(1) **Multi-argument representation:**
`take:dobj(PRP,bus,•)`, `arrive:subj(bus,•,•)`, `have:subj(bus,stop,•)`,
`slow:dobj(driver,bus,•)`, `maneuver:dobj(driver,bus,•)`

(2) **Multi-argument representation with supersenses:**
`take:dobj(•,ARTIFACT,•)`, `arrive:subj(ARTIFACT,•,•)`, `have:subj(ARTIFACT,EVENT,•)`
`slow:dobj(PERSON,ARTIFACT,•)`, `maneuver:dobj(PERSON,ARTIFACT,•)`

(3) **Multi-argument representation with participant labels:**
`take:dobj(•,bus,•)`, `arrive:subj(bus,•,•)`, `have:subj(bus,bus_stop,•)`,
`slow:dobj(bus_staff,bus,•)`, `maneuver:dobj(bus_staff,bus,•)`

The supersense and participant label representation look similar in terms of common concepts. However, while the supersense representation assigns broad and general concepts, the participant label representation annotates domain-specific labels. For example, instead of recognizing the word *driver* as a general *PERSON*, the participant label representation identifies it as *bus_staff*.

Moreover, the application of supersenses results in more generalization since more specific terms are mapped to the same category than this is the case for the role labels. For the *riding a bus* scenario, the argument *time* in `know:subj(PRP,time,•)` and the argument *stop* in `have:subj(bus,stop,•)` correspond to the role labels *time/date* and *bus_stop*, whereas the supersense *EVENT* captures both arguments. The question remains whether this generalization also results in better performance.

Both representations omit arguments where no general role or supersense is available. In contrast, the raw multi-argument representation only replaces personal pronouns and does not omit extracted information.

4 Visualization of Narrative Chains

This chapter presents *FactBro*, a platform that enables users to explore events and narrative chains in text documents without requiring any expert knowledge. FactBro annotates and highlights events in the document and offers a user-friendly interface to explore the different narrative chains contained in the respective document.

The combination of natural language methods and information visualization techniques further enables the use of such an application for the purpose of getting a broad and fast overview of the most important information in a large document. For that reason, an outline always presents the current visible facts in the document. This is achieved by using an automatic alignment algorithm that synchronizes facts in the outline and events visible in the document. Section 4.1 gives an overview for each component of the user-interface and presents details about the alignment algorithm. The utility of such an application is evaluated in Section 4.2.

4.1 Event Browser Overview

FactBro[27] consists of two interlinked views: (1) *the document view* and (2) the *document outline*. Figure 4.1 shows the user-interface (UI).

Figure 4.1: UI of FactBro showing the document view (left side) and event outline (right side).

[27] Online video available at `https://youtu.be/mSI7qm_o-VQ` (accessed August 2016).

Document View

The document view gives an overview of the document and its content. For each document the verb fragment of the event is annotated and colored in blue. Once the user hovers over an annotated verb the associated arguments are highlighted for multi-argument events. By hiding additional information per default the overflow of visual stimuli is reduced.

The application has support for all event representations proposed in Chapter 3, which can be enabled through the drop-down menu shown in Figure 4.1. This feature controls the granularity of the presented events in the document view and the outline. The verb-dependency pair event representation annotates only verbs, whereas the multi-argument representation also adds additional arguments to the outline.

FactBro can visualize pre-deployed document collections or process and represent events and narrative chains for new text. The user can change the current active collection in the drop-down menu below the document view and explore it with the *next* and *previous* document button.

The platform uses the Eventos framework as a basis and adds an additional visual layer for the processed documents and the extracted information.

Document Outline

Starting from an overview, the user can utilize the outline to further explore the document. Its main objective is to browse through the interlinked knowledge structures that represent the underlying events in the document. The outline shows more events than present in the current view-port of the document, but always marks the current visible events in bold. In this way, the user can focus on the current part of the document, while having a broader overview about upcoming events.

For each event in the outline, a representative mention replaces entities represented by personal pronouns like *she* or *we*, whenever coreference information is available for a certain entity. For the representative mention, the most frequent coreferring mention is selected considering only lemmatized noun mentions. This feature increases the informativeness of the outline and gives the reader direct reference to the antecedent the pronoun refers to without checking the document.

When the user scrolls the document pane, the outline pane adjusts itself automatically according to the currently visible events in the document. This feature is called *scroll-link*, which is especially useful for markdown editors in order to match the markdown-preview with its markdown source.

The simplest syncing approach is to keep the relative position of both scrollbars in sync. This means, if the document pane is scrolled to 20% of its height, the outline pane is also scrolled 20% of its height. However, the content between the two panes is very disproportional since a sentence can yield multiple events. Therefore, the pane scrollbars can easily go out of sync. The platform implements a more elaborate approach that uses invisible markers between both panes. While this approach yields decent results, it is far more complex and should be avoided if the content is more or less proportional between both panes.

The algorithm works as follows. Invisible markers are inserted between all event triggers in the document view and additional matching markers are added to the outline view. When the document pane is scrolled, it is determined, which of these markers are currently visible in the view-port. The outline pane is then adjusted to show the same hidden markers. An additional offset between both panes is also taken into account to make the result more accurate.

The mechanism for the scroll alignment works in both directions. So, when the outline is scrolled the document view is also adjusted accordingly (and vice-versa). The *directional* button can be used to disable the two-way scrolling for the outline scrollbar. In case the feature breaks, the user can still deactivate it and scroll manually.

Narrative chain view

In contrast to the outline, the narrative chain view allows to focus on a specific actor in the document and enables to explore the actions associated with this actor. At a first glance, the view shows a grouped list of actors referenced throughout the document. Each group refers to a set of coreferring mentions in the document. When a group is selected, the events associated with this actor will be shown. Figure 4.2 presents the narrative chain view for the story given below.

> *Harris was poor. He decided to rob a bank. He held up the bank at gunpoint and stole thousands of dollars. He escaped in his getaway car. Harris could not outrun the police and was caught.* Source: ROCStories corpus [Mostafazadeh et al., 2016]

The story contains one narrative chain with *Harris* as the main actor. The personal pronoun *he* also represents *Harris* in the story and is therefore visualized in the same actor list. However, no narrative chain could be resolved for *bank*, because the coreference resolution system failed in extracting it as mention. Also no chain is created for *police* since it is only mentioned once.

When a user selects a narrative chain as shown on the right side of Figure 4.2, the sequence of events associated with this chain is visualized. The individual events are ordered according to their occurrences in the document. For this example, the temporal order of the events is in line with the occurrence in the document. However, the order in text does not necessarily have to match the real-world order. Humans tend to use common-sense knowledge to reason about the temporal order of events in addition to the information available in the text. This is even possible, if the information cannot be inferred from the document. For instance, *eating food* in a restaurant usually precedes *leaving it*. A natural extension would therefore be to include temporal ordering information to the visualization. Such an extension is discussed in the outlook in Section 7.2.

The narrative chain for *Harris* consists of eight events starting with a predicative adjective that captures Harris current situation i.e. *he was poor*.

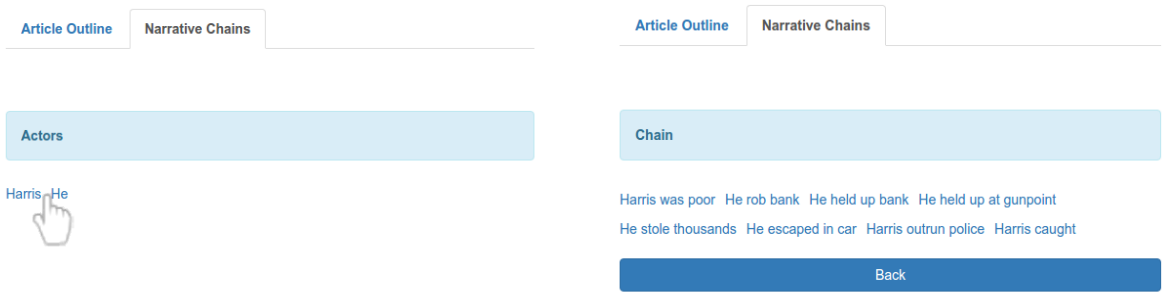


Figure 4.2: The figure demonstrates the narrative chain view for the given story above. The left side of the figure shows the actor overview, whereas the right side lists the individual events associated with the respective actor.

Subsequently, each event describes a major happening in the story until Harris gets finally caught. By clicking on an event the source sentence is shown and highlighted as additional context in the document view. However, even without the source sentences it is possible to deduct the story just by looking at the sequence of events.

4.2 Evaluation

The goal of this experiment is to assess the overall quality difference of a system that makes use of an event outline in comparison to one that does not. This is done via an *extrinsic evaluation* [Clark et al., 2010] that measures the quality of the event outline by looking at its impact on the performance of the overall system. This is in contrast to an *intrinsic evaluation*, where the evaluation only considers an isolated part of the system. A variation of an extrinsic evaluation is conducted, which not only includes the system as a whole, but also includes users and their interactions. The quality of the system is then indirectly evaluated by considering (1) the whole system with the event outline and (2) the system without the additional outline.

The objective of the experiment is to test whether the outline feature gives a broad and fast overview of the documents and supports users in finding facts in large documents. The following presents the procedures for the experiment, the experiment's results and an analysis of those results.

Procedure

For the evaluation, 40 users are asked to perform a standardized task i.e. finding answers in documents to a set of test questions. For each question the time taken to find the answer in the document is measured. All of these users are computer science students between 20 and 30 years with a good command of English. The participants have not seen the application before and they all received the same introduction to the tool prior to the evaluation (approximately 3 minutes).

The experiment is conducted with a *between-subject design* [Gray and Bjorklund, 2014] that has two different groups of users. The first group is called the *treatment group* and is allowed to make use of the event outline, whereas the outline is being disabled for the second group. Both groups are equal in their size and users are distributed randomly. A major disadvantage that comes along with the between-subject design is that it often requires a large number of participants to generate reliable data.

The presence of the event outline feature is considered as an *independent variable* of the experiment. In this context, the second group is called the *control group*, because the independent variable being tested cannot influence the results for this group. This procedure helps to isolate the independent variable's effect on the experiment. On the other hand, the time taken to complete the task represents the *dependent variable*[28] of the experiment.

The documents used in this evaluation are obtained from the *Simple English Wikipedia*[29] through the *Wikimedia API*. The Simple English Wikipedia was selected because it only uses basic words, simple grammar and shorter sentences. This aims to minimize the influence of language skills

[28] Dependent variables are variables whose values are predicted using the independent variables.
[29] Simple English Wikipedia project page: `https://simple.wikipedia.org/wiki/Main_Page` (accessed June 2016).

in this task and should increase the reliability of the results. For example, a non-native speaker should be able to complete the task while not being blocked by his language skills.

The experiment includes a subset of 16 documents that are marked as *very good articles* by the Wikipedia's editors. These articles are characterized by their correct style according to the requirements of very good articles[30]. This property helps to identify comprehensive articles containing many facts. On average each document contains 109 sentences.

The events shown in the outline and documents presented in the document view are generated from the plain text of the Wikipedia articles. Thus, additional Wikimedia markup such as image captions, references, tables and info boxes are removed from the documents in a preprocessing step.

Question generation and question wording is a difficult and crucial task. Several work has been focusing on automatic question generation [Rus et al., 2007; Aldabe et al., 2006]. However, automatically generated test items have to be post-edited and the implementation of an automatic question generation system is beyond the scope of this thesis. Hence, the set of test questions is created manually in advance. For each document, a random sentence with at least one term is selected. The sentence is then transformed into a question by changing its order. Questions generated in this way need to meet the following requirements:

(1) The question should omit unnecessary and irrelevant material.

(2) The question should avoid giving clues to the correct answer e.g. "... is called **an**?".

(3) There is only one valid answer to the question in the whole document.

(4) The answer consists of one word that also appears in the relevant sentence of the document.

If the generated question does not meet the criteria above, another random sentence is selected until a valid question can be created. As an example, consider the sentence *Percy knows that Ares has tricked him when he finds the bolt in his backpack* taken from the article about the novel *The Lightning Thief*[31]. This sentence will be further transformed into the question *Where does Percy find the bolt?* with the word *backpack* as correct answer. In addition, irrelevant parts of the sentence are omitted in the question.

For a number of questions there are multiple possible answers, which can often only be resolved given sufficient context. A common example is the question *Where is the Taj Mahal?*[32]. Besides to the *mausoleum in Agra*, there is the *Taj Mahal casino* in Atlantic City, New Jersey. Such questions are not uncommon, because entities are often ambiguous. However, in this experiment only the term Agra is accepted as an answer for a document about the Taj Mahal, because it is the only word that appears in the document and answers the question. Other answers that might be correct in general, but do not appear in the document are therefore not accepted as correct answers.

After a first test, the summary sections from the documents were removed since these paragraphs often contain clues leading to the correct answer. Also a few misleading questions were corrected that caused participants to give wrong answers. To not affect the results of the experiment, the pilot test results are excluded from overall evaluation.

[30] Requirements for a very good article: `https://simple.wikipedia.org/wiki/Wikipedia:Requirements_for_very_good_articles` (accessed June 2016).

[31] Source for the example sentence: `https://simple.wikipedia.org/wiki/The_Lightning_Thief` (accessed June 2016).

[32] The example is taken from Clark et al. [2010].

During the realization of the experiment, no systematic error in the measurement process that correlates with the feature under test was detected. In this case, a systemic error could be i.e. caused by a participant group that contains significantly more native-speakers than the other group. As all participants in the experiment are non-native speakers and randomly distributed into groups, it is assumed that the results are unaffected. Another systematic error could be the use of different hardware setups for the evaluation. A setup that includes a mouse with a scroll wheel could be superior against a setup with no scroll wheel in terms of task time. The same problem applies to the usage of displays, which are different in their size. However, all participants use the same hardware for the evaluation.

All question and answer pairs as well as the source sentences and their position in the document are listed in the Appendix in Section A.1.

Results

For the results, only correct answers are considered among all participants. An answer is deemed to be correct, if it is an exact or partial match of the gold standard answer, ignoring case. For example, the answer *Lung Cancer* matches the gold standard answer *cancer* as partial match.

Figure 4.5a shows the document number as a function of the time taken to find the answer. The individual results are averaged over the total number of participants using the geometric mean. The geometric mean is an arithmetic mean after taking the logarithm of each value as illustrated in Equation 4.1.

$$\Big(\prod_{i=1}^{N} x_i\Big)^{1/N} = exp\Big[\frac{1}{N} \cdot \sum_{i=1}^{N} \ln x_i\Big] \tag{4.1}$$

The values x_i characterize the underlying data series that has the size N. The geometric mean was favored over the arithmetic mean here, because the latter is highly sensitive to extreme values. Consider the data series $x = [3, 4, 5, 6, 8082]$. The corresponding arithmetic mean is calculated as $\overline{X} = \frac{1}{5}\sum x_i = 1620$, which does not tell anything about the level of individual values. This behavior makes the arithmetic mean much more prone to sampling errors.

In addition, most confidence interval formulas make the assumption that the sampling distribution of the estimator follows approximately a *standard normal distribution*. However, time data as measured in this experiment is almost always positively skewed. This observation is also expressed in the difference between the arithmetic and geometric mean i.e. the geometric mean is lower than the arithmetic mean (compare Figure 4.5b and Figure 4.5a). If the data would be normally distributed, the two values would be almost the same.

Note that the sampling error is negligible for sufficiently large N. At least for $N \to \infty$, the mean of sample sets is normally distributed, even if the underlying data is not normally distributed. Thus, the following equation is approximately true regardless of the shape of the population distribution:

$$\overline{X} \sim \mathcal{N}(\mu, \frac{\sigma}{\sqrt{N}}) \tag{4.2}$$

where $\mu = E(X_k)$ is the expected value and $\sigma^2 = var(X_k)$ models the variance. This observation is a direct consequence of the central limit theorem (without proof). Though, for small N the approximation error is large.

To reduce the influence of this error a log-transformation is applied to the data series[33]. Howell [2012] found this transformation to be one of the best transformations for time data. The geometric mean and the geometric standard deviation of the resulting log-normal distributed population are used to calculate the 95 percent (quantile) confidence intervals. The confidence intervals and estimates are finally back transformed to the original units for interpretation. One consequence of calculating the confidence intervals from the transformed data is that the intervals are not symmetric around the geometric mean.

The chart in Figure 4.5a highlights the benefit of the performed data transformation. It is divided into two parts. The dark gray bars describe the average time taken to find the answer by using the document outline. The light gray bars in contrast show the average time taken without the respective feature. The average time is calculated with the geometric mean and is tabulated in seconds for both data series. The bold numbers located above the bars of each document represent the sentence indices of the sentences that contain the answers. These numbers are referred to as *answer-sentence* indices in the following.

The Y error bars in the negative and positive direction show the measurement uncertainty of the measurements for a confidence range of 95 percent. For instance, one can be 95 percent confident that the average time with the outline feature for the first document is between 96.21 and 133.49 seconds. Likewise, the average time without the outline feature for the first document is with a probability of 95 percent between 217.98 and 245.14 seconds.

The chart in Figure 4.5b shows the same information, but averages the time with the arithmetic mean instead. Here, the difference observed between the two groups for document 12 is not statistically significant, as the error margin overlaps. The difference in Figure 4.5a in contrast is statistically significant for the same document. Similar observations can be made for document three. This shows that the data transformation is beneficial to reduce the sampling error in this experiment.

The chart in Figure 4.3 summarizes the accumulated results. It highlights the overall time taken for all users and documents. The total time series is approximately normally distributed for each group. Thus, the arithmetic mean is used to average the results. The Y error bars show a 95 percent confidence interval, which is symmetric around the arithmetic mean. The individual times as measured for both groups appear in Table A.2 and Table A.3 of the Appendix in Section A.2.

[33] The analysis and log-transformation is implemented in the statistical programming language R [R Development Core Team, 2008]. The associated code can be found in the Appendix in Section A.3.

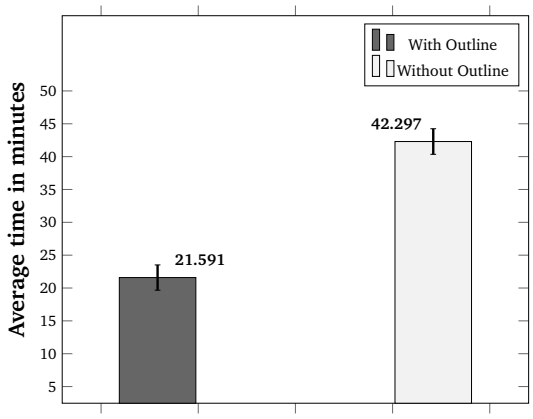
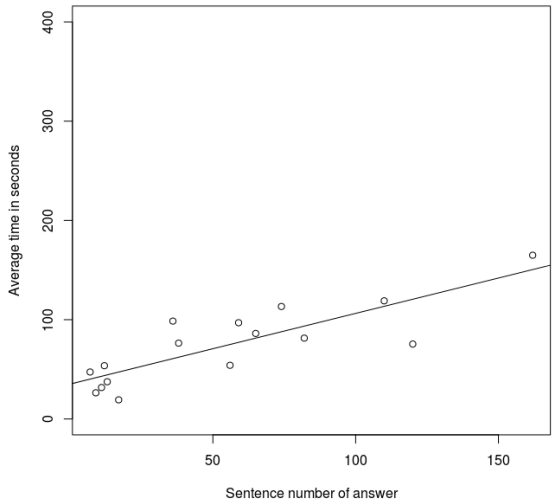
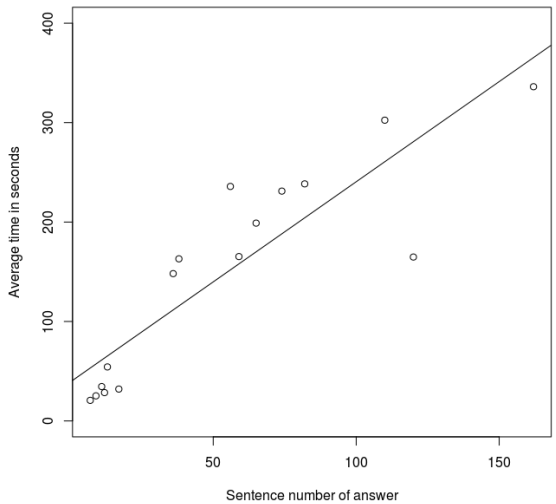


Figure 4.3: The chart shows the cumulative results of the experiment. Results are averaged for all users and documents using the arithmetic mean and are tabulated in minutes. The dark gray bar illustrates the results for the treatment group, whereas the light gray bar shows the results for the control group. The black numbers above both bars represent the average time.

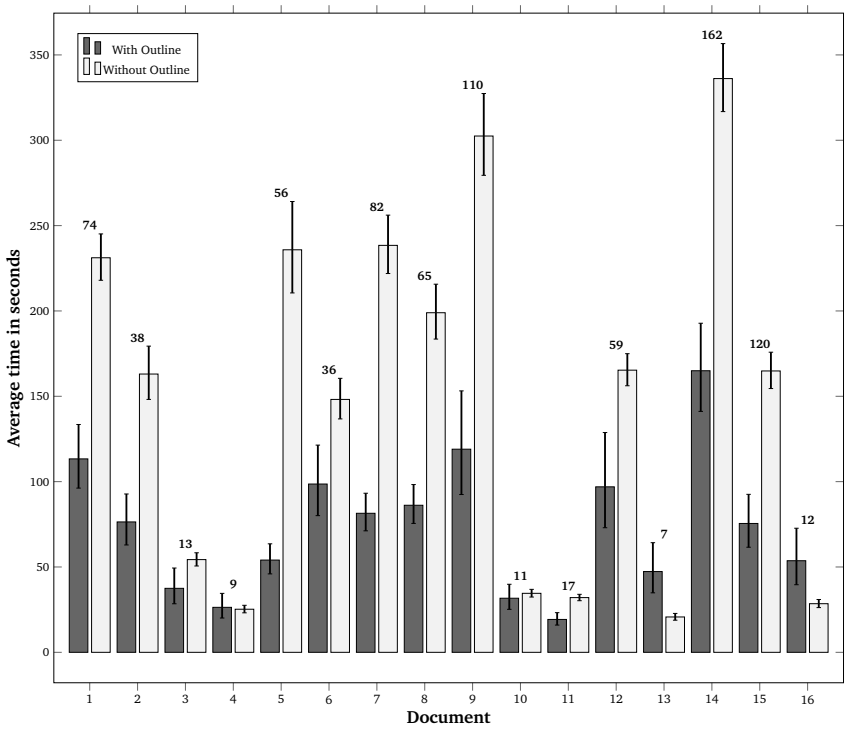


(a) Treatment group (r = 0.8376, p-value = 5.148^{-05}).

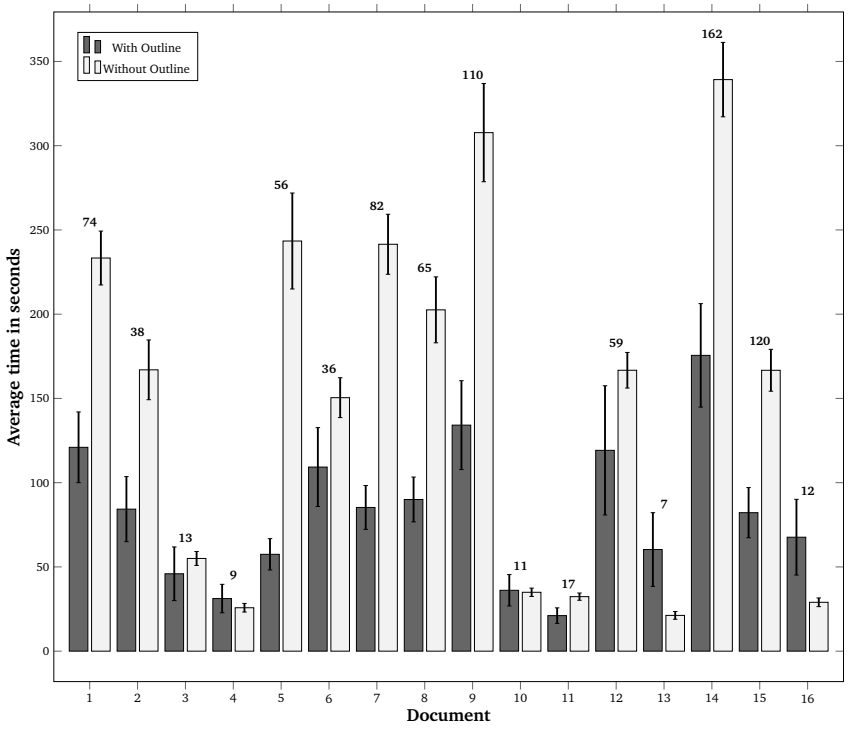


(b) Control group (r = 0.8836, p-value = 5.67^{-06}).

Figure 4.4: The scatter plots show the correlation between the answer-sentence index and the average time taken to find the answer tabulated in seconds. The black line represents the line of best fit.



(a) Geometric mean.



(b) Average mean.

Figure 4.5: The chart shows the average time for each document. The individual results are averaged over the total number of participants. The black number above each group of bars represent the index of the answer sentence.

Discussion

This section analyses the results of the experiment. No errors such as the occurrence of confounding factors were identified.

The results in Figure 4.3 clearly indicate that the event outline supports users in finding facts in documents. Moreover, participants using the outline were twice as fast than those without the respective feature. A comparison between the two test groups in the chart of Figure 4.5a reveals differences.

In general, the measured time for documents in which the relevant sentence occurred at a higher index is significantly lower for the treatment group than for the control group. This finding implies that the outline feature had a positive effect for these documents. That is, participants with the outline were significantly faster than participants not using it on documents, where the answer is placed at the end.

However, there seems to be almost no supporting effect for documents with small answer-sentence indices i.e. for documents where the answer can be found in the first few sentences of the document such as *document 13* or *document 16*. Although the answer to the question could be found in the upper part of the outline, users without the outline were faster.

In this cases, a weakness of the event outline becomes apparent. The question for *document 13* i.e. *Who received the Congressional Gold Medal?* serves for illustration purposes. Because the outline represents event arguments by the head of each constituent, the associated fact in the outline is extracted as *He [Graham] received honors including Medal* from the answer sentence. The participants on the other hand were focused on the whole phrase *Congressional Gold Medal* and therefore overlooked the correct fact in the outline. This took additional time and required the participants to recheck the document. Similar observations can be made for *document 16*. The issue could be resolved by incorporating additional dependents like compounds to each argument. The fact would then become *He [Graham] received honors including Congressional Gold Medal*.

In general, users tended to use the document as additional context to recheck their findings. This time is negligible for larger documents, but has a big impact on shorter ones.

The results further suggest an association between the average time and the sentence index of the answer for both groups. The scatter plots in Figure 4.4a and Figure 4.4b show the average time as a function of the answer-sentence index. Both plots give the evidence of a strong positive correlation. To support this finding the *Pearson correlation coefficient* denoted as r [Pearson, 1895] is calculated. This coefficient is a measure of the strength of a linear association between two variables and can take a range of values from $+1$ to -1. The stronger the correlation, the closer the Pearson coefficient will be to either $+1$ or -1 depending on whether the correlation is positive or negative. See Section A.3 of the Appendix for implementation details.

The value of r is 0.8376 with a p-value of 5.148^{-05} for the treatment group. This is a strong positive correlation, which means that high X variable scores go with high Y variable scores (and vice versa). The number also indicates that there is just a small variation around the line of best fit. The correlation is even stronger for the control group with $r = 0.8836$ and p-value $= 5.67^{-06}$.

The scatter plot for the control group contains one major outlier, which also varies from the line of best fit for the treatment group. This data point is associated with *document 15*, which is an article about the planet *Saturn*. The reason for the variation in both groups is attributed to the question that belongs to this document that is *What is Hyperion?*. This question differs from the other questions in that it gives direct evidence about the term to search for in the document. This

tempted users to scan the document only for this term without actually reading the document. As a result, the average time spent to find the answer to this question was less compared to the length of the document.

Conclusion

The FactBro platform was introduced as a tool for visualizing narrative chains and events for giving a broad and fast overview of large documents. For this purpose, an automatic alignment algorithm was proposed that always shows the current visible facts in a document outline. Finally, a large user study with 40 participants was conducted to show the utility of such a platform.

Overall, the experiment succeeded in showing that the outline feature with events supports users in finding facts in large documents. It enables users to scan the document for certain facts within a short amount of time. On average, users who had access to the outline feature were twice as fast than users without it.

The system also shows a significant influence for documents, where the answer to the question is placed at the end of the document. In this case, the outline system achieves an improvement by a factor of 3 regarding the time to find the fact. In contrast, almost no improvement was measured for documents in which the answer is found within the first few sentences.

User feedback after the evaluation was very positive. Different applications could benefit from the use of such a technology. For example, Wikipedia might use an additional event outline with the document as context. For this use case, the outline could be grouped into paragraphs leading to a faster overview of the individual sections of Wikipedia. Furthermore, the combination of such an event outline with a navigation structure could lead to a better overview for large documents.

Future work includes the incorporation of additional argument dependents, like compounds to the event representation, rather than just using the head of each phrase (e.g. *Congressional Gold Medal* instead of *Medal*). While this feature leads to more sparsity for the script induction systems, it helps to recognize multi-word expression in the outline. A second evaluation with these enhancements might show improvements for documents in which the answer can be found in the first few paragraphs. The visualizing of narrative events in their temporal order will be tackled in future work.

The platform is publicly available and open-source[34].

[34] The project page is available at `http://uli-fahrer.de/thesis/` (accessed August 2016).

5 Statistical Script Models

The script models presented in this chapter tackle the problem of script induction by learning narrative chains from text collections. Given a sequence of narrative events, the goal is to predict the most likely event that has happened at some point in the sequence.

Models for the prediction of narrative events are typically divided in two steps. In the first step, a text collection with document annotations is processed in order to find narrative chains in each document. In a second step, the narrative chains are used as training data to infer relationships between everyday life events in order to learn common-sense knowledge automatically.

The following sections describe each step in detail and discuss the various possible choices in each step.

5.1 Extracting Narrative Chains

For the experiments, different event representations are evaluated in order to learn common-sense knowledge. These representations were introduced in Chapter 3 and range from simple verb-dependency pair events to more complex multi-argument events.

According to previous work [Chambers and Jurafsky, 2008; Jans et al., 2012], the narrative events are identified with a coreference resolution system and a dependency parser. The Eventos framework uses both technologies as a basis and extracts narrative chains from documents in the required representation (see Section 3.2).

Although the dependency parser and coreference system make mistakes, the training includes all extracted chains. The closed-domain corpora used in this setup are rather small and experiments show that the models yield better results when being trained on the whole set of chains. However, for open-domain corpora it might be beneficial to select a subset of the extracted chains from each document. Jans et al. [2012] proposed and evaluated three different selection strategies:

(1) Select **all narrative chains** with two or more events linked by common protagonists.
   This strategy produces the largest amount of training data, but may contain noise. Especially short chains may contain preprocessing errors.

(2) Select only the **longest narrative chain** for each document.
   This selection strategy includes the chain from the key protagonist of the story. That chain may be of higher quality and is more likely to represent a real script. This strategy is especially useful when a lot of training data is available.

(3) Select all **narrative chains** consisting of five or more events.
   This strategy is the most balanced approach trying to combine the benefits of the previous two selection strategies.

After extracting the narrative chains, a subsequent step would be the temporal ordering of events since the occurrence of events in text does not necessarily represent the real-world order. However, this task goes beyond the scope of the presented thesis and is therefore considered as future work and discussed in Section 7.2.

5.2 Learning from Narrative Relations

This section provides an overview of the different script models explored in this thesis. The presented models can be grouped into language-model and vector-space-model-based approaches. The language-model-based approaches derive a probability distribution over events to estimate a *maximum likelihood estimate (MLE)* [Manning and Schütze, 1999, p. 197]. The vector-space-based approaches use vectors to represent events in a document.

Language Model based Script Induction

Single Protagonist Model

The single protagonist model is an adaption of the bigram model proposed in Jans et al. [2012]. Unlike the model presented by Chambers and Jurafsky [2008], this approach takes the ordering between events in a document into account.

The input of this model will be a narrative chain $c = c_1, ..., c_n$ and a position m that shows where the new event should be added in the sequence. The events can be either verb-dependency pair events such as $c_i = (v_i, d_i)$ with $d_i \in \{\text{subj}, \text{dobj}, \text{iobj}\}$, or multi-argument events such as $c_i = v_i{:}d_i(e_{subj:i}, e_{dobj:i}, e_{iobj:i})$. The model predicts the event $\hat{e}$ that maximizes the scoring function $S_b(e, c, m)$ with

$$S_b(e, c, m) = \underbrace{\sum_{k=1}^{m} \log P(e|c_k)}_{s_1} + \underbrace{\sum_{k=m+1}^{n} \log P(c_k|e)}_{s_2} \tag{5.1}$$

$$\hat{e} = \text{argmax}_{e \in V}\ S_b(e, c, m) \tag{5.2}$$

where n is the length of the partial chain c from which new events are inferred. The left sum s_1 describes the probability of an event being observed following all the events before it in the chain. Likewise, the right sum s_2 describes the probability being observed preceding all the events after it in the chain. Both sums contribute independently to the score $S_b(e, c, m)$.

The conditional probability $P(e_2|e_1)$ is the learned bigram probability of seeing e_2 after e_1, given the occurrence of e_1. It is estimated as:

$$P(e_2|e_1) = \frac{P(e_1, e_2)}{P(e_1)} \tag{5.3}$$

$$= \frac{C(e_1, e_2)}{\sum_{e'} C(e_1, e')} \tag{5.4}$$

where $C(e_1, e_2)$ is the bigram-count describing the number of times e_1 has been observed prior e_2 in a training corpus. Note that the collection of events is no longer an unordered set as in Chambers and Jurafsky [2008]. Therefore, $C(e_1, e_2) \neq C(e_2, e_1)$ applies to the bigram-count C.

The applied strategy to collect the bigram-count may vary. For regular bigrams, the occurrence of two adjacent events in a narrative chain is counted. For example, given the following narrative chain consisting of three verb-dependency pair events (`enter,subj`), (`bring,dobj`), (`order,subj`), the bigrams would be extracted as ((`enter,subj`), (`bring,dobj`)) and ((`bring,dobj`), (`order,subj`)). However, Jans et al. [2012] found out that event bigrams with one intervening event outperforms vanilla bigrams. This approach is called skip-ngram modeling [Guthrie et al., 2006] and reduces data sparsity since more observations are made. Although skip-bigrams can be used as a counting method, the bigram probability will still be derived as in the classical language model. The skip-bigrams are only utilized to increase the size of the training data. For instance, given the same narrative chain as mentioned above, the 1-skip-bigram counting strategy would generate ((`enter,subj`), (`order,subj`)) as additional event besides to the bigram events ((`enter,subj`), (`bring,dobj`)) and ((`bring,dobj`), (`order,subj`)).

Imagine the chain corresponds the actions performed when visiting a restaurant: *After entering the diner, the waitress takes the customer to the table and he orders a beer*. While the bigram strategy assumes that the customer is brought to the table after entering the restaurant, the skip-gram approach allows to skip this event and directly proceed with the ordering. This approach is most similar to the scripts as presented by Schank and Abelson [1977] that also allow to omit events in the sequence. The single protagonist model is referred to as the `Bigram` model in the following.

There are many unseen bigrams in the rather small corpus used in the evaluation. Thus, discounting is essential to achieve decent performance with the script models on such a dataset. In general, two variants for discounting exist: (1) *backoff* and (2) *interpolation*.

The backoff model estimates the conditional probability with a lower n-gram model if the higher n-gram is not present in the language model. Equation 5.5 shows the backoff approach for the bigram model as used in the script models.

$$P(e_2|e_1) = \begin{cases} P(e_2|e_1) & \text{if } C(e_1, e_2) > 0 \\ P(e_2) \cdot a & otherwise \end{cases} \tag{5.5}$$

If a certain bigram probability $P(e_1, e_2)$ is not present in the language model, the unigram model is used as a backoff. The backoff weight a preserves the overall probability distribution.

However, higher and lower order n-grams have different advantages and disadvantages. For instance, bigrams are sensitive to context, but have sparse counts. In contrast, unigrams consider no context, but have robust counts. This observation motivates the usage of interpolation, which combines the individual distributions as shown in Equation 5.6.

$$P(e_2|e_1) = \lambda \cdot P(e_2|e_1) + (1 - \lambda) \cdot P(e_2) \quad \text{with} \quad 0 \leq \lambda \leq 1 \tag{5.6}$$

Previous work used absolute discounting and a backoff approach for their models [Rudinger et al., 2015b; Ahrendt and Demberg, 2016]. This approach is based on the assumption that high counts are more reliable than low counts. Therefore, it subtracts some small but fixed amount d from all observed counts and redistributes it proportionally based on the observed events. The value of d lies between 0 and 1 and can be estimated via a maximum likelihood estimate and

leave-one-out testing [Manning and Schütze, 1999]. While absolute discounting provides accurate estimates for frequent n-grams, it is not suitable for small samples.

Empirically, interpolated algorithms tend to perform better than their backoff variants [Chen and Goodman, 1996]. To validate this finding for script models, an interpolated version of Ney's absolute discounting [Ney and Essen, 1991] is added to each language-model-based approach. Additionally, the Witten-Bell discounting [Witten and Bell, 2006] is evaluated in its backoff and interpolation version. This discounting algorithm is easy to compute and robust for small corpora [Chen and Goodman, 1996].

One of the most popular discounting methods is the modified Kneser-Ney discounting [Chen and Goodman, 1996]. It is an extension of the absolute discounting and proves to be effective for higher and lower order n-grams. This discounting method was originally proposed as a backoff algorithm by Kneser and Ney [1995] and both variants are included in the evaluation.

Unigram Model

The unigram model ranks events according to their unigram probability as calculated from a set of training documents. This model corresponds to a bag-of-word model, which assumes that events occur independently. It can be further trained with different event representations using bigram or skip-bigram counting. This model was first employed as a baseline by Pichotta and Mooney [2014]. They find that the `Unigram` model is essentially as good as the `Bigram` model. Rudinger et al. [2015b] confirmed this finding.

Weighted Single Protagonist Model

The weighted single protagonist model is similar to the `Bigram` model, but the distance between events in a document is taken into account. The approach is based on the assumption that the closeness of events describes some association between them. This kind of assumption has not been proposed for narrative events before. It is inspired by the underlying concept of word association measures, which assumes that the closeness of words in text is an indicator for some kind of relationship between them. In the same manner, statistical tests to measure the strength of word similarity assume that similarity between words follows through word co-occurrence.

The weighted single protagonist model maximizes the objective function $S_{wb}(e, c, m)$ as given in Equation 5.7.

$$S_{wb}(e, c, m) = \sum_{k=1}^{m} w(c_k) \cdot \log P(e|c_k) + \sum_{k=m+1}^{n} w(c_k) \cdot \log P(c_k|e) \tag{5.7}$$

$$w(e) = \frac{1}{d(e)} \tag{5.8}$$

Similar to the `Bigram` model, the events before and after the insertion point m contribute independently to the overall score. However, each probability derived from the context c_k is weighted with its inverse distance to the held-out event at position m. Equation 5.8 illustrates the weighting. Such a scoring has not been used before for narrative learning, but is a natural extension.

Figure 5.1 further exemplifies the weighting for a partial chain with seven events and tabulates the distance d for each event. Due to the equivalence of ranking in the evaluation, the sum of the individual weights does not necessarily need to add up to one. It holds

$$\hat{e} = \text{argmax}_{e \in V} \frac{1}{Z} \cdot S_{wb}(e, c, m) = \text{argmax}_{e \in V} S_{wb}(e, c, m) \tag{5.9}$$

where Z is a normalization factor. The model is referred to as `Weighted-Bigram` in the following.

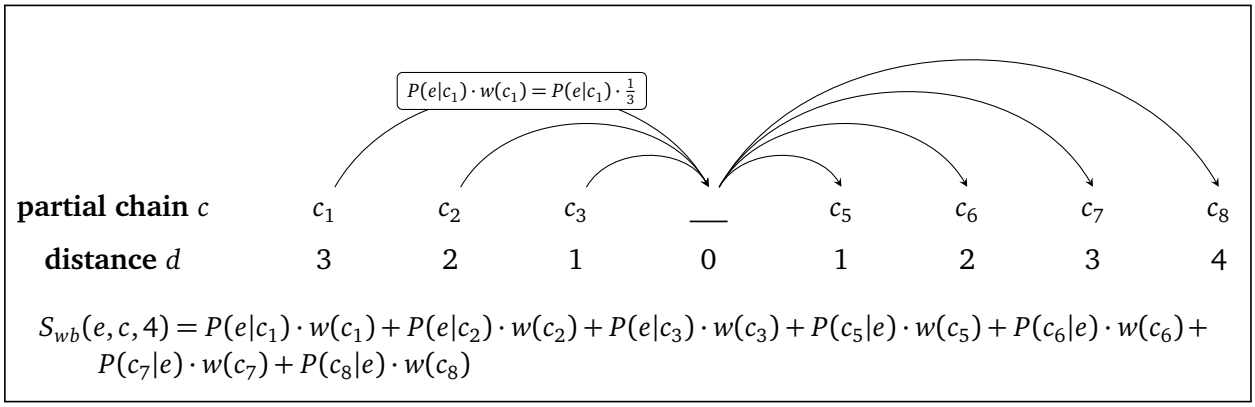


Figure 5.1: Illustration of the scoring approach for the `Weighted-Bigram` model.

Script Induction with Dense Vectors

Mikolov et al. [2013] describe a method to learn dense vector representations of words (word embeddings) from large corpora that capture relationships and similarities between words. Such representations seem to capture many different semantic and syntactic similarities. For instance, the vector operations $W(king) - W(man) + W(woman)$ yield a vector that is very close to the vector $W(queen)$. The example indicates that the similarity between *king* and *man* is similar to the similarity of *queen* and *woman*. Mikolov et al. [2013] made the implementation available as the word2vec framework[35].

Word2vec consumes a sequence of text with a sliding window of a fixed length. The algorithm starts with a random initialization and transforms the words in a vector of fixed length by minimizing the error using a linear combination of the context. Thereby, a vector representation is learned for each word using the vector representations of the surrounding words within the window.

The framework contains two different methods: (1) *continuous bag of words (CBOW)* and (2) *continuous skip-gram model*. Although the continuous skip-gram model is slower than CBOW, it performs better for infrequent words. A closer look at the differences between both algorithms explains this finding. CBOW predicts target words from their context words, whereas the skip-gram model implements the inverse and predicts context words from the target word. The skip-gram model therefore treats each pair of context and target word as a new observation and creates more training instances from limited amount of data as a consequence.

[35] Project page: `https://code.google.com/archive/p/word2vec/` (accessed June 2016).

Consider the sentence *the quick brown fox jumped over the lazy dog*[36]. The first step is to define the words and the context in which they appear. Here, context refers to the window of words to the left and to the right of a word. However, in general the context can be defined in any meaningful way. Using a window size of 1 yields the dataset ([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), ..., where the first entry of the tuple represents the context and the second entry the target word. The skip-gram model inverts these pairs and predicts each context word from its target word. For instance, it tries to predict *the* and *brown* from *quick*.

Granroth-Wilding and Clark [2016] trained a neural network to learn a non-linear composition of verbs and arguments into an event representation. The word vectors of their neural network are initialized with word embeddings generated from narrative chains.

Similarly, the embedding script model presented here learns word embeddings from narrative chains. To derive the embeddings with word2vec, each verb-dependency pair event is treated as a word and each narrative chain as a sentence. In this way, word2vec learns vector representations for verb-dependency pair events, so that those that appear in similar chains are close together.

The model is trained using the continuous skip-gram model and hierarchical softmax sampling that performs better for infrequent words. Equation 5.10 shows the scoring function.

$$S_w(e, c, m) = cosine\Big(\sum_{i=1}^{n} W(c_i), W(e)\Big) \tag{5.10}$$

The variable e represents the pair event candidate to be scored, c_i the given context of length n and W the vector associated with its given event. The candidate e is scored by the *cosine similarity* [Manning and Schütze, 1999, p. 541] of its verb-dependency vector $W(e)$ to the context vector. This context vector is derived as the sum of the vectors from W for each verb-dependency pair event of the partial chain c.

The model is referred to as `Word2Vec-Event` in the following.

[36] The example is taken from the TensorFlow manual hosted at `https://www.tensorflow.org/versions/r0.9/tutorials/word2vec/index.html` (accessed June 2016).

6 Evaluation

This chapter focuses on the evaluation of the systems described in the previous chapters. The performance of these systems is assessed with a comparative measure that is commonly applied to evaluate script models. The evaluation also includes a detailed analysis for the influence of several model parameters. In addition, a qualitative evaluation is conducted that seeks to explain what kind of knowledge is learned from a closed-domain corpus and further shows the common types of errors made by the systems.

6.1 Evaluation Task

The various script models presented in Section 5.2 are evaluated on the *narrative cloze test* [Chambers and Jurafsky, 2008]. This method is inspired by the classical psychological cloze test that is used to evaluate human for language proficiency [Taylor, 1953]. Given a sentence with a word missing, the test requires a subject to fill in the blank. For instance, the word *brings* is the correct solution for the gap in the cloze test: *The waitress* _____ *the beer to the table*.

Similarly, the narrative cloze test is a sequence of narrative events from which one event has been removed. Chambers and Jurafsky [2008] designed the test as a "comparative measure to evaluate narrative knowledge" and it is "not meant to be solvable by humans". Figure 6.1 shows an example of such a cloze test with a partial narrative chain associated with *hair* as protagonist.

The exact definition of the narrative cloze test depends on the event representation used in a script system as stated by Pichotta and Mooney [2016]. For example, Chambers and Jurafsky [2008], Jans et al. [2012] and Ahrendt and Demberg [2016] evaluate inference of held-out verb-dependency pair events. In contrast, Pichotta and Mooney [2016] and Pichotta and Mooney [2014] evaluate on the task of guessing a full multi-argument event, given all other events in a document.

The narrative cloze test used in this evaluation is as follows. Given a partial chain of events $e_1, ..., e_n$ from a document and the position m of some held-out event e_m, attempt to predict the missing pair event, given the other events in the chain. The model can either be trained on multi-argument events or verb-dependency pair events, but has to predict pair events in both cases. This definition is similar to the cloze test as used by Jans et al. [2012], where the position of the missing event is known.

The cloze tests are auto-generated from the dependency parses and coreference system. This is another major change with respect to the original narrative cloze test, where event chains are manually verified. But in contrast to Jans et al. [2012] and Pichotta and Mooney [2014], only script-relevant verbs with their arguments are used as held-out events. This constraint makes the cloze test results even more meaningful.

She led me to the sink and **washed** my **hair**, then brought me back to **comb it** out.
Once my **hair** was **dried** and **styled**, she **tidied it** up with a hair straightener.

Cloze test: (wash, dobj) (comb, dobj) (dry, subj) _____ (tidy, dobj)
Answer: (style, subj)

Figure 6.1: Example for a narrative cloze test as used in the evaluation for the models trained on verb-dependency pair events (Source: InScript corpus [Modi et al., 2016]).

6.2 Experimental Setup

The script models are trained on documents from the InScript corpus[37] [Modi et al., 2016]. InScript is a closed-domain corpus of 910 quality-checked stories containing on average 12 sentences each. The stories were collected via the Amazon Mechanical Turk platform[38], which allows to present an online task to humans. *Turkers* were asked to describe a scenario in form of a story "as if explaining it to a child" by using only a minimum of 150 words [Modi et al., 2016]. The corpus consists of ten different scenario categories with about 90 stories for each scenario type.

The InScript corpus contains additional text annotations like basic linguistic information such as tokenization, parts of speech and dependency labels for each sentence. Furthermore, all noun phrase heads in the corpus are annotated with a participant role and all verbs are annotated with an event label, which indicates whether this verb is script-relevant. Additionally, the text is annotated with coreference chains between noun phrases. However, for the purpose of this evaluation, narrative events and narrative chains are extracted automatically, instead of relying on gold annotated coreference chains and events.

The datasets are rather small i.e. between 87 and 133 stories. Thus, following Rudinger et al. [2015a], leave-one-out testing [Manning and Schütze, 1999] at document level is performed. For each fold of training, all narrative chains in the held-out test document are extracted. Every event from a narrative chain that has been annotated as script-relevant generates a new narrative cloze test. Narrative chains with less than two events are not considered for the cloze test generation.

The difficulty of the narrative cloze test varies depending on the context around the missing event. For some cases it might be likely to predict a single event. However, for most of the cases there is no confident single event that is likely to occur. Thus, the task is more about ranking the missing event high than predicting a single event. Pichotta and Mooney [2014] also argue that a script system is best evaluated by its top inferences. The performance of the script model is therefore evaluated with the following metric:

(1) **Recall at Rank**

Recall at rank (R@N) measures the fraction of cloze tests where the system predicts the missing event in the top N of its ranked list [Jans et al., 2012]. The value ranges from 0 to 1, where 1 indicates perfect system performance. It is defined in terms of a cloze collection C consisting of $|C|$ partial chains. For each partial chain c with missing event e, the function $rank(c)$ represents the rank of the event e in the guess list for the narrative chain c:

[37] Corpus is available for download at `http://www.sfb1102.uni-saarland.de/?page_id=2582` (accessed June 2016).
[38] Amazon Mechanical Turk Platform: `https://www.mturk.com/mturk/welcome` (accessed June 2016).

$$\frac{1}{|C|} \cdot \Big| \{c \mid c \in C \wedge rank(c) \leq N\} \Big| \tag{6.1}$$

Other metrics to evaluate script models like the *average rank* as proposed by Chambers and Jurafsky [2008] or the *mean average precision* [Manning and Schütze, 1999] are strongly influenced by miss-ranked events. Since these metrics were also not reported in prior work for the InScript corpus, their reporting does not aid in comparison. Recall at rank in contrast allows to assess the performance of the script induction systems by their best inferences.

Similar to Ahrendt and Demberg [2016], a document threshold $d = 5$ is applied for the ranking. That is, every event that occurs in less than d distinct documents during training will be ranked after every event whose count meets the threshold.

6.3 Results

Table 6.1 shows the average R@10 and R@1 measure for the different script models over all scenarios of the InScript corpus. The results are further divided into four categories, each representing another event representation.

Although the script models are trained on multi-argument events, the task requires to infer verb-dependency pair events. However, only the language-model-based approaches support multi-argument representations. Although, the outlook in Section 7.2 discusses the support of argument words in the vector representations to learn a representation of verbs and arguments together.

The scores are generated using the best performing settings for each system. For each language-model-based approach, the bigram-count C and the discounting algorithm is tabulated. In this context, the shortcut *WB* stands for Witten-Bell discounting, *AD* represents Ney's absolute discounting and *MKN* stands for the modified Kneser-Ney discounting. The postfixes *I* and *B* denote interpolated and backoff variants of the algorithms above respectively. The following describes the best performing setting for each model.

The `Weighted-Bigram` model uses bigram counts and Witten-Bell discounting. It is furthermore trained with a pseudo event <s> and </s>, which marks the beginning and the end of a narrative chain. This feature is referred to as *event marker (EM)* in the following. The rest of the setup is arranged as described in Section 5.2.

The `Bigram` model is configured as stated in Ahrendt and Demberg [2016] to allow for the comparability of the results. The model uses Ney's absolute discounting and skips up to the entire length of the chain. This means that the bigram count $C(e_1, e_2)$ is incremented if e_1 and e_2 occur anywhere within the same narrative chain. The results for this model in Table 6.1 correspond to the re-implementation of their system, whereas values marked with (*) correspond to the results reported by Ahrendt and Demberg [2016]. The variation is attributed to the different preprocessing steps and is further discussed in Section 6.4.

The best performing `Word2Vec-Event` model is using a skip-gram model with hierarchical softmax sampling, a window size of 1 and a vector size of 200. This configuration is able to capture the context of one narrative event around a target event. Implementation details and the overall setup are described in Section 5.2.

Finally, the Unigram model is used as competitive and informed baseline on this task. Table 6.1 contains the results for this model using bigram and skip-bigram counts for training.

Figure 6.2 shows the results for each of the scenarios from the InScript corpus separately. The results were generated with verb-dependency pair events for training and for held-out inference. The light gray and dark gray bars illustrate the R@10 and R@1 measure respectively. The description labels are as follows.

- **V1:** Weighted-Bigram model with event marker.
- **V1B:** Unigram model trained on bigrams.
- **V2:** Bigram model.
- **V2B:** Unigram model trained on all skip-bigrams.
- **V3:** Word2Vec-Event model.

Table 6.2 illustrates the results of the different discounting methods for the language-model-based approaches. Each discounting algorithm is evaluated in its backoff and interpolated variant. In order to make the effect of the discounting visible, the additional event markers are omitted for the Weighted-Bigram model. However, the modified Kneser-Ney discounting was not applicable to the skip-all counting variant. This happens when the count-of-count statistics of the training data is not suitable for KN discounting. In these cases, the cell in the table is left blank (-).

Table 6.3 tabulates the vector size as a function of the window size for the Word2Vec-Event model. Each entry contains the R@10 and R@1 measure for a fixed vector and window size.

	Model	Count C	Disc.	Average Recall at Rank	
				R@10	R@1
Pair events	Weighted-Bigram+EM	bigram	WB-I	**0.39**	**0.067**
	Weighted-Bigram	bigram	WB-I	0.35	0.063
	Unigram	bigram	-	0.27	0.029
	Bigram	skip all	AD-B	0.30/0.34*	0.056/0.018*
	Unigram	skip all	-	0.27/0.29*	0.051/0.040*
	Word2Vec-Event	-	-	0.33	0.079
Multi-argument	Weighted-Bigram+EM	bigram	WB-I	**0.33**	**0.060**
	Weighted-Bigram	bigram	WB-I	0.28	0.050
	Unigram	bigram	-	0.24	0.030
	Bigram	skip all	AD-B	0.25	0.042
	Unigram	skip all	-	0.24	0.049
Supersense	Weighted-Bigram+EM	bigram	WB-I	**0.35**	**0.061**
	Weighted-Bigram	bigram	WB-I	0.31	0.058
	Unigram	bigram	-	0.25	0.039
	Bigram	skip all	AD-B	0.27	0.050
	Unigram	skip all	-	0.25	0.054
Role labels	Weighted-Bigram+EM	bigram	WB-I	**0.35**	**0.067**
	Weighted-Bigram	bigram	WB-I	0.31	0.057
	Unigram	bigram	-	0.25	0.043
	Bigram	skip all	AD-B	0.28	0.048
	Unigram	skip all	-	0.24	0.050

* value as reported in [Ahrendt and Demberg, 2016]

Table 6.1: Best average performance of the different script models on the documents of the InScript corpus over all scenarios. The performance is measured as R@10 and R@1 and is generated using the best configuration for each system. The table contains the results for all event representation introduced in Section 5.1.

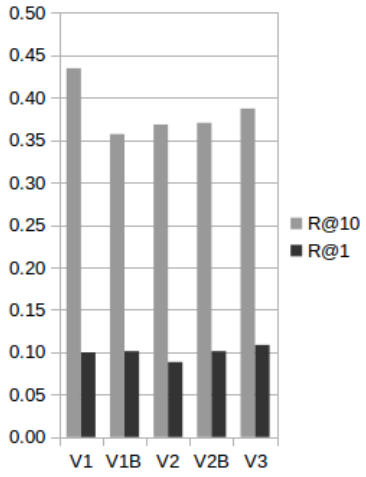
Model	Count C	Absolute Discounting (AD)				Witten-Bell Discounting (WB)			
		Backoff		Interpolate		Backoff		Interpolate	
		R@10	R@1	R@10	R@1	R@10	R@1	R@10	R@1
Weighted-Bigram	bigram	0.326	0.064	0.332	0.064	**0.350**	**0.063**	**0.350**	**0.063**
Bigram	skip all	0.303	0.056	0.309	0.057	0.306	0.055	0.314	0.056

Kneser-Ney Discounting (MKN)			
Backoff		Interpolate	
R@10	R@1	R@10	R@1
0.330	0.060	0.348	0.064
-	-	-	-

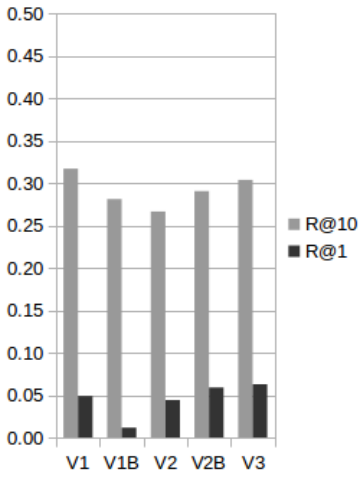
Table 6.2: Average performance of the language-model-based systems on the documents of the InScript corpus over all scenarios using different discounting methods.

Vector Size	Window Size									
	1		2		3		4		5	
	R@10	R@1	R@10	R@1	R@10	R@1	R@10	R@1	R@10	R@1
50	0.299	0.074	0.309	0.072	0.306	0.069	0.305	0.064	0.301	0.065
100	0.312	0.077	0.312	0.074	0.308	0.076	0.306	0.071	0.304	0.072
150	0.321	0.078	0.319	0.077	0.311	0.073	0.309	0.073	0.305	0.069
200	**0.329**	**0.079**	0.320	0.077	0.314	0.075	0.312	0.073	0.306	0.070
250	0.325	0.070	0.320	0.070	0.314	0.070	0.308	0.072	0.305	0.072

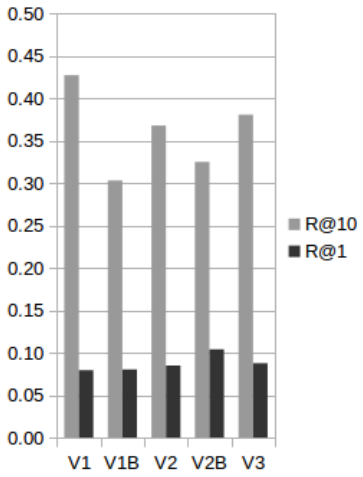
Table 6.3: Average performance of the `Word2Vec-Event` model on the documents of the InScript corpus over all scenarios tabulating the vector and window size.



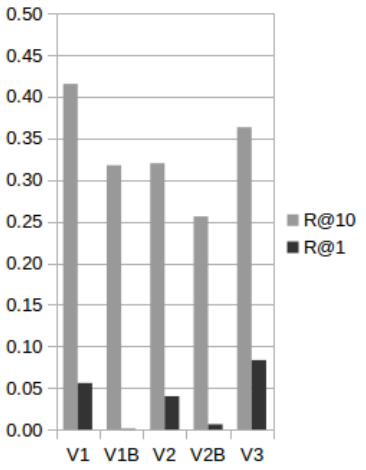
(a) Grocery



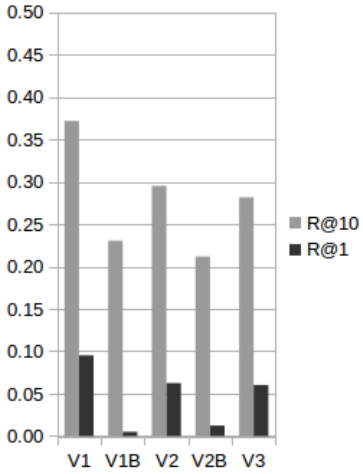
(b) Flight



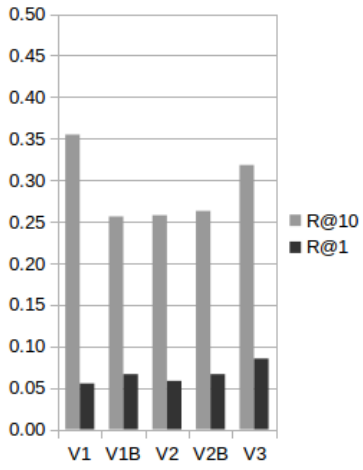
(c) Bus



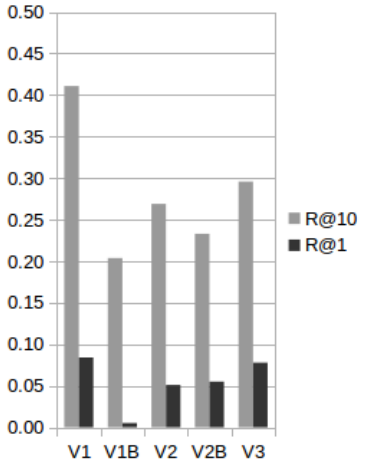
(d) Bicycle



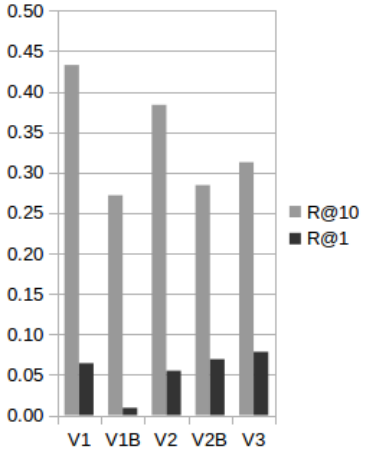
(e) Haircut



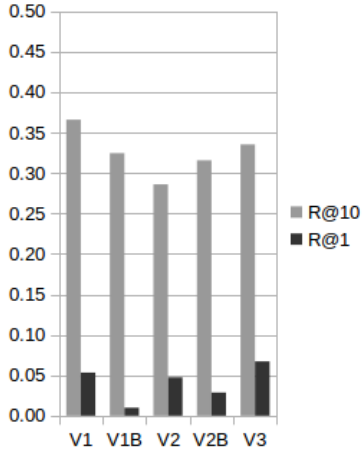
(f) Bath



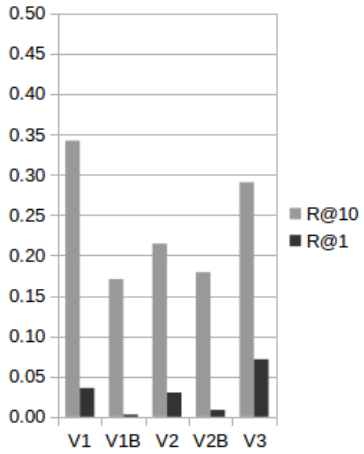
(g) Cake



(h) Library



(i) Train



(j) Tree

Figure 6.2: Best system comparison per category using verb-dependency pair events. `Weighted-Bigram` model + EM: V1, `Unigram` model trained on bigrams: V1B, `Bigram` model: V2, `Unigram` model trained on all skip-bigrams: V2B and `Word2Vec-Event` model: V3.

6.4 Discussion

The first part of the discussion deals with the analysis of the results for the verb-dependency pair event representation in Table 6.1 and Table 6.2. The second part then interprets the performance of the script models using multi-argument representations and concludes with a comparison between the usage of simple pair events and multi-argument events for event inference.

Verb-dependency pair events

Almost all systems in their best configuration perform above the associated unigram baseline on R@10 and R@1. The sole exception is the difference between the `Bigram` model and its corresponding baseline, which is not significant[39] at $p \leq 0.05$ on R@1 (t-value: 0.671 and p-value: 0.26). This observation stands in contrast to the finding of Pichotta and Mooney [2014], where the `Unigram` model is essentially as good as the `Bigram` model. However, Pichotta and Mooney trained on an open-domain corpus and considered every event as a test case for the cloze test. Thereby, the `Unigram` model is able to yield decent performance for test cases with common events such as `(have,subj)` or `(do,subj)`. In contrast, the closed-domain evaluation conducted here only considers script-relevant events for the cloze test. These test cases do not necessarily have to occur frequently in the corpus. Thus, the effect of the event frequency prior is reduced.

The best R@10 performance of 0.39 was achieved by the `Weighted-Bigram + EM` model, improving 12 points over the informed `Unigram` baseline. This model also achieves a 9% absolute improvement compared over the state-of-the-art `Bigram` model that reaches a R@10 score of 0.30. Overall, the model beats all other systems by a large margin on R@10.

[39] Unless otherwise stated: The dependent t-test for paired samples is used to determine if the two series of data are significantly different from each other.

The Weighted-Bigram model without the additional event marker reaches a R@10 value of 0.35 and therefore also outperforms the Bigram model on this measure. While the event marker feature gives 4 points performance improvement on R@10 for the Weighted-Bigram model, the result is not significant at $p \leq 0.05$ on R@1 (t-value: -1.144 and p-value: 0.141). Hence, both weighting-based models attain the same performance with 0.067 and 0.063 for R@1. This evidence suggests that the weighting has no influence on the top rank. However, both settings reach a R@1 score twice as high than the informed Unigram baseline with 0.029. Moreover, the improvement between the Weighted-Bigram model and the associated Unigram baseline of 8% is much larger than the 3% margin between the Bigram model and its baseline.

The improvement of the event marker feature reveals that narrative chains tend to have stereotypical events that start and end the sequence of common actions. This observation is consistent with the experience of common sense scenarios that have fixed entry conditions for the underlying situation. For example, the scenario of *riding a bus* tends to start with the action of *going to the bus stop* or *taking the bus*.

The values for the Bigram and Unigram models reported by Ahrendt and Demberg [2016] slightly differ from the results produced with the re-implementation of the same system. This variation is not surprising since the narrative chains for training and testing are different due to different preprocessing. Thereby, the number of cloze tests per category varies. For example, when the preprocessing miss to extract a script-relevant subject complement like *swim* in *he likes to swim*, the system is not tested on this event. In addition, small variations in the ranking functions could contribute to the difference as well.

In order to show the improvement of the weighting as an isolated feature, the Bigram model is trained with the interpolated Witten-Bell discounting method instead of applying an absolute discount. This allows a direct comparison of the Weighted-Bigram model using the same discounting. The counting strategy C remains different for both models, because the weighting feature is not appropriate for skip-bigram counting. Table 6.2 shows the results for the Bigram model trained with the Witten-Bell discounting. The model achieves an absolute improvement with a value of 0.31 on R@10 of 1% compared to the usage of Ney's absolute discounting algorithm. Thus, the improvement of the weighting feature can be identified as 4%. While the bigram counting strategy produces much less training data than the skip-gram modeling, the Weighted-Bigram model with vanilla bigrams performs better than the Bigram model using skip-bigrams.

Although the data is sparse, the Word2Vec-Event model outperforms the Bigram model with a R@10 value of 0.33 and a R@1 value of 0.079 as well. Both, R@10 and R@1 show an absolute improvement of 3% and 2% over the corresponding values for the Bigram model. Hence, all proposed models perform better than one of the state-of-the-art script induction systems using verb-dependency pair events for training. More importantly, the event embedding model also performs better than the Weighted-Bigram model on R@1. However, the difference between R@1 for this model and the Weighted-Bigram model using the event marker feature is not significant at $p \leq 0.05$ (t-value: 1.65 and p-value: 0.066). Incorporating event marker into the event embedding representation might increase the performance on R@10 as well and yield similar results as the Weighted-Bigram + EM model. The assessment of this assumption in a second evaluation is left for future work.

Overall, the model that performs best is the Weighted-Bigram + EM model, trained on all chains, using interpolated Witten-Bell discounting and bigram counting. This model also significantly outperforms the results of the Bigram model reported by Ahrendt and Demberg [2016].

As stated in Section 5.2, Chen and Goodman [1996] found out that interpolated discounting algorithms tend to perform better than their backoff variants for classical language models using probability distribution over sequences of words. This raises the question whether similar observations can be made for language models using event sequences. The results in 6.2 show that this finding only applies to the modified Kneser-Ney discounting on R@10 for the InScript corpus. There are no further statistically significant differences.

Besides, Witten-Bell and the modified Kneser-Ney discounting perform better than Ney's absolute discounting, which was commonly used in previous work [Ahrendt and Demberg, 2016; Rudinger et al., 2015a]. For instance, the `Weighted-Bigram` model with Witten-Bell discounting in its backoff variant achieves an absolute improvement of 3% on R@10 over the usage of Ney's absolute discounting. This improvement may be due to the fact that absolute discounting overly penalizes events that occur only once or twice, while Witten-Bell discounting is robust for small and noisy corpora. A possible solution to this problem would be to have separate discounting values for rare events.

However, there is no difference between Witten-Bell discounting and the modified Kneser-Ney discounting for the `Weighted-Bigram` model. This observation may be attributed to the rather small corpus used in the evaluation. This analysis clearly demonstrates that the Witten-Bell and modified Kneser-Ney discounting techniques are more suitable for small corpora than Ney's absolute discounting.

Table 6.3 reveals that target events for the `Word2Vec-Event` model are best described with their direct neighbors i.e. a window size of 1 and a vector size of 200. This observation is in accordance with the improvement of the weighting feature using vanilla bigrams as a basis. The bigram counting strategy only considers adjacent event pairs for training and the weighting feature assigns the highest score to the direct neighbors of the held-out events and penalizes long-distance events. Thus, this approach is additionally based on the assumption that a target event is better described by a smaller context and results confirm this assumption.

Multi-argument events

In the evaluation of the multi-argument events with supersenses, the `Weighted-Bigram` model using the event marker feature performs better than all other script models with a R@10 of 0.35 and a R@1 of 0.061. This observation demonstrates that the event marker and the weighting feature are useful extensions for a richer semantic representation. Moreover, the supersense representation shows large improvements over the usage of raw multi-argument events for all script models, which is an indicator for its ability to generalize well over the dataset.

Both weighting-based script models beat the informed baseline by a large margin, whereas the difference between the `Bigram` model and its associated baseline is rather small. This shows that the `Unigram` model is still not able to exploit and benefit from frequently occurring multi-argument events. Instead, the performance drops, because the richer representation leads to more sparsity.

At a first glance, the participant label representation seems to generalize more than the supersense events, though the difference in R@10 and R@1 is marginal. It shows that there is no significant difference in the performance of the `Bigram` model and the `Unigram` baseline on R@10 compared to the multi-argument representation using supersenses. Finally, the `Weighted-Bigram + EM` model with a R@10 score of 0.35 and a R@1 score of 0.067 is the best performing system for the participant label representation.

In comparison with the supersense representation, the effort to generate participant label events is immense, while both representations perform equal good on the InScript dataset. Thus, there is little payoff regarding the required resources. In addition, domain-specific participant labels are required to replace arguments with their abstract role within the script. The participant label representation is therefore not applicable to open-domain corpora consisting of thousands of different scripts. The supersense approach in contrast functions in a totally unsupervised way that allows to acquire supersense events with minimal effort that yield decent performance.

Overall, the script models utilizing multi-argument representations perform worse than the models trained with simple pair events. Thus, the general assumption that richer semantic representations improve the inference of held-out events is not confirmed. However, this does not necessarily mean that such representations are not suitable for representing common-sense knowledge.

One issue for the multi-argument event representation may be the data sparsity problem. However, the problem may also be attributed to the training algorithm. The current approach calculates raw co-occurrence counts from multi-argument events resulting in poor generalization. Instead, the script models need to utilize the representation in a way that allows more generalization and flexible inference. As an improvement the relationships between entities in multi-argument events can be incorporated in the learning algorithm. According to Pichotta and Mooney [2014] such a relationship may be captured through the overlapping entities for two multi-argument events. Their results show slight improvement for the prediction of simple pair events using multi-argument events for training. A combination of their training algorithm with the supersense or participant label representation may further increase the improvement. This approach is described as future work in the outlook in Section 7.2.

6.5 Qualitative Evaluation

The goal of this qualitative evaluation is to provide a feeling for the knowledge that is learned from a closed-domain corpus with the `Weighted-Bigram` model and verb-dependency pair events as presented in Section 5.2. For this purpose, two example stories from different scenarios of the InScript corpus are selected. The script model is then used to induce a narrative chain that maximizes the probability of the events in the longest chain extracted from the example story. The event marker <s> is used as initial pseudo event. Subsequently, events are added to the new chain maximizing the probability of the sequence given the previous events as context. The example story is excluded from training. This approach enables to analyze whether the thereby induced subchain contains meaningful and script-relevant events, but also reveals whether the events match the overall context[40].

Figure 6.3 shows the selected stories for the qualitative evaluation. The events corresponding to the longest chain are marked in bold. The blue boxed words represent the best narrative chain that is automatically induced from the longest chain. For each boxed word, the prediction order is annotated as subscript number.

The first story is taken from the *haircut scenario* of the InScript corpus. The longest chain contains 12 events and corresponds to the mention *she* representing the hairdresser. Based on this chain, the highest probability narrative chain of length five is induced.

[40] The approach is similar to the *Shannon game*, which is about predicting the next letter in a sequence of letters.

The script model assigns the highest score to the *welcome* event (check in)[41]. This event is a reasonable component of the *haircut script* and is also annotated as script-relevant. In addition, the model succeeds in predicting this event first, given the pseudo starter event as context. The *check in* event is usually one the first events that occur when visiting a barbershop. The script model demonstrates to possess this common-sense knowledge too.

The second most likely event in the chain is *confirm* (talk haircut). This event is again script-relevant and its occurrence after the *check in* event is plausible, because the customer first needs to agree with the hairdresser about the type of service. The third event in the chain is *put* (put on cape), which is script-relevant and fits in the overall context.

Although the fourth and fifth event are script-relevant, their order does not match the real-world order. Humans would commonly agree that the hair is usually washed before it is conditioned. This observation is a common problem with script induction systems, which rely on the order of events in the document. However, it is not the order of the event occurrences in the text that matters. Script systems should take the real-world order into account. Otherwise, associations are learned that would never occur in that order in the real-word. For instance, an informed system should not predict the *live* event, given a partial chain that contains the *die* event.

This discussion also raises the question, whether the text order should be part of the cloze test. For the InScript corpus, the textual occurrence of events seems to match the real-world order in most of the cases. This is probably due to the *turkers* who wrote their stories in a linear fashion. Though, the gap between the real-world order and the order of the event occurrences in the text is much larger for corpora consisting of complex written stories. The problem of classifying the temporal relation between events is further discussed in the outlook in Section 7.2.

The next story that will be analyzed is taken from the *flight scenario*. The longest chain of this document is associated with the *we* mention and consists of eight events. The induced chain of length three is boxed and highlighted in blue. While the second event *booked* (get ticket) is script-relevant, the two other events such as *have* and *decided* are not.

Rudinger et al. [2015a] suggest that incorporating object information into the event representation could improve the performance of the model. With this information, a script model would be able to distinguish for example between the events *have tickets* and *have seat*. Note that in contrast to *have tickets* the event *have seat* is script-relevant for this scenario. However, without additional argument information, *have* will dominate due to its frequent nature in natural language.

[41] The annotated terms correspond to the event annotation labels used by the InScript corpus.

As I was getting ready for work this morning, I noticed in the mirror that my hair was getting a bit longer than I'd like. I resolved to go get a haircut after work. While I was at work, I researched salons. I decided on one while I was at lunch, and called to make an appointment for 5 pm. I got there about 15 minutes early, and informed the receptionist that I had an appointment. She ***welcomed***$_1$ *me and* ***asked*** *me to* ***have*** *a seat. I saw and perused a few magazines and hairstyle books while I waited, and actually found a style I liked more than what I had already had in mind. Once the stylist was done with her previous client, she* ***came*** *to* ***greet*** *me. She* ***asked*** *what I wanted and I presented the picture I found. She* ***said*** *that was no problem, and took me back to the styling chair. She* ***washed***$_5$ *and* ***conditioned***$_4$ *my hair and* ***put***$_3$ *a cape on me. She* ***confirmed***$_2$ *with me again what I wanted, and set off to cutting. I sat as tiny bits of hair fell all around me. Once she was done, she* ***blow*** *dried and styled my hair. It looked just like the picture. That's how I got a hair cut.*

Source: Haircut scenario 95 from Modi et al. [2016]

My wife decided she wanted to go to the beach last summer. We ***decided***$_1$ *the best way to get there would be an airplane. We* ***booked***$_2$ *the tickets 3 months in advance to ensure we would* ***have***$_3$ *tickets to board the plane to* ***take*** *us to our vacation destination. When the day finally came fly on the airplane, we* ***made*** *sure we* ***arrived*** *at least an hour and a half early. There are many delays before you actually get on board an airplane. First, you have to go through a metal detector and wait in long lines just to get to the metal detectors. Then you have to check your luggage. After that, you have to go to the terminal and provide proof of a ticket. Once you get on an airplane, you have to prepare for turbulence. Other than that, flying on an airplane is a safe and efficient way to travel. When we* ***landed****, we* ***went*** *straight to the beach.*

Source: Flight scenario 49 from Modi et al. [2016]

Figure 6.3: Example stories of the InScript corpus as used in the qualitative evaluation. The longest chain is marked in bold. The boxed words correspond to the best narrative chain of length n.

7 Conclusion and Future Work

7.1 Conclusion

This thesis tackled the problem of script induction by learning narrative chains from text collections as introduced by Chambers and Jurafsky [2008], but with a focus on event inference. In order to emphasize the contribution of this work, each chapter is briefly summarized.

Chapter 3 introduced Eventos as a flexible extraction framework for narrative chains. The unsupervised system allows to extract various event representation ranging from simple verb-dependency pair events to richer semantic representations. In this context, the supersense representation was proposed as an alternative to the participant label events that rely on abstract script roles. In contrast, supersense events are extracted in a pure unsupervised way that does not require explicit participant labels acquired from expert annotators. The application of supersenses as generalization method for narrative learning has not been used in previous work.

In Chapter 4, the value of narrative events in giving a broad and fast overview for large documents was shown. In a user study with 40 participants, users were asked to find facts to questions in different documents. Results show that users were twice as fast with an outline of narrative events extracted from the document than the control group without this feature. Especially for longer documents, participants utilizing the outline were about three times faster when the answer is placed near the end of the document.

Finally, in Chapter 5 and Chapter 6 different script models for learning common-sense knowledge were presented and evaluated on the narrative cloze test. The best performing system trained on verb-dependency pair events significantly outperforms prior script induction approaches using the same simple pair events for training. This script system utilizes a novel event weighting concept and applies a training algorithm, which is based on the assumption that narrative chains tend to start and end with stereotypical actions. The significant improvements further confirm this assumption and demonstrate that the theoretical concept of common entry and exit conditions to a script can be incorporated into an automatic script induction system. The evaluation also reveals that held-out events from documents are better described by a smaller surrounding event context.

Furthermore, the novel supersense event representation yields similar results to the expensive participant label representation and in contrast is applicable for open-domain corpora.

7.2 Future Work

This work addressed the problem of automatically learning knowledge of event sequences from text. Future research directions emerged during this study that are out of the scope of this thesis, but should be handled in future work. This research includes some higher level concepts related to the extraction of narrative chains, but also extensions to the existing approaches in order to improve the performance of the developed systems.

Ordering Narrative Events

The task of *ordering narrative events* describes the problem of temporally ordering sets of narrative events centered around a common protagonist. The script induction systems proposed here rely on the assumption that the textual order of events follows the temporal order. However, this condition is not always satisfied. For example, the second event in (1) and (2) precedes the first [Moens, 1987].

(1) Max fell. John pushed him.

(2) John went to visit Mary. He had bought her some flowers.

Several other relationships between two narrative events are possible, e.g. the *simultaneous relation*, which captures temporally overlapping events. In some cases, the order of narrative events does not matter, e.g. when adding of ingredients for baking. However, for script induction systems it is essential to learn relationships from narrative chains that reflect the real-world order rather than the occurrence of events in text. While the margin between the textual order and temporal order of events for close-domain corpora is rather small, it significantly differs for corpora like newspaper collections. But also for small closed-domain corpora the script models learn wrong associations as shown in the qualitative evaluation in Section 6.5.

An additional processing step preceding the script model training therefore is the learning of a partial temporal ordering of the extracted events. However, in the context of learning relationships from narrative chains only the *before* relation is relevant.

Chambers et al. [2007] present a machine-learning approach using a support vector machine (SVM) to extract the temporal relation between pairs of events. Future work includes the incorporation of such an approach to the event extraction pipeline in order to improve the performance of the script induction systems.

Context Expansion

According to Bruner [1986], humans use two different modes of thinking to interpret and understand the world around: (1) *narrative thinking* and (2) *paradigmatic thinking*. While narrative thinking describes the learning and using of relationships between different events, paradigmatic thinking is concerned with categorizing knowledge according to commonalities e.g. cats and dogs are both animals.

The learning of narrative chains already implements the first concept for script models. However, to date there has been no approach for using paradigmatic relations to improve the performance

of script models. As natural language is ambiguous, several sequences of events can describe the same situation. For example, the narrative chains *arrive, start, win* and *reach, begin, beat* both express the scenario of succeeding in a competition. While current approaches are able to learn the relations between the individual events, they cannot recognize that both chains describe the same situation. However, with access to knowledge about semantically related events, the model would be able to connect both chains to the same underlying narrative context. Paradigmatic thinking in contrast allows humans to detect these commonalities.

In this context, Biemann and Riedl [2013] proposed the metaphor of two-dimensional text, which represents language in two dimensions. The first dimension captures the syntagmatic relations between language elements like grammatical relations and the second dimension models paradigmatic relations, which describe elements that can be substituted with each other. This principle has its root in Saussure's structural linguistics hypothesis and is illustrated in Figure 7.1. Based on this idea, Biemann and Riedl introduce the *JoBimText framework*[42], which provides a software solution for automatic text expansion using contextualized distributional similarity.

The framework allows to expand the verb fragments for each event and provides a list of verbs occurring in similar context. The training algorithm of the script model can then use this information to connect the verb fragments with the expanded context. For each observed event pair (e_1, e_2), the expanded verbs of e_1 can be conditioned with e_2 and the expanded verbs of e_2 can be conditioned with e_1 as well.

Table 7.1 shows the top three expanded verb fragments for the aforementioned narrative chain using the English JoBimText model computed on a 100 million sentences corpus using Stanford dependency parser for context representation. For the first event *arrive*, the associated *reach* event is not among the first three elements. In contrast, the *begin* event is placed first for its corresponding *start* event. Likewise, the *beat* event is ranked third for the *win* event.

Using this approach the script model is able to condition the events in the second narrative chain, if it only observes the first chain during training. Thereby, it classifies the different appearing events with the same underlying situation. The same approach can be applied to the arguments of the raw multi-argument events in order to reduce data sparsity.

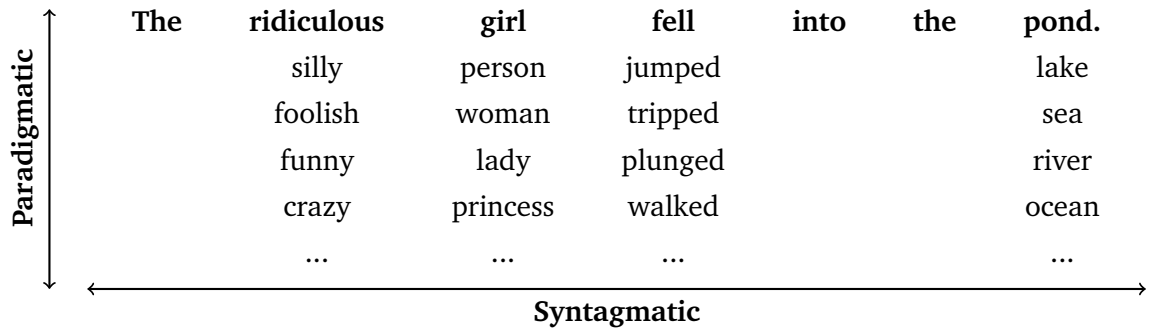


Figure 7.1: Illustration of the metaphor of two-dimensional text showing the difference between surface words and terms[43].

[42] Project page: `http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/` (accessed August 2016).
[43] Source: `https://courses.nus.edu.sg/course/elltankw/history/Vocab/B.htm` (accessed August 2016).

arrive#VB		start#VB		win#VB	
Jo	**Score**	**Jo**	**Score**	**Jo**	**Score**
travel#VB	119	begin#VB	509	clinch#VB	241
land#VB	99	continue#VB	246	lose#VB	220
depart#VB	97	want#VB	142	beat#VB	164

Table 7.1: Top three similar words for each event in the *competition chain*. The values were generated with the JoBimText framework using the English Stanford dependency parser model on a 100 million sentences corpus.

Learning Entity Substitution

The verb-dependency pair event representation is not capable to capture interactions between different entities. Consider the following example:

(1) Tom fights Jerry in the last match. He is regarded as the underdog of the tournament and knows Jerry very well.

Narrative chain for *Tom*: `(fight,subj)`, `(regard,subj)`, `(know,subj)`

Narrative chain for *Jerry*: `(fight,dobj)`, `(know,dobj)`

The pair event representation for both protagonists is given below the example. It fails in capturing the most important facts of the story such as *Tom and Jerry are fighting with each other* and that *Tom knows his opponent*. The events `(fight,subj)` and `(fight,dobj)`, as well as `(know,subj)` and `(know,dobj)` are totally unrelated to each other.

In contrast, richer semantic representations like the multi-argument representation capture interactions between different entities. The narrative chain for *Jerry* using multi-argument events becomes then: `fight:dobj(Tom,Jerry,•)`, `know:dobj(Tom,Jerry,•)`. However, modeling raw multi-argument events was not superior to simple pair events and even the more generalized representations such as supersense and participant label events could not achieve similar performance in the evaluation. The problem may be attributed to the gathering of raw co-occurrence statistics from multi-argument events. Given the two multi-arguments `ask:subj(Mary,Bob,•)` and `answer:subj(Bob,•,•)` the bigram statistics only considers the occurrence for the involved entities i.e. *Bob* and *Mary*.

Pichotta and Mooney [2014] in contrast proposed another training strategy that is based on entity substitution. It achieves improved performance on predicting verb-dependency pair events using multi-argument events for training. Their training algorithm is illustrated in Algorithm 1 that captures relationships in multi-argument events between entities by their overlapping entities. The algorithm gathers co-occurrence counts from narrative events and starts with an initialization step, where $N(e_1, e_2)$ describes the number of times e_1 has been observed prior to e_2 and evs represents the set of all multi-argument events. The function $coocurEvs$ returns all pairs of co-occurring events for a document d, whereas the set of all documents is given by $documents$. Thus, instead of training over sequences of events for each entity, the algorithm uses one event sequence per document.

Algorithm 1 Learning with entity substitution (Source: Pichotta and Mooney [2014]).

```
for e_1, e_2 ∈ evs do
    N(e_1, e_2) ← 0
end for
for D ∈ documents do
    for e_1, e_2 ∈ occurEvs(D) do
        for σ ∈ subs(e_1, e_2) do
            N(σ(e_1), σ(e_2)) ← N(σ(e_1), σ(e_2)) + 1
        end for
    end for
end for
```

The function *subs* consumes two multi-arguments e_1 and e_2 and returns all variable substitutions σ from entities involved in the events e_1 and e_2 to a set of four variables $\{x, y, z, o\}$. The variables x, y, and z represent arbitrary entities, whereas the fourth variable o models entities not shared between both events. The substitution follows the following rules:

(1) The substitution maps coreferent entities to the same variable in $\{x, y, z\}$.

(2) No distinct entity is mapped to the same variable in $\{x, y, z\}$.

(3) Entities not shared between both events are mapped to a dummy variable o.

(4) Arguments not filled with an entity are mapped to itself.

After gathering the substituted co-occurrence counts, the conditional probability can be derived as before. The adaption of this training algorithm is one of the major focuses for future work. By using this training algorithm, the supersense and participant label representation may achieve much better results and might also outperform simple pair events.

Script Model Extensions

The following presents a variety of script model improvements.

Mixture model

The language-model-based approaches already provide decent performance in the task of learning common-sense knowledge from text collections automatically. However, while the `Weighted-Bigram` model outperforms the traditional `Bigram` model using skip-gram modeling, it still lacks the ability to omit events in a sequence of events.

Assume the `Weighted-Bigram` model learns the following narrative chain from common actions performed by a waitress: *take order*, *bring order*, *do payment*. The script model would not be able to react to a new situation, where the customer carries the food to the table on his own as it is often the case in fast-food restaurants. In such a situation, the *bring order* event is missing and the chain for the waitress only consists of *take order* and *do payment*. However, the script model never conditioned the events together during training. Thus, the occurrence of both events receives a low score[44], although the scenario is quite common.

[44] In case the model uses a backoff approach, the pair receives the unigram probability normalized with some backoff constant.

In order to allow flexible inference, the script model needs to be able to skip events in the sequence. This could be achieved with the n-skip-bigram modeling approach, which also collects pairs of events that occur with n events intervening between them. For example, this approach would also learn a relationship between the *take order* and the *do payment* event. However, a pure skip-gram model proves not suitable for this task as shown in the evaluation. In contrast, the `Weighted-Bigram` model with simple bigram counting yields better results but is not able to skip events.

Therefore, a mixture model is proposed aiming to combine the strength of both models. This mixture model interpolates a pure bigram model with a skip-bigram model and represents the conditional probability $P(e_2|e_1)$ as

$$\lambda \cdot P_{bigram}(e_2|e_1) + (1-\lambda) \cdot P_{skip}(e_2|e_1) \qquad 0 \leq \lambda \leq 1 \tag{7.1}$$

, where P_{bigram} describes the conditional probability using bigram counting and P_{skip} models the learned bigram probability using skip-bigram counting. The parameter λ determines the influence of each model on the overall score. It can be tuned for each model and event representation over a development set. In order to find the best possible solution, λ can be estimated with values from 0 to 1 and a step size of 0.01.

In general, the mixture model allows the combination of various different language models. A possible extension is to mix a supersense or participant label event model with a script model trained on raw multi-argument events. Thereby, an event pair that also matches in its raw form could receive a higher score than those just matching the generalized representation.

However, the utility of such a mixture model has to be assessed in a new evaluation.

Multi-argument word2vec event model

The `Word2Vec-Event` model successfully learned embeddings from verb-dependency pair events extracted from closed-domain corpora and showed decent performance in the narrative cloze test. However, the model lacks the ability to learn embeddings from multi-argument event representations. Incorporating both, the verb and the argument fragments of a multi-argument event in the training context allows to learn a vector representation of verbs and arguments together. The following example illustrates the word2vec sentence representation based on multi-argument events:

(1) **Multi-argument representation:**
`get:subj(PRP,snack,•)`, `watch:subj(PRP,scenery,•)`, `enjoy:subj(PRP,•,•)`, `see:subj(PRP,place,•)`, `arrive:subj(PRP,hour,•)`, `pick:dobj(friend,PRP,•)`

(2) **Word2Vec sentence representation:**
`get:subj PRP:arg snack:arg watch:subj PRP:arg scenery:arg enjoy:subj PRP:arg see:subj PRP:arg place:arg arrive:subj PRP:arg hour:arg pick:dobj friend:arg PRP:arg`

Similar to Granroth-Wilding and Clark [2016], each verb and argument headword functions as both, context word and target word for the word2vec skip-gram model. The vector representation for a multi-argument event e is then defined by the sum of the argument vectors and the verb-dependency pair event vector as shown in Equation 7.3. The scoring function S_w is similar to the previous approach, but uses the enhanced vector representation:

$$S_w(e, c, m) = cosine\Big(\sum_{i=1}^{n} comb(c_i), comb(e)\Big) \tag{7.2}$$

$$comb(e = v{:}d(e_{subj}, e_{dobj}, e_{iobj})) = W(v{:}d) + W(e_{subj}) + W(e_{dobj}) + W(e_{iobj}) \tag{7.3}$$

Appendix

A User Study

A.1 Documents and Questions

Table A.1 lists all documents used in the user study. It includes the question and the answer for each document and also tabulates the sentence index of the answer. The entire document is accessible with the document name joined via underscores and the following link: `https://simple.wikipedia.org/wiki/#{underscore-document-name}#`.

#	Article Name	#Sent	#Answer	Question	Answer
1	The Lightning Thief	131	74	Where does Percy find the bolt?	backpack
2	Hanami	44	38	Where are the gift trees planted?	Washington
3	Violine	122	13	How old is the modern violin in years?	400
4	Ana Ivanovi	64	9	Where does Ana Ivanovi come from?	Serbian
5	American Airlines Flight 11	78	56	When did CNN interrupted their commercial?	08:49
6	Anna Kournikova	63	36	Who gave Anna K. the best double pair award?	WTA
7	Jessica Alba	87	82	Which political party does Jessica A. belong to?	Democratic
8	Powderfinger	88	65	When did Bishop leave the band?	1992
9	Red Hot Chili Peppers	162	110	Who rejoined the band?	Frusciante
10	Bobby Robson	99	11	What made Robson die?	Cancer
11	Bloc Party	110	17	Who plays the drums?	Tong
12	Jupiter	130	59	When did NASA's Voyager 1 probe went to Jupiter?	1979
13	Billy Graham	145	7	Who received the Congressional Gold Medal?	Graham
14	City of Manchester Stadium	171	162	Who performed the first concert in the stadium?	RHCP
15	Saturn	153	120	What is Hyperion?	moon
16	Mourning Dove	112	12	What sound gives the doves its name?	woo-oo-oo-oo

Table A.1: Document references with question answer pairs.

A.2 Descriptive Statistics

#	Geometric Mean	95% Confidence Interval		Max. Time in s	Min. Time in s
		lower	upper		
1	113.33	96.21	133.49	231.43	52.17
2	76.39	62.91	92.77	192.05	37.30
3	37.49	28.45	49.39	136.26	12.23
4	26.36	20.1	34.55	79.02	8.59
5	54.05	45.96	63.57	90.49	25.39
6	98.60	80.08	121.41	220.76	35.71
7	81.46	71.19	93.21	153.25	41.55
8	86.46	75.52	98.3	160.20	46.92
9	119.01	92.46	153.18	247.36	18.01
10	31.69	25.19	39.87	94.89	10.26
11	19.27	15.97	23.24	43.87	10.34
12	96.96	73	128.77	300.10	39.53
13	47.32	34.86	64.24	200.51	20.46
14	164.97	141.15	192.81	297.43	6.27
15	75.49	61.57	92.57	165.55	16.18
16	53.67	39.61	72.71	186.54	18.50

Table A.2: Individual results of the user study for the treatment group.

#	Geometric Mean	95% Confidence Interval		Max. Time in s	Min. Time in s
		lower	upper		
1	231.16	217.98	245.14	322.93	190.55
2	163.04	148.18	179.39	266.36	103.53
3	54.32	50.59	58.33	71.56	34.22
4	25.25	23.14	27.55	38.63	17.03
5	235.83	210.56	264.13	369.90	113.25
6	148.17	136.76	160.53	217.97	77.89
7	238.43	221.93	256.15	304.83	152.28
8	198.96	183.53	215.69	298.75	151.10
9	302.49	279.47	327.39	462.15	227.17
10	34.57	32.39	36.90	46.82	25.14
11	32.06	30.25	33.99	44.85	25.48
12	165.31	156.18	174.97	220.14	132.95
13	20.72	18.82	22.8	31.59	13.15
14	336.14	316.80	356.65	440.35	258.18
15	164.85	154.55	175.84	246.21	131.95
16	28.49	26.21	30.96	40.03	19.55

Table A.3: Individual results of the user study for the control group.

A.3 Evaluation Metric Implementations

Calculation for the confidence interval of a geometric mean

```
# Calculates the confidence interval of a geometric mean
# with alpha = 0.05. Usage: v = c(1.5,2.5,7,4); gm(v)
# Arguments: x vector of numeric values.
gm <- function(x) {
  # Calculate the geometric mean
  gm1 = mean(log(x), na.rm = T)
  # Lower bound
  cil = exp(gm1 - (1.96 * (sd(log(x), na.rm = T) / sqrt(length(x)))))
  # Upper bound
  ciupp = exp(gm1 + (1.96 * (sd(log(x), na.rm = T) / sqrt(length(x)))))
  vec = c(round(cil, 2), round(ciupp, 2))
  return (vec)
}
```

Scatter plot calculation

```
# Vector with the sentence index of the answers
x = c(74,38,13,9,56,36,82,65,110,11,17,59,7,162,120,12)
# Times with the outline feature
y1 = c(
    113.3289563756,76.3933290845,37.4900478064,26.3572783024,54.0495305509,
    98.6004207197,81.4575916051,86.1606510193,119.0099113883,31.6932739519,
    19.2650299498,96.9569250775,47.3225145322,164.9691917148,75.4943081358,
    53.6684531426)
# Times without the outline feature
y2 = c(
  231.160098354,163.0374201949,54.3232420166,25.2490176775,235.8292109,
  148.165916911,238.4254213069,198.9608480099,302.4854697313,34.5689433476,
  32.0623210613,165.3105315447,20.7159459428,336.1384676832,164.8477554888,
  28.4909123659)
# Run tests
cor.test(x,y1, method = "pearson")
cor.test(x,y2, method = "pearson")
# Plot scatter plot for treatment group
plot(x,y1, xlab = "Sentence number of answer",
  ylab = "Average time in seconds", ylim=c(0,400))
abline(lm(y1 ~ x))
# Plot scatter plot for control group
plot(x,y2, xlab = "Sentence number of answer",
  ylab = "Average time in seconds", ylim=c(0,400))
abline(lm(y2 ~ x))
```

Bibliography

S. Ahrendt and V. Demberg. Improving event prediction by representing script participants. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 546–551, San Diego, California, June 2016. Association for Computational Linguistics.

R. Akerkar. *Introduction to Artificial Intelligence*. New Delhi: Prenctice- Hall of India Private Limited, 2005.

I. Aldabe, M. L. de Lacalle, M. Maritxalar, E. Martinez, and L. Uria. Arikiturri: An automatic question generator based on corpora and NLP techniques. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, ITS '06, pages 584–594, Jhongli, Taiwan, June 2006. Springer-Verlag.

P. M. Andersen, P. J. Hayes, A. K. Huettner, L. M. Schmandt, I. B. Nirenburg, and S. P. Weinstein. Automatic extraction of facts from press releases to generate news stories. In *Proceedings of the 3rd Conference on Applied Natural Language Processing*, ANLC '92, pages 170–177, Trento, Italy, 1992. Association for Computational Linguistics.

G. Angeli, M. J. Johnson Premkumar, and C. D. Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China, July 2015. Association for Computational Linguistics.

C. F. Baker, C. J. Fillmore, and J. B. Lowe. The Berkeley Framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, volume 1 of *ACL '98*, pages 86–90, Montreal, Quebec, Canada, 1998. Association for Computational Linguistics.

N. Balasubramanian, S. Soderland, Mausam, and O. Etzioni. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1721–1731, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

C. Biemann and M. Riedl. Text: Now in 2D! A framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1), Apr 2013.

G. H. Bower, J. B. Black, and T. J. Turner. Scripts in memory for text. *Cognitive Psychology*, 11: 177–220, 1979.

J. Bruner. *Actual Minds, Possible Worlds (The Jerusalem-Harvard Lectures)*. Harvard University Press, Cambridge, MA, October 1986.

N. Chambers. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

N. Chambers and D. Jurafsky. Unsupervised learning of narrative event chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, ACL '08, pages 789–797, Columbus, Ohio, USA, June 2008. Association for Computational Linguistics.

N. Chambers and D. Jurafsky. Unsupervised learning of narrative schemas and their participants. 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 2 of *ACL '09*, pages 602–610, Singapore, August 2009. Association for Computational Linguistics.

N. Chambers, S. Wang, and D. Jurafsky. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 173–176, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

D. Chen and C. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics.

S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

J. C. K. Cheung, H. Poon, and L. Vanderwende. Probabilistic frame induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 837–846, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.

K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th Annual Meeting on Association for Computational Linguistics*, ACL '89, pages 76–83, Vancouver, BC, Canada, 1989. Association for Computational Linguistics.

A. Clark, C. Fox, and S. Lappin, editors. *The Handbook of Computational Linguistics and Natural Language Processing*. Blackwell Handbooks in Linguistics. Wiley-Blackwell, 2010.

K. Clark and C. D. Manning. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1405–1415, Beijing, China, July 2015. Association for Computational Linguistics.

M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, volume 10 of *EMNLP '02*, pages 1–8, Philadelphia, PA, USA, July 2002. Association for Computational Linguistics.

B. Coppola, A. Gangemi, A. M. Gliozzo, D. Picca, and V. Presutti. Frame detection over the semantic web. In *The Semantic Web: Research and Applications, 6th European Semantic Web Conference*, pages 126–142, Heraklion, Crete, Greece, May 2009. Springer.

R. E. Cullingford. *Script Application: Computer Understanding of Newspaper Stories*. PhD thesis, Department of Computer Science, Yale University, New Haven, CT, 1978.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. Gate: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, ACL '02, Philadelphia, PA, USA, July 2002. Association for Computational Linguistics.

G. F. DeJong. An overview of the FRUMP system. In W. G. Lehnert and M. H. Ringle, editors, *Strategies for Natural Language Processing*, pages 149–176. Lawrence Erlbaum, Hillsdale, NJ, 1982.

L. Del Corro and R. Gemulla. ClausIE: Clause-based open information extraction. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 355–366, Rio de Janeiro, Brazil, 2013. ACM.

C. Fellbaum. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA, 1998.

D. Ferrucci and A. Lally. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

C. J. Fillmore. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech*, 280(1):20–32, 1976.

J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

T. Gamerschlag, D. Gerland, R. Osswald, and W. Petersen. *Frames and Concept Types: Applications in Language and Philosophy*. Studies in Linguistics and Philosophy. Springer International Publishing, 2013.

D. Graff, J. Kong, K. Chen, and K. Maeda. English Gigaword. Linguistic Data Consortium, Philadelphia.

M. Granroth-Wilding and S. Clark. What happens next? Event prediction using a compositional neural network model. In *Proceedings of the 30th Conference on Artificial Intelligence*, pages 2727–2733, Phoenix, Arizona, USA, Feb 2016. AAAI Press.

P. Gray and D. F. Bjorklund. *Psychology*. Worth Publishers, 2014.

G. Grefenstette and P. Tapanainen. What is a word, what is a sentence? Problems of tokenization. In *Proceedings of 3rd conference on Computational Lexicography and Text Research*, COMPLEX '94, pages 79–87, Budapest, 1994.

D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation*, LREC '06, pages 1–4, Genoa, Italy, 2006.

Z. S. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

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

D. C. Howell. *Statistical Methods for Psychology*. Wadsworth Cengage Learning, 2012.

B. Jans, S. Bethard, I. Vulić, and M.-F. 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*, pages 336–344, Avignon, France, April 2012. Association for Computational Linguistics.

M. John, S. Lohmann, S. Koch, M. Wörner, and T. Ertl. Visual analytics for narrative text - visualizing characters and their relationships as extracted from novels. In *Proceedings of the 7th International Conference on Information Visualization Theory and Applications*, volume 7, pages 29–40, Rome, Italy, February 2016. SciTePress.

S. Jänicke, G. Franzini, M. F. Cheema, and G. Scheuermann. Visual text analysis in digital humanities. *Computer Graphics Forum*, Jun 2016.

A. Kampmann, S. Thater, and M. Pinkal. A case-study of automatic participant labeling. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology*, pages 97–105, Duisburg-Essen, Germany, October 2015. German Society for Computational Linguistics and Language Technology.

D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *Proceedings of the Conference on Information Visualization*, IV '06, pages 9–16, Washington, DC, USA, 2006. IEEE Computer Society.

R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, ICASSP '95, pages 181–184, Detroit, Michigan, USA, May 1995.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

H. J. Levesque. The Winograd schema challenge. In *Logical Formalizations of Commonsense Reasoning*, pages 21–23. American Association for Artificial Intelligence, March 2011. AAAI Spring Symposium.

I. Mani. *Advances in Automatic Text Summarization*. MIT Press, Cambridge, MA, USA, 1999.

C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Human Language Technology Workshop*, HLT '94, pages 114–119, San Francisco, USA, 1994. Association for Computational Linguistics.

Mausam, M. Schmitz, R. Bart, S. Soderland, and O. Etzioni. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 523–534, Jeju Island, Korea, 2012. Association for Computational Linguistics.

W. S. McCulloch and W. Pitts. Neurocomputing: Foundations of research. chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press, Cambridge, MA, USA, 1988.

M. McTear. *The Articulate Computer*. Oxford: Blackwell, 1987.

T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. volume abs/1309.4168, 2013.

M. Minsky. A framework for representing knowledge. Technical report, Cambridge, MA, USA, 1974.

A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 641–648, Corvallis, Oregon, USA, June 2007. ACM.

A. Modi, T. Anikina, S. Ostermann, and M. Pinkal. Inscript: Narrative texts annotated with script information. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, LREC '16, Paris, France, May 2016. European Language Resources Association.

M. Moens. *Tense, Aspect and Temporal Reference*. PhD thesis, Centre for Cognitive Science, University of Edinburgh, 1987.

N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, and J. Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics.

H. Ney and U. Essen. On smoothing techniques for bigram-based natural language modelling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 825–828, Toronto, Canada, 1991. IEEE.

K.-H. Nguyen, X. Tannier, O. Ferret, and R. Besançon. Generative event schema induction with entity disambiguation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 188–197, Beijing, China, July 2015. Association for Computational Linguistics.

K. Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242, 1895.

S. Petrov, D. Das, and R. McDonald. A universal part-of-speech tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, LREC '12, Istanbul, Turkey, May 2012. European Language Resources Association.

K. Pichotta and R. Mooney. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '14, pages 220–229, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

K. Pichotta and R. J. Mooney. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, AAAI '16, pages 2800–2806, Phoenix, Arizona, February 2016. AAAI Press.

J. Pustejovsky, J. M. Castaño, R. Ingria, R. Saurí, R. Gaizauskas, A. Setzer, and G. Katz. TimeML: Robust specification of event and temporal expressions in text. In *Fifth International Workshop on Computational Semantics*, IWCS '05, Stanford, CA, USA, 2003. AAAI Press.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008.

N. Reiter, A. Frank, and O. Hellwig. An NLP-based cross-document approach to narrative structure discovery. *LLC*, 29(4):583–605, 2014.

R. Rudinger, V. Demberg, A. Modi, B. Van Durme, and M. Pinkal. Learning to predict script events from domain-specific text. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 205–210, Denver, Colorado, June 2015a. Association for Computational Linguistics.

R. Rudinger, P. Rastogi, F. Ferraro, and B. Van Durme. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP '15, pages 1681–1686, Lisbon, Portugal, September 2015b. Association for Computational Linguistics.

D. Rumelhart. *Notes on a schema for stories*, pages 211–236. Academic Press, Inc, 1975.

E. Ruppert, J. Klesy, M. Riedl, and C. Biemann. Rule-based dependency parse collapsing and propagation for German and English. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology*, pages 58–66, Duisburg-Essen, Germany, October 2015. German Society for Computational Linguistics and Language Technology.

V. Rus, Z. Cai, and A. C. Graesser. Experiments on generating questions about facts. In *Computational Linguistics and Intelligent Text Processing: 8th International Conference*, CICLing '07, pages 444–455, Mexico City, Mexico, February 2007.

D. Rusu, J. Hodson, and A. Kimball. Unsupervised techniques for extracting and clustering complex events in news. In *Proceedings of the 2nd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 26–34, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

R. C. Schank and R. P. Abelson. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. The Artificial intelligence series. Psychology Press, 1977.

N. Schneider and N. A. Smith. A corpus and model integrating multiword expressions and supersenses. In *The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL '15, pages 1537–1547, Denver, Colorado, USA, May 2015.

B. Settles. ABNER: An open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1 of *NAACL '03*, pages 134–141, Edmonton, Canada, May 2003. Association for Computational Linguistics.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, pages 3104–3112, Montreal, Quebec, Canada, December 2014.

W. Taylor. Cloze Procedure: A New Tool for Measuring Readability. *Journalism Quarterly*, 30: 415–433, 1953.

K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1 of *NAACL '03*, pages 173–180, Edmonton, Canada, May 2003. Association for Computational Linguistics.

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

I. H. Witten and T. C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Trans. Inf. Theor.*, 37(4):1085–1094, Sept. 2006.

S. M. Yimam, H. Ulrich, T. von Landesberger, M. Rosenbach, M. Regneri, A. Panchenko, U. F. Franziska Lehmann, C. Biemann, and K. Ballweg. new/s/leak information extraction and visualization for an investigative data journalists. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, Berlin, Germany, August 2016. Association for Computational Linguistics.

M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud '10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.