Script Acquisition: A Crowdsourcing and Text mining approach

A Dissertation submitted towards the degree

Doctor of Philosophy

in the Faculty of Philosophy

Computational Linguistics Department

of Saarland University

submitted by

Lilian Diana Awuor, Wanzare

from Homabay, Kenya

Saarbrücken, December 2019

Dean Faculty of Philosophy: Prof. Dr. Heinrich Schlange-Schöningen

First reviewer (Supervisor): Prof. Dr. Manfred Pinkal

Second reviewer: Prof. Dr. Alexander Koller

Third reviewer: Dr. Michael Roth

Date of final colloquium: 19th December 2019

Abstract

According to Grice's (1975) theory of pragmatics, people tend to omit basic information when participating in a conversation (or writing a narrative) under the assumption that left out details are already known or can be inferred from commonsense knowledge by the hearer (or reader). Writing and understanding of texts makes particular use of a specific kind of common-sense knowledge, referred to as *script knowledge*. Schank and Abelson (1977) proposed *Scripts* as a model of human knowledge represented in memory that stores the frequent habitual activities, called scenarios, (e.g. *eating in a fast food restaurant*, etc.), and the different courses of action in those routines.

This thesis addresses measures to provide a sound empirical basis for high-quality script models. We work on three key areas related to script modeling: *script knowledge acquisition, script induction* and *script identification in text*. We extend the existing repository of script knowledge bases in two different ways. First, we crowdsource a corpus of 40 scenarios with 100 event sequence descriptions (ESDs) each, thus going beyond the size of previous script collections. Second, the corpus is enriched with partial alignments of ESDs, done by human annotators. The crowdsourced partial alignments are used as prior knowledge to guide the semi-supervised script-induction algorithm proposed in this dissertation. We further present a semi-supervised clustering approach to induce script structure from crowdsourced descriptions of event sequences by grouping event descriptions into paraphrase sets and inducing their temporal order. The proposed semi-supervised clustering model better handles order variation in scripts and extends script representation formalism, Temporal Script graphs, by incorporating "arbitrary order" equivalence classes in order to allow for the flexible event order inherent in scripts.

In the third part of this dissertation, we introduce the task of scenario detection, in which we identify references to scripts in narrative texts. We curate a benchmark dataset of annotated narrative texts, with segments labeled according to the scripts they instantiate. The dataset is the first of its kind. The analysis of the annotation shows that one can identify scenario references in text with reasonable reliability. Subsequently, we proposes a benchmark model that automatically segments and identifies text fragments referring to given scenarios. The proposed model achieved promising results, and therefore opens up research on script parsing and wide coverage script acquisition.

Kurzzusammenfassung

Gemäß der Grice'schen (1975) Pragmatiktheorie neigen Menschen dazu, grundlegende Informationen auszulassen, wenn sie an einem Gespräch teilnehmen (oder eine Geschichte schreiben). Dies geschieht unter der Annahme, dass die ausgelassenen Details bereits bekannt sind, oder vom Hörer (oder Leser) aus Weltwissen erschlossen werden können. Besonders beim Schreiben und Verstehen von Text wird Verwendung einer spezifischen Art von solchem Weltwissen gemacht, welches auch Skriptwissen genannt wird. Schank und Abelson (1977) erdachten Skripte als ein Modell menschlichen Wissens, welches im menschlichen Gedächtnis gespeichert ist und häufige Alltags-Aktivitäten sowie deren typischen Ablauf beinhaltet. Solche Skript-Aktivitäten werden auch als Szenarios bezeichnet und umfassen zum Beispiel Im Restaurant Essen etc.

Diese Dissertation widmet sich der Bereitstellung einer soliden empirischen Grundlage zur Akquisition qualitativ hochwertigen Skriptwissens. Wir betrachten drei zentrale Aspekte im Bereich der Skriptmodellierung: Akquisition ition von Skriptwissen, Skript-Induktion und Skriptidentifizierung in Text. Wir erweitern das bereits bestehende Repertoire und Skript-Datensätzen in 2 Bereichen. Erstens benutzen wir Crowdsourcing zur Erstellung eines Korpus, das 40 Szenarien mit jeweils 100 Ereignissequenzbeschreibungen (Event Sequence Descriptions, ESDs) beinhaltet, und welches somit größer als bestehende Skript-Datensätze ist. Zweitens erweitern wir das Korpus mit partiellen ESD-Alignierungen, die von Hand annotiert werden. Die partiellen Alignierungen werden dann als Vorwissen für einen halbüberwachten Algorithmus zur Skriptinduktion benutzt, der im Rahmen dieser Dissertation vorgestellt wird. Wir präsentieren außerdem einen halbüberwachten Clusteringansatz zur Induktion von Skripten, basierend auf Ereignissequenzen, die via Crowdsourcing gesammelt wurden. Hierbei werden einzelne Ereignisbeschreibungen gruppiert, um Paraphrasenmengen und der deren temporale Ordnung abzuleiten. Der vorgestellte Clusteringalgorithmus ist im Stande, Variationen in der typischen Reihenfolge in Skripte besser abzubilden und erweitert damit einen Formalismus zur Skriptrepräsentation, temporale Skriptgraphen. Dies wird dadurch bewerkstelligt, dass Equivalenzklassen von Beschreibungen mit "arbiträrer Reihenfolge" genutzt werden, die es erlauben, eine flexible Ereignisordnung abzubilden, die inhärent bei Skripten vorhanden ist.

Im dritten Teil der vorliegenden Arbeit führen wir den Task der SzenarioIdentifikation ein,

also der automatischen Identifikation von Skriptreferenzen in narrativen Texten. Wir erstellen einen Benchmark-Datensatz mit annotierten narrativen Texten, in denen einzelne Segmente im Bezug auf das Skript, welches sie instantiieren, markiert wurden. Dieser Datensatz ist der erste seiner Art. Eine Analyse der Annotation zeigt, dass Referenzen zu Szenarien im Text mit annehmbarer Akkuratheit vorhergesagt werden können. Zusätzlich stellen wir ein Benchmark-Modell vor, welches Textfragmente automatisch erstellt und deren Szenario identifiziert. Das vorgestellte Modell erreicht erfolgversprechende Resultate und öffnet damit einen Forschungszweig im Bereich des Skript-Parsens und der Skript-Akquisition im großen Stil.

Ausführliche Zusammenfassung

Gemäß der Grice'schen (1975) Pragmatiktheorie neigen Menschen dazu, grundlegende Informationen auszulassen, wenn sie an einem Gespräch teilnehmen (oder eine Geschichte schreiben). Dies geschieht unter der Annahme, dass die ausgelassenen Details bereits bekannt sind, oder vom Hörer (oder Leser) aus Weltwissen erschlossen werden können. Das folgende Beispiel ist ein Fragment eines Blog-Textes über einen Restaurantbesuch.

(1) *(. . .) we drove to Sham Shui Po and looked for a place to eat. (. . .) One of the restaurants was fully seated [so we] chose another. We had 4 dishes—Cow tripe stir fried with shallots, ginger and chili. 1000-year-old-egg with watercress and omelet. Then another kind of tripe and egg—all crispy on the top and soft on the inside. Finally calamari stir fried with rock salt and chili. Washed down with beers and tea at the end. (. . .)*

Während der Text in Beispiel (1) offensichtlich über einen Restaurantbesuch spricht, werden viele Ereignisse, die zu einem Restaurantbesuch gehören, ausgelassen, wie z.B. einen Platz finden, hinsetzen, Essen bestellen. Das Gleiche gilt für Partizipanten, wie z.B. den Ober, die Karte, oder die Rechnung. Ein menschlicher Leser wird auf der Grundlage von Weltwissen annehmen, dass alle diese Elemente Teil des erwähnten Ereignisses sind, obwohl der Text diese implizit lässt.

Besonders beim Schreiben und Verstehen von Text wird Verwendung von einer spezifischen Art solchen Weltwissens gemacht, welches auch Skriptwissen genannt wird. Schank und Abelson erdachten Skripte als ein Modell menschlichen Wissens, welches im menschlichen Gedächtnis gespeichert ist und häufige Alltags-Aktivitäten sowie deren typischen Ablauf beinhaltet. Information dieser Art wird oft als gegeben gesehen und muss vom Leser aus dem Text inferiert werden. Skriptwissen über ein spezifisches Szenario ermöglicht es uns, solche nicht erwähnten Ereignisse, die entweder vor oder nach einem erwähnten Ereignis geschehen, und ebenso unerwähnte Partizipanten zu inferieren. Für Textverstehens-Systeme, die keinen Zugang zu entsprechendem Weltwissen haben, stellt diese Implizitheit eine nicht-triviale Hürde dar. Skriptwissen kann solchen Systemen dabei helfen, bessere Textrepräsentationen zu lernen, die für viele Sprachverstehens-Tasks benötigt werden.

Ein Skript ist eine Struktur, die verschiedene Arten von Informationen über ein bestimmtes Szenario beinhaltet, und setzt sich aus den folgenden Elementen zusammen:

- **Ereignisse (Events)**

  Die einzelnen Aktionen, die im Rahmen eines Skripts passieren können: Das Restaurant betreten, darauf warten, zum Platz gebracht zu werden, Essen bestellen, etc.

- **Partizipanten**

  Die Personen und Objekte, die in die Skriptereignisse involviert sind: Restaurant, Karte, Essen, Ober etc.

- **Beziehungen zwischen Ereignissen**

  Skript-Ereignisse besitzen eine natürliche partielle Ordnung: Man betritt das Restaurant, bevor man Essen bestellen kann. Einige Skript-Ereignisse schaffen durch ihre partielle Ordnung erst Voraussetzungen für das Stattfinden späterer Ereignisse: Der Ober muss das Essen zum Tisch bringen, bevor es gegessen werden kann; man muss zuerst auf die Karte schauen, bevor man von der Karte bestellen kann.

Welche Information über Skripte wird benötigt, um sie für das Sprachverstehen (Natural Language Understanding; NLU) benutzen zu können? Benötigt wird vor allem Information über die Ereignis- und Partizipantentypen und deren Beziehungen zueinander, d.h. die Rolle, die bestimmte Partizipanten in bestimmten Ereignissen spielen. Ereignis- und Partizipantentypen müssen mit der Information assoziiert sein, auf welche Art ein gegebenes Ereignis linguistisch realisiert werden kann. Die Sätze, die diese verschiedenen Arten der Realisierung repräsentieren, stellen eine Verbindung zur Ereignisrepräsentation dar. Als letztes wird auch temporale Information benötigt, die darstellt, in welcher Reihenfolge Ereignisse typischerweise im Skript auftauchen. Wir betrachten drei zentrale Aspekte im Bereich der Skriptmodellierung: *Akquisition von Skriptwissen, Skript-Induktion* und *Skriptidentifizierung in Text*.

Akquisition von Skriptwissen. Es existieren aktuell keine großen Skriptdatenbanken. Wir erweitern bestehende Datensätze auf zwei Arten: Erstens benutzen wir Crowdsourcing zur Erstellung eines Korpus, das 40 Szenarien mit jeweils 100 Ereignissequenzbeschreibungen (Event Sequence Descriptions, ESDs) beinhaltet, und welches somit größer als beste-

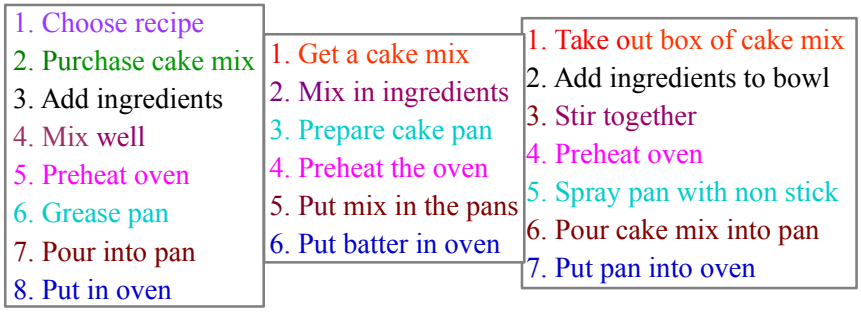


Figure 1: Beispiel-ESDs für das `Kuchenbacken`- Szenario.

hende Skript-Datensätze ist. Abbildung 1 zeigt drei Beispiel-ESDs für das `Kuchenbacken` - Szenario. Jede Ereignissequenzbeschreibung wurde von einem Arbeiter auf Amazon Mechanical Turk[1] verfasst. Zur Veranschaulichung werden Ereignisbeschreibungen, die das gleiche Ereignis realisieren, in einer Farbe gedruckt. Zweitens erweitern wir das Korpus mit partiellen ESD-Alignierungen, die von Hand annotiert werden. Die partiellen Alignierungen werden dann als Seed-Daten für einen halbüberwachten Algorithmus zur Skriptinduktion benutzt, der im Rahmen dieser Dissertation vorgestellt wird. Arbeitern wurde eine Source- und eine Target - ESD präsentiert, und sie wurden daraufhin gebeten, hervorgehobene Beschreibungen der Source - ESD mit denjenigen der Target - ESD zu verlinken, die semantisch ähnlich waren. Dabei konnten single-Target, d.h. *eins - zu - eins*, oder *multiple-Target-Annotationen*, d.h. eins - zu - viele, vorgenommen werden. Um die Anzahl an Seed-Daten zu minimieren, die für das halbüberwachte Skript-Induktions-Modell benötigt werden, benutzen wir verschiedene Kriterien, um die informativsten Ereignisbeschreibungen für das Alignment auszusuchen. Mit einem unüberwachten Clustering-Verfahren identifizieren wir automatisch solche Ereignisbeschreibungen, die sich nicht gut clustern lassen, sogenannte *Outlier*. Wir sammeln Alignments für diese Outlier per Crowdsourcing und integrieren diese als Instanzen in ein halbüberwachtes Modell zur Skriptinduktion.

Skriptinduktion. Im zweiten Teil dieser Dissertation betrachten wir das Problem der automatischen Induktion einer Ziel-Skript-Repräsentation von einem Korpus von ESDs. Wie vorher erwähnt, besteht ein Script aus Ereignissen, Partizipanten und den Beziehungen

[1] https://www.mturk.com/

zwischen Ereignissen. Als Konsequenz sollte die induzierte Skriptstruktur *Ereignisse, Partizipanten* und *temporale* Information zwischen Paaren von Ereignissen umfassen. Zusätzlich sollte Paraphraseninformation modelliert werden. Paraphrasen machen die verschiedenen linguistischen Variationen von Ereignissen und Partizipanten deutlich. Wir betrachten zwei Hauptaufgaben im Bereich der Skriptinduktion: Ereignis-Paraphrasierung und Ereignis-Ordnung. Wir präsentieren außerdem einen halbüberwachten Clusteringansatz zur Induktion von Skripten, basierend auf Ereignissequenzen, die via Crowdsourcing gesammelt wurden. Hierbei werden einzelne Ereignisbeschreibungen gruppiert, um Paraphrasenmengen (die Ereignistypen repräsentieren) und der deren temporale Ordnung abzuleiten. Semantisch ähnliche Ereignisse werden dem selben Ereignis-Cluster zugeordnet (ähnlich sind z.B. *"mix well", "mix in ingredients"* und *"stir together"* in Abbildung 1). Clustering-Algorithmen stellen im Gegensatz zu bisherigen Ansätzen eine weniger rigide Möglichkeit der Skriptinduktion dar, durch die Benutzung von semantischer und positionsabhängiger Ähnlichkeitsinformation. Durch das Hinzufügen von Positionsinformation als zusätzlichem Feature kann das Clustering auf Ordnungsinformation zurückgreifen, ohne dass es Einschränkungen im Bezug auf temporale Ordnung ausgesetzt ist. Wir benutzen die partiellen Alignments, die per Crowdsourcing gesammelt wurden, als Vorwissen, um den Clustering-Algorithmus anzuleiten. Wir zeigen, dass bereits die Benutzung einer kleinen Anzahl solcher Alignments die Cluster-Qualität im Vergleich zu State-of-the-Art-Modellen substantiell verbessert und eine geeignete Basis für die Induktion temporaler Ordnung darstellt.

Die prototypische Ordnung von Ereignissen in ESDs erlaubt es, die temporale Ordnung direkt aus den ESDs zu lernen. Skriptereignisse sind standardmäßig geordnet, aber diese Ordnung ist bis zu einem gewissen Grad flexibel. Beim Backen eines Kuchens zum Beispiel, kann man den *Ofen vorheizen* ("preheat oven"), bevor oder nachdem man *das Blech eingefettet hat* ("grease pan") (Abbildung 1). Nachdem Ereignisbeschreibungen eines Szenarios in Mengen gruppiert wurden, die die szenariospezifischen Ereignistypen abbilden, konstruieren wir temporale Skriptgraphen (TSG), indem wir die prototypische Abfolge von Ereignistypen bestimmen. TSGs ermöglichen es, von der feingranularen Repräsentation von Skripten in ESDs weg zu abstrahieren und eine globale Skriptrepräsentation herzuleiten. Die Knoten in einem TSG stehen für die Ereignistypen (Cluster) aus dem Clusteringschritt, und eine gerichtete Kante zwischen zwei Knoten zeigt die typische temporale Abfolge an. Die temporale Abfolge wird aus der partiellen Ordnung der Ereignistypen abgeleitet und

indiziert nicht unbedingt die unmittelbare Abfolge, sondern eher den Fall, dass der erste Ereignistyp typischerweise vor dem zweiten stattfindet. Eine Kante vom Cluster E zum Cluster E' zeigt also an, dass E typischerweise vor E' passiert, mit der Möglichkeit, dass dazwischen andere Ereignisse stattfinden. Der vorgestellte Clusteringalgorithmus ist im Stande, Variationen in der typischen Reihenfolge in Skripten besser abzubilden und erweitert damit einen Formalismus zur Skriptrepräsentation, temporale Skriptgraphen. Dies wird dadurch bewerkstelligt, dass Equivalenzklassen von Beschreibungen mit "arbiträrer Reihenfolge" genutzt werden, die es erlauben, eine flexible Ereignisordnung abzubilden, die inhärent bei Skripten vorhanden ist. Auf einem Paraphrasing-Task übertrifft unser Ansatz alle bisherigen Modelle, während er im Bereich der Vorhersage von temporaler Ordnung immer noch sehr gut funktioniert.

Ein Hauptproblem der Methode, die in dieser Dissertation vorgestellt wird, ist die Skalierbarkeit: Temporale Skriptgraphen werden basierend auf einzelnen Szenarien bottom-up gelernt. Sie repräsentieren nur Fragmente des umfassenden Skriptwissens, welches von Menschen in alltäglicher Kommunikation genutzt wird. Um dies zu quantifizieren, präsentieren wir eine Abdeckungs-Studie, um die Skalierbarkeit unserer Skript-Akquisition und der Induktion zu demonstrieren. Wir führen eine Studie auf den ROC-Stories (Mostafazadeh et al., 2016) durch, deren Resultate suggerieren, dass ein Skriptmodell, das mit unserer Methode gelernt wird, einen großen Anteil der Ereignisstrukturen abdecken kann, die in thematisch uneingeschränkten narrativen Texten vorkommen.

Skriptidentifizierung in Text. Menschen beziehen sich in einem Text normalerweise auf unterschiedliche Skript-Szenarien, und die vorkommenden Szenarien können in verschiedenen Teilen eines Textes auftauchen. Im dritten Teil dieser Dissertation führen wir die Aufgabe der Szenarioidentifikation in Texten ein, d.h. der Identifikation von Referenzen auf Skripte in narrativen Texten. Skripterkennung ist ein wichtiger erster Schritt für eine großflächige, Daten-getriebene Skriptinduktion für Aufgaben, die eine Anwendung von Skriptwissen erfordern. Um ein besseres Verständnis für die Probleme bei der Skripterkennung zu bekommen, erstellen wir einen Benchmark-Datensatz mit annotierten narrativen Texten, in denen einzelne Segmente im Bezug auf das Skript, welches sie instantiieren, markiert wurden. Dieser Datensatz ist der erste seiner Art. Als Ziellabels benutzen wir die Szenarien von allen öffentlich verfügbaren Szenariolisten. Unser Datensatz besteht aus

Texten des Spinn3r Personal Stories Korpus. Um sicherzustellen, dass unser Datensatz eine ausreichende Anzahl an relevanten Sätzen beinhaltet, d.h. Sätzen, die sich auf Szenarien in unserer Sammlung beziehen, wählen wir solche Texte aus, die eine hohe Affinität zu mindestens einem unserer Szenarien besitzen.

Ein Hauptproblem bei der Annotation von Segmenten mit Szenario-Labels ist die Frage, ab wann man entscheidet, dass ein Satz sich auf ein Szenario bezieht. Für den Task, den wir uns im Rahmen dieser Doktorarbeit ansehen, betrachten wir nur Segmente, die explizit bestimmte Aspekte des Skriptwissens realisieren, welche über Skript-evozierende Ausdrücke hinausgehen (d.h. mehr als ein Ereignis und Partizipant müssen explizit realisiert sein). Beispiel (2) unten zeigt ein Textsegment mit minimaler Szenarioinformation für das Einkaufen - Szenario, mit 2 erwähnten Ereignissen. In Beispiel (3) ist nur ein evozierender Ausdruck gegeben, weshalb das Beispiel nicht annotiert wird.

(2) ✓`going grocery shopping`

...We also **stopped at a small shop** near the hotel to **get some sandwiches** for dinner...

(3) ✗`paying for gas`

... A customer was heading for the store to **pay for gas** or whatever,...

Die Annotatoren errreichen akzeptable Übereinstimmung. Eine Analyse der Annotation zeigt, dass Szenario-Erwähnungen in Texten mit annehmbarer Zuverlässigkeit identifiziert werden können.

Als Letztes stellen wir ein Benchmark-Modell vor, welches Textfragmente automatisch erstellt und deren Szenario identifiziert. Wir präsentieren ein zweistufiges Verfahren, das etablierte Methoden der Topik-Segmentierung und Textklassifikation kombiniert. Für die Segmentierung machen wir die Annahme, dass eine Szenarioänderung durch eine Verschiebung der lexikalischen Kohäsion modelliert werden kann. Wir identifizieren Segmente, die zu spezifischen Skripts oder Szenarien gehören könnten, mit Hilfe einer Topiksegmentierung, unter der Annahme, dass Szenarien als Distributionen über Topics approximiert werden können. Nach der Segmentierung wird ein überwachter Klassifikator dafür genutzt, Szenario-Labels für jedes gefundene Segment vorherzusagen. Das vorgestellte Modell erreicht erfolgversprechende Resultate und öffnet damit einen Forschungszweig im Bereich des Skript-Parsens und der Skript-Akquisition im großen Stil.

To mum, dad, and hubby

for your unwavering love and support.

Acknowledgment

I look back with awe at the amount of support I received that made this project achievable. First, I would like to thank my lead supervisor, Prof. Manfred Pinkal for giving me the chance to join his project in SFB, together with Dr. Stefan Thäter. I am particularly thankful for the feedback, Manfred you always pushed me to do more and get out of my comfort zone. The journey was definitely manageable knowing that I could always count on you, Manfred, for constructive criticism and feedback. Michael Roth, your timely arrival in Saarbrücken was the much needed break that I needed in order to finish my thesis. Thank you for the ideas and feedback. I would also like to thank Alessandra Zarcone for constructive brainstorming sessions and paper writing.

In a special way, I would like to thank my colleagues with whom we shared office space. Simon Osterman and Ashutosh Modi, you were the best comrades one would ever ask for. Always ready to step in and provide intuitive feedback during impromptu brainstorming sessions. Simon, thank you for helping me out with German language and for translating the German parts of this thesis. I am grateful to Andrea Horbach and Annemarie Friedrich, who helped me overcome my fear of deep water and for giving me support whenever I needed. Andrea, thank you for accompanying me countless times to the Saarland foreign office. And Diana Steffen, your are special, thank you for not tiring in answering my questions and helping me out with German bureaucracy. I would also like to appreciate Ursula Kröner and Gabriele Reibold for their support.

This work would not have been complete without the help of all student assistants that I had a chance to interact with over the years: Leonie Harter, David Meier, Christine Schäfer, Georg Seiler, Dai Quoc Nguyen, Stefan Grünewald, Sophie Henning, Tatjana Anikina and Florian Pusse. I greatly value the effort you put in during the many annotation and programming tasks and for always speaking out your mind in order to improve the process.

What words can I use to thank the Studentenwerk im Saarland - Kindergarten community. You are an amazing team. I had so much peace of mind knowing that my boys are happy and taken care of with utmost tenderness.

I would like to thank the Saarbücken Adventist church for providing a community of friends that I could rely on, share and worship together. To my friends in Saarbrücken, thank you

for easening my stay. Dorothy and Milcah, you have been like sisters to me and I am grateful. To my friends in Kenya, you have been of great support.

I am thankful for having a strong family support system around me. To mum and dad, thank you for always believing that I could do anything that I set my mind to, and for your prayers. I thank my sister, Marrieanne, for checking on me and urging me to keep pushing on. To my brothers, Gordon and Jeff, my cousins Molline and Evans and the rest of the family - thank you for your unwavering love and support. I thank my in-laws for their support and prayers throughout this journey. To Rosy, Raziah, Evelyne and Leila, thank you for helping out and taking care of the house. To my husband, Alfred, your support is beyond what words can express. I am blessed to have you in my life. To my boys, Aldean and Nafrel, your presence in my life has seen me though.

This research was supported by the German Research Foundation (DFG) as part of SFB 1102 'Information Density and Linguistic Encoding' and the Cluster of Excellence on "Multimodal Computing and Interaction" (MMCI) of the German Research Foundation (DFG). I would like to thank the whole SFB fraternity for providing an environment where every member can contribute, collaborate and flourish.

Above all, I can do all things through Christ who is my strength.

Contents

Part I

Introduction and background

Chapter 1

Introduction

People communicate predominantly through languages, like English, German and Swahili. Language is one of the early things you learn in life. Language enables communication within and across cultures and helps in resolving misunderstandings. With language, you can express your thoughts, desires and queries. People talk to friends, colleagues, strangers, even to pets; people talk face-to-face or over the phone among others. Thus, we live in a world where communication in natural language is a central theme.

The ability to speak a language is made possible with the required knowledge of that particular language. Having mastery of a language entails more than just knowing the words and their meanings; more than knowing how to combine the words in order to transmit thoughts and ideas, but also entails the " ability to decode complex sets of concepts that are not necessarily literal or tangible" (Fromkin and Rodman, 1998). The "complex set of concepts" can be referred to as *common knowledge* about the world. Common knowledge about the world and the techniques for forming obvious reasoning from this knowledge are called *common sense* (Davis, 1990); examples are , the knowledge that "Lemons are sour"[1], or "If a father has a son, then the son is younger than the father and remains younger for his entire life."[2].

According to Grice's theory of pragmatics (Grice, 1975), when people communicate, they

[1] Wikipedia

[2] http://www.psych.utoronto.ca/users/reingold/courses/ai/commonsense.html

tend to leave out many basic information, with the assumption that the information is common knowledge and need not be explicitly mentioned. The left out information is expected to be known or can be inferred from commonsense knowledge. For instance, when you here conversation (1), it is assumed that the speakers share the context of `eating out in a cafeteria`: that one would need to *order the food*, and *pay for the food*, and so on; and that the second speakers mentioned having forgotten the wallet because he/she understands what is expected in `eating out in a cafeteria` context. Likewise, when you utter sentence (2), we understand that you were in a restaurant, and you *ordered the food*, the food was *brought* to you and you *ate it*; even without explicitly saying so. We also know that the food was tasty as it makes sense for the "it" in the sentence to refers to the food (soya gyros), rather than the ordering. Commonsense knowledge is the implicit, shared knowledge, that allows people to communicate effectively.

(1) Speaker 1: It's 12:30, are you joining for lunch?
Speaker 2: I forgot my wallet in the house.

(2) I ordered soya gyros. It was tasty!

The big question is, can computers also possess this shared knowledge? Computer systems are becoming more and more fused into everyday life of people. They have been integrated in day-to-day life for personal use, in businesses, by administrations , in entertainment, in schools and in hospitals. People are interested in machines helping them out in achieving their day-to-day goals. An interesting question for computational linguists and Natural Language Processing (NLP) community is: can machines also understand and talk natural language? A major research focus in NLP is on ways of building systems that are able to understand natural language. There are systems that are able to process text for a given task. For example email filters that filter out spam from nonspam messages, smart email categorizers which puts emails into relevant folders, spell checkers that correct orthographic errors in text, search engines for retrieving information based on a query written in natural language e.g. Google search [3], Bing[4], Yahoo search[5], e.t.c; translation tools for translating a source language to some other target language e.g. Google Translate[6], Bing Translator [7],

[3]https://www.google.com/
[4]http://www.bing.com/
[5]https://search.yahoo.com/
[6]https://translate.google.com/
[7]http://www.bing.com/translator/

Linguee [8] e.t.c.; speech-to-text tools that convert spoken language into text and the converse text-to-speech tools that convert text into spoken language - they are currently integrated in most smart mobile devices. There are also smart personal assistants that use speech-to-text technology enabling users to issue spoken instructions and questions in natural language. For example Alexa[9] from Amazon allows users to issue spoken requests to send a text message, play music, order items online and even answer questions like "Alexa, how do you make chocolate chip cookies?" e.t.c.; Siri[10] (just like Alexa) allows users to issue spoken requests to it, like "Remind me to feed the fish at 7 a.m.", or "Find a coffee shop nearby". Natural language understanding and production often requires common sense knowledge. Imagine you were able to have the conversation illustrated in example 3 with your phone.

(3) You: "I need to fly to Kenya."
Phone: "I recommend Ethiopian airlines."
You: "I do not like their meal!"

It is common knowledge that "meals are served on long distance flights", and you should not be surprised when "having a meal" is mentioned as part of `flying in an airplane`. Building systems with broad coverage commonsense knowledge is still an open and challenging task in NLP. There have already been attempts to bring domain specific commonsense knowledge to machines. Recently, Google launched its new smart assistant, Google Duplex, (Leviathan and Matias, 2018), capable of conducting natural language conversations aimed at completing specific closed-domain tasks like `scheduling a haircut appointment` or `calling a restaurant`. An example conversation by the Google Duplex illustrated how the system `schedules a haircut appointment`; by asking for available time slots, confirming and agreeing on the time of appointment. Apart from linguistic capabilities, the computer system needs commonsense knowledge about the typical aspects required in order to book an haircut appointment; for instance, information that a haircut appointment typically has a date and time, and the type of services offered in a hair salon, e.t.c. The Google Duplex system was trained on a large collection of anonymized phone conversations about the given tasks. Thus, in order to bring commonsense knowledge

[8] https://www.linguee.com/
[9] https://developer.amazon.com/alexa?cid=a
[10] https://www.apple.com/ios/siri/

to computers, reliable commonsense knowledge resources from where computer systems could learn are required.

As noted by Grice (1975), commonsense knowledge is typically not mentioned in text and the reader or listener is still able to understand the text as people share commonsense knowledge within and across different social settings. Text understanding is a non trivial task to machines, as machines do not have this shared commonsense knowledge. There are many linguistic resources describing facts about the world, for instance, information that James Madison was the forth president of United States of America, or that Berlin is the capital of Germany. Knowledge describing facts about the world are often written down or recorded in a medium that is easily sharable, like books, newspaper articles, scientific papers, Wikipedia, among others. Commonsense knowledge usually resides in the human mind, and is not always recorded in a shareable state. It is therefore difficult to find linguistic resources to extract commonsense knowledge, as commonsense knowledge is usually left implicit in text. There have been attempts to extract commonsense knowledge from text: Schubert (2002), Schubert and Tong (2003), Gordon et al. (2009), Gordon (2010), Tandon et al. (2014) (see Section 2.2 for details), but the results are still incomplete and unreliable. Havasi et al. (2007) also notes that "people tend not to provide information which is obvious or extraneous" and, therefore, "it is difficult to automatically extract common-sense statements from text, and the results tend to be unreliable".

(4) *(...) we drove to Sham Shui Po and looked for a place to eat. (...) One of the restaurants was fully seated [so we] chose another. We had 4 dishes—Cow tripe stir fried with shallots, ginger and chili. 1000-year-old-egg with watercress and omelet. Then another kind of tripe and egg—all crispy on the top and soft on the inside. Finally calamari stir fried with rock salt and chili. Washed down with beers and tea at the end. (...)*[11]

Consider the text in Example (4), showing a typical text from a blog that mentions `eating in a restaurant`. It's only written that they had 4 dishes, drunk beer and tea. Many things that are involved while `eating in a restaurant` are left out. Information about what took place before can be inferred even though they are not explicitly mentioned in the

[11] text extracted from Personal Stories Spinn3r corpus (Gordon et al., 2009)

passage, for instance, *finding a table*, *sitting down*, *ordering food* e.t.c. Likewise, information about what took place after can also be inferred, for instance, *paying for the food*, *leaving the restaurant* e.t.c. Many things that are involved in `eating in a restaurant` are not mentioned and so cannot be easily learned from the text. For machines, this inference is a non-trivial task as they lack the necessary commonsense knowledge and this knowledge is not easy to acquire. As previously illustrated in example (1) and (2), this kind of knowledge is necessary for effective communication. Bringing this kind of knowledge to machines if one of the central challenges in computational linguistics and text understanding.

1.1 Scripts

As illustrated by the blog text in Figure 4, the writer of the blog assumes that the reader can infer the events that took place in the restaurant, and only mentions relevant aspects of the activity. This shows that people posses commonsense knowledge of prototypical activities and can make inferences based on this knowledge. Commonsense knowledge about prototypical activities is referred to as **Script Knowledge**. Schank and Abelson (1977) proposed **Scripts** as a model of human knowledge represented in memory that stores frequent habitual routines, and the different courses of action in those routines, intended to achieve a particular goal (Barr and Feigenbaum, 1981). For example, `eating in a restaurant`, `going to the dentist`, `taking a bus` e.t.c. We refer to these type of activities as *Scenarios*. Figure 1.1 shows an example script for `eating in a restaurant` scenario. A **Script** is a structure showing various information about a given scenario and has the the following elements:

- **Events**

  These are the actions that would take place in a given script: *enter restaurant, wait to be seated, order food*, e.t.c.

- **Participants**

  These are the persons and objects involved in the script events: *restaurant, menu, food, waiter* etc.

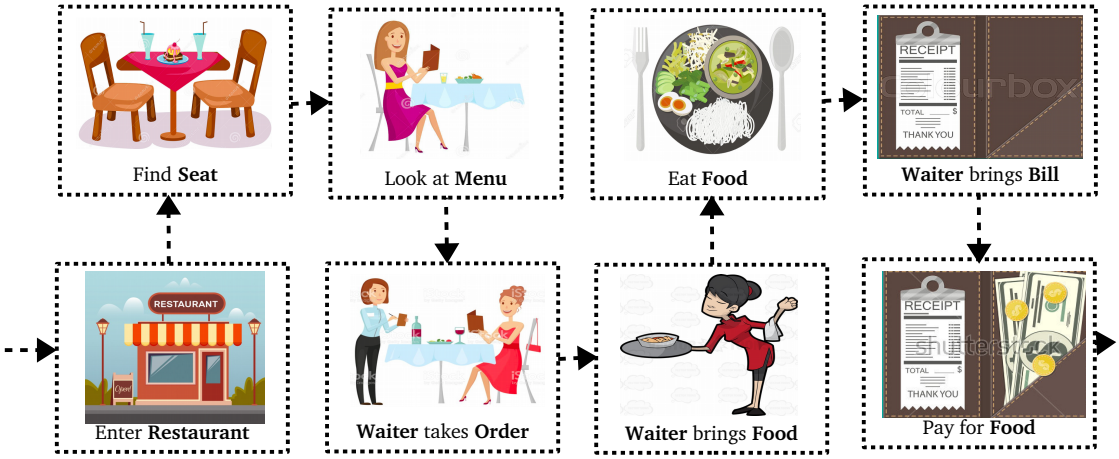


Figure 1.1: Example `eating in a restaurant` script showing internal script structure.

- **Relationships among events**

  Script events have typical partial ordering among them: you have to *enter the restaurant* and *sit down* before you can *order food*. Because of the partial ordering, some script events are enabling conditions for succeeding events: the *waiter* has to *bring the food* to your table before you can *eat it* or you have to *look at the menu* before you can *order from the menu*.

Thus, a script can be viewed as a set of events with partial ordering among them, each event involving one or more participant(s). Induction of script structure for various scenarios is one of the contributions of this thesis (see Section 4 for details).

Previous research have shown that people learn and use scripts from a very tender age, Searleman and Herrmann (1994), such that when exposed to a particular context, the relevant script is activated and used as a reference for interpreting and understanding the given context, Brown (1992), Bozinoff (1982).

Imagine you are shown Figure 1.2 and asked to tell a story about it. You could say: there is a man in a black half coat holding a piece of paper and standing next to a table; a lady and a gentleman are seated, each holding red books e.t.c. This description would be correct but not necessarily meaningful. But with relevant script knowledge, it is quite obvious that the image is about `eating in a restaurant` scenario; the man with a black half coat is probably the *waiter* taking their *order*, the gentle man seated is probably looking

[12] image from https://www.offset.com/photos/

Figure 1.2: Example `eating in a restaurant` scenario [12]

at the *menu*, as the lady makes her *order*. Dalli (1991) also stated that scripts are useful in helping one give an appropriate response in a given situation. Consider Figure 1.3, the appropriate answer to what one *orders in a restaurant* has to be limited to edible things. The inappropriate answer arises from the lack of relevant script knowledge on the scope of things i.e. participants, to be ordered in a restaurant. With the relevant script knowledge about `eating in a restaurant`, the girl would have known the scope of what can be ordered in a restaurant.

Figure 1.3: Example `eating in a restaurant` misunderstanding[13]

Challenges in script acquisition

Scripts have complex internal structures, thus posing various challenges when inducing the script structure from texts. In the first part of my thesis, we will look at the following challenges:

[13] image adapted from https://funnyjunk.com/channel/oc-comic-makers/Fancy+restaurant/aLyaLnY/

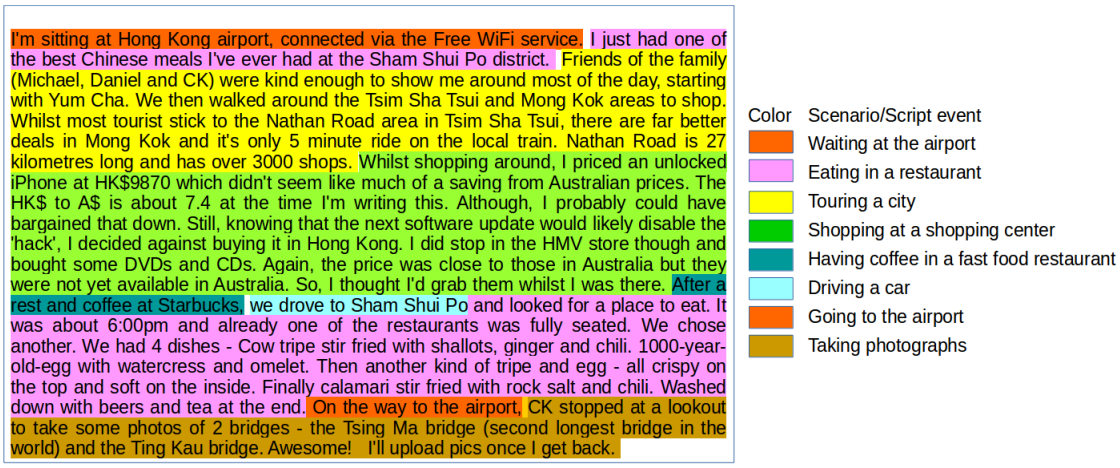


Figure 1.4: Example text[14] showing multi-script instantiation

- various linguistic realization of script events

  The same script event can be linguistically realized in different ways; some are semantically easier to identify e.g. *order food* and *place your order* in `eating in a restaurant` scenario are semantically similar, while others are similar only within the context of a given script and harder to identify e.g. *walk up the ramp* and *enter the plane* refer to the same event in `flying in an airplane` scenario. We refer to this similarity as *script-specific equivalence*.

- flexible order of execution of events

  Some scripts tend to have a stronger temporal order among events than others, for example to `have a hair cut`, *you book an appointment, go to the saloon, have your hair cut and washed* e.t.c, while in `cleaning up a flat`, the order of events might vary a lot, (*sweeping the floor*, *taking out garbage*, *vacuuming the carpet* can occur in various orders), leaving a lot of room for order variation in text.

- ordering of script events in text different from the prototypical order

  Script events are referred back and forth, thus, the order of the events in text is not necessarily the order in which the events in the given script occur. In Figure 1.4, *waiting at the airport* (in red) is mentioned before *going to the airport*.

- optional script events

[14] text extracted from Personal Stories Spinn3r corpus (Gordon et al., 2009)

EAT-AT-RESTAURANT Script

Props: (Restaurant, Money, Food, Menu, Tables, Chairs)
Roles: (Hungry-Persons, Wait-Persons, Chef-Persons)
Point-of-View: Hungry-Persons
Time-of-Occurrence: (Times-of-Operation of Restaurant)
Place-of-Occurrence: (Location of Restaurant)
Event-Sequence:
first: Enter-Restaurant Script
then: if (Wait-To-Be-Seated-Sign or Reservations)
then Get-Maitre-d's-Attention Script
then: Please-Be-Seated Script
then: Order-Food-Script
then: Eat-Food-Script unless (Long-Wait) when Exit-Restaurant-Angry Script
then: if (Food-Quality was better than Palatable)
then Compliments-To-The-Chef Script
then: Pay-For-It-Script
finally: Leave-Restaurant Script

Figure 1.5: Example showing script variants for `eating in a restaurant` scenario Schank (1999)

some script events could be optional, i.e. could be left out when executing a given script. For instance in Figure 1.5, having a *reservation* is an optional even when `eating in a restaurant`. To identify optionality is difficult as we have seen that events could be left out in text. The challenge is to identify which is which.

- different granularity of script events

  Different variants of a script may include different granularity of script events, for instance, while `checking in at the airport`, you can say that you *went through security check*, or you can say that you *removed your shoes*, *passed through the scanner* and *had someone look through your luggage* to mean the same thing.

Gordon and Van Durme (2013) also noted that the rate of occurrence of particular types of events in text may not represent the real-world frequencies of those events.

One major challenge of script acquisition from text is the issue of *implicitness*, i.e. many script events are not mentioned and are left implicit. In this dissertation, We look at narrative texts as a source of script knowledge. In narrative texts, it is common that people will write about more than one scenario. We look at the following challenges regarding identification of scripts in texts:

- multiple scripts being instantiated

Multiple scenarios could be instantiated in the same document. People are able to keep track of which events relate to which scenarios. Consider Figure 1.4, there are several scenarios mentioned in the text, and different portions of the same scenario mentioned in different parts of the text. For instance, the writer talks about having eaten the best Chinese food at the beginning of the text, goes on to mention other events and elaborates on what they ate towards the end of the text.

- closely related scripts

  One scripts can be closely related to another script i.e. the two would typically share some participants and events. For example, `eating in a restaurant` and `eating in a fast food restaurant` have similar event and participants, taking the subway and taking the train, e.t.c. Closely related scripts that share script events and participants can be interleaved in text making it difficult to separate which script event belongs to which scenario.

- overlapping scripts

  In Figure 1.4, while `touring the city`, the person is also `shopping` at the same time, thus the two scenarios are overlapping. Another example is when `planting a tree in the garden`, you are simultaneously `working in the garden`, though `working in the garden` could also involve other things like `weeding plants`.

- sub scripts

  Scripts can have sub-scripts. A sub-script is completed as part of a more general script. There are sub-scripts that always overlap with the more general script, for instance, `painting a room` is a sub-script of `renovating a room`. On the other hand `boiling milk` could be part of `making coffee`, but one can also `boil milk` for some other reason not as part of `making coffee`. In text, a script can occur as part of another unrelated script, for instance, you can `review a conference paper` while `eating in a restaurant`, the two not being related.

- where does a script begin and end?

  What triggers the start of a script, or a transition to another script? For instance in Figure 1.4, `driving a car` is mentioned, "we drove to Sham Shui Po". It could

be a script on its own, or an event, *going to the restaurant by car*, in `eating in a restaurant` scenario.

In order to have computer systems that have natural language understanding abilities, both linguistic capabilities, and commonsense knowledge, in particular Script knowledge, should be included.

1.2 Research questions

Scripts form a major part of the shared commonsense knowledge that is necessary for text understanding. We have already shown that interesting parts of scripts are usually left our in text (see Section 1 for details) as they can be inferred from commonsense knowledge. We address the following major research questions about scripts:

- How can we acquire script knowledge?

  On the one hand, the manual creation of wide-coverage script knowledge bases is infeasible, due to the size and complexity of relevant script knowledge. On the other hand, texts typically refer only to certain steps in a script and leave a large part of this knowledge implicit, relying on the reader's ability to infer the full script in detail. Crowdsourcing of linguistic descriptions of event patterns has been proposed in script acquisition; Singh et al. (2002), Regneri et al. (2010), Li et al. (2012). This thesis presents a large-scale crowdsourced collection and annotation of explicit linguistic descriptions of event patterns, to be used for the automatic acquisition of high-quality script knowledge (see Chapter 3 for details).

- How can we induce script-knowledge from the crowdsourced script corpora?

  The crowdsourced linguistic descriptions of event patterns cover various ways that a single event can be verbalized. Script induction answers the question of how the target script knowledge representation can be acquired from the event patterns. We propose a semi-supervised clustering technique that groups event descriptions representing the same event into event clusters using a small set of prior relational knowledge between event descriptions that should or should not belong to the same event

cluster (see Chapter 4 for details). The event sets provide a connection between the event representation and the linguistic realizations of events.

- What can be identified as a script in text?

  Texts typically refer to multiple scenarios and the referred scenarios can occur in different sections of the text. Thus, scenario identification is an important prerequisite for tasks that require text understanding. In order to answer the question of what can be identified as a script in text, we collect scenarios from all previous literature (see Chapter 2) and look at ways of identifying passages in texts where these scripts are instantiated. Gordon and Swanson (2009) identified and collected personal stories from the Internet that could be a potential resource for mining commonsense knowledge. We use these texts as a resource for extracting text segments instantiating scripts . We curate a benchmark dataset for the task of scenario identification and classification in text and propose a benchmark model that reveals some of the challenges in identifying script-references in text. (see Chapters 5 and 6 for details)

1.3 Thesis structure

This thesis looks at several aspects related to script acquisition and script learning.

Part I: Introduction and background

In the first part of this dissertation, we introduce the concept of scripts and provide motivation for the application of script knowledge in text understanding. In Chapter 2, review previous work on three aspects of scripts that should be considered in order to bring script knowledge to machines: *Script acquisition*: from where do we get script knowledge? *Script representation*: in what format can script knowledge be represented? *Script induction*: what methods can be used to automatically derive the target script representation from corpora?

1.3.1 Part II: Script acquisition and induction

The second part of this dissertation describes how we addressed the tasks of script acquisition, induction and representation. Chapter 3 describes our methodology for acquiring script knowledge via crowdsourcing. We present DeScript, a large-scale crowdsourced collection and annotation of explicit linguistic descriptions, Event sequence descriptions (ESDs), of script events, to be used for the automatic acquisition of high-quality script knowledge. We discuss the construction of our gold paraphrase sets and provide some challenging cases. We also describe how we identify event sequences that are particularly challenging and how we obtained crowdsourced alignments between event descriptions for use in semi-supervised script induction. Chapter 4 expounds our model for script induction from ESDs. We present a semi-supervised model that automatically groups event sequences expressing the same scenario into event clusters and induces the prototypical event order from the event clusters. Finally, we provide a brief study on coverage of script knowledge that can be obtained by the combination of existing script resources in terms of ESD collections. The study is based on the ROC-stories (Mostafazadeh et al., 2016), which is a corpus of topically unrestricted short narrative texts.

1.3.2 Part III: Scenario identification in text

The third part of this dissertation describes a novel task on identifying script references in narrative texts. In Chapter 5, we describe the task of scenario identification in texts and provide a motivation for the usefulness of the knowledge of which script is being referred for tasks that require text understanding. We give details of how we do the annotation, and the process of creating the gold segment labels. We analyze the annotated corpus and illustrate challenging cases that led to disagreements between annotators. In Chapter 6, we describe a benchmark model for the automatic segmentation and labeling of sentences according to the scenarios they reference.

1.3.3 Part IV: Thesis summary and future work

In Chapter 7, we provide a a summary of our work and give a brief overview of possible directions of future work around script-related applications.

Contributions of this thesis

The main contributions of this thesis are:

- A large-scale crowdsourced collection and annotation of explicit linguistic descriptions of event patterns, to be used for the automatic acquisition of high-quality script knowledge. Part of the results reported in Chapter 3 were published in Wanzare et al. (2016); joint work with Alessandra Zarcone, Stefan Thater and Manfred Pinkal.
- A state-of-the-art semi-supervised clustering algorithm for inducing script structure from texts describing particular scenarios; including a flexible graphical representation of scripts that allows equivalence classes to support order variation inherent in scripts. Part of the results reported in Chapters 3 and 4 were published in Wanzare et al. (2017); joint work with Alessandra Zarcone, Stefan Thater and Manfred Pinkal.
- A novel task of detecting and classifying segments from narrative texts about given scenarios, together with a first dataset of narrative texts which have annotations at sentence level according to the scripts they instantiate, as well as a benchmark model for text segmentation and scenario classification of sentences that achieves promising first results. Part of the results reported in Chapters 5 and 6 were published in Wanzare et al. (2019); joint work with Michael Roth and Manfred Pinkal.

Relevant publications. The below publications report part of the research conducted as part of this dissertation. I was the first author of these publications and was responsible for the core parts of the research (data collection, modeling, experiments and implementations).

Wanzare, L.D.A., Zarcone, A., Thater, S. & Pinkal, M. (2006). DeScript: A crowdsourced Corpus for the Acquisition of High-Quality Script Knowledge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia.

Wanzare, L.D.A., Zarcone, A., Thater, S. & Pinkal, M. (2017). Inducing Script Structure from Crowdsourced Event Descriptions via Semi-Supervised Clustering. In *Proceedings*

of the Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem 2017), Valencia, Spain.

Wanzare, L.D.A, Roth, M. & Pinkal, M. (2019). Detecting Everyday Scenarios in Narrative texts. In *Proceedings of the Second Workshop of Storytelling (StoryNLP@ACL 2019)*, Florence, Italy.

Chapter 2

Background

When carrying out everyday activities, having a conversation, watching movies or reading novels or newspapers, we make abundant use of *Script knowledge*. Schank and Abelson (1977) proposed *Scripts* as a model of human knowledge represented in memory that stores the frequent habitual routines (e.g. *baking a cake*, *eating in a fast food restaurant*, *feeding a dog*, etc.), and the different courses of action in those routines. Scripts represent knowledge of standardized event sequences describing typical everyday activities, such as *baking a cake* or *eating in a fast food restaurant*.

Scripts are a rich way of representing knowledge about everyday stereotypical activities. Script knowledge guides expectations in text understanding and makes missing events and referents in a discourse accessible. For example, if we hear someone say "I wanted to bake a cake on Sunday but I realized that the mixer is broken", we can understand the statement only with relevant script knowledge about `baking a cake`. Let us say we would like a personal assistant for indoor activities to understand the statement, then we would have to render information about the intermediate steps in `baking a cake` in order for the assistant to understand why the given information, *the mixer is broken*, is relevant in this context.

This chapter is structured as follows: Section 2.1 motivates the use of script knowledge in tasks requiring natural language understanding. I describe an overview and related work of acquisition of script knowledge in Section 2.2. conclude the chapter by highlighting existing script corpora from where script knowledge can be learned in Section 2.3.

2.1 Scripts and their applications

Script knowledge plays an important role for the computational modeling of cognitive abilities in particular for natural language processing, but making this kind of knowledge available for use in modeling is not easy. Script knowledge has the potential to support NLP tasks. The early script-based systems focused on showing how script knowledge is useful for story understanding and question answering. Next, we review a few representative examples of script-based system for story understanding.

Schank and Riesbeck (1981) present some miniature script-based systems. For example, Script Applier Mechanism (SAM, Cullingford (1977, 1981)), used script knowledge to understand simple invented stories and newspaper articles about pre-specified domains. SAM was a proof-of-concept system for the usefulness of script-knowledge for text understanding.

Mueller (1998) improved on SAM and developed the ThoughtTreasure platform. ThoughtTreasure contained (among other commonsense knowledge) knowledge of about 100 scripts, and included information about the events, roles, entry conditions, goals and emotions associated with the given script, duration and costs of the script. ThoughtTreasure attempted to understand stories by simulating the *states* and *events* described in the story. Mueller (1998) was able to model states and events in stories using two main agent categories, *planning* agents that indicate the steps to achieve a given goal and *understanding* agents which try to make sense of a given input text. For instance, when a sentence *He woke up* is input into the system, the event is *waking up*. Similarly, in the sentence *Jim poured shampoo on his hair*, the event is *pour shampoo*. Given the event *waking up*, the *sleep* understanding agent forwards the *sleep* panning agent to *awake* state. Likewise, given the event *pour shampoo*, the *shower* understanding agent is activated and in turn forwards the *shower* planning agent to *ready-to-lather* state. Script knowledge guides the understanding and planning agents by providing information on what are the typical events and temporal order of events in the given situations.

Mueller (2004) investigated the use of script knowledge in understanding texts involving various scenarios. Mueller (2004) built a script classifier that used script knowledge to identify which scripts were being referred to in the given text. Their also had a commonsense reasoning component that used script knowledge to make inferences and to fill out

missing details that were not mentioned in the input text but can be inferred given the script. A major downside was that their system was incapable of handling multiple scripts simultaneously.

Distributed SCript processing and Episodic memoRy Network (DISCERN, Miikkulainen (1993)) was a subsymbolic neural network that was trained on an artificially generated corpus based on three broad scenarios (restaurant visits, shopping and traveling). DISCERN used script knowledge learned during training to infer missing roles and events when reading script-based stories during testing. Similar to the system by Mueller (2004), DISCERN could not handle multiple scripts simultaneously. Such systems show the importance of script knowledge for text understanding.

Similar to story understanding and question answering, script knowledge has been applied to story telling (Schank., 1991) and Narrative Intelligence[1](Li, 2015). Script knowledge has also been applied to story generation. Liu and Singh (2002) built the MakeBelieve story generation system entirely based on the commonsense knowledge (that included Script knowledge) crowdsourced by Singh et al. (2002) (see Section 2.3).

Several robotic systems have applied script-like knowledge in improving robot understanding of instructions in natural language. Gupta and Kochenderfer (2004) described a crowdsourced collection of sequences of actions about various indoor activities (see Section 2.3) that represent scripts that could be carried out by an indoor robot. The crowdsourced script-like knowledge has been integrated into household robots to enable the robots infer proper decisions (Kunze et al., 2010) and to understand human instructions (Lu et al., 2016). Script knowledge has also been applied to anaphora resolution (Rahman and Ng, 2011), information extraction (Rau et al., 1989) and recognition of complex activities, e.g. *assembling furniture, food preparation, etc*, in videos (Rohrbach et al., 2012).

Recently, Ostermann et al. (2018a) released a novel dataset for machine comprehension to demonstrate the application of script knowledge in machine comprehension (Ostermann et al., 2018a) and question answering. The dataset included several questions that would potentially require script knowledge in order to correctly answer. For instance, given a text about `going to the sauna`, script knowledge can be useful in answering a question asking about *where did they sit?*, answer *on the bench*, event without *the bench* being

[1]Narrative Intelligence is the ability to explain narratives, comprehend and make inferences about narratives and produce responses about narratives

mentioned explicitly in the text. Research on script knowledge acquisition and processing is a vital part in developing next generation text understanding systems. This dissertation aims at building robust and scalable script knowledge acquisition and inductions techniques for natural language processing tasks.

2.2 Bringing Scripts to machines

If we had a repository of script knowledge, we first need to identify the script being talked about in a given text. It is quite common that a given text passage expresses more than one script at different parts of the text. The task of identifying the script being referred to could involve, first segmenting the text into parts expressing different scripts, then identifying the script being expressed in the various segments. Challenges arise when identifying scripts in texts, given the complex script structure. Scripts can have sub-scripts and also more than one script can be referred to simultaneously in a given piece of text, making it difficult to demarcate where the reference to a script begins or ends in text (see Section 1.1 for more details on challenges in script acquisition).

After we identify the script in text, we then locate the script elements (events and participants) mentioned in the text and map them to their respective script events and participants. The task of mapping events and objects in texts to their respective script events and participants is called *script parsing* (Ostermann et al., 2017). Figure 2.1 shows how script events can be aligned with narrative texts for script parsing.

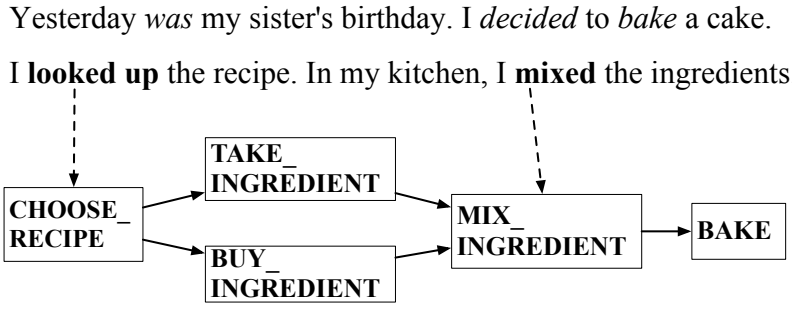


Figure 2.1: Aligning script-events with narrative texts (Ostermann et al., 2017)

What information about scripts do we precisely need in order to use scripts for NLU tasks? We need to have information about the event and participant types and the relation between

them i.e. the roles that the participants play in the given event. The event and participant types need to have paraphrase information describing the various ways the given event can be linguistically realized. Paraphrase information within the script context is wider than the standard notion of paraphrase. For instance, in `flying in an airplane` scenario, *board plane* and *walk up the ramp* refer to the same script event although they are in no was equivalent semantically. Consider also the event *open the door*. It could be a paraphrase to *open the microwave* in `heating food in a microwave scenario` or a paraphrase to *open the washing machine* in `doing laundry` scenario. This shows that scenario information is important as events and participants are located within a given scenario. Lastly, we need to have temporal information indicating the typical order in which the events in the script occur.

Currently, there is no large repository of script knowledge (see Section 2.3 for details on existing data). This dissertation will address the task of script knowledge acquisition (Chapter 3), script induction (Chapter 4) and script identification in text (Chapter 5). Various methods for script knowledge acquisition and induction have been proposed. In the following section, I review the relevant methods that have been proposed, focusing on the feasibility of the methods for acquisition of script knowledge.

2.2.1 Script Acquisition

The first natural step to bringing script knowledge to computers is building reliable linguistic knowledge bases about scripts. Script acquisition looks at ways of obtaining sufficiently large knowledge bases containing script knowledge from where machines can induce script structure.

I review the major methods of script knowledge acquisition:

Handcrafting script knowledge

The Early systems mentioned in Section 2.1, relied on handcrafted script knowledge: SAM system (Cullingford, 1981) used handcrafted script knowledge about `eating in a restaurant` scenario to understand stories about restaurant visits. ThoughtTreasure (Mueller, 1999) had a manually encoded knowledge base for about 100 scripts. The script knowledge included information about the events, participants (roles and physical objects), entry and post con-

ditions of the scripts, results and emotions after performing the script, among others. John (1992) manually encoded six scripts as tree structures and used the tree structures to generate an artificial corpus of script-based stories to train and test their story understanding system (Story Gestalt model).

Similar projects that hand-craft script-like knowledge include Gordon (2001) collection of common-sense activities. Gordon (2001) described representations for about 770 commonsense activities. Each activity representation contained information about the events, places, people and things involved in the given activity. The activity representations were meant for browsing image collections but could also be viewed as a resource for learning script knowledge. Figure 2.2 shows an example representation of `going to a restaurant` activity. We can see that several script events e.g. *finding a seat, ordering food*, etc., are not mentioned under the events in `going to a restaurant` (Figure 2.2).

Going to a restaurant for a meal	
Events:	Eating & drinking, Paying bills
Places:	Railroad dining cars, Restaurants
People:	Cooks, Restaurant workers, Waiters, Waitresses
Things:	Cash registers, Condiments, Dining tables, Menus, Silverware, Table settings & decorations, Tableware

Figure 2.2: Activity representation for `going to a restaurant` scenario

FrameNet (Baker et al., 1998) also contains special frames called *scenario frames* that encode script knowledge. For instance, Figure 2.3 shows a scenario frame for `Operate vehicle` scenario which more or less equates with `driving a car` scenario. `Operate vehicle` scenario has 3 subframes: *getting vehicle underway, operating the vehicle* towards some destination and finally *cause to land* meaning bringing the vehicle to a stop. The subframes represent the events necessary for realizing the given scenario represented by the scenario frame.

Handcrafting script knowledge is challenging and to some part practically infeasible for building broad coverage scrip-based systems. First the final script knowledge base solely relies on the individual background of the workers. Also, there exits a vast amount of

Operate vehicle Scenario :	This frame describes an Operator interacting with a Vehicle to transport him/herself (and possibly others) from a Source to a Goal. The three primary stages are preparation for canonical operation, operation of the vehicle proper, and bringing the vehicle back to a state of rest, where it may continue to move but not in its typical fashion.
Core FEs:	*Area, Goal, Operator, Path, Source, Vehicle*
Subframes:	***Getting vehicle underway, Operate vehicle, Cause to land***

Figure 2.3: *Operate Vehicle* scenario frame

script knowledge and it would be challenging to handcraft all possible scripts. Scripts also have complex internal structure thus making it difficult to handcraft all possibilities. Script events and participants can be linguistically expressed in various ways. Paraphrase information about script events is important and capturing this variability is difficult with handcrafting. A scenario can have several variants (e.g. for `eating in a restaurant`, the script variants could include, eating in a fast-food, or fancy or drive-in restaurant). Certain script events trigger preference for one script variant over another. Script induction systems come naturally with preference information about scripts and script events while handcrafting does not come with preference information.

Extracting script knowledge from text

An important line of research attempts to leverage existing large text corpora to learn script-like knowledge. Chambers and Jurafsky (2008) learned *narrative chains* from text by identifying co-occurring events that share participants using point-wise mutual information. The *narrative chains* are composed of verb sequences plus their dependency information. Figure 2.4 shows an example narrative event chain that involves a shared protagonist X. Chambers and Jurafsky (2009) built typed narrative chains by including sets of argument types representing a single role in an event chain. For instance, in Figure 2.5, the left hand side shows three typed chains. In the second typed chain, the argument types representing the role in blue could be *criminal*, *suspect* etc. Chambers and Jurafsky (2009) built narrative schemas by merging typed chains. Figure 2.5 shows an example of how three typed chains are merged into a single partially ordered narrative schema.

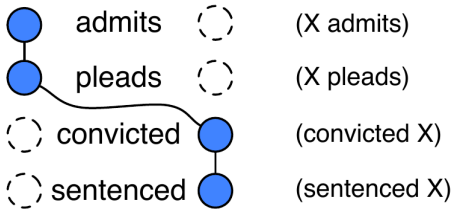


Figure 2.4: Example event chain with a single protagonist X (Chambers and Jurafsky, 2009)

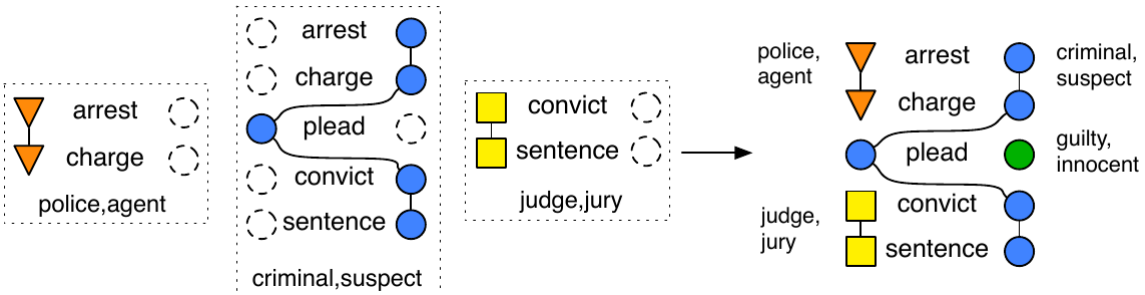


Figure 2.5: Merging typed chains into a single unordered Narrative Schema. (Chambers and Jurafsky, 2009)

Following the seminal work of Chambers and Jurafsky (2008, 2009) on the induction of script-like *narrative schemas* from large, unlabeled corpora of news articles, a series of models have been presented for improving the induction method. For example: Jans et al. (2012) studied different ways of selecting event chains and used skip-grams for computing event statistics, Pichotta and Mooney (2014) employed richer event representations, exploiting the interactions between multiple arguments to extract event sequences from a large corpus. Also, models have been presented for exploring alternative data sources like narrative texts from the web, for script learning, for example, Manshadi et al. (2008); Gordon (2010) mined commonsense knowledge from stories describing events in day-to-day life. Tandon et al. (2014, 2017) built a commonsense knowledge base containing fine-grained relations about sense-disambiguated nouns and adjectives by combining pattern-based techniques and semi-supervised label propagation to extract assertions from web contents. Ryu et al. (2010) applied syntactic-patterns and probabilistic machine learning to automatically extract activity knowledge from eHow.com[2] articles. Chu et al. (2017) extracted and semantically organized procedural knowledge from WikiHow[3] and built task-frames showing task phrases, e.g. painting a wall, with their associated attributes, hierarchical and tempo-

[2]www.ehow.com
[3]www.wikihow.com

ral relation. Abend et al. (2015) proposed an edge-factored model to induce the temporal order of events in cooking recipes, but their model is limited to scenarios with an underlying linear order of events. Rudinger et al. (2015) formulated the script learning task as a language modeling task and extract events from stories found in Dinners from hell[4]. Tandon et al. (2015) built activity-frames representing information about human activities e.g. `climbing a mountain` that were extracted from narrative texts as well as from movies and TV series scripts.

All these approaches aim at high recall, resulting in a large amount of wide-coverage, but noisy schemas. The extracted events do not have paraphrase information on the various ways the given event can be linguistically realized. Also, the models produce verb sequences plus dependency information not related to a specific script or scenario. We can also not rely on the temporal order information given in the texts as events in texts are not necessarily in the order in which the events in the script occur. Building robust script parsers require information about the relation of events and participants to a given scenario and the paraphrase information of script events and participants.

To the best of our knowledge, majority of the corpus-based methods for script knowledge acquisition from texts do not have information relating the extracted event sequences with the appropriate scenarios. One path of research tries to relate extracted events to specific topics (scenarios) by first creating topic-specific datasets from the Web, then extracting topic-specific event schemas from them. Kasch and Oates (2010); Rahimtoroghi et al. (2016) follow this path and extract topic-based event schemas from the built topic-specific datasets. Kasch and Oates (2010) focused on `eating in a restaurant` domain as proof-of-concept for their approach, while Rahimtoroghi et al. (2016) focused on learning contingent relations between events and not on script acquisition and induction. Both methods did not provide paraphrase information on the extracted events nor any information on temporal order between events. Their analysis shows that we could extract reliable script information by having multiple narratives describing the same scenario. This establishes a basis for further study on script knowledge acquisition from narrative texts. In the second part of this dissertation, we look at ways of script knowledge acquisition from narrative texts by identifying text segments referring to given scenarios (see Chapter 5).

[4] www.dinnersfromhell.com/

Crowdsourcing script knowledge

Crowdsourcing. Brabham (2008) describes crowdsourcing as a method for accessing a large pool of human workforce, mostly non-expert workers, "through an open call for proposals". Crowdsourcing is increasingly becoming a quick and affordable method for data acquisition and annotation for many NLP tasks. Online interfaces e.g. Amazon Mechanical Turk[5], can distribute the workforce among many non-experts. As the workforce is made up of non-experts, the tasks to be done have to be broken down to what can be done by them. Snow et al. (2008) demonstrated the effectiveness of using crowdsourcing for a variety of NLP tasks such as affect recognition, word similarity, recognizing textual entailment, temporal event recognition, and word sense disambiguation. Crowdsourcing has also been applied in acquisition of linguistic corpora (Wang et al., 2012), paraphrase generation (Chen and Dolan, 2011; Burrows et al., 2013) and acquisition of commonsense knowledge (Open Mind Common Sense project, OMCS, Singh et al. (2002)).

Script knowledge acquisition via crowdsourcing. Scripts represent the stereotypical events that normally take place when accomplishing a given scenario. It would be non trivial to ask crowd workers to provide script information as crowdsourcing is done by non-experts who maynot have an understanding on what scripts are. To make the task doable for the workers, we do not ask them to write a script but rather ask them to write down the steps that they would typically do in order perform a given activity. We later induce the script structure from the collected descriptions. Each step is described with a short sentence, and the sequences of steps describe the events in a given scenario. We call these **Event Sequence Descriptions**, ESDs. Figure 2.6 shows example ESDs for `eating in a fast food restaurant` scenario. ESDs are generic descriptions of how an activity (scenario) usually takes place. Each ESD is provided by a different worker hence the ESDs are of different granularity.The units that are used for induction are simple sentences. The sentences provide a great resource for obtaining paraphrase information of the events in the scenarios. You also can rely on the temporal order of events as provided in the ESDs. There is also a fixed relation between event descriptions and scenarios i.e. each ESD belongs to exactly one scenario. The ESDs provide texts that explicitly refer to scripts, and possibly also include the mundane aspects of scripts that would normally not be mentioned in text.

[5] www.mturk.com

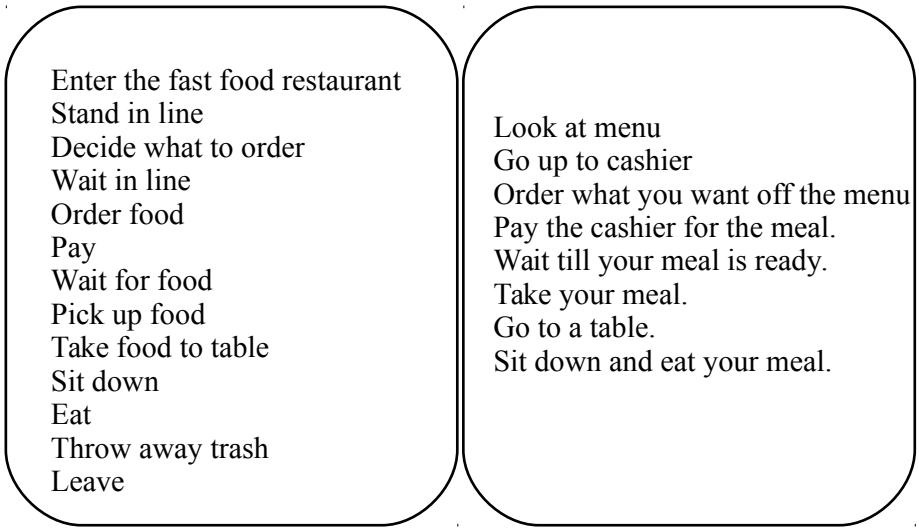


Figure 2.6: Example ESDs for `eating in a fast food restaurant` scenario

Crowdsourcing ESDs allows us to acquire script information for specific scenarios, thus building script-specific linguistic material from where we can induce script knowledge.

Recently, there has been research on using crowdsourcing for acquisition of script knowledge. The Honda Research Institute project, Open Mind Indoor Common Sense (OMICS, Gupta and Kochenderfer (2004)) cloned the OMCS project (Singh et al., 2002) and crowdsourced commonsense knowledge about indoor activities. The task was oriented towards activities for indoor robots e.g. `watering indoor plants, answering the doorbell, cleaning the floor,` etc. They asked workers to write down short sentences describing how they would accomplish an activity. For instance "The task *water indoor plants* involves the steps:_____" (Gupta and Kochenderfer, 2004). Their task was limited to only indoor activities. Li et al. (2012) also crowdsourced ESDs for narrative intelligence and provided named characters for the participants in the scenarios e.g. for `eating in a fast food restaurant` scenario, event descriptions could be *Mary looks at the menu, Mary decides what to order*, etc. They only collected activity descriptions for about 9 scenarios. Most similar work to our crowdsourcing approach is the work of Regneri et al. (2010), in the SMILE[6] project, who collected event sequence descriptions for about 22 everyday scenarios. We describe details on existing script corpora in Section 2.3 and give details of our crowdsourcing approach for script acquisition in Chapter 3.

[6]http://www.coli.uni-saarland.de/projects/smile/

My friend Margo and I went to McDonald's to have lunch together before we went to the movies. We went in around lunch time and got in line, patiently waiting our turn to order. When it was our turn, *I ordered a cheeseburger without pickles*, a large order of French fries and a medium Diet Coke. Margo ordered a grilled chicken sandwich, a small order of fries, an apple pie and large Sprite. We quickly paid the cashier and moved to the side, eagerly awaiting our meal. After only a few minutes, the employees placed our trays on the counter top for us to take. We stopped at the service station and grabbed some napkins and ketchup for our fries. Margo found a table for us in the corner near a window. We sat down and ate our food while talking about the movie we were going to go see after we ate lunch.

Figure 2.7: An excerpt from a story on `eating in a fast food restaurant` scenario

Instead of collecting stories from the web, you can crowdsource narrative texts with respect to a given scenario as a basis for script acquisition. Previous work on crowdsourcing narrative texts include work by Modi et al. (2016); Ostermann et al. (2018a). They crowdsourced narrative texts for several scenarios asking users to write a story about a given scenario "as if they were explaining to a child". The workers were encouraged to include the mundane aspects of a scenario that would normally be left out in narrative texts. Figure 2.7 shows an excerpt from a story about `eating in a fast food restaurant` scenario. ESDs about given scenarios differ from narrative texts in that ESDs describe event and participant types rather than giving specific references e.g. *order food* as opposed to *I ordered a cheese burger without pickles* in the narrative text example (Figure 2.7). The participants are types, *food, order* instead of particular foods (*chips, burger, etc.*). The events in narrative texts can be inverted, i.e. the order of the events mentioned in texts does not necessarily indicate the order in which the events in the scenario typically occur. In contrast, we can rely on the order of events as provided in ESDs. It is also common in narrative texts, as compared to ESDs, to find more than one instance of an event (e.g. ordering is mentioned twice), and also mentions of other scenarios before, after or during the current scenario, e.g. `going to the movie` is mentioned at the beginning and at the end of the passage in Figure 2.7. Such crowdsourced texts can be a resource for learning scripts, but to our knowledge this has not been done.

A related research is work by Mostafazadeh et al. (2016) who built a corpus for understanding stories by crowdsourcing short everyday commonsense stories about arbitrary topics.

They proposed to use it for the evaluation of script knowledge models, and it may also turn out to be a valuable resource for script learning, although to our knowledge this has not yet been attempted.

This dissertation uses crowdsourcing of ESDs as a method of script knowledge acquisition. Chapter 3 describes our methodology for acquiring quality script knowledge by crowdsourcing event sequence descriptions for various scenarios.

Script acquisition from other modalities

Images have also been used in the acquisition of script knowledge. Bosselut et al. (2016) induced prototypical event structure in an unsupervised way from a large collection of photo albums with time-stamped images and captions. This method is limited by the availability of albums for "special" events such as `wedding` or `barbecue`, in contrast to everyday, trivial activites such as `making coffee` or `going to the dentist`.

2.2.2 Script Induction from ESDs

As pointed out in Section 1.1, a script is a structure showing information about the events, participants and the temporal order. *Script induction* involves ways of automatically acquiring the target script knowledge representation from ESDs. Consider the internal structure of a script (see Section 1.1), what principally needs to be learned in order to arrive at script representations? We need to learn information about the events, participants and the temporal order given two script events. We also need to have information about the various ways in which the script events and participants can be paraphrased.

In this section we focus on previous work on script induction from event sequence descriptions. Closest to the approach adopted in this dissertation is the work of Regneri et al. (2010) (henceforth referred to as RKP). RKP developed a model for extraction of explicit script structure from the ESDs using Multiple Sequence Alignment (MSA) (Durbin et al., 1998). Sequence alignment can be viewed as the task of aligning points shared by two sequences in order to re-write one sequence into another sequence. In script sense, Multiple Sequence Alignment aligns multiple ESDs with each other: event descriptions expressing the same event are aligned with each other. Multiple Sequence Alignment relies on the assumption that the temporal position of an event description in an ESD provides a

strong cue for the event type it describes, and vice versa. RKP represented the learned script knowledge as Temporal Script Graphs (TSG). Temporal script graphs are partially ordered structures whose nodes are sets of alternative descriptions denoting the same event type, and whose edges express temporal precedence. Figure 2.8 shows an example TSG for `baking a cake` scenario. As can been seen, the nodes are made up of alternative descriptions that denote a given event e.g. *pour cake mix in bowl, add ingredients to bowl* and *add cake ingredients* are linguistic realizations for the same event. Frermann et al. (2014) present a Bayesian generative model for joint learning of event types and ordering constraints. Their model shows that flexible event order in scripts can be suitably modeled but paraphrase information was not well captured. Modi and Titov (2014) focused on event ordering between script-related predicates, using distributed representations of predicates and arguments induced by a statistical model. They obtained paraphrase sets as a by-product, namely by creating an event timeline and grouping together event mentions corresponding to the same interval. Orr et al. (2014) learn scripts as Hidden Markov Models (HMM) and induced a graphical representation of scripts with transition probabilities learned from corpora. Their model did not capture the flexible event ordering in scripts. For comparison, all these methods employ the same dataset for evaluation, that is, the SMILE corpora (Regneri et al., 2010) and part of the OMICS dataset (Singh et al., 2002; Gupta and Kochenderfer, 2004) (see Section 2.3 for details on these corpora). We also use the same dataset to allow for a direct comparison. We will discuss further these models and explain our robust and flexible model for script induction in Chapter 4.

In this work, we also adopt TSGs to represent script knowledge. TSGs represent the temporal order information and paraphrase information in form of paraphrase sets. Complex formalism, like Finite Automata (Hopcroft et al., 2001) and Petri nets (Petri, 1966), would be worth exploring but left for future work. TSGs are not Finite Automata. Automata convey what must occur in a certain place, but you can not state that an event must occur at certain points using TSGs. Automata are in principle relevant and would be interesting for script representation but we do not consider them for this dissertation. ESDs give information about the partial ordering between events, without concretely providing information about what must go in between. For instance, if A, B and C are events in a given script, from ESDs we can have one ESD with ABC and another with AC. What is B? is it an optional event in the scenario? Is it an event that must occur in the scenario but just left out?

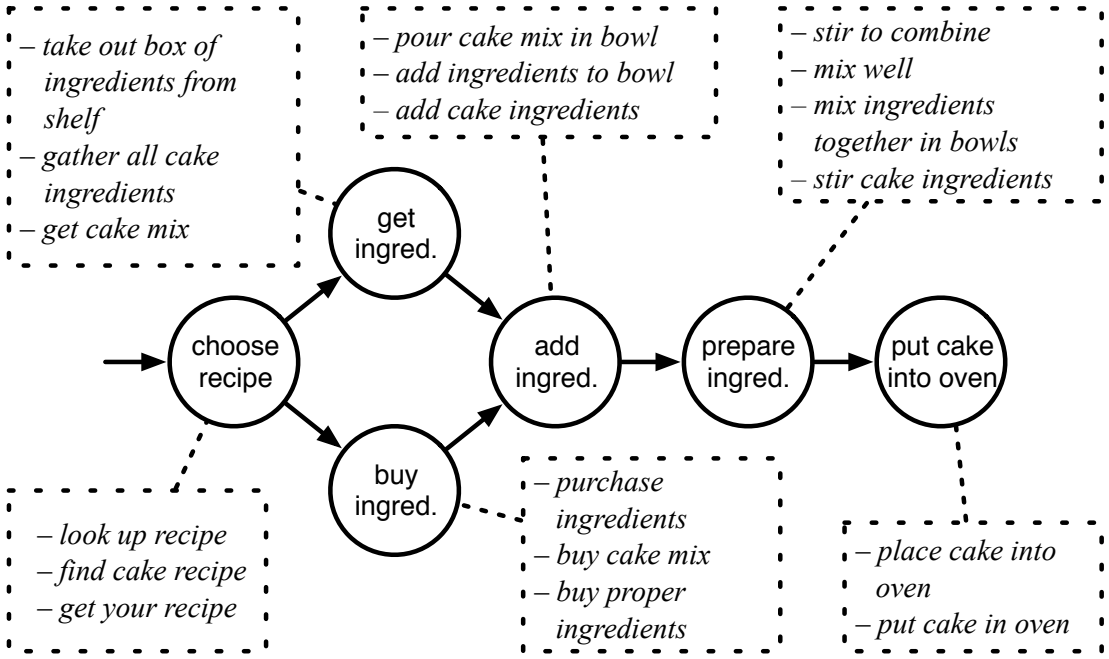


Figure 2.8: Example TSG for `baking a cake` script

TSGs are a direct way of representing the partial ordering information that can be derived from ESDs.

We extend RKP work, first by Crowdsourcing ESDs for more scenarios (Chapter 3), second, by proposing a flexible model for script induction from ESDs to better handle order variation in scripts and third, by extending their script representation formalism, Temporal Script graphs, by incorporating "arbitrary order" equivalence classes in order to allow for the flexible event order inherent in scripts (see Chapter 4).

Alternative representations for script knowledge

In this section, I highlight alternative representations of script-like knowledge that have been applied to crowdsourced datasets.

ConceptNet (Liu and Singh, 2004; Havasi et al., 2007; Speer and Havasi, 2012) is a semantic network representations that is made up of (among others) the crowdsourced commonsense knowledge learned from the OMCS (Singh et al., 2002) data set. *Concepts* are the basic nodes in ConceptNet and consists of a noun phrase, verb phrase, adjectival phrase or prepositional phrase. In script terms, concepts represent event or participant types. Concepts are related by an edge indicating the type of relation. Figure 2.9 shows example concepts with relations between pairwise concepts, e.g *used for, location of, prerequisite*

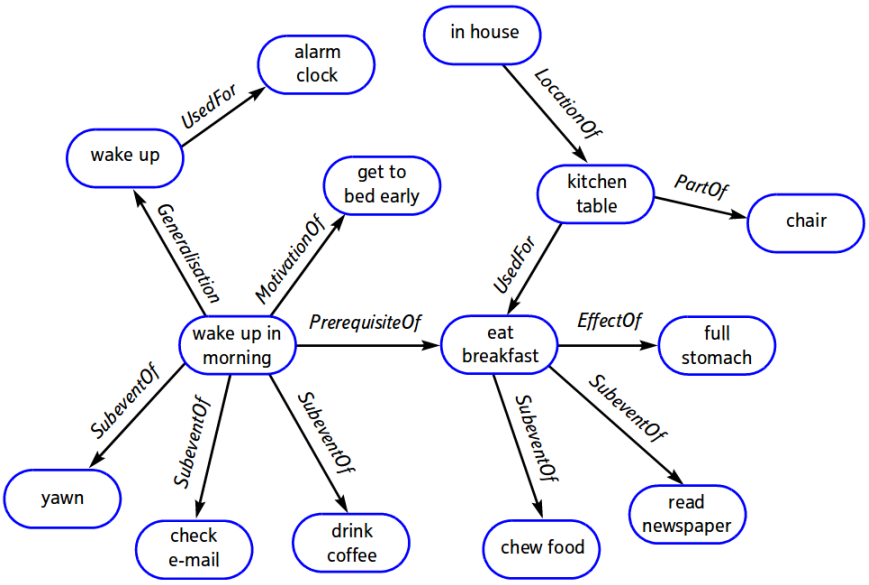


Figure 2.9: An excerpt from ConceptNet's semantic network of commonsense knowledge (Liu and Singh, 2004)

for, etc. The relation between concepts is not handled in this dissertation on the general level that is in ConceptNet.

Gupta and Pedro (2005) represented their crowdsourced indoor ESDs (OMICS, Gupta and Kochenderfer (2004)) as multidimensional semantic nets called PraxiNet. PraxiNet has two major types of objects that can be compounded, a *situation* which is the conjunction of an object and property e.g. floor slippery, window broken, coffee cup empty. and a *response* which is the conjunction of an *action* and an *object* (see Figure 2.10). In script terms, an event can be represented as a response, the action being the event type and the objects as the participant types.

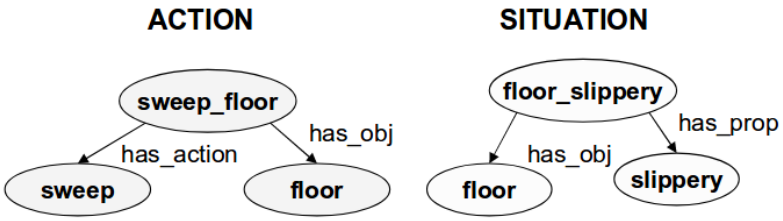


Figure 2.10: Example representation of a *situation* and a *response* in Praxinet. (Gupta and Pedro, 2005)

The LifeNet (Singh and Williams, 2003) knowledge base is a graphical representation of commonsense knowledge also learned from OMCS (Singh et al., 2002). The nodes are

egocentric propositions that have temporal and atemporal links between them. For example, when you *put your foot at the brake pedal*, it is likely that you will *stop the car*; when you *put a key in the ignition*, it is likely that you will *drive a car*; etc. The collected event relations are not linked to a particular scenario and tend to be noisy. Causal or enabling relations between events are relevant for script knowledge, but will not be handled in this dissertation.

2.3 Script Corpora

Table 2.1 shows a summary of the existing script knowledge bases in form of ESDs with their approximate sizes that have so far been built. Regneri et al. (2010); Singh et al. (2002); Gupta and Kochenderfer (2004); Li et al. (2012) collected the ESDs via crowdsourcing. In addition, Regneri (2013) crowdsourced a domain specific corpus focusing on the cooking domain. The collected ESDs were much longer and more complex as compared to their previous work (Regneri et al., 2010). Raisig et al. (2009) recruited ninety German native speakers from Humboldt University Berlin to participate in the script collection task. Out of the 60 scenarios they collected, they choose 30 scenarios for their study, and later merged ESDs from the same scenario into a single "gold" ESD.

In Chapter 3, we extend the existing repository of script knowledge bases by crowdsourcing ESDs for both new scenarios, as well as for old scenarios based on previous work by Raisig et al. (2009), Regneri et al. (2010) and Singh et al. (2002). Our choice of scenarios was motivated by the act that we wanted a representative set of scenarios with varying degrees of complexity and covering a range of everyday activities.

ESD Corpora	Total scenarios	Total ESDs	ESD per Scenario
SMILE (Regneri et al., 2010)	22	386	5 to 24
Cooking (Regneri, 2013)	53	2500	50
OMICS (Singh et al., 2002; Gupta and Kochenderfer, 2004)	175	9044	14 to 122
Raisig et al. (2009)	30	450	15
Li et al. (2012); Li (2015)	9	500	30 to 79

Table 2.1: Existing corpora focusing on scripts

Apart from ESDs, there are several text collections that are a potential source for script knowledge acquisition. Modi et al. (2016); Ostermann et al. (2018a) crowdsourced stories instantiating script knowledge (InScript and McScript respectively). They encouraged the crowdworkers to include mundane aspects of scripts that would normally be left out in a narrative text. InScript contains 10 scenarios with about 100 stories each while McScript contains 110 scenarios with about 20 stories per scenario.

The ROC-stories database (Mostafazadeh et al., 2016) consists of 50,000 short narrative texts, collected via Mechanical Turk. Workers were asked to write a 5-sentence length story about an everyday commonsense activity, and they were encouraged to write about "anything they have in mind" to guarantee wide distribution across topics. These stories may turn out to be a valuable resource for script learning, although to our knowledge this has not yet been attempted.

Gordon and Swanson (2009) employed statistical text classification in order to identify narrative texts about personal stories from the Spinn3r[7] dataset (Burton et al., 2009). The extracted personal stories corpus contains about 1.5 Million stories. It serves as a potential source of naturalistic texts providing script knowledge.

This dissertation looks at scenario-based text segmentation and classification in narrative texts from the subset of Spinn3r corpus created by Gordon and Swanson (2009). We employ statistical text classification using InScript and McScript corpora as anchors in order to identify Spinn3r texts that address specific scenarios (see Chapter 5 for details).

2.4 Summary

In this chapter, I provide background information on scripts. From early systems that handcrafted script knowledge, to methods that leverage existing text corpora for script knowledge acquisition and finally to techniques that crowdsource event sequence descriptions that provide script-specific linguistic material that explicitly refer to scripts. I have also given a summary of existing script knowledge bases. We have seen that the existing script knowledge bases are not sufficient and there is still need for reliable knowledge bases from where script knowledge can be learned.

[7] http://www.icwsm.org/data/

Part II

Crowdsourcing Script Knowledge and Script Induction

Chapter 3

DeScript: A corpus Describing Script Structure

In this chapter, we present a large-scale crowdsourced collection and annotation of explicit linguistic descriptions of event patterns, to be used for the automatic acquisition of high-quality script knowledge. This dissertation is part of a larger research effort where we seek to provide a solid empirical basis for high-quality script modeling by inducing script structure from crowdsourced descriptions of typical events, and to investigate methods of text-to-script mapping, using naturalistic texts from crowdsourced stories, which describe real-life experiences and instantiate the set of scripts covered by our collection (Modi et al., 2016) (see Section 3.5). Predecessors of our work are the OMICS and SMILE corpora (Singh et al., 2002; Regneri et al., 2010), containing multiple event-sequence descriptions (ESDs) for specific activity types or *scenarios* (see Section 2.3).

We have taken measures to provide a sound empirical basis for high-quality script models, by extending existing corpora in two different ways. First, we crowdsourced a corpus of 40 scenarios with 100 ESDs each (see Section 3.1), thus going beyond the size of previous script collections. Second, we enriched the corpus with partial alignments of ESDs, done by human annotators. The partial alignments are employed in our semi-supervised script-induction model. We will discuss our methodology for the alignment collection in Section 3.3 and details of the semi-supervised script induction model in Chapter 4. For evaluation,

borrowing a book from the library	going to the swimming pool	taking a child to bed
changing batteries in an alarm clock	going to the theater	taking the underground
baking a cake	having a barbecue	washing one's hair
cooking pasta	ironing laundry	washing the dishes
doing laundry	making a bonfire	checking in at an airport
fueling the car	making coffee	cleaning up a flat
getting a hair cut	ordering a pizza	eat in fast food restaurant
going bowling	planting a tree	making scrambled eggs
going grocery shopping	play tennis	paying with a credit card
going on a train	renovating a room	taking a bath
flying in an airplane	repairing a flat bicycle tire	take a driving lesson
going to a funeral	riding on a bus/taking a bus	taking a shower
going to the dentist	sewing on a button	going to the sauna
	sending food back (in a restaurant)	

Table 3.1: The 40 scenarios in DeScript

we fully aligned event descriptions for 10 scenarios and created gold event clusters for each of the 10 scenario. We give details of the gold-standard creation in Section 3.2. The result is a corpus of partially-aligned generic activity descriptions, the **DeScript**[1] corpus (**De**scribing **Script** Structure). More generally, DeScript is a valuable resource for any task involving alignment and paraphrase detection of events.

We compare our crowdsourced ESDs to the crowdsourced collection of narrative texts, InScript (Modi et al., 2016), which is also part of our larger research effort to provide a solid empirical basis for high-quality script modeling, in Section 3.5.

3.1 ESD Collection

There are a number of corpora providing scripts (see Section 2.3) with varying numbers of ESDs per scenario but we needed a more representative and uniform database of scenarios controlled for our purpose. We selected 40 scenarios for which we wanted to crowdsource ESDs. Section 3.1.1 describes the criterion we used to select these scenarios. In Section 3.1.2, I describe the setup of the crowdsourcing experiment and the analysis of the collected data.

[1]The corpus is publicly available for scientific research purposes at this url: `http://www.sfb1102.uni-saarland.de/?page_id=2582`

3.1.1 Choice of scenarios

Our choice of scenarios was motivated by the fact that we wanted a representative set of scenarios with varying degrees of complexity and covering a range of everyday activities. Apart from new scenarios that we came up with, we also included scenarios based on previous work by Raisig et al. (2009), Regneri et al. (2010) and Singh et al. (2002) (see Section 2.3). We acquired ESDs for both new and old scenarios. Table 3.1 lists the 40 scenarios that was used. DeScript contains ESDs for 40 scenarios, with approximately 100 event sequences per scenario. We discuss more on the collected data in Section 3.1.3. The selected set of scenarios show interesting variability and complexity as highlighted below:

- Common scenarios: scenarios thought to have a general shared knowledge among many users but still possessing interesting variability: `eating in a fast food restaurant`, `borrowing a book from the library`, `riding on a bus`, `going grocery shopping`, e.t.c.
- Expert knowledge scenarios: scenarios requiring some amount of expert knowledge: `renovating a room`, `sewing a button`, `planting a tree`, `repairing a flat bicycle tire`, e.t.c.
- Complex scenarios: scenarios that are thought to have complex event patterns: `going to a funeral`, `going to the dentist`, `borrowing a book from the library`, `taking a driving lesson`, e.t.c.
- Simple scenarios: scenarios that are thought to have simple event patterns: `washing ones hair`, `taking a shower`, `ironing laundry`, `paying with a credit card`, e.t.c.
- Flexible pattern scenarios: scenarios with a considerable degree of variability: `going to a funeral`, `going to a funeral`, `taking a child to bed`, e.t.c.
- Fixed pattern scenarios: scenarios with a considerable fixed order of events: `eating in a restaurant`, `flying in an airplane`, e.t.c.
- We included scenarios relating to various aspects of everyday activities including:

  - Leisure activities: `going to the sauna`, `eating in a restaurant`, `going bowling`, `going to the theater`, e.t.c.
  - Cleaning and house chores: `doing laundry`, `ironing laundry`, `washing the dishes`, `cleaning up a flat`, e.t.c.
  - Cooking: `baking a cake`, `cooking pasta`, `making coffee`, `making scrambled eggs`, e.t.c.
  - Sports: `playing tennis`, `going swimming`, `going bowling`, e.t.c.
  - Outdoor activities: `having a barbecue`, `making a bonfire`, `planting a tree`, e.t.c.
  - Using public transport: `going on a train`, `flying in an airplane`, `riding on a bus`, `taking the underground`, e.t.c.
  - Personal grooming: `taking a bath`, `taking a shower`, `washing one's hair`, `getting a haircut`, e.t.c.
  - Making transactions: `paying with a credit card`, `going grocery shopping`, `ordering pizza`, `fueling a car`, e.t.c
  - Do-It-Yourself: `sewing a button`, `changing batteries in an alarm clock`, `repairing a flat bicycle tire`, e.t.c.
  - Renovation: `renovating a room`, e.t.c.

- We included scenario clusters that have semantic relations among them:
  - Event holonyms: one scenario being part of a more general scenario: `paying with a credit card` could be part of `eating in a restaurant`, `going grocery shopping`, e.t.c., `checking in at an airport` is part of `flying in an airplane`.
  - Related scenarios: scenarios having similar events and are part of some more general scenario: `going on a train`, `taking the underground` and `riding on a bus` are part of a more general scenario of using public transportation, `taking a bath`, `taking a shower` and `washing one's hair` are part of a more general scenario of grooming oneself.

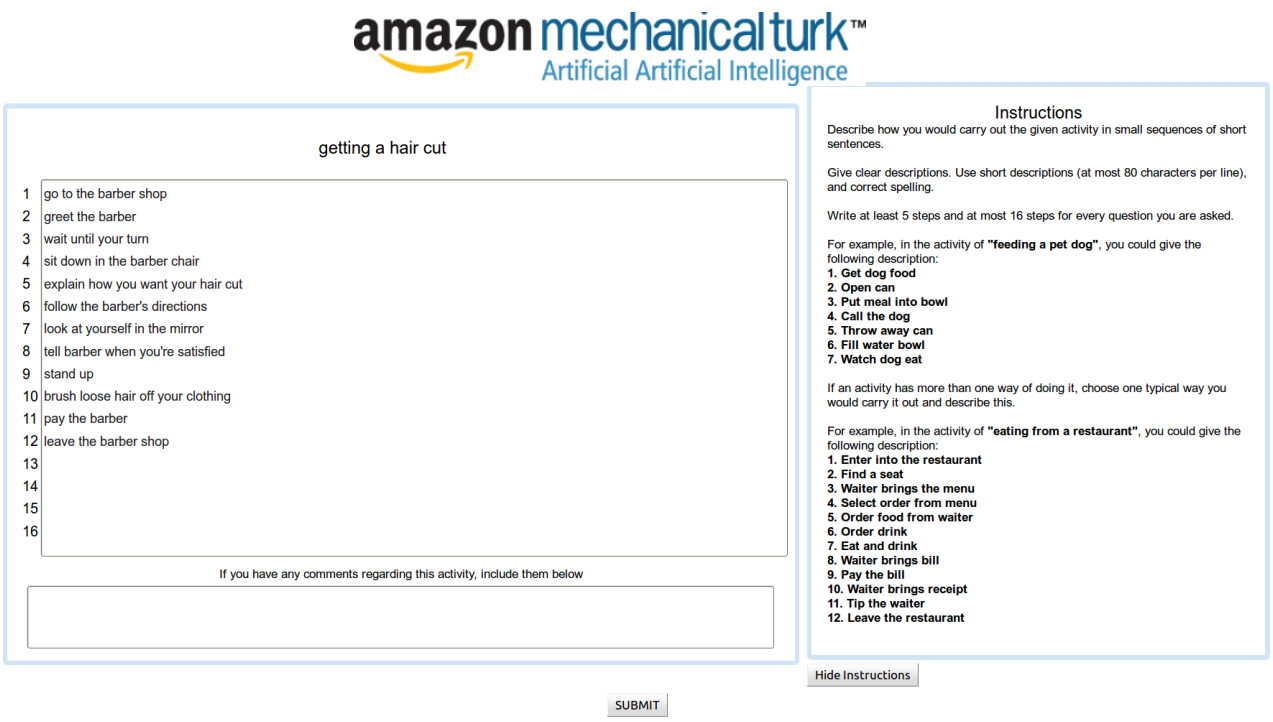


Figure 3.1: Experimental setup: M-Turk Interface

3.1.2 Experimental setup

There are a number of crowdsorcing platforms e.g. Amazon's Mechanical Turk[2] (M-Turk) (Barr and Cabrera, 2006), Prolific[3], Figure-eight[4] that use human-in-the-loop for machine learning tasks,annotation, e.t.c. We opted to use Amazon's Mechanical platform, henceforth referred to as M-Turk, as it has been well established and is reliable.

A total of 320 workers (native speakers of English) described everyday activities in small sequences of short sentences. Each worker could write at most one ESD per scenario, and between 5 and 16 event descriptions per ESD. They were paid 0.20 USD per ESD and took on average 2.78 minutes per ESD. After a pilot study on 10 scenarios with 10 ESDs per scenario, we collected the full corpus of 40 scenarios with 100 ESDs per scenario.

3.1.3 Corpus description

We collected 100 ESDs per scenario, totaling to c.a. 4000 ESDs for 40 scenarios, with more than 30,000 event descriptions. Once the data was collected, it was manually checked to

[2]https://www.mturk.com/
[3]https://prolific.ac/
[4]https://www.figure-eight.com/

ESD 1

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

ESD 2

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

ESD 3

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

Figure 3.2: Example ESDs `baking a cake`

remove ESDs that had unclear language or where the worker misunderstood the task. In total we discarded 7% of the collected ESDs and had on average 93 ESDs per scenario.

Figure 3.2 shows example ESDs for `baking a cake` scenario. We can see that the ESDs differ in granularity (e.g. ESD 1 has 16 event descriptions while ESD 2 has only 7 event descriptions). Some script events are left implicit in ESD 2 (e.g. *set timer*,*bake cake*, e.t.c.) There are also interesting variation in the way events are linguistically realized (e.g. in ESD 1, *spray pan with non stick or grease* refers to the same script event as *grease pan* in ESD 3). It can also be noted that some event descriptions are more general than others (e.g. *prepare the cake pan* in ESD 2 could in principle include *grease pan* in ESD 3). It is also interesting to observe that the order of script events varies across ESDs (e.g. *grease pan* happens before *preheat oven* in ESD 3 while in ESD 1 *preheat oven* happens before *spray pan with non stick or grease*).

Analysis on collected data

Although the collected dataset has high conformity, there are interesting scenario-specific differences, which can be captured by different metrics. Next we explain the metrics that we used to measure differences among scenarios:

No. of word types This is the number of unique words in a document i.e. the word count after lemmatization (or stemming in some cases). Scenarios with higher number of word types tend to be more complex.

Avg. word tokens per word type This is the average number of tokens for every word type in a given scenario. Scenarios with higher number of tokens per word type are more homogeneous and have lower lexical diversity as people tend to use the same words on average.

Avg. word type per ESD This is the average number of word types per ESD in each scenario. More word types indicates scenarios with higher degree of diversity as each participant tends to use different sets of words.

Avg. tokens(types) per event This is the average number of tokens(types) per event descriptions in a given scenario. Scenarios whose events are described with more tokens(types) tend to involve more participants and such event descriptions are more complex and harder to process.

Avg. events per ESD This is the average number of events descriptions per ESD in a given scenario. Scenarios with longer ESDs tend to be more diverse.

Type-token-ratio (TTR) This is the ration of number of unique words (word types) and number of unique tokens (all words) in a document. Higher TTR shows a higher degree of lexical variation in a document.

ESD word-type overlap This is the proportion of overlap of word types between pairs of ESDs in a given scenario (averaged over all pairs of ESDs). We compute this using Dice:

$$type\_overlap(ESD_1, ESD_2)) = \frac{2*|types(ESD_1) \cap types(ESD_2)|}{|types(ESD_1)|+|types(ESD_2)|}$$

Table 3.2 gives an overview of the collected data. Overall, `taking a shower` has the largest number of events per ESD on average (10.96), followed by `baking a cake` (10.66), while `taking the underground` and `sending food back (in a restaurant)` have the lowest number of events per ESD on average, 7.56 and 6.44 respectively. A possible reason could be that `baking a cake` is generally a more complex scenario as compared to `sending food back (in a restaurant)`. A second possible reason, that is

also noted by Raisig et al. (2009), is that people tended to use more event descriptions for frequent[5] activities and fewer event descriptions for less frequent activities. `Taking a shower` and `baking a cake` can be considered as common and frequent scenarios, while `sending food back (in a restaurant)` is a less frequent activity.

Linguistic diversity

It is interesting to note that there was high lexical variability in the way one event in a given scenario was linguistically realized. Figure 3.3 illustrates the linguistic variations for events in `baking a cake` scenario: *grease pan*, *prepare cake pan* and *spray cake pan with non stick* all refer to the same event, similarly *get a cake mix* and *take out box of cake mix*. We address this challenge of grouping together semantically similar events into event cluster in Chapter 4.

Easier scenarios show lower vocabulary variance than more complex ones as measured by the type-token ratio (TTR) per scenario: e.g. `taking a shower` has the smallest vocabulary variance with a TTR of 0.07 and shortest event descriptions (on average 3.87 tokens per event description). `renovating a room` has the highest TTR of 0.16 and is among scenarios with the longest event descriptions on average at 4.90 tokens, with the longest being `sending food back (in restaurant)` at 5.52 tokens.

We can see that these numbers correlate to the complexity of scenarios as mentioned in Section 3.1.1, where commonsense and simple scenarios (e.g. `taking a shower`, `baking a cake`, `doing laundry`, `paying with a credit card`, e.t.c.) have lower vocabulary variance (TTR between 0.07 - 0.09) as compared to expert and complex scenarios (e.g. `taking a driving lesson`, `renovating a room`, `going to a funeral`, `cleaning up a flat`, e.t.c) with TTR between 0.14 - 0.16.

Complexity and homogeneity

Different scenarios differ with regard to the amount of knowledge workers share about them. For example, ESDs for `taking a shower` and `renovating a room` differ not only for their TTR and length, but also in their homogeneity: ESDs for `taking a shower` are

[5] "frequent" refers to the number of times participants engaged in the given event i.e. once a year, once a week, everyday e.t.c

Scenario	Avg. Events per ESD	Word types	Avg. tokens per word type	Avg. word type per ESD	Avg. tokens per event	Type token ratio	Avg. ESD word Type overlap	Avg. word type per event
taking a shower	***10.96***	*281*	***14.2***	27.6	*3.87*	*0.07*	***0.46***	*3.81*
baking cake	10.66	425	11.03	32.56	4.63	0.09	0.35	4.54
fueling the car	10.05	369	12.89	30.29	4.83	0.1	0.36	4.71
doing laundry	10.57	372	12.64	30.06	4.78	0.08	0.35	4.69
making scrambled eggs	10.07	471	10.61	34.01	5.12	0.09	0.36	5.01
taking a bath	10.13	345	11.57	41.56	4.1	0.09	0.35	4.05
going grocery shopping	10.05	404	11	***47.26***	4.7	0.09	0.41	4.63
eating in a fast food restaurant	9.77	304	11.76	28.09	3.98	0.09	0.39	3.92
cooking spaghetti/pasta	9	367	11.13	42.12	4.68	0.08	0.38	4.6
going to the dentist	9.15	454	8.91	30.86	4.71	0.11	0.33	4.65
having a barbecue	8.83	505	7.76	27.75	4.58	0.13	0.25	4.5
going bowling	9.11	416	9.23	28.86	4.58	0.11	0.35	4.52
going to the theater	8.35	400	8.8	37.05	4.44	0.11	0.31	4.36
washing one's hair	8.61	319	10.24	23.77	4.13	0.1	0.35	4.13
flying in an airplane	9.86	430	7.4	29.78	4.04	0.13	0.32	3.99
getting a hair cut	8.66	399	9.56	29.37	4.84	0.1	0.36	4.78
going to the swimming pool	8.36	393	8.34	24.43	4.17	0.12	0.3	4.13
going to a funeral	7.85	517	6.97	27.25	4.59	0.14	0.26	4.5
ironing laundry	8.13	342	11.33	26.18	4.96	0.09	0.37	4.89
washing the dishes	7.98	344	10.91	25.94	4.9	0.09	0.36	4.82
renovating a room	7.72	***610***	*6.15*	26.91	4.91	***0.16***	*0.2*	4.84
making coffee	7.91	303	11.37	22.82	4.59	0.09	0.36	4.47
going to the sauna	8.12	407	7.69	24.79	4.19	0.13	0.27	4.14
ordering a pizza	7.47	379	9.11	25.37	4.86	0.11	0.31	4.78
taking a child to bed	7.54	322	10.2	25.46	4.63	0.09	0.33	4.59
borrowing a book from the library	7.29	294	11.29	22.84	4.69	0.09	0.38	4.59
sewing on a button	7.58	350	10.7	24.92	5.31	0.12	0.34	5.09
going on a train	7.41	318	9.33	22.13	4.21	0.1	0.39	4.14
play tennis	7.4	446	8.27	38.83	5.25	0.12	0.3	5.11
cleaning up a flat	8.52	450	6.77	27	4.36	0.15	0.24	4.3
repairing a flat bicycle tire	7.55	415	8.76	24.73	5.23	0.11	0.26	5.05
planting a tree	6.95	384	8.85	23.76	4.94	0.11	0.37	4.79
riding in a public bus	7.13	316	10.25	22.97	4.74	0.11	0.41	4.6
taking a driving lesson	7.61	443	7	25.37	4.57	0.14	0.28	4.5
checking in at an airport	7.48	352	8.51	25.28	4.55	0.12	0.3	4.5
making a bonfire	6.68	492	7.1	26.32	5.45	0.14	0.26	5.34
changing batteries in an alarm clock	6.63	305	9.78	*21.05*	4.69	0.1	0.38	4.59
paying with a credit card	6.73	363	8.06	23.63	4.73	0.09	0.29	4.65
taking the tube/underground	7.56	298	8.87	23.94	4.43	0.11	0.32	4.37
sending food back (in a restaurant)	*6.44*	370	8.74	25.14	***5.52***	0.11	0.3	***5.36***
Average	*8.35*	*385*	*9.58*	*28.2*	*4.66*	*0.11*	*0.33*	*4.58*

Table 3.2: DeScript corpus analysis

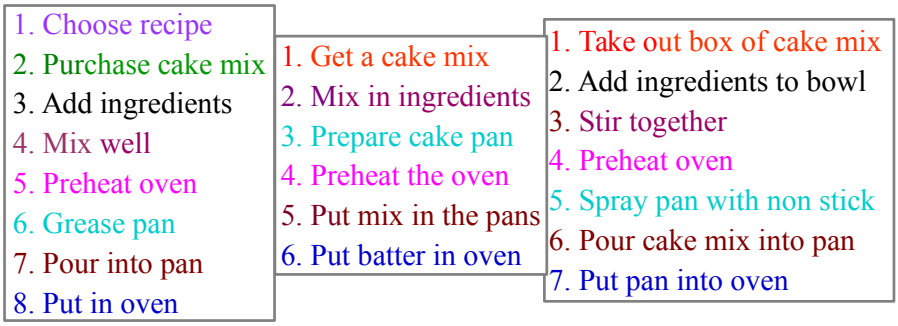


Figure 3.3: Example ESDs `baking a cake` showing lexical variability

most similar to one another (average Dice ESD word-type overlap of 0.46) while ESDs for `renovating a room` are least similar to one another (average Dice ESD word-type overlap of 0.2). While everyone has common knowledge about *taking a shower*, *renovating rooms* is not something we all share expertise about or do in the same way.

Scenario relatedness

Apart from variance within scenarios, it also interesting to compare similarity between scenarios. We want to ascertain whether or not semantically related scenarios are also lexically similar (e.g. `checking in at the airport` and `flying in an airplane` or `taking a shower, taking a bath` and `washing one's hair`). To do this, we employ a simple method for lexical analysis that checks the proportion of overlap of word types between scenarios using Dice coefficient. We collapse all ESDs for a given scenario into a single scenario document and get the word types for each scenario.

Table 3.3 summaries the results of the pairwise scenario similarities as measured by Dice coefficient over word types. The first 10 entries indicate the top 10 most similar scenario pairs while the lower 10 indicate the 10 least similar scenario pairs. We get an average of 0.24 type overlap. Most of the results can straightforwardly be explained. `Taking a bath, washing ones's hair` and `taking a shower` are most semantically similar as more than half of the vocabulary used in `taking a bath` is also used in both `taking a shower` and `washing ones's hair`. `Flying in a plane, going on a train, riding on a bus` and `taking the underground` are all part of a more general scenario of using public transportation , also share similar vocabulary (type overlap between 0.41-0.47).

Scenario 1	*Scenario 2*	*Word Type Overlap*
taking a bath	taking a shower	0.56
flying in a plane	checking in at an airport	0.55
washing one's hair	taking a shower	0.51
washing one's hair	taking a bath	0.51
going to the swimming pool	going to the sauna	0.49
flying in a plane	going on a train	0.47
cooking pasta	making scrambled eggs	0.47
going on a train	riding on a bus	0.46
taking the underground	going on a train	0.45
playing tennis	going bowling	0.45
going on a train	making scrambled eggs	0.12
washing one's hair	sending food back (in a restaurant)	0.11
going to the dentist	ironing laundry	0.11
cleaning up a flat	planting a tree	0.11
cleaning up a flat	going to a funeral	0.10
cleaning up a flat	paying with a credit card	0.10
sending food back (in a restaurant)	cleaning up a flat	0.10
cleaning up a flat	borrowing a book from the library	0.10
cleaning up a flat	checking in at an airport	0.10
cleaning up a flat	taking a driving lesson	0.08
Average overall scenario pairs		0.24

Table 3.3: Table showing scenario relatedness

Scenario holonyms, e.g. `flying in a plane` vs. `checking in at an airport` are also highly similar (typer overlap above 0.5.), as the two share many event and participant types. `Paying with a credit card` is part-of many other transaction scenarios, e.g. `ordering a pizza, fueling a car, eating in a fast food restaurant, going grocery shopping`, e.t.c,. `Paying with a credit card` compared with other transaction scenarios are also relatively similar (type overlap between 0.34-0.40).

3.2 Gold Standard Alignment Annotation

As indicated earlier, the DeScript corpus was collected to provide solid empirical basis for high-quality script modeling by inducing script structure from ESDs. In Section 3.1, I gave empirical analysis on the collected ESDs. The corpus does not only consist of the ESDs, but also a gold standard to be used for evaluating the script induction algorithm (see Chapter 4). The gold standard is composed of paraphrase sets each containing event descriptions that express the same script event. Four experts, all computational linguistics students, were trained for the task and were presented with a source and a target ESD. They were to link all event descriptions in the source ESD with all event descriptions in the target ESD that were semantically similar, with respect to the given scenario, to those in the source ESD. Every ESD in a given scenario was paired with every other ESD in the same scenario. In the end, all similar event descriptions were aligned and the full alignments were used to group the event descriptions into gold paraphrase sets. A small subset of the gold standard alignments were used to evaluate the quality of the crowdsourced alignments (see Section 3.3).

In creating the gold paraphrase sets, we assumed script-specific functional equivalence, that is, we instructed the annotators to group event descriptions serving the same function in the script as semantically similar (e.g, *scan bus pass*, *pay for fare* and *show driver ticket* are grouped into the same paraphrase set in the `riding on a bus` scenario, as they all represent the *pay* event). Each paraphrase set was annotated with an event label that indicated the event being expressed in the given paraphrase set (e.g. in Figure 3.4, paraphrase sets are represented with dashed-line rectangles, *choose recipe* and *buy ingredients* are example event labels for `baking a cake`). Event labels were harmonized with those used in the annotation of stories in Modi et al. (2016). We harmonized the labels as this dissertation

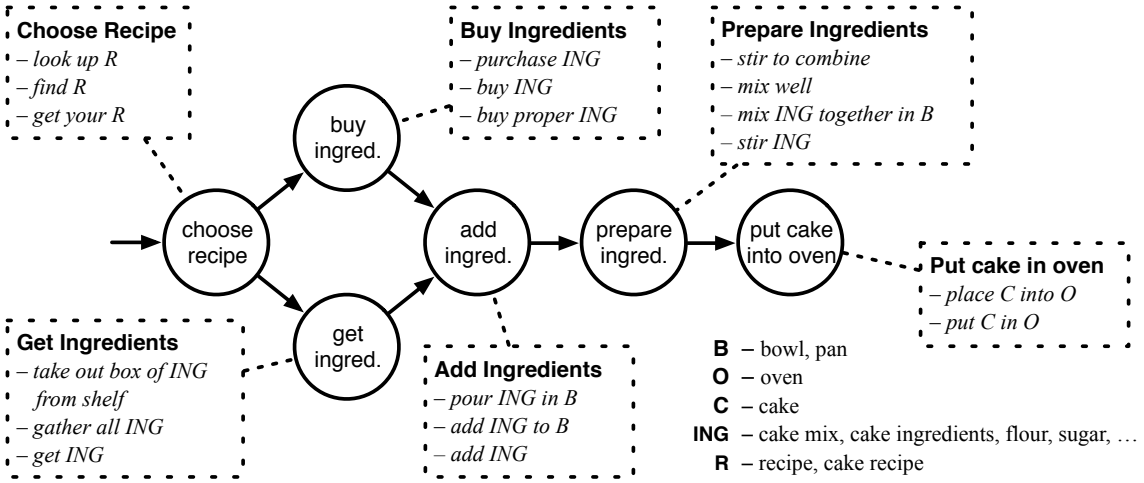


Figure 3.4: Example of an induced script structure for the `baking a cake` scenario.

is part of a larger research effort seeking to provide solid empirical basis for high-quality script modeling, including text-to-script mapping using more naturalistic texts (Ostermann et al., 2017).

Scenario	EDs		Gold
	annotated	excluded	sets
baking a cake	513	29	20
borrowing a book from the library	389	46	16
flying in an airplane	504	46	24
getting a haircut	418	52	23
going grocery shopping	505	95	18
going on a train	357	45	12
planting a tree	344	41	13
repairing a flat bicycle tire	363	40	16
riding on a bus	358	23	14
taking a bath	479	29	20
Total/Average	4230	446 (10.5%)	18

Table 3.4: Gold alignment annotation: the annotated EDs for each scenario, the excluded EDs for each scenario (singletons or unrelated events) and the number of gold paraphrase sets obtained.

Paraphrase sets that were singletons (e.g. in `baking a cake`, *return to oven* occurred only once) were considered not representative of the events in the scenario, and hence were removed based on the gold annotation. Likewise, event descriptions that were not related to the script (e.g., in `riding on a bus`, *pray your bus is on time*) or that were related to the scenario but not really part of the script (e.g., in `baking a cake`, *store any leftovers*

in the fridge), were also removed. In the end, approximately 10.5% of the annotated event descriptions were excluded. Table 3.4 indicates the number of excluded event descriptions and the number of gold clusters per scenario.

The result of the gold standard annotation is a rich resource of full alignments for all event descriptions in 500 ESDs (10 scenarios with 50 ESDs each) grouped in gold paraphrase sets, each containing different linguistic variations of the events in the scenario.

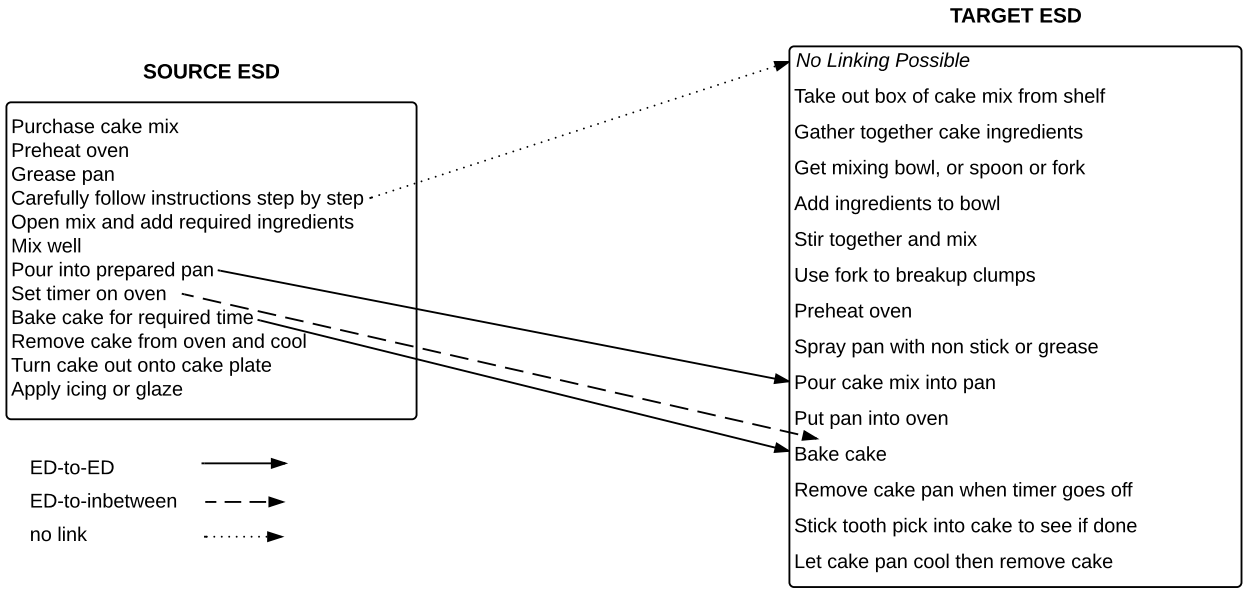
3.3 Alignment Annotation

A second step in the data collection enriched the ESD corpus with partial alignment information, to be used as seed data in semi-supervised clustering of event descriptions into semantically similar paraphrase sets (see Chapter 4 for details). The partial alignment information was also crowdsourced via M-Turk. We chose a representative set of 10 scenarios, with approximately 100 ESDs each, to be used in the alignment study.

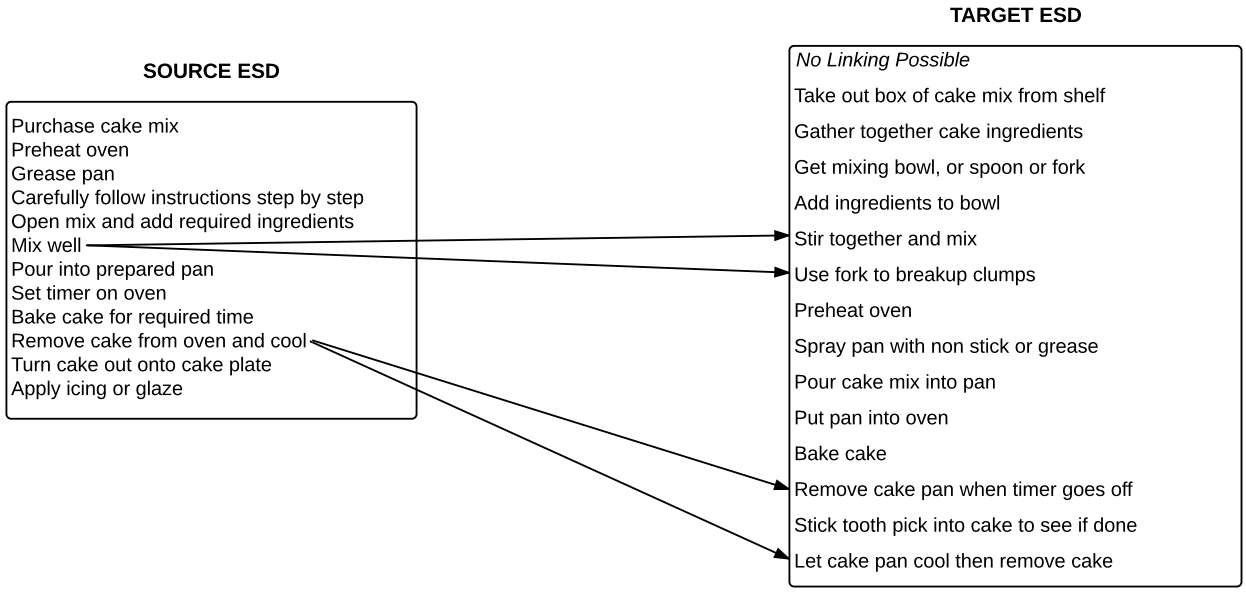
The workers were presented with a source and a target ESD and asked to link highlighted descriptions from the source ESD with those event descriptions from the target ESD that were semantically similar to those in the source ESD (see Figure 3.5). They had the option of either finding a **single-target** description in the target ESD or **multiple-target** descriptions.

Single target

In the simplest case, workers linked one event description in the source ESD to one event description in the target ESD (**ED-to-ED link**, e.g. *pour into prepared pan* $\rightarrow$ *pour cake mix into pan* in Figure 3.5a). If the target ESD did not contain any matching event description, the workers could either select a position between two event descriptions on the target side where the source event would usually take place (**ED-to-in-between links**, e.g., *set timer on oven* could take place between *put pan into oven* and *bake cake*), or they could indicate that no linking at all is possible (**no-links**). This latter option is useful in case of spurious events (e.g., *carefully follow instructions step by step* is not really an event) but also for alternative realizations of a script (e.g. *paying cash* vs. *with a credit card*).



(a) Single-target case



(b) Multiple-target case

Figure 3.5: Examples of possible annotations for the `baking a cake` scenario.

Multiple target

If the workers felt that the source event was described in more detail in the target ESD compared to the source ESD, they could link the source event description to more than one event description in the target ESD (e.g. *mix well* → *stir together and mix, use fork to break up clumps*, which can be broken down to two or more **ED-to-ED links**, see Figure 3.5b). Also when linking the source description to multiple descriptions in the target ESD,

workers could choose in-between positions (**ED-to-in-between links**).

Seed data selection

Davidson et al. (2006b) introduced measures for determining the quality of seed data. Of particular relevance are *informativeness* and *coherence*. Informativeness that determines if the chosen constraints contain extra information that the clustering algorithm cannot determine on its own. *Coherence* determines if the chosen constraints agree within themselves and are consistent with the true underlying similarity. They argue that highly informative and coherent constraints would provide the highest performance gain to the clustering algorithm. When constraints are informative, nearer data points that should not be in the same cluster have a cannot-link constraint between them while far apart data points that should be in the same cluster have a must-link constraint between them.

In order to minimize the amount of seed data that would be needed for semi-supervised clustering, we employed several criteria for choosing the most informative event descriptions to be aligned. Informative seeds are expected to have the strongest corrective effect on the induction algorithm. Thus, the event descriptions to be aligned should be the borderline cases that are the most difficult for the algorithm. In order to select the borderline cases, henceforth called *outliers*, we used two methods. First, we ran the Affinity Propagation clustering algorithm (Frey and Dueck, 2007) while varying the preference parameter that determines the cost of creating clusters, thus leading to different configurations of clusters (i.e. varying number of clusters and cluster sizes). As illustrated in Figure 3.6, decreasing the preference parameter increases the cost of creating clusters leading to a configuration with fewer clusters. Each item in the respective columns represents the cluster centers (as determined by AP).

We chose those event descriptions that changed their neighbors and cluster centers as the number of clusters increased or decreased. Consider Figure 3.7, on the left hand side, we have two small clusters, *take a number* has *make payment* and *listen for total price* as its nearest neighbors, and *make payment* is the cluster center. In an ideal case, when the cluster size increases, a well clustered item should maintain its neighbors. On the right hand side, when we decrease the preference parameter, the size of the clusters increase. *make payment* and *listen to total price* are still neighbors while *take a number* has changed its neighbors.

Preference : -1	**Preference : -0.04**	**Preference: 0**
walk into the restaurant	walk into the restaurant	walk into the restaurant
go to counter	go to counter	go to counter
look at menu board	look at menu board	look at menu board
decide what you want	decide what you want	decide what you want
wait in line	wait in line	wait in line
i order it	i order it	i order it
	get condiments	get condiments
pay	pay	pay / make payment
	take receipt	take receipt
wait	wait for order to be done	wait for food / wait for order to be done
get food	get food	get food
	sit down at table	sit down at table
take food to table	take food to table	take food to table
eat	eat	eat
	dispose of trash	dispose of trash
		return tray
	leave	leave

Figure 3.6: Varying preference parameter in Affinity Propagation

take a number is considered an outlier as the clustering algorithm has difficulty in properly clustering it.

Secondly, we chose those event descriptions that were not well clustered as measured by the Silhouette index (Rousseeuw, 1987), which takes into account the average dissimilarity of an item to the members of its cluster and its lowest average dissimilarity to the members of the other clusters. Consider Figure 3.8 the outlier data points (in red) have a higher average dissimilarity to members of the cluster. The stable data points have a lower average dissimilarity to members of the cluster. The two criteria used would select the most difficult cases for the algorithm, that is, cases whose true alignment is expected to be most informative. Figure 3.9 shows example clusters for `baking a cake` scenario and the found outliers. For instance, in the last cluster, which mainly consists of descriptions of the "putting cake in oven" event, *put the mix in the pans* and *store any leftovers in the fridge* clearly are outliers. The true alignment of such outliers are expected to have the strongest corrective effect on

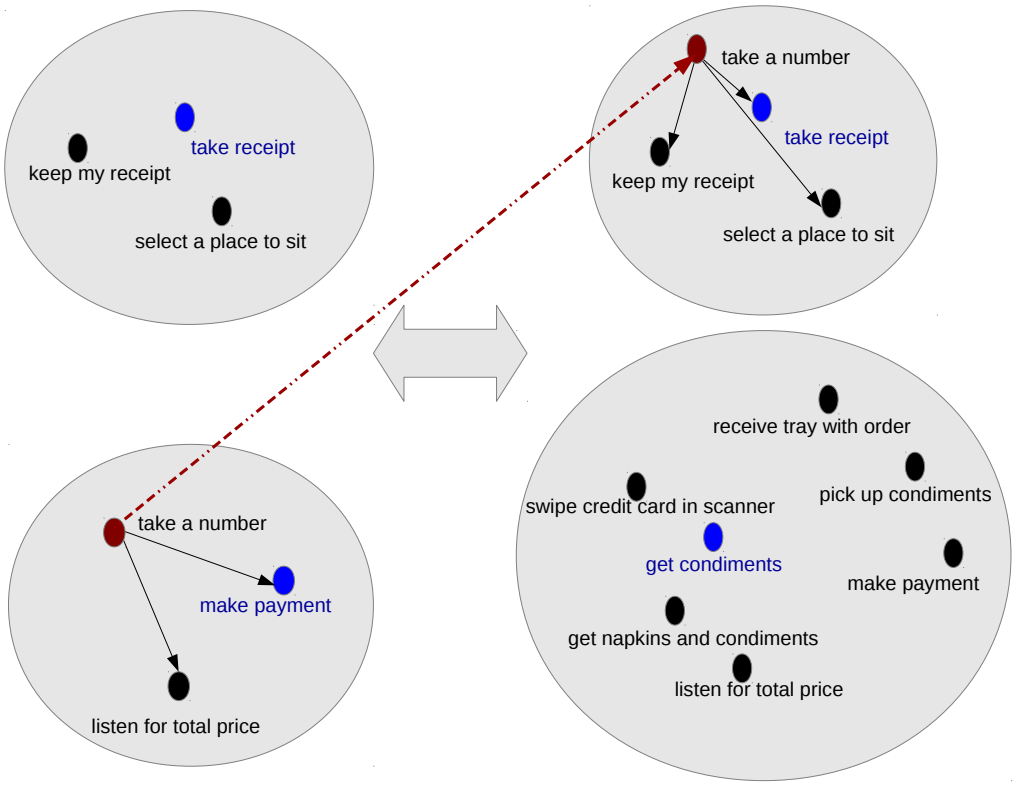


Figure 3.7: Choosing outlier and stable seeds: Varying preference parameter in Affinity Propagation

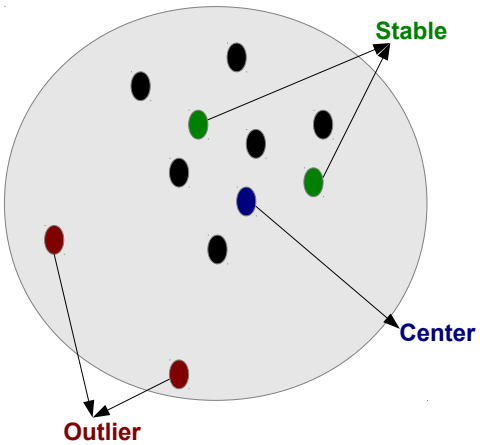


Figure 3.8: Choosing outlier and stable seeds: Silhouette index measure

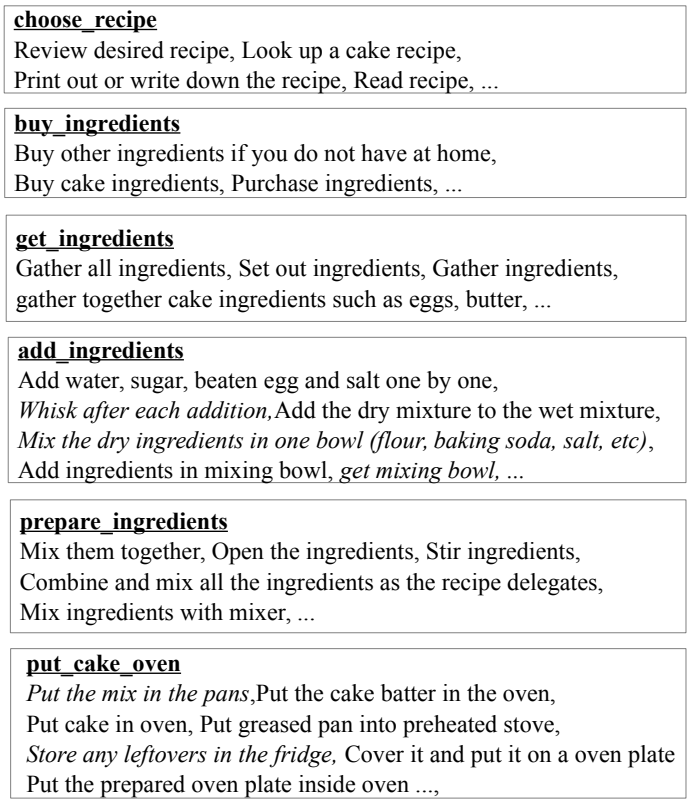


Figure 3.9: Example clusters (event labels are underlined, outliers are in italics).

the induction algorithm.

We also included event descriptions that were not outliers, henceforth called *stable cases*, that were used as a baseline when evaluating the inter-annotator agreement to show the difficulty of the outliers. The stable cases were randomly selected from those event descriptions that were not outliers. We expect more annotator agreement in stable cases as compared to the outliers. Approximately 20% of the event descriptions per scenario were selected as outliers and 10% were selected as stable cases and presented to the workers to align with another event[6].

Additionally, we included *gold seeds*, that is a small subset of the event descriptions aligned by experts as part of our gold standard annotation (see Section 3.2), to be used for an evaluation of the workers' annotation. 5% of the event descriptions per scenario were included as gold seeds. The workers were not aware of what alignments were outliers, stable cases or gold seeds.

Each source ESD containing outlier, stable or gold event descriptions was matched with 3

[6]Note that we refer to the number of descriptions per scenario. The annotated alignments are less than the 1% of all possible alignments (approx. 0.1-0.2%).

target ESDs. Each source-target ESD pair was annotated by 3 different annotators. In total approximately 600 outlier event descriptions, 300 stable event descriptions and 120 gold seeds were selected for each scenario and presented to workers for annotation. 292 workers (native speakers of English) took part in the annotation study. They were paid 0.35 USD per ESD and took on average 1.05 minutes per ESD.

Scenario	ED-to-ED links	ED-to-in-between links	No-links	Total links
baking a cake	1836	831	448	3115
borrowing a book from the library	1787	1146	254	3187
flying in an airplane	1676	1219	192	3087
getting a haircut	1887	926	292	3105
going grocery shopping	1863	997	251	3111
going on a train	1937	884	285	3106
planting a tree	1902	845	343	3090
repairing a flat bicycle tire	1500	1019	612	3131
riding on a bus	2226	750	114	3090
taking a bath	1898	891	314	3103
Total	18512	9508	3105	31125

Table 3.5: All links drawn for all source-target ESD pairs by all annotators. Note: the first column includes both ED-to-ED links from *single-target* cases and each single ED-to-ED link (arrow) in *multiple-target* cases.

3.4 Data Analysis and Evaluation

The alignment links we collected (Table 3.5) show an interesting degree of variability across scenarios and across the ESDs within a scenario. A high number of *ED-to-in-between links* and *no-links* shows that not all events are verbalized in every ESD for the same scenario: the source ESD may contain optional events which are either not explicitly mentioned in the target ESD (ED-to-in-between links) or event descriptions in source ESD that are not considered events in the scenario (no-links). Recall that workers could either find a single-target description in the target ESD or multiple-target description. They showed a strong preference for single-target links (which were chosen 30021 times) over multiple-target links (which were chosen 238 times).

We distinguish between *one-to-one* alignments, that is, cases which all three annotators considered to be *single-target* cases, since they used exactly one link between source and target ESD (including *in-between-links* and *no-links*), and *one-to-many* alignments, that is

cases which at least one annotator considered to be *multiple-target* cases, using more than one link between source and target ESD. Note that, as annotators preferred single-target links over multiple-target links, most source event descriptions are annotated as *one-to-one* alignments (Table 3.6).

Many workers were involved in the alignment annotation task and not all of them aligned the complete set of highlighted event descriptions. For this reason, we computed agreement as the number of times where the majority of workers agreed in an alignment instance, instead of the Kappa Fleiss (1971).

Scenario	**One-to-one alignments**			**One-to-many alignments**		
	tot	maj. agreem.	% agreem.	tot	overlap	avg Dice
baking a cake	1091	885	0.81	37	28	0.38
borrowing a book from the library	1081	718	0.66	55	30	0.4
flying in an airplane	1081	904	0.84	14	13	0.36
getting a haircut	1003	810	0.81	13	12	0.44
going grocery shopping	993	856	0.86	28	23	0.41
going on a train	982	810	0.82	34	32	0.45
planting a tree	1000	822	0.82	21	18	0.63
repairing a flat bicycle tire	991	708	0.71	29	17	0.2
riding on a bus	1074	914	0.85	28	25	0.46
taking a bath	1101	933	0.85	26	24	0.38
Total	10397	8360	0.80	285	222	0.41

Table 3.6: Number of source event descriptions that all workers annotated as single-target (*one-to-one* alignments), and number of descriptions where at least one chose the multiple-target option (*one-to-many* alignments), with majority counts and overlap counts.

One-to-one alignments

Figure 3.10 shows how well the workers agreed with each other in the annotation of the *outliers* and *stable cases* for *one-to-one* alignments. As expected, on average, workers tended to agree more in *stable cases* as compared to *outliers*. The relatively low agreement figures for `borrowing a book from the library` can be explained by the variety and complexity of the scenario: e.g. given a source event description *find the book on the shelf*, and *select a book* and *take the book off the shelf* on the target ESDs, it is not obvious if the source event is most similar to *find the book on the shelf*, *take the book off the shelf* or *in between* the two (Figure 3.11 (C)).

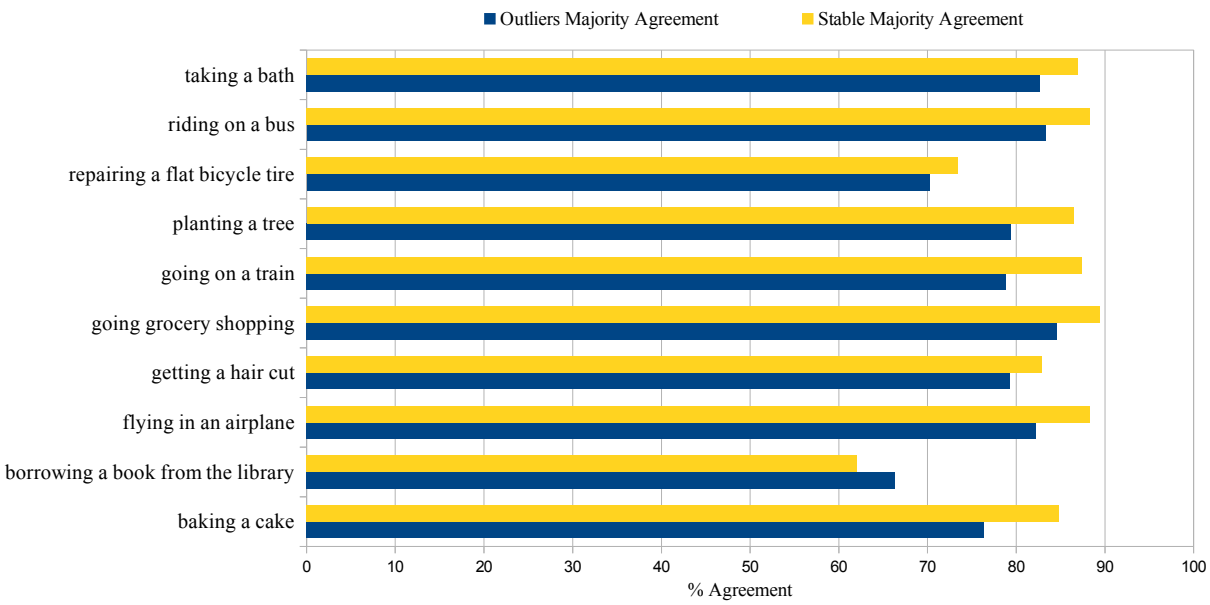
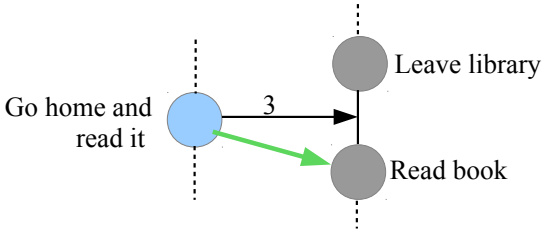
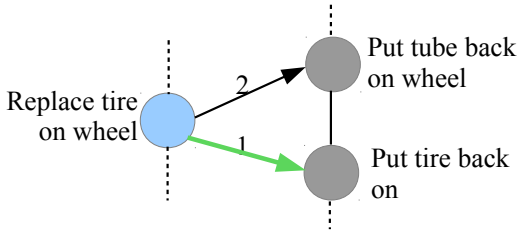


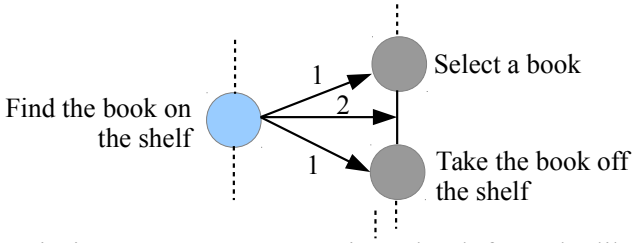
Figure 3.10: Worker agreement for *outliers* and *stable cases* in *one-to-one* alignments.



(A) All agree: Borrowing a book from the library



(B) Majority agreement: Repairing a bicycle



(C) Majority agreement: Borrowing a book from the library

Figure 3.11: Example alignments among workers (**green** $\longrightarrow$ represents gold alignment)

One-to-many alignments

In the *one-to-many* cases, 79% of event description instances have at least partially overlapping annotations, that is, there is an overlap in the alignments of two or all three annotators

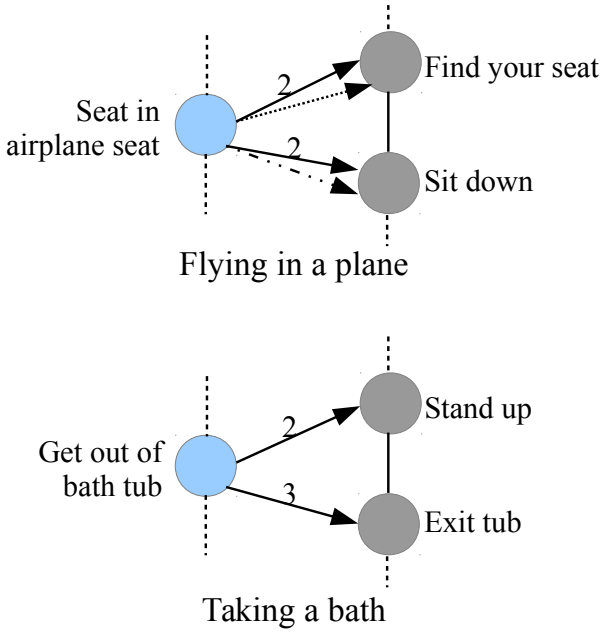


Figure 3.12: Example agreements for *one-to-many* cases

(see Table 3.6). We calculated the average Dice (indicating the degree of overlap between two sets) for all possible pairs of worker annotations for a given *one-to-many* alignment. The `borrowing a book from the library` scenario has the highest number of *one-to-many* alignments (55) and the highest number of non-overlapping alignments (25), and `repairing a flat bicycle tire` scenario has the lowest Dice score (0.2). The two scenarios are among those with the highest complexity and variability in how the script could be carried out.

The overall average Dice is 0.41, that is, the agreement is quite good on corresponding core events, although there may be disagreement about the precise sequence in the target ESD that corresponds to a given source event. For instance, in Figure 3.5b workers may agree on the corresponding core event (as *mix well* is semantically similar to *stir together and mix*), but they may not agree on the corresponding span, whether it is most similar only to *stir together and mix* or it also entails *use fork to break up clumps*. In Figure 3.12, workers could not agree on the span of *seat in airplane seat* in `flying in a plane` scenario. One worker aligned that *seat in airplane seat* spans from *find seat* to *sit down* while the remaining workers aligned *seat in airplane seat* to only *find seat* or *sit down*. In the second example, two workers agreed that *set out of bath tub* spans from *stand up* to *exit tub* in `taking a bath` scenario while the third worker only aligned *get out of bath tub* to

exit tub.

Gold seeds

Recall that besides the outliers (the difficult cases) and the stable cases (which were used as a baseline), we also included gold seeds for evaluation purposes, in order to compare the workers' annotation against expert annotation. Unsurprisingly, majority agreement in both *stable cases* and *gold seeds* was higher than agreement on *outliers* (87% for *gold seeds*, 82% for *stable cases* and 78% for *outliers*), showing that, while our outlier selection method effectively selects more challenging cases, the quality of the annotation is still very satisfactory.

Agreement between the worker's majority vote and the gold annotation was 81%. The cases where the workers did not agree with the gold annotation also illustrate the inherent complexity of the scripts. Figure 3.11 (A) and (B) illustrate difficult cases where the workers did not agree with the gold alignments. For example, in `borrowing a book from the library`, the workers aligned *take the book home* in the source ESD to the position between *leave library* and *read book* in the target ESD, while the experts aligned the same description to *leave library*. In `repairing a bicycle scenario`, it is not clear whether *replace tire on wheel* is only similar to *put tire back on* or also similar to *put tube back on wheel*. This shows that the workers were not necessarily wrong in all the cases where they did not agree with the *gold seeds*.

No-links

Interestingly, the workers tended to use *no-links* for those event descriptions that were not really events (e.g., in `baking a cake`: *carefully follow instructions step by step*) or event descriptions that were unrelated to the given scenario (e.g., in `riding on a bus`: *sing if desired*). The event descriptions annotated as *no-links* by the workers tend to overlap with those marked by experts as unrelated event descriptions in the gold standard. These cases typically involve event descriptions that are misleading and should not be part of the script. This shows that certain event descriptions are spurious, cases which we cannot expect a clustering algorithm to group in any meaningful way.

3.5 Comparison with the InScript Stories

As mentioned at the beginning of this chapter, this dissertation is part of a larger research effort where we seek to provide a solid empirical basis for high-quality script modeling. As part of this larger project, Modi et al. (2016) crowdsourced a corpus of simple scenario-related stories, the InScript corpus (Narrative Texts **In**stantiating **Script** structure). They asked workers on M-Turk to write down stories narrating recent real-life experiences instantiating specific scenarios (e.g. `eating in a restaurant`). The induced script structure from the ESDs will be used to investigate methods of text-to-script mapping, as well as to model the instantiation of script structures in naturalistic texts from the crowdsourced stories, as depicted in Figure 3.13.

Modi et al. (2016) created *script templates* that described script-specific event labels and participant labels for each scenario which were used to annotate the stories (e.g. event labels in `going to a restaurant`: *get_restaurant*, *take_seat, look_menu* and participant labels: *restaurant, waiter, menu*). They annotated event-denoting verbs in the stories with the event labels and participant-denoting NPs with the participant labels. Event labels used in the annotation of stories in the InScript corpus were harmonized with the gold paraphrase sets from the DeScript corpus (Section 3.2) to reach a one-to-one correspondence.

We compared the two resources with regard to their lexical variety, which is higher in the narrative texts than in the ESDs. We chose not to use the type-token ratio (TTR), as it is known to be sensitive to text length, and in this case the narrative texts are generally longer and would result in very low TTR values for the InScript data. Instead, we compared the lexical variety using the Measure of Textual Lexical Diversity (MLTD, McCarthy and Jarvis (2010)), which computes the mean length of word strings that are needed to maintain a set threshold level of lexical variation. We used a threshold TTR value of 0.71, which was empirically set by the authors (high MTLD corresponds to high lexical variety). We noted that the narrative texts have higher MTLDs across all scenarios, ranging between 40 and 47, as compared to ESDs with MTLDs ranging between 26 and 44 (Figure 3.14). That is, in the narrative texts more tokens are needed to reach the set TTR of 0.71; hence, the narrative texts are more lexically diverse in comparison to the ESDs.

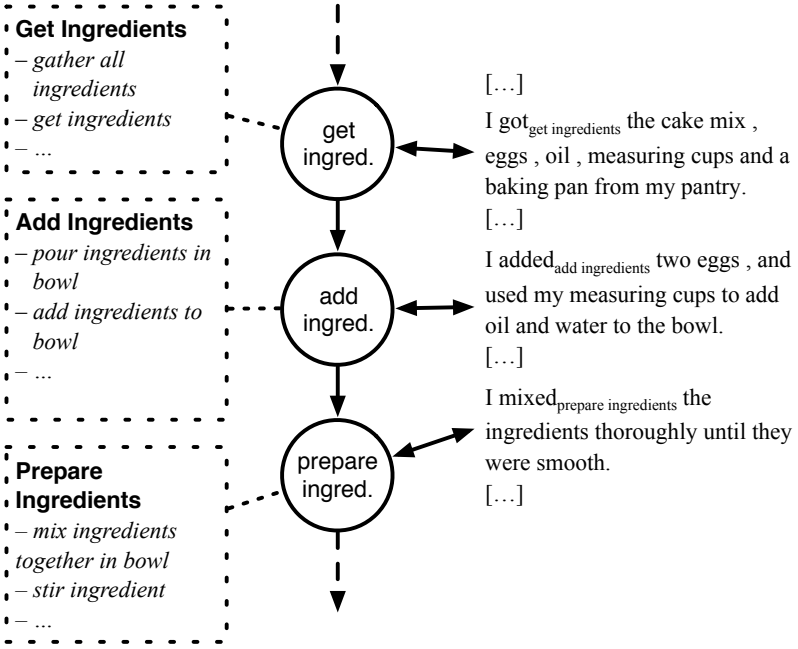


Figure 3.13: Connecting DeScript and InScript: an example from the BAKING A CAKE scenario (InScript participant annotation is omitted for better readability).

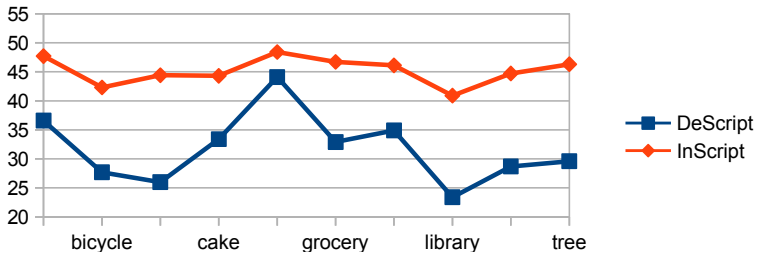


Figure 3.14: MTLD values for DeScript and InScript, per scenario.

As expected, the DeScript corpus, a collection of generic descriptions of script-related activities, has a lower lexical diversity compared to the InScript corpus, which in turn contains naturalistic texts describing real-life experiences.

3.6 Summary

We collected a corpus of 3,948 event sequence descriptions (40 different scenarios, approximately 100 different event sequence descriptions descriptions per scenario), ranging from simpler ones to ones that show interesting variation with regard to their granularity, to the

events described, and to different verbalizations of the same event within a scenario. The corpus, which is to our knowledge the largest collection of event sequence descriptions available, is enriched with partial alignment information on difficult event descriptions. *Multiple-target* annotations and *in-between* links are of particular interest, because they can capture differences between event descriptions in terms of granularity and optionality of events.

We also collected full alignments by experts for 10 different scenarios (50 event sequence descriptions per scenario), grouped into labeled paraphrase sets, to be used in the evaluation of semi-supervised clustering of event descriptions. We expect that the crowdsourced corpus and the gold standard alignment set will provide a sound basis for high-quality script models and will be used as a valuable resource for any task involving alignment and paraphrases of events.

Chapter 4

Script Induction

In this chapter, we present a method for script-learning by inducing script structure from the crowdsourced Event sequence descriptions (ESDs) from Chapter 3 . *Script induction* is about the methods to automatically induce the script representations from corpora. Consider the internal structure of a script as specified in Chapter 1: a script is composed of *events, participants* and the *relationship* between events. Accordingly, the induced script structure should be able to model the events, and participants in a script, and the temporal order among events. In addition, we also should model information about paraphrases. Paraphrases show the various linguistic realizations of events, and serve as a link between the script representation and the linguistic realizations.

Figure 4.1 illustrates the task of inducing the target script representation from ESDs. The extraction of structured script information from these descriptions can be viewed as the task of grouping event descriptions into paraphrase sets exploiting semantic and positional similarities, then inducing the script structure from the paraphrase sets. Semantically similar events are grouped into the same event cluster (e.g. in Figure 4.1: *mix well, stir together* and *mix in ingredients* are semantically similar). We can also rely on the prototypical order of events as given by the ESDs, thus we are able to learn the temporal order between events from the ESDs. This is different from narrative texts where we cannot always rely on the order of events in the narrative to by the typical order of events in the given script as the order of narration may be different.

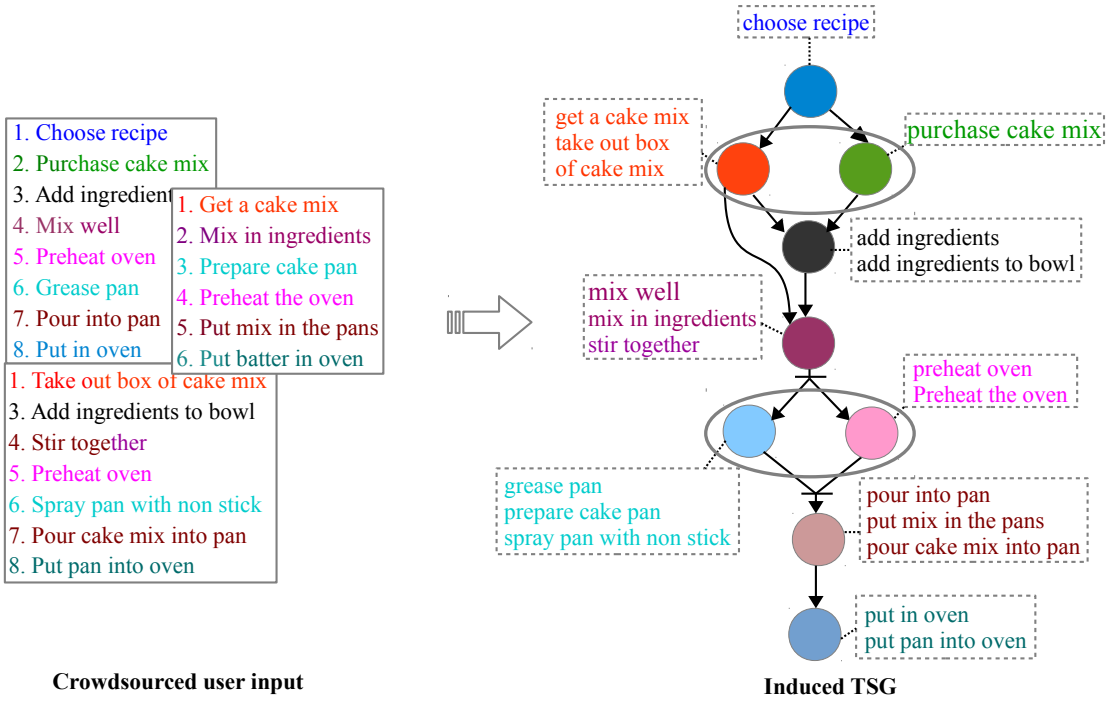


Figure 4.1: Example induced script structure from ESDs for `baking a cake` scenario

In this chapter, we provide empirical methods for inducing script structure from ESDs, relying on both semantic and positional information of the respective event descriptions. Event paraphrasing in the context of script modeling is challenging as event descriptions referring to the same script event can be semantically dissimilar. For example, in Figure 4.2, *board plane* and *walk up the ramp* are functionally similar in the `flying in a plane` scenario and should be grouped in the same paraphrase set. We will first build flexible methods for grouping functionally similar event descriptions into paraphrase sets and show that the quality of the built paraphrase sets is improved by including semi-supervised methods into our system. After building the paraphrase sets, we shall exploit the temporal information provided in the ESDs to induce temporal order between paraphrase sets and build a graph structure in terms of script graph as a full representation of the script.

This chapter proceeds as follows: Section 4.1 gives background and related work on script induction from ESDs. Section 4.2 gives an introduction and motivation to clustering as an alternative method for script induction. Section 4.3 describes in detail our semi-supervised model for script induction from event sequence descriptions. In Section 4.4, we describe the data we used in the experiments, which is based on the data collection provided in Chapter 3. We evaluate our model and compare it to previous work in Section 4.5. In Section 4.6

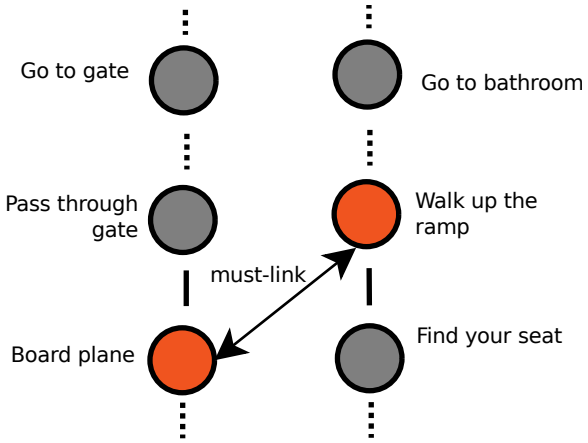


Figure 4.2: Example script-specific semantic similarity in `flying in an airplane` scenario

we address the concern of scalability of our methodology by providing an assessment of the coverage of existing script resources, and an estimate of the concrete costs for their extension.

4.1 Background on Script induction from ESDs

Several methods for the automatic acquisition of script knowledge have been proposed. Seminal work by Chambers and Jurafsky (2008, 2009) provided methods for the unsupervised wide-coverage extraction of script knowledge from large text corpora. The extracted events do not have paraphrase information on the various ways the given event can be linguistically realized. The models produce verb sequences plus dependency information not related to a specific script or scenario. Additionally, we can also not rely on the temporal order information given in the texts as events in texts are not necessarily in the order in which the events in the script occur. Texts typically only mention small parts of a script, banking on the reader's ability to infer missing script-related events. The task is therefore challenging, and the results are quite noisy (see Chapter 2 for an overview of previous work). Building robust script parsers require information about the relation of events and participants to a given scenario and the paraphrase information of script events and participants.

The starting point of out work is the approach proposed in Regneri et al. (2010) (henceforth "RKP"). The collected event sequence descriptions provide generic descriptions of a given

scenario in concise telegram style (see Section 2.3 for a description of the corpus). Regneri et al. (2010) used Multiple Sequence Alignment (MSA, Durbin et al. (1998)) to induce script structure by aligning semantically similar event descriptions across different ESDs. Sequence alignment is the task of re-writing a given sequence into another sequence. The re-writing process includes deletion and insertion of items in one sequence in order to transform it into another sequence. In can also be viewed as the task of aligning points shared by the two sequences while indicating points that need insertion or deletion. Similar points in the two sequences are aligned and zero elements representing possible insertions or deletions are used as place holders for the non-aligned points. The left side of Figure 4.3 shows how semantically similar events are aligned using MSA and zero elements used to represent event descriptions that are missing in some of the ESDs.

Roughly speaking, the paraphrase sets gained from the alignment step correspond to the script's event types, while their default temporal order is induced from the order of the event descriptions in the ESDs. Based on these paraphrase sets, RKP extracted high-quality script knowledge for a variety of different scenarios, in the form of Temporal script graphs (TSGs). Temporal script graphs are partially ordered structures whose nodes are sets of alternative descriptions denoting the same event type, and whose edges express temporal precedence. RKP derived TSGs from the results of the MSA by adding an edge from an event cluster a to another event cluster b, if at least there was some ESD where an event description in a directly happened before an event description in b. They further performed a post processing step, pruning out nodes that were too sparse and merging duplicate nodes, in order to simplify the built TSG. The right side of Figure 4.3 shows a picture of the temporal script graph induced from the paraphrase clusters from the alignment step.

The choice of MSA was motivated by the effect of positional information on the detection of scenario-specific paraphrases: event descriptions occurring in similar positions in ESDs tend to denote the same event type. The precision of MSA-based script extraction is impressive, but MSA has a fundamental drawback. MSA makes far too strong an assumption about the temporal ordering information in the ESDs. It does not allow for crossing edges and thus must assume a fixed and invariable event order. Script events are temporally ordered by default, but the ordering of events in a script is to some degree flexible. For example, when baking a cake, one can *preheat the oven* before or after *greasing the pan*. MSA does not allow for crossing alignments, and thus is not able to model order variation:

s1	s2	s3
enter restaurant	∅	enter restaurant
go to counter	walk to counter	∅
∅	∅	look at menu
make selection	∅	decide what you want
place an order	place order	order at counter

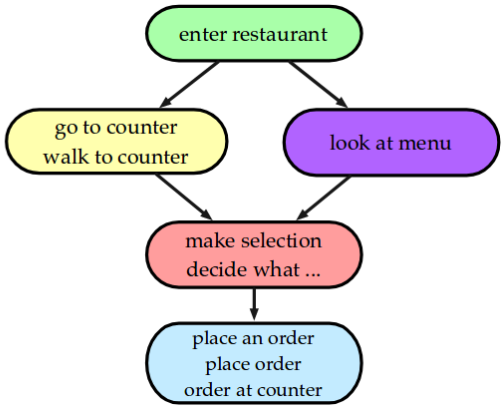


Figure 4.3: Example induced TSG from ESDs for `eating in a fastfood restaurant` scenario using MSA (Regneri, 2013)

this leads to an inappropriately fine-grained inventory of event types.

Lastly, a main concern with the approach in RKP is scalability: temporal script graphs are created scenario-wise in a bottom-up fashion. They represent only fragments of the rich amount of script knowledge people use in everyday communication. In Section 4.6 we address this concern with the first assessment of the coverage of existing script resources, and an estimate of the concrete costs for their extension.

While RKP employ Multiple Sequence Alignment (MSA), we use a *semi-supervised clustering approach* for script structure induction (see Section 4.3 for details). We propose clustering as an alternative method to overcome the rigidity of the MSA approach. We use a distance measure based on both semantic similarity and positional similarity information, making our clustering algorithm sensitive to ordering information, while allowing for order variation in the scripts. In the next Section, We provide an overview of clustering as a method for script induction and motivate our semi-supervised clustering approach to script induction from ESDs.

4.2 Clustering

As noted before, script events are temporally ordered by default, but the ordering of events is to some extent flexible. For example, in Figure 4.4 one can *preheat the oven* before or after *greasing the pan* when `baking a cake`. Likewise one can *put pan on burner* before or after *cracking the eggs* when `making scrambled eggs`, or *put filter* and *coffee beans in coffee maker* before or after *adding water to reservoir* when making coffee. The extraction

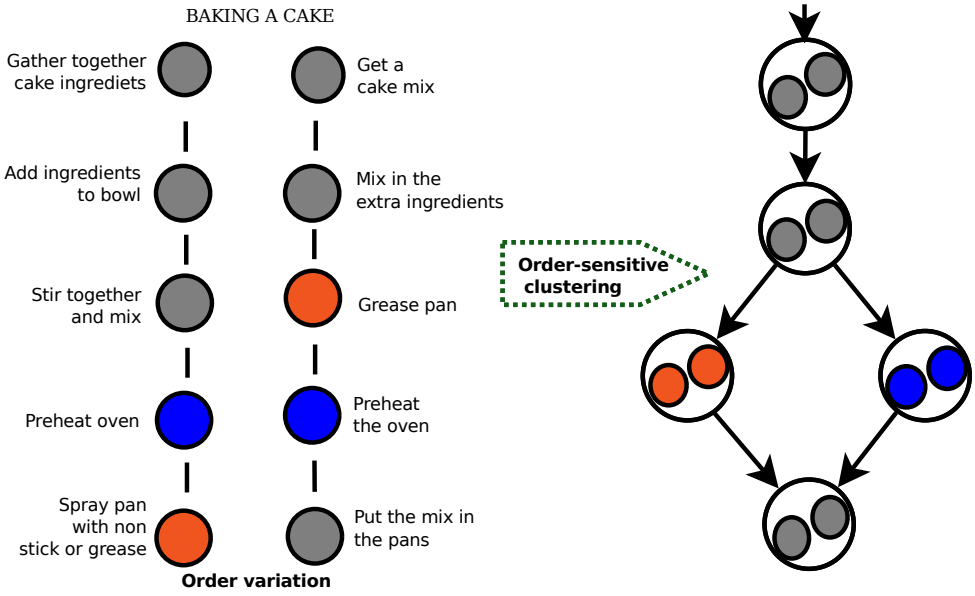


Figure 4.4: Example modeling of order variation for `baking a cake` scenario

of script information from ESDs can be seen as the task of grouping event descriptions referring to the same script event into paraphrase sets, then inducing the script structure from the paraphrase sets. We can therefore formulate the first task in script induction from ESDs as a *Clustering* task. Clustering algorithms offer a less rigid approach to script induction using both semantic and positional similarity information. By adding positional information as an additional feature, clustering can be made sensitive to order information without strictly imposing the temporal ordering constraints.

Figure 4.4 illustrates the task of using clustering algorithms for script induction, resulting in event descriptions expressing the same script event being grouped into event clusters, i.e. paraphrase sets. Clustering algorithms can be made sensitive to ordering information, but do not use it as a hard constraint, and therefore allowing for an appropriate treatment of order variation.

We have experimented with several clustering algorithms. Figure 4.5 shows an example output of the best-performing algorithm, Affinity Propagation (AP, Frey and Dueck (2007)). The outcome is still quite noisy. For instance, in the last cluster, which mainly consists of descriptions of the *putting cake in oven* event, *put the mix in the pans* and *store any leftovers in the fridge* clearly are outliers. The noise is to some degree due to the complex nature of script structures in general, but it is also the price one has to pay for a gain

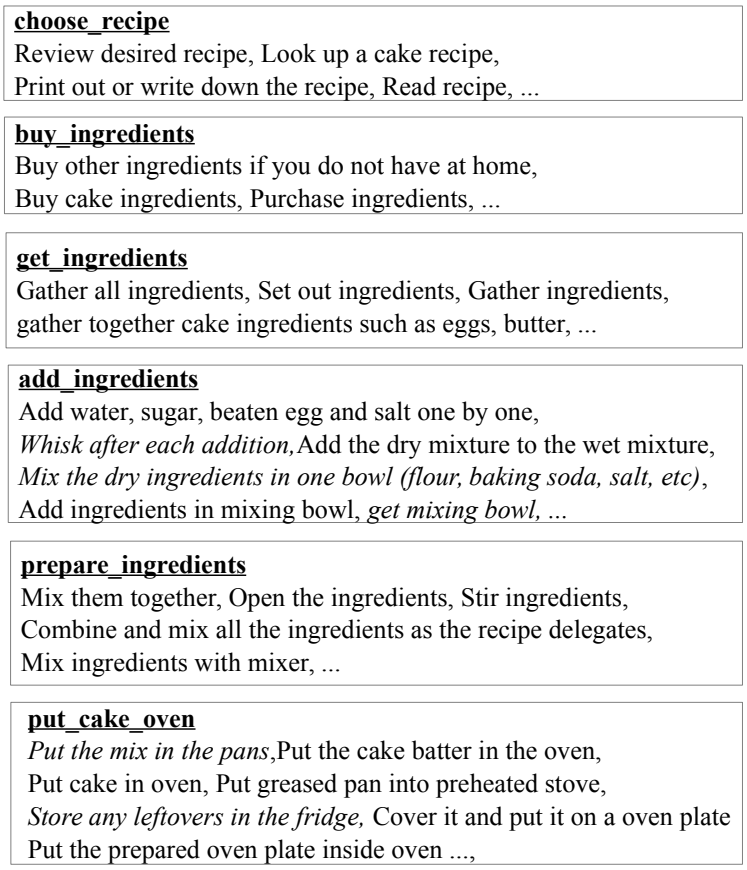


Figure 4.5: Example clusters (event labels are underlined, outliers are in italics).

in flexibility of event ordering.

Clustering accuracy depends on the reliability of similarity estimates, but scenario wise paraphrase relations are often based on scenario-specific functional equivalence, which cannot be easily determined using semantic similarity, even if complemented with positional information. Consider Figure 4.2 (highlighted before), in `flying in a plane` scenario, it is challenging for any semantic similarity measure to predict that *walk up the ramp* and *board plane* are functionally similar with respect to the given scenario and should be put in the same event cluster. In order to improve clustering performance, we can exploit semi-supervised methods for clustering.

Semi-supervised clustering

Semi-supervised clustering involves including a small number of *label* information into the clustering process. Labeled information mainly comes in two flavors:

- Partial categorized data - in this case, the final set of cluster labels is known in ad-

vance and the true cluster assignments for some data points are known. The task is to group the remaining data points into the appropriate clusters using the already known partial assignments.

- Partially constrained data - in this case, the constraints among pairs of data points are known. The constraints can be expressed in terms of *must-link* constraints (if they are believed to belong to the same cluster) denoted by $M = (i, j)$ or *cannot-link* constraints (if they are believed to belong to different clusters) denoted by $C \neq (i, j)$. Such partially constrained data points are also referred to as *instance-level* relational constraints.

For the event paraphrasing task, we do not know the final set of event cluster for a given scenario at the beginning, and therefore partially categorized data cannot be straightforwardly obtained. We opted for the latter approach as it is appropriate for crowdsourcing. It is intuitive to ask workers to indicate which pairs of event descriptions are similar (or not) rather than asking them for class labels, thus its cheaper and faster to get constraints between any two event descriptions.

Semi-supervised clustering has be applied to different real-world problems, such as image segmentation Givoni and Frey (2009); Asafi and Cohen-Or (2013), detection of population stratification Liu et al. (2013), among others. Previous work on clustering (Wagstaff and Cardie, 2000; Klein et al., 2002; Givoni and Frey, 2009; Asafi and Cohen-Or, 2013) has also shown that clustering performance can be enhanced by incorporating a small number of instance-level relational constraints. Instance-level constraints are often fast and cheap to obtain and can sometimes be automatically collected. The choice of which instance-level relational constraint to use is particularly crucial. Klein et al. (2002) showed that heuristically guided selection of constraints perform better than a random selection of constraints.

Must-link constraints are inherently transitive (Klein et al., 2002). That is, if i and j are aligned to be most similar, $M = (i, j)$, and j and k are likewise aligned, $M = (j,k)$, then it follows that i and k must also be similar, $M = (i,k)$, and (i, j,k) for an equivalence set. On the other hand *cannot-link* constraints are non-transitive and their implications are not as easy to interpret (Klein et al., 2002; Asafi and Cohen-Or, 2013). Due to such differences between *must-link* and *cannot-link* constraints, they should be incorporated differently.

Instance-level relational constraints can be incorporated into clustering in various ways.

Davidson and Basu (2007) describes two broad approaches of incorporating instance-level relational constraints:

- constraint-based- where we modify the standard unsupervised clustering algorithms to account for the constraints directly. The constraints are used to "bias the search for an appropriate clustering of the data" (Davidson and Basu, 2007).
- distance-based - where we adapt the similarities (distances) between the data points to better align with the constraints, and then apply an unsupervised clustering algorithm on the adapted similarities.

In this dissertation, we chose to use the latter approach of adapting similarities. This allows us to be more flexible and much more modular in optimizing the distance adaptation step independent of the clustering algorithm being used. The existing crowdsourced alignments could also contain conflicting constraints, i.e. a pair of event description could have both a *must-link* and a *cannot-link* constraint. Hence, we do not want to strictly enforce the crowdsourced constraints.

Klein et al. (2002) include the *must-link* constraints by shortening the distance between pairs of constrained data points, and then applying the all-pairs-shortest-paths algorithm on the adapted matrix. (more details of Klein et al. (2002) in the next section). For *cannot-link* constraints, they set the distance between the pairs of constrained data points to a large constant number and used a proximity based clustering algorithm to indirectly propagate the *cannot-link* constraints. Asafi and Cohen-Or (2013) also adopt a similar method for handling *must-link* constraints and propose a method that treats *cannot-link* constraints as additional features. They employed the idea of diffusion maps (Nadler et al., 2005) to compute new distances between data points that account for *cannot-link* constraints, and incorporate the new distances to the original distance matrix as additional features (more details of Asafi and Cohen-Or (2013) in the next section).

We adopt a similar method for handling *must-link* constraints. Klein et al. (2002) used a proximity-algorithm, Constrained Complete-Link, CCL. In CCL one is required to give the number of clusters in advance. Instead, We used Affinity Propagation (Frey and Dueck, 2007) that does not require one to specify the expected number of clusters in advance, but rather uses a preference parameter that determines the cost of creating clusters to discover

the most appropriate number of clusters. For the cannot-link constraints, We use a similar approach to (Asafi and Cohen-Or, 2013) (see Section 4.3 for the proposed model).

We automatically identify event descriptions that are likely to cause alignment problems (called *outliers*), crowdsource alignments for these items and incorporate them as instance-level relational seeds into the clustering process. The alignments are in the form of *must-links*, that is, event descriptions that are semantically similar are aligned (see Section 3.3 for details of the alignment collection).

4.3 Semi-supervised script induction

Our script-induction technique extends previous work, first, by using semi-supervised techniques to handle script-specific semantic similarity, second, by proposing a flexible model to better handle order variation in scripts and third, by extending previous script representation formalism, Temporal Script Graphs, by incorporating "arbitrary order" equivalence classes to capture event types with variable order and optionality that is inherent in scripts.

I first present a semi-supervised clustering method to induce script events from ESDs using crowdsourced alignments as seeds (Section 4.3.1). In Section 4.3.2, I describe how we calculate the underlying distance matrix based on semantic and positional similarity information. Later in Section 4.3.3, I describe the induction of temporal order for the script events, which turns the set of script events into a temporal script graph (TSG).

4.3.1 Incorporating relational seeds in semi-supervised clustering

We use the crowdsourced alignments between event descriptions, (see Section 3.3 for details of the alignment collection), as instance-level relational seeds for clustering, more specifically as *must-link constraints*, requiring that the linked items should go into one cluster. We incorporate the constraints into the clustering process following the method in Klein et al. (2002): that is adapting the input distance matrix in a pre-processing step, rather than directly integrating the constraints into the clustering algorithm. This makes it possible to try different adaptation strategies, independently of the specific clustering algorithm, and the adapted matrices can be straightforwardly combined with the clustering algorithm of choice.

Adding must-link constraints

Klein et al. (2002) handle must-link constraints by modifying the input matrix D in the following way: if two instances i and j are linked by a must-link constraint, then the corresponding entry $D_{i,j}$ is set to zero, which forces i and j to be grouped into the same cluster by the underlying clustering algorithm. In addition, distance scores for instances in the

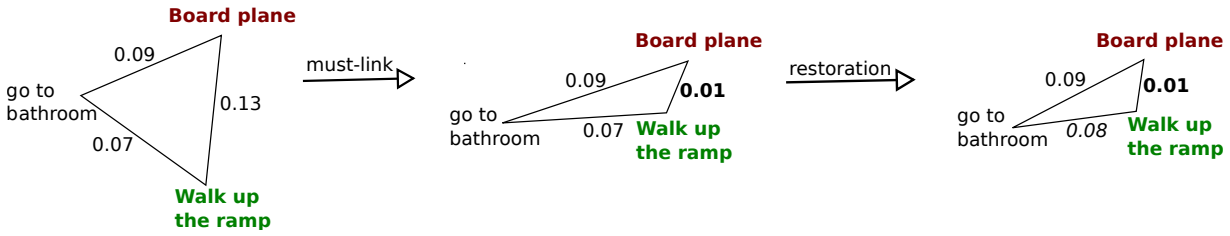


Figure 4.6: Triangular-inequality violation

neighborhood of i or j are affected: if the distance is reduced for one pair of instances, triangle-inequality may be violated. Triangle-inequality states that the sum of any two sides of a triangle should be greater than the third side. Consider Figure 4.6, once the distance between *walk up the ramp* and *board plane* is reduced, triangle-inequality does not hold anymore, ($0.07 + 0.01 = 0.08$, is less than 0.09). An all-pairs-shortest-path algorithm propagates must-link constraints to other instances in D that restores triangle inequality ($0.08 + 0.01 = 0.09$, is equal to 0.09).

We use a modified version of Klein et al. (2002) approach. First, as the crowdsourced information may not be completely reliable, the clustering algorithm should be able to override it. We thus do not set $D_{i,j}$ to zero but rather to a small constant value d, that is the smallest non-identity distance value occurring in the matrix (Algorithm 1, lines 7 to 10). Second, we exploit the inherent transitivity of paraphrase judgments to derive additional constraints: if (i, j) and (j, k) are must links, we assume the pair (i, k) to be a must link as well, and set the distance to d (Algorithm 1, line 1). After the additional constraints are derived, the all-pairs-shortest-path algorithm is applied to the input matrix as in Klein et al. (2002) (Algorithm 1, line 11).

We experimented with various state-of-art clustering algorithms including Spectral Clustering and Affinity Propagation (AP). The results presented in Section 4.5 are based on AP, which proved to be most stable and provided the best results.

Algorithm 1: Imposing and propagating *must-link* constraints

Input: $n \times n$ distance matrix D
Input: *must-link* constraints M
Output: Adapted $n \times n$ distance matrix D'

1 $M' \leftarrow$ transitive closure of M
2 $A \leftarrow \emptyset$
3 **for** $pair(i, j)$ ***in*** M' **do**
4 $\quad A_i \leftarrow nn(i, t_{mindist}, t_{minsize}, t_{maxsize})$
5 $\quad A_j \leftarrow nn(j, t_{mindist}, t_{minsize}, t_{maxsize})$
6 $\quad A \leftarrow A \cup A_i \times A_j$
7 $d \leftarrow$ smallest non-zero value in D
8 **for** $pair(i, j)$ ***in*** $M' \cup A$ **do**
9 $\quad D_{i,j} \leftarrow d$
10 $\quad D_{j,i} \leftarrow d$
11 $D' \leftarrow$ All-pair-shortest-path(D)
12 **return** D'

Incorporating Nearest Neighbors We also wanted to automatically derive more constraints and experimented with incorporating constrains derived from data points that are close to points in a must-link relationship. When an instance i is linked to j by a must-link constraint, we derive additional must-link constraints for instances which are very close to i or j. This is to make sure that close instances do not end up in different clusters. This could happen, if one of the instances in the must-link is an outlier, which is very dissimilar to the other one and its closest neighbors. (Algorithm 1, lines 3 to 6)

For each pair i and j in a must-link constraint, the algorithm derives additional must-link constraints by computing the cross-product of the set of nearest neighbors A_i and A_j of i and j, respectively (Algorithm 1, lines 3 to 6). The computation of nearest neighbors is dependent on three parameters: a distance threshold $t_{maxdist}$ that makes sure that only instances which are very close to the target instance are returned, and two thresholds $t_{minsize}$ and $t_{maxsize}$ that constrain the admissible number of instances to be returned. In our experiments, we obtained best results for a maximal distance of 0.09, a minimal size of 2 and a maximal size of 7. If at least 2, but at most 7 neighbor instances lie within the minimal distance, the set of neighbors is returned unchanged. The $t_{minsize}$ parameter is motivated by the possibility that the semantic closeness of single instances is not sufficient as evidence for the presence of a natural cluster: *enter the airport* and *enter the plane* may be semantically similar, but describe different events. In this case, the empty set is returned. The

$t_{maxsize}$ parameter limits the maximal number of instances to be returned, to avoid that the influence of individual noisy must-links becomes too strong.

Adding cannot-link constraints

Algorithm 2: Imposing and propagating *cannot-link* constraints

Input: n by n distance matrix $\acute{D}$
Input: $\alpha \leftarrow$ getAlpha($\acute{D}$)
Input: Indices of event description $A = \{a_1, a_2, \ldots, a_n\}$
Input: *cannot-link* constraints
1 **for** $pair(i, j)$ ***in*** CL **do**
2 $NN \leftarrow$ get nearest neighbors(i, j, A)
3 $DD \leftarrow$ getDiffusionDistance($\acute{D}$)
4 $\vec{V} \leftarrow$ calculateVectorV(i, j, A)
5 $CD \leftarrow$ calculateConstrainedDistance($\vec{V}$)
6 $\hat{D} \leftarrow$ adaptDistance($\alpha, \acute{D}, CD, NN$)
7 **return** $\hat{D}$

Asafi and Cohen-Or (2013) introduced an interesting model of adding *cannot-link* constraints by augmenting the D' matrix with additional coordinates that respect the *cannot-link* constraints. Each additional coordinate is derived from *cannot-link* constraint. Our method for adding *cannot-links* is based on a similar methodology. In our model, we focus on adapting the similarities of the data points that are nearest to points in a cannot-link relationship, as our goal is to get more crisp cluster-boundaries. Equation 4.1 shows our model.

$$\hat{D}^p_{i,j} = \acute{D}^p_{i,j \in NN} + \sum_{c=1\ldots n} (\alpha . D^{(c)}_{i,j})^p \tag{4.1}$$

where $\acute{D}$ is the output matrix after adding *must-link* constraints and α is a scale parameter used to determine the how much influence the *cannot-link* constraints have on $\acute{D}$. α can range between the smallest and largest distance in the input distance matrix. $D^{(c)}_{i,j})$ is the additional coordinate derived from each *cannot-link* constraint pair (added one at a time) and is computed as:

$$D^{(c)}_{i,j}) = |v_i - v_j| \tag{4.2}$$

$$v_i = \frac{\varphi(i,c_2) - \varphi(i-c_1)}{\varphi(i,c_2) + \varphi(i,c_1)} \tag{4.3}$$

where c_1 and c_2 are the event descriptions in the *cannot-link* relationship, and i is the index of all other data points not in this *cannot-link* relationship. In vector $\vec{V}$, the event descriptions in the *cannot-link* relationship, c_1 and c_2, are assigned extreme values of 1 and -1. Thus points closer to c_1 get values closer to 1 and points closer to c_2 get values closer to -1.

The φ function is the distance between points in the Diffusion Map (Nadler et al., 2005), which we call diffusion distance, *DD*, derived from $\acute{D}$ and is given as $\varphi(x,y) = |\Psi_t(x) - \Psi_t(y)|$. Diffusion maps are a low dimensional representation of a distance matrix by using the first few k eigenvectors and eigenvalues. We used the R package $diffusionMap()$ to calculate n k-dimensional coordinates, $\Psi_t(x) = (\lambda_1^t\psi_1(x), \lambda_2^t\psi_2(x), \ldots, \lambda_k^t\psi_k(x))$, for each n in $\acute{D}$, where λ_1^t are the eigenvalues and $\psi_i(x)$ are the eigenvectors of $\acute{D}$'s Diffusion map.

Determining the number of event clusters.

AP uses a parameter p, which influences the cluster granularity without determining the exact number of clusters beforehand. There is considerable variation between the optimal number or clusters between scenarios, depending on how many event types are required to describe the respective activity patterns. We use an unsupervised method for estimating scenario-specific settings of p, using the mean Silhouette Coefficient (Rousseeuw, 1987). This measure balances optimal inter-cluster tightness and intra-cluster distance, making sure that the elements of each cluster are as similar as possible to each other, and as dissimilar as possible to the elements of all other clusters. We run the unsupervised AP algorithm for each scenario with different settings of p and select the number resulting in the highest total Silhouette Coefficient as the optimal value for p.

4.3.2 Similarity Features

First we present the structure that we use to represent an event in a given event description, followed by an explanation of how we combine semantic and positional similarity information to obtain the distance measure that capture the similarities between event descriptions.

Event representation

In previous work, events have been denoted using various representations. The most notable ones are using single verbs or verbs with one or two arguments (Chambers and Jurafsky, 2008, 2009; Regneri et al., 2010; Modi and Titov, 2014). Verbs play a central role as the main trigger of script events . Multi-argument event representations have been shown to be richer, and captures better the interplay among events in the same scenario (Pichotta and Mooney, 2014). We also represent events using a multi-argument event representation that includes the particle of the verb, only if the combination of verb-particle is found in WordNet[1] as a compound verb. In ESDs, the subject is mostly left implicit and is normally the main protagonist, thus not included in the event representation as it does not provide any new information. We include the main verb of an event description, and direct object, and indirect object(s) and the particle of the verb i.e. *verb_{particle}*{direct object, indirect object(s)*}*. Figure 4.7 shows how event descriptions in an ESD from `baking a cake` scenario would be represented.

#	**Event description → Event representation**
1	Getting ingredients ready → get (ingredients)
2	Put all ingredients in bowl → put_in (ingredients, bowl)
3	Stir all ingredients until well mixed → stir (ingredients)
4	Pour mixture into a greased pan → pour_into (mixture, pan)
5	Put greased pan into preheated stove → put_into (pan, stove)
6	Set timer → set (timer)
7	Bake cake → bake (cake)

Figure 4.7: Example event representations for `baking a cake` scenario

RKP noted common verbs that appear in many event descriptions and have lower semantic contribution, e,g. *get, turn, put, go*, etc. For such verbs, the presence of the direct and / or indirect object helps indicate the event that is taking place e.g *get ingredients* in `baking a cake` scenario and *get towel* in `taking a shower` scenario. Also, the meaning of the head verb changes with the inclusion of particle, e.g. *put in , put off, put on,* etc.

We parse each event description using an adapted parser by RKP, and lemmatize the verbs and nouns. RKP adapted the parser by training the parser on sentences with the subject

[1] www.wordnet.princeton.edu/

omitted, to mimic the subject-less sentences in event descriptions. For spell correction, we restrict the output of the spell checker to the vocabulary used in the given scenario. RKP showed that this is useful for resolving ambiguities and guiding the spell checker. We used a simple model for co-reference resolution that takes advantage of the redundant nature of the ESDs corpus, (Bloem et al., 2012). We compute selectional preferences from the whole corpus to determine of the selectional preferences of the head verbs. During co-reference resolution, we consider only the *n* closest preceding nouns as candidate antecedents. For our experiments, setting *n* to at 4 gave us the best results. Some event descriptions are complex i.e. two or more verbs. We did not perform any event splitting as few event descriptions (less than 10 %) were complex enough to warrant further preprocessing.

We use the described event representation for all similarity features, apart from vocabulary and positional similarity.

Semantic Similarity

We inspect different models for word-level similarity, as well as methods of deriving phrase-level semantic similarity from word-level similarity. We use pre-trained Word2Vec (w2v) word vectors (Mikolov et al., 2013) and vector representations (rNN) by Tilk et al. (2016) to obtain word-level similarity information. The rNN vectors are obtained from a neural network trained on large amounts of automatically role-labeled text and capture different aspects of word-level similarity than the w2v representations. We also experimented with WordNet/Lin similarity (Lin, 1998), but an ablation test (Table 4.1) showed that it was not useful.

To derive phrase-level similarity from word-level similarity, we employ the following three different empirically informed methods:

Centroid-based similarity. This method derives a phrase-level vector for an event description by taking the centroid over the word vectors of all content words in the event description. Similarity is computed using cosine.

Alignment-based similarity. Following RKP, we compute a similarity score for a pair of event descriptions by a linear combination of (a) the similarity of the head verbs of the two event descriptions and (b) the total score of the alignments between all noun phrases in the

two descriptions, as computed by the Hungarian algorithm (Papadimitriou and Steiglitz, 1982).

Vocabulary similarity. We use the approach in Fernando and Stevenson (2008) to detect paraphrases and calculate semantic similarities between two event descriptions p_1 and p_2 as:

$$sim_{vocab}(\vec{p}_1, \vec{p}_2) = \frac{\vec{p}_1 W \vec{p}_2{}^T}{|\vec{p}_1||\vec{p}_2|} \tag{4.4}$$

where W is an $n \times n$ matrix that holds the similarities between all the words (vocabulary) in the two event descriptions being compared, n being the length of the vocabulary, and $\overrightarrow{p_1}$ and $\overrightarrow{p_2}$ are binary vectors representing the presence or absence of the words in the vocabulary. Combining these three methods with the three word-level similarity measures we obtained a total of 8 different features[2].

Positional Similarity Feature

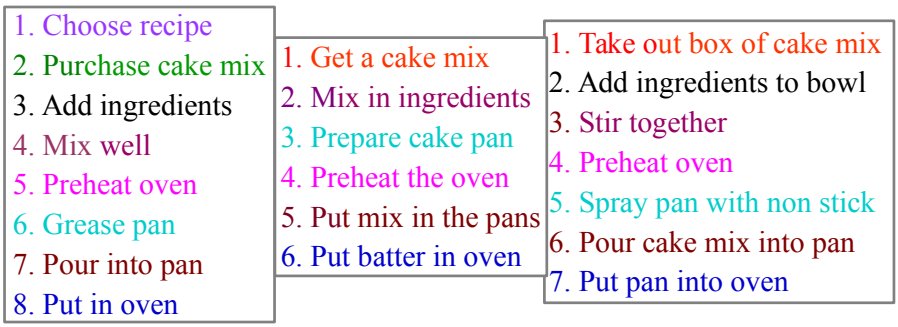


Figure 4.8: Example ESDs `baking a cake` showing positional similarity

In addition to the semantic similarity features described above, we also used information about the position in which an event description occurs in an ESD. The basic idea here is that similar event descriptions tend to occur in similar (relative) positions in the ESDs. For example, in Figure 4.8, *adding ingredients* event tends to occur at the beginning of the ESDs while *putting cake pan in oven* event tends to occur towards the end of the ESDs, *preheating the oven* and *greasing the cake pan* events are relatively at the center of the ESDs.

[2]The *centroid* method can not be combined with Lin similarity.

Positional similarity between two event descriptions is given as:

$$sim_{pos}(n_1, n_2) = 1 - abs\left(\frac{n_1}{T_1} - \frac{n_2}{T_2}\right) \tag{4.5}$$

where $n_1$and n_2 are the positions of the two event description and T_1 and T_2 represent the total number of event descriptions in the respective ESDs.

Positional similarity helps to tear apart similar event descriptions that express different script events. For instance, in `washing dishes`, *Open the dish washer door* (when putting in dirty dishes) and *Open dishwasher* (when removing clean dishes), are two similar event descriptions that refer to different event types and are typically used in different parts of the ESDs. Also, positional similarity brings together event descriptions that are not similar but typically occur in similar positions in the ESDs. For instance, *peel off clothes* and *undress* in `taking a shower` can be identified by the clustering algorithm as expressing the same script event because they tend to occur in similar positions.

Combination

We linearly combine our 9 similarity features into a single similarity score, where the weights of the individual features are determined using logistic regression trained on the development set of the RKP paraphrase dataset which covers 4 scenarios (see Section 4.4). We run an ablation test by considering all possible subsets of features, and found that the combination of the following five features performed best:

- centroid-based, alignment-based and vocabulary similarity with w2v vectors
- centroid-based similarity with rNN vectors
- position similarity

Ablation RKP built an paraphrase corpus containing judgments on pairs of events descriptions as either paraphrase or not paraphrase (see Section 4.4 for details). We use 4 held-out scenarios for training a logistic regression model using our similarity features. Table 4.1 shows an ablation study on our similarity features. We observe that positional similarity information is relevant. The fact that removing single semantic similarity features has only moderate effect, is due to the redundancy in the feature set. Table 4.2 shows

the classification results for predicting whether two event descriptions are paraphrases ot not. We obtain an average F-Scrore of 0.797 with `making coffee` having the least score and `eating in a fastfood restaurant` having the highest score. We noted that the dataset was skewed towards having more negative examples than positive examples.

Feature	Accuracy	F_1-score
centroid+w2v	0.86	0.78
position	**0.82**	**0.74**
alignment+lin	0.86	0.77
alignment+w2v	0.84	0.76
vocabulary+lin	0.86	0.79
vocabulary+w2v	0.86	0.79
centroid+rNN	0.84	0.77
All features	0.86	0.79

Table 4.1: Ablation test (predicting paraphrase or not based on dataset by RKP

scenario	F1
telephone	0.778
bus	0.872
shower	0.727
iron	0.750
foodback	0.878
coffee	*0.667*
microwave	0.727
vending	0.757
scramble	0.867
fastfood	**0.952**
Average	**0.7970**

Table 4.2: Classification results (paraphrase or not) on RKP test set using the 5 best features

4.3.3 Temporal Script Graphs

After clustering the event descriptions of a given scenario into sets representing the scenario-specific event-types, we build a Temporal Script Graph (TSG) by determining the prototypical temporal order between event types. TSGs allow us to abstract away from the fine grained representation of script events found in ESDs and to construct a global representation of a script. The nodes in the TSG represent the event types (clusters) from the cluster-

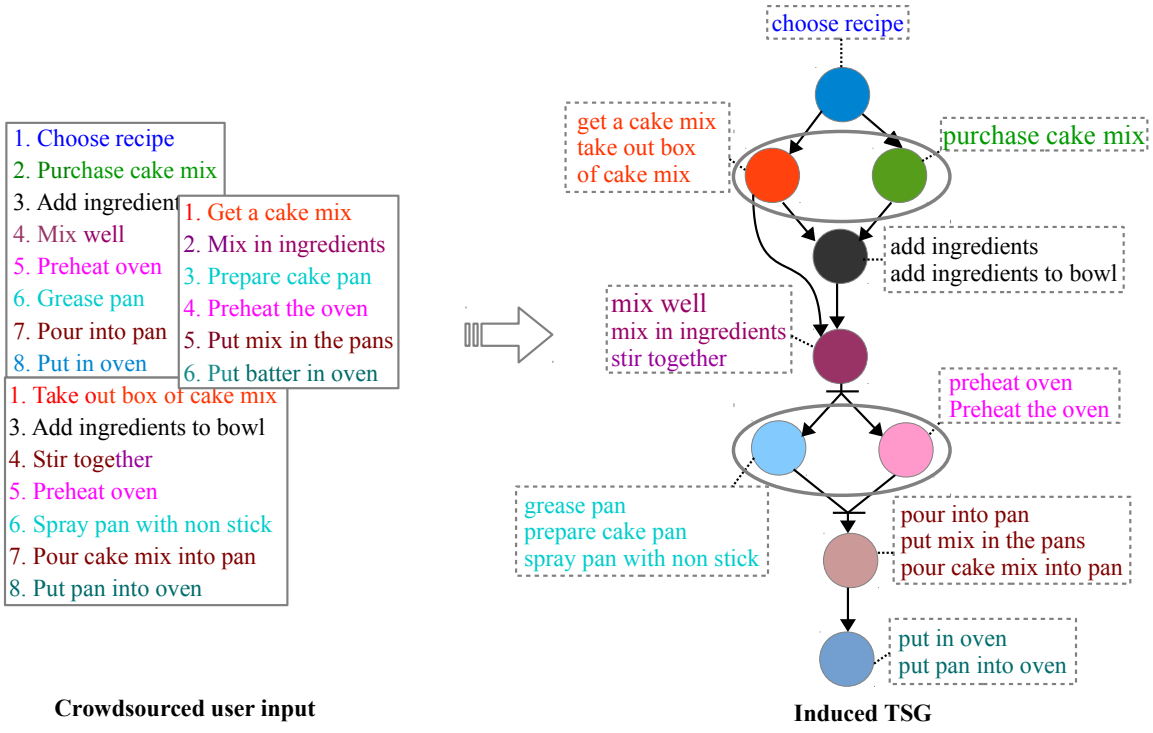


Figure 4.9: Example induced TSG for `baking a cake` scenario.

ing step, while the existence of an edge between two nodes indicates temporal precedence. Temporal precedence is derived from the partial ordering over the event types and does not necessarily imply *immediate* temporal precedence but rather the preceding event typically happens before the succeeding event. An edge from a cluster E to a cluster E' indicates that E typically precedes E', with the possibility of other events in between.

We induce the edges as follows. We say that an ESD *supports* $E \rightarrow E'$ if there are event descriptions $e \in E$ and $e' \in E'$ such that e precedes e' in the ESD. In a first step, we add an edge $E \rightarrow E'$ to the graph if there are more ESDs that support $E \rightarrow E'$ than $E' \rightarrow E$. In a second step, we compute transitive closure, i.e. we infer an edge $E \rightarrow E'$ in cases where there are clusters E, E', E'' such that $E \rightarrow E''$ and $E'' \rightarrow E'$. Finally, we form "arbitrary order" equivalence classes from those pairs of event clusters which have an equal number of supporting ESDs in either direction and are not yet connected by a directed temporal precedence edge. This is an extension of the concept of a temporal script graph used in RKP, in order to allow for the flexible event order assumed by our approach. Figure 4.9 shows an example script structure for `baking a cake` scenario with nodes representing events in the scenario linked to paraphrase sets of semantically similar linguistic descriptions of the same event. For example, the event descriptions *grease pan* and *preheat oven*

are two event types which are members of the same equivalence class, expressing that the event descriptions in either paraphrase set are not paraphrases, but may occur in any order.

4.4 Data

Next, I highlight the resources we used for the experimental studies, namely the datasets of ESDs, the gold standards and the crowdsourced alignments between event descriptions. More details on the ESD and alignments collection (the *DeSript corpus*) can be found in Chapter 3.

Datasets and gold standards. Three crowdsourced collections of activity descriptions in terms of ESDs are available: the OMICS corpus (Gupta and Kochenderfer, 2004), the SMILE corpus (Regneri et al., 2010) and this dissertation's DeScript corpus (Section 3.1). Sections 4.3 - 4.5 of this chapter focus on a subset of ESDs for 14 scenarios from SMILE and OMICS, with on average **29.9** ESDs per scenario. In RKP, and in the follow-up studies by Frermann et al. (2014) and Modi and Titov (2014), as well as in this dissertation, 4 of these scenarios were used as development set and 10 as test set.

RKP provided two gold standard datasets for this subset: the *RKP paraphrase dataset* contains judgments for 60 event description pairs per scenario, the *RKP temporal order dataset* contains 60 event description pairs that are separately annotated in both directions, for a total of 120 datapoints per scenario. They built the paraphrase (temporal order) corpus by asking annotators to label a pair of event descriptions as paraphrase (happens_before) or not paraphrase (does not happen_before).

In order to directly evaluate our models for clustering quality, we also created a *clustering gold standard* for the RKP test set, adopting the experimental setup in Section 3.2: we asked three trained students of computational linguistics to independently annotate the scenarios with gold standard alignments between event descriptions in different ESDs referring to the same event. Every ESD was fully aligned with every other ESD in the same scenario. Based on the alignments, we derived gold clusters by grouping the event descriptions into gold paraphrase sets (17 clusters per scenario on average, ranging from 10 to 23).

In addition, we used a subset of 10 scenario with 25 ESDs each from DeScript with gold

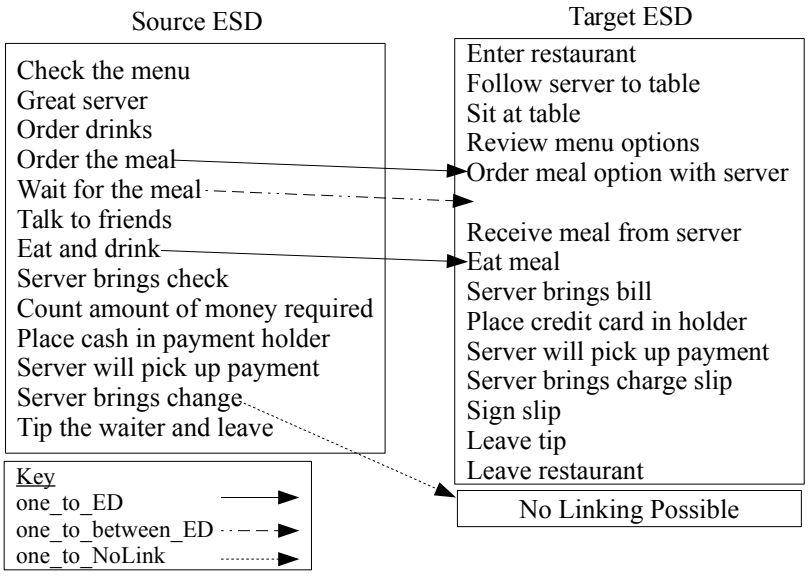


Figure 4.10: Example for alignment options

clusters (Section 3.2), to evaluate our models and to demonstrate that our method is independent of the specific choice of scenarios.

Crowdsourced alignments. To provide seed data for the semi-supervised algorithm, we crowdsourced alignments between ESDs. First, we identified challenging event descriptions (called *outliers*), which we expected to be particularly informative and help improve clustering accuracy. A complementary type of event descriptions was obtained by selecting those event descriptions that were considered well clustered (called *stable cases*) (see Section 3.3 for details of the seed selection strategy).

Workers in Mechanical Turk were presented with pairs of ESDs and were asked to select a description in the target ESD denoting the same script event as a highlighted description in the source ESD. Figure 4.10 shows the aligments that were collected: *one-to-ED*: the source and target ED denote the same script event, *one-to-between-ED*: the source ED denotes a script event that is omitted in the target ESD but would typically take place in a position between EDs in the target ESD, and *one-to-NoLink*: No possible alignment could be made. The workers could also select more than one event descriptions from the target ESD, if the source event was described in more detail in the target ESD, *one-to-many link*. We aimed at collecting two sets of high-quality seeds based on outliers and stable cases, respectively, each summing up to 3% of the links required for a total alignment between all

pairs of scenario-specific ESDs (6% links in total). To guarantee high quality, we accepted only items where three (out of up to four) annotators agree. We checked the annotators' reliability by comparing their alignments for stable cases against the gold standard and rejected the work on 3% of the annotators.

We collected alignments for 20 scenarios: for the test scenarios of the SMILE+OMICS dataset, and for those in the clustering gold standard of DeScript.

Constraint selection

From the set of partial alignments, we extracted pairwise instance-level constraints for use as seed data in semi-supervised clustering. We extracted *must-link* constraints from all worker alignments where at least two workers agreed on a *one-to-ED link* or overlapped in a *one-to-many* link. In order to effectively test the effect of the constraints, we further divided the must-link constraints into alignments that were from outlier event descriptions, alignments that were from stable event descriptions, alignments that were from cases where all the three workers agreed in a *one-to-ED link* and alignments that were from cases where a majority (at least two) workers agreed. Cases where each worker had a different alignment were considered noisy and hence not used. For each of the categories, we further constructed the transitive closure of the set of pairwise constrained event descriptions to get must-link paraphrase sets. That is, if i and j are aligned to be most similar, $M = (i, j)$, and j and k are likewise aligned, $M = (j,k)$, then it follows that i, j and k must belong to the same must-link paraphrase set.

Recall that workers were only asked to align event descriptions that were semantically similar. It is a realistic task to ask workers if two event descriptions must-link, but rather unnatural to ask for cannot-links. We derive the *cannot-link* constraint pairs from the worker alignments. If a set of event description come from the same ESD, we assumed that each event description represents a unique script-event and thus derived a *cannot-link* relationship between all pairs of event descriptions in the same ESD. It was also the case that if two or more workers agreed on a *no link* or *in-between link*, then there exists an implicit *cannot-link* between the aligned source event description and all event descriptions in the target ESD.

In order to make the maximum use of the partial alignments, we also adjusted the cannot-

link constraints to include the implications from the must-link constraints. If their exists a cannot-link pair $C \neq (k, j)$ and a must-link pair $M = (i,k)$, we include $C = (i, j)$ to the set of cannot-link constraints (Liu et al., 2013).

Resolving Inconsistent constraints

Despite the fact that there has been much reported success on the use of pairwise instance-level constraints in semi-supervised clustering, Davidson et al. (2006a) noted that if constraints are poorly specified, they could have negative effect on the clustering algorithm. As the partial alignments were crowdsourced, there is likely to be inconsistencies among the alignments due to differences in worker judgments or errors. We are relying only on a majority vote, but still there may be cases where majority decisions of the workers are incorrect. We cannot treat this in general, the minimal thing that we do is to sort out contradictory annotations. Inconsistent constraints are likely to conflict with other constraints and also likely to be inconsistent with the true underlying similarity. In order to minimize the negative effects that could be brought about by such inconsistencies, we dropped constraints if there exists a pair i, j such that both $C \neq (i, j)$ (*cannot-link*) and $M = (i, j)$ (*must-link*) are true.

4.5 Evaluation

In this section, we address the tasks that we evaluated our model on i.e. the paraphrasing task and the temporal ordering task. We compare the performance of our model against previous state-of-art models and show that on the paraphrase task, our approach outperforms all previous proposals, while still performing very well on the task of temporal order prediction.

4.5.1 Experimental Setup

We applied different versions of our semi-supervised clustering algorithm to the SMILE+OMICS dataset. In particular, we explored the influence of positional similarity, of the number of seeds (from 0 to 3%), as well as the proportion of the two seed types (outlier vs. stable). As a baseline, we ran the unsupervised clustering algorithm based on semantic similarity only.

We evaluated the models on the tasks of event-type induction, paraphrase detection, and temporal order prediction, using the respective gold standard datasets (see Section 4.4).

Cluster quality.

First, we evaluated the quality of the induced event types (i.e. sets of event descriptions) against the SMILE+OMICS gold clusters. We used the B-Cubed metric (Bagga and Baldwin, 1998), which is calculated by averaging per-element precision and recall scores. Amigó et al. (2008) showed B-Cubed to be the metric that appropriately captures all aspects of measuring cluster quality such as cluster homogeneity and cluster completeness, among others.

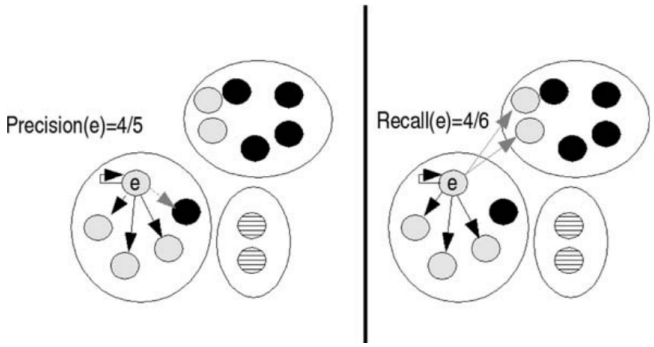


Figure 4.11: B-Cubed cluster quality metric (Amigó et al., 2008)

B-Cubed metric is calculated by averaging per-element precision and recall scores as illustrated in Figure 4.11. For example, on the left side, for event *e*, the precision is 0.8 because out of the cluster elements, 3 belong to the same cluster as *e*. Similarly for the right side, for event *e*, recall is 0.667 because 2 other elements that should be in the same cluster as *e* are in a different cluster.

Additionally, we examined the dependents of cluster quality on various factors namely: the effect of using stable vs outlier cases, using different seed-data based on worker agreement, adding additional seed-data from neighboring data points and incorporating cannnot-links constraints to the cluster quality.

Paraphrase detection.

For direct comparison with previous work, we tested our model on RKP's binary paraphrase detection task(see Section 4.5.2). The model classifies two event descriptions as

paraphrases if they end up in the same cluster. We computed standard precision, recall and F-score by checking our classification against the RKP paraphrase dataset.

Temporal order prediction.

We tested the quality of the temporal-order relation of the induced TSG structures using the RKP temporal order dataset as follows. For a pair of event descriptions (e, e'), we assume that (1) e precedes e', but not the other way round, if $e \in E$ and $e' \in E'$ for two different clusters E and E' such that $E \rightarrow E'$. (2) e precedes e' *and* vice versa, if $e \in E$ and $e' \in E'$, and E and E' are different clusters, but part of the same equivalence set. For instance, in Figure 4.14 *get into the tub* $\in$ *get in shower* precedes *adjust to cold or hot water* $\in$ *adjust temperature* and vice versa, because *get in shower* and *adjust temperature* are in the same equivalence set. In all other cases (in particular, if e and e' are members of the same cluster), we assume that precedence does not hold. We computed standard precision, recall and F-score by checking our classification against the RKP temporal order dataset.

Baselines For evaluating the clustering quality and paraphrase detection, the baseline is the unsupervised system with only semantic features and zero constraints added (USC) and the unsupervised system with both semantic and positional features and zero constraints added (USC+Position).

For the paraphrase detection and temporal order prediction tasks, we also compared to previous work by (Regneri et al., 2010; Modi and Titov, 2014; Frermann et al., 2014).

4.5.2 Results

The main results of our evaluation are shown in Table 4.3. The last three rows show results for three of our model variants: unsupervised clustering with both semantic and positional information (USC+Position), semi-supervised clustering with positional information and only 3% outlier constraints (SSC+Outlier) and with the best-performing ratio of constraint types (SSC+Mixed, with 2% outliers and 1% stable cases). Row 4 shows the results for our unsupervised clustering baseline with semantic similarity only (USC). ar For comparison against previous work, we added the results on the paraphrase and temporal ordering tasks of the MSA model by RKP, the Hierarchical Bayesian model by Frermann et al. (2014)

	Clustering	Paraphrasing			Temporal Ordering		
Model	B-Cubed	Precision	Recall	F-score	Precision	Recall	F-score
Regneri et al. (2010)	–	0.645	**0.833**	0.716	0.658	0.786	0.706
Modi and Titov (2014)	–	–	–	0.645	0.839	**0.843**	**0.841**
Frermann et al. (2014)	–	0.743	0.658	0.689	0.85	0.717	0.776
Baseline: USC	0.525	0.738	0.593	0.646	0.736	0.712	0.722
USC+Position	0.531	0.76	0.623	0.675	0.789	0.766	0.775
SSC+Outlier	0.635	0.781	0.751	0.756	0.858	0.791	**0.822**
SSC+Mixed	**0.655**	**0.796**	0.756	**0.764**	**0.865**	0.784	**0.822**

Table 4.3: Results on the clustering, paraphrasing and temporal ordering tasks for state-of-the-art models, our unsupervised (USC) and semi-supervised clustering approaches (SSC)

and the Event Embedding model by Modi and Titov (2014) (for review on previous work see Chapter 2). On all three tasks, our best-performing model is SSC with mixed seed data (SSC+Mixed). We can see that the use of both positional information and mixed seed data in the distance measure has substantial effects on the quality of the results. Our best model outperforms RKP by 4.8 points (F-score) on the paraphrasing and by 11.6 points on the temporal ordering task. Interestingly, the performance gain is exclusively due to an increase of precision in both tasks (13.6 and 20 points, respectively). Our system comes close, but does not beat Modi and Titov (2014) on their state-of-the-art model for temporal ordering, but outperforms it on the paraphrase task by almost 14 points F-score. Table 4.4 shows scenario-wise results on the paraphrasing and temporal ordering tasks. The most challenging scenarios for the model for the paraphrasing task were `making coffee` (0.655), `making scrambled eggs` (0.624) and `eating in a fast food restaurant` (0.647) . The ESDs in these scenarios had varying granularity and also included variants of the given scenario which contributed to the low performance of the system. For instance, some ESDs had more granular events than others, e.g. in `making scrambles eggs`, *add salt*, *add pepper* as seperate events or *add salt pepper and hot sauce* as a single event, in `eating in a fast food restaurant`, *listen to cashier repeat order, listen for total price, look at order number, count change,* etc., each appearing in a single ESD only). Some ESDs described `making coffee` in a pot while others using a coffee maker. Likewise, some ESDs described `making scambled eggs` in the microwave while others in a frying pan.

	Paraphrasing: F-Score			**Temporal Ordering: F-Score**		
Scenario	RKP	SSC_Out.	SSC_Mix.	Modi	SSC_Out.	SSC_Mix.
bus	0.740	**0.822**	0.784	0.884	**0.947**	0.931
coffee	0.650	0.593	**0.655**	0.702	0.766	**0.885**
Fastfood	0.590	0.626	**0.647**	**0.889**	0.863	0.834
Ret. Food	0.710	0.930	**0.944**	**0.910**	0.746	0.750
Iron	0.670	0.725	**0.757**	**0.834**	0.732	0.777
Microwave	0.750	0.923	**0.866**	**0.864**	0.839	0.793
Scr. Eggs	**0.690**	0.671	0.624	**0.787**	0.758	0.736
Shower	0.780	0.755	**0.783**	0.821	0.821	**0.828**
Telephone	**0.890**	0.850	0.858	0.882	**0.930**	0.817
Vending	0.690	0.664	**0.720**	**0.882**	0.814	0.816
Average	0.716	0.756	**0.764**	**0.841**	0.822	0.817

Table 4.4: Scenario wise results on paraphrasing and temporal ordering tasks for state-of-art models (RKP, Regneri et al. (2010)), (Modi, Modi and Titov (2014)) and our semi-supervised clustering approaches (SSC

Effect of worker agreement In our first experiments, we contrasted the contribution of alignments where all workers agreed with the alignments where the majority of workers agreed. It was noted that full agreement gives better results than majority agreement. Even if there are more seeds from majority cases, cluster performance was better for some scenarios, but for most scenarios, the cluster quality was negatively affected. There was high inconsistencies in the seed data with only majority vote. Inconsistencies arise when there exists a pair i, j such that both $C \neq (i, j)$ (*cannot-link*) and $M = (i, j)$ (*must-link*) are true (see previous section). The negative effect of seed data was also pointed out by Davidson et al. (2006b), where non informative and inconsistent seeds were likely to decrease the cluster quality.

We therefore aimed at collecting only high-quality seeds. To guarantee high-quality, we accepted only items where three (out of up to four) annotators agree on a particular alignment.

Effect of stable vs outlier cases We tested the level of improvement got when using alignments from outlier cases as opposed to alignments from stable cases. Workers tend to agree more in stable cases as compared to outliers cases and it took more annotation rounds to get high quality outlier seeds. In one case, we randomly added must-link constraints from outlier alignments, in the second case, we randomly added must-link constraints from

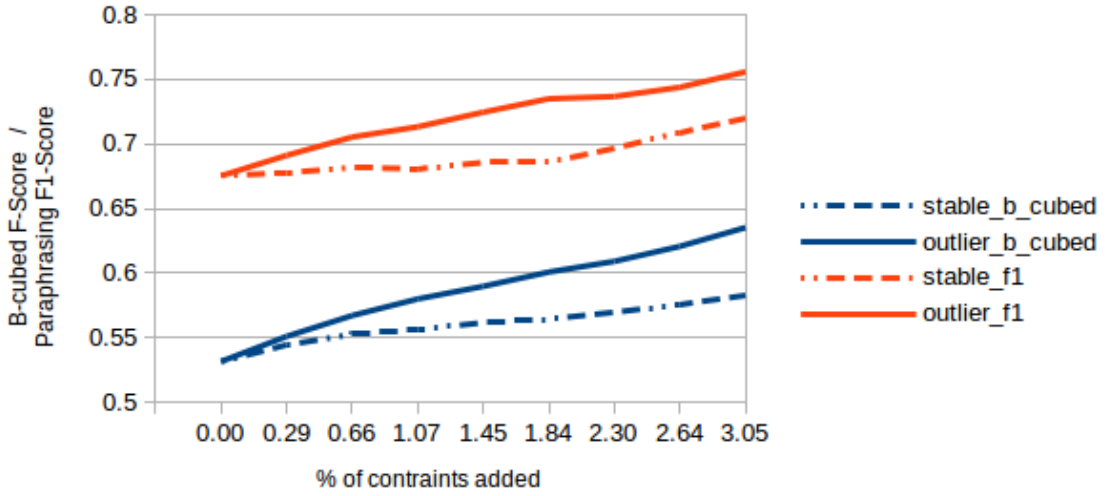


Figure 4.12: Effect of stable vs outlier seeds

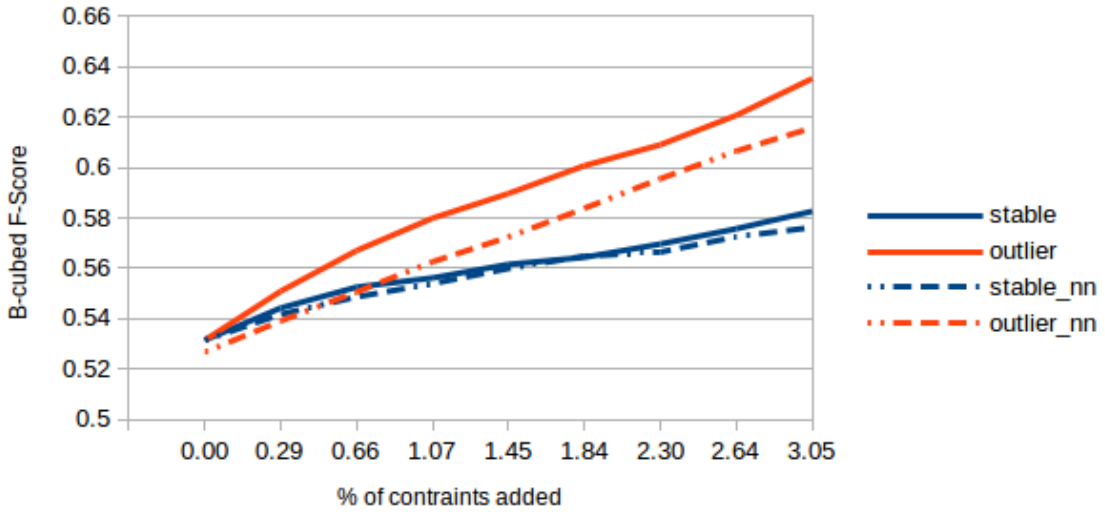


Figure 4.13: Effect of incorporating additional Nearest Neighbors seeds

stable case alignments.

Outlier cases tend to achieve higher results as compared to stable cases. Figure 4.12 enables a closer look on the effect in dependence of the number of seed links added. The outlier curve grows faster than the stable curve (for both clustering and paraphrasing tasks). The informed selection of seed data pays off as fewer seeds are needed to get significant improvements in both the clustering and paraphrasing task.

Effect of adding NN We also evaluated the effect of including other event descriptions that are nearest to the event descriptions in a must-link constraint. For each event description in a must-link constraint, we found its n nearest neighbors that were less than t_1 distance apart, and the size of n had to be greater than a threshold $t_2$to qualify to be included

to expand the must-link pair.

We found out that automatically added constraints do not improve the end results, both for stable and outlier alignments. Figure 4.13 shows that the decrease in performance is more prominent in the outlier cases than in stable cases. Intuitively, this is not surprising as data points neighboring an outlier case are probably misleading (e.g. in `flying in a plane` scenario, *walk up the ramp* might be closer to *walk to gate* or *walk to plane* but the three should be in different clusters).

Effect of cannot-link constraints As we extracted many cannot-link constraints, we only added those cannot-link constrained pairs whose similarity is greater than a set threshold (0.3 in our experiments). If the similarity between two cannot-link pairs is less than the given threshold, then the information that the two belong to different clusters is already provided by the similarity score. We experimented with adding cannot-link constraints before and after adding the must-links constraints, and randomly adding cannot-link and must-link constraints. In the first case we added the cannot-link constraints before resolving the conflicts with must-link constraints. In the second case, we added cannot-link constraints after dropping the cannot-link pairs that conflict with some must-link pair. In the third case, we dropped the conflicting pairs from the set of cannot-link constraints and must-link constraints.

The results showed that adding cannot-links does not improve cluster quality but does not make the performance worse either. It could be argued that, because we did not explicitly ask for cannot-links but rather derived cannot-links from the provided must-links, the information that any pair of event descriptions should not be in the same cluster is already provided by the similarity scores. It would be interesting to see if there would be improvement in performance if we particularly select event pairs that look semantically similar (e.g. *walk to gate* and *walk to plane*) but should belong to different clusters and ask for cannot-links. This part is left out for future work.

4.5.3 Discussion

The largest and most consistent performance gain of our model is due to use of crowdsourced alignment information. Figure 4.14 shows the induced TSG for `taking a shower`

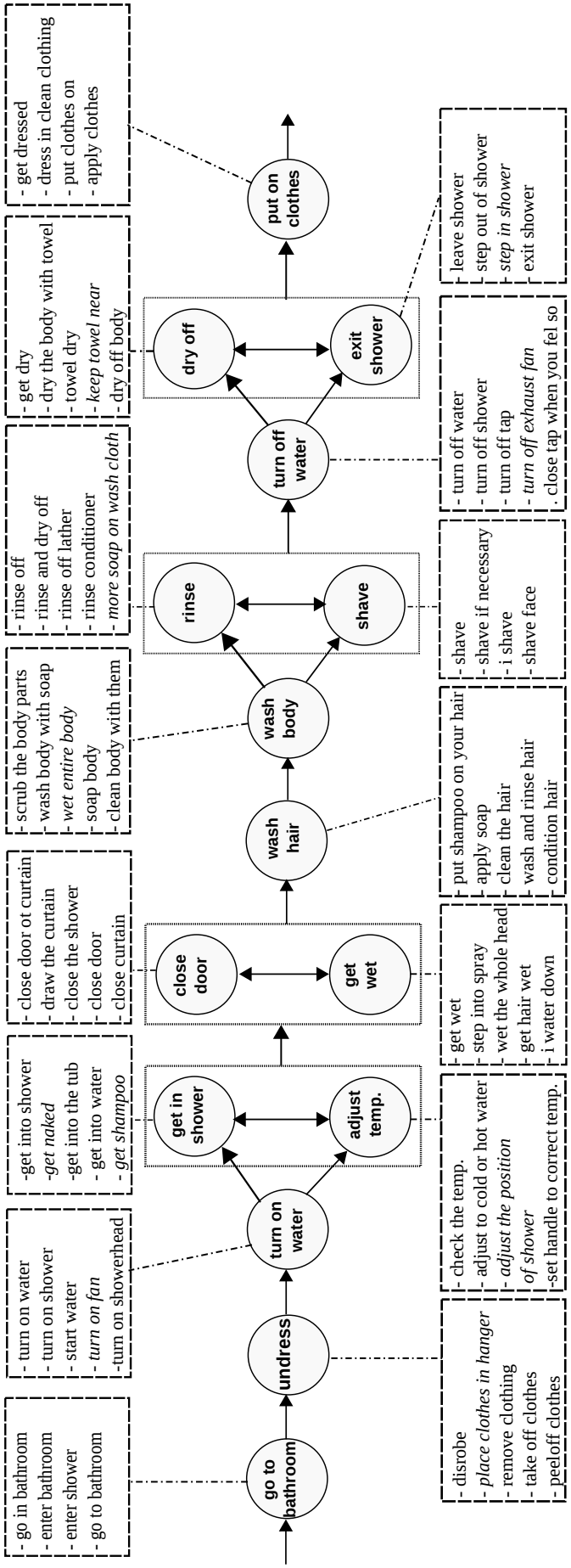


Figure 4.14: Example TSG output by our model for `taking a shower`.

scenario, with script-specific paraphrases captured by our best model. The model was able to capture a wide variety of lexical realizations of `undress`, including *peel off clothes, disrobe, remove clothes* etc., and similarly for `dress`, where we get *get dressed, apply clothes, put on clothes*, while these ended up in different clusters in the baseline model (e.g. *get dressed* was clustered together with *shampoo hair* cluster). There are still some incorrect classifications (indicated with italics in Figure 4.14); note that these are often near misses rather than blatant errors. The near misses were mostly event descriptions that were written by one ot two workers and might be too fine grained to be in their own cluster (e.g. *adjust position of the shower, place clothes in hanger, keep towel near,* etc.).

A leading motivation to use clustering instead of MSA was the opportunity to model flexible event order in script structures. Our expectations were confirmed by the evaluation results. In addition, our system actually makes extensive use of the option of flexible event ordering, as shown by the example TSG in Figure 4.14. In `taking a shower` scenario, one can *adjust the temperature* before or after *getting in the shower*. One can also *dry off* before or after *exiting the shower*. The equivalence classes also denote underspecification,i.e. some of the events in the equivalence cluster may or may not occur while performing the given scenario. For instance, look at the equivalence class (*rinse, shave*), it is possible that one may or may not *shave* while `taking a shower`.

Positional information substantially contributes to the quality of the derived TSGs. While the model using semantic similarity features only put *peel off dresses* in the `dress` cluster, positional similarity helped placing it correctly in the `undress` cluster, as it appears in the initial segment of its ESD. Positional information sometimes also caused wrong clustering decisions: *place cloth in hanger* typically occurs directly after `undressing`, and thus ended up in the `undress` cluster.

As described above, we collected alignments for outliers and for stable cases and tried several outlier-to-stable ratios. Outliers were much more effective than stable cases, as they improved recall by adjusting cluster boundaries to include scenario-specific functional paraphrases that were semantically dissimilar. Interestingly, adding a small number of stable cases leads to a slight improvement, but adding more stable cases leads to a performance drop, and using only stable cases does not improve the unsupervised baseline at all. Figure 4.15 shows how the model improves as more constraints are added. We tried to reduce the amount of manual annotation in several ways. The decision to derive additional must-

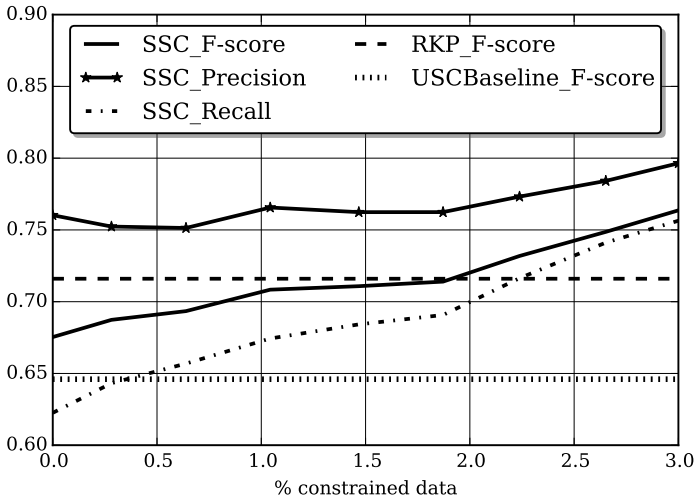


Figure 4.15: Paraphrase detection results for RKP, for our Unsupervised baseline (USC) and for our best Semi-supervised model (SSC+Mixed)

links using transitivity paid off: F-score consistently improves by about 1 point F-score. To further increase the set of seeds, we experimented with propagating the links to nearest neighbors of aligned event descriptions, but did not see an improvement. Also, we tried to use alignments obtained by majority vote, which however led to a performance drop, showing that using high quality seeds is crucial.

To make sure that our results are not dependent on the selection of a specific scenario set, we evaluated our model also on the DeScript gold clusters. The results were comparable: B-Cubed F-Score improved from 0.509 (RKP: 0.525) to 0.662 (RKP: 0.655). As the DeScript corpus provides 100 ESDs per scenario, we were also able to test whether an increased number of input ESDs also improves clustering performance. In Figure 4.16, we observe no effect with 50 ESDs, and only a slight (less than 1 point) improvement with the full 100 ESD dataset (while increasingly adding the outlier constraints). With zero constraints added, clustering performance is better with 25 ESDs and worse with 100 ESDs. As we add more outlier seeds, clustering performance with 100 ESDs quickly improves. This could partly be attributed to high redundancy of event descriptions in 100 ESDs as compared to 25 ESDs.

We also experimented with deriving additional constraints by computing the cross-product of the set of nearest neighbors to the pairs i and j in a must-link constraint, but the effects were not consistent as spurious neighbors resulting from errors in semantic similarity reduced the overall performance of the model. We conducted exploratory study on us-

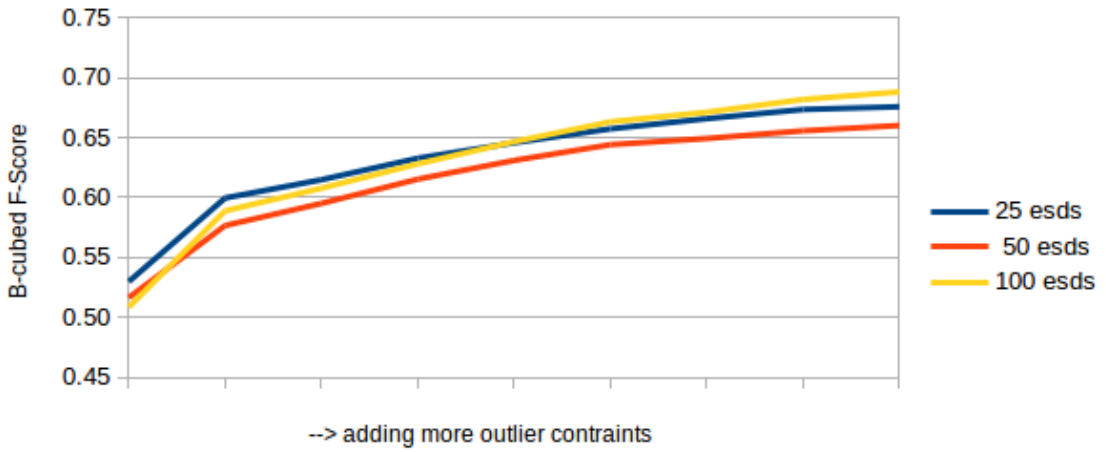


Figure 4.16: Effect of increasing the number of ESDs

ing cannot-link constraints, but we were unable to collect sufficient links to show concrete effects of including cannot-links. This can still be explored as future work.

Although we did not optimize for temporal ordering, our semi-supervised models rivals the state-of-art Modi and Titov (2014) model, which did optimize for this specific task, and outperformed the other two models. Also, from Table 4.3, it can be noted that the better we get at paraphrase detection, the better we get at temporal ordering.

4.6 Costs and Coverage

We have demonstrated that crowdsourcing enables acquisition of reliable script knowledge with substantially higher quality than existing methods. But how does the method scale? Can we expect to obtain a script knowledge database with sufficiently wide coverage at reasonable costs?

The process of script extraction requires crowdsourced data in terms of (1) ESDs and (2) seed alignments. To complete 3+3% high-quality alignments for the 10 DeScript scenarios via Mechanical Turk, workers spent a total of 37.5 hours, with an average of 3.75 hrs per scenario, ranging from 2.5 (GOING GROCERY SHOPPING) to 7.52 hrs (BAKING A CAKE)[3]. It took on average 2.78 hrs to collect 25 scenario-specific ESDs, that is 6.63 hrs of data acquisition time per scenario.

The costs per scenario appear moderate. But how many scenarios must be modeled to achieve sufficient coverage for the analysis of script knowledge in natural-language texts?

[3]We used the DeScript scenarios as reference because they come with equal numbers of ESDs, while the ESD sets in the SMILE+OMICS corpus vary considerably in size.

Jessica needs milk. Jessica wakes up and wants to eat breakfast. She grabs the cereal and pours some into a bowl. She looks in the fridge for milk. There is no milk in the fridge so she can't eat her breakfast. She goes to the store to buy some milk comes home and eats breakfast.
MAKE BREAKFAST: **C**
GOING GROCERY SHOPPING: **P**

Figure 4.17: Example ROC-story with scenario annotation.

Answering this question is not trivial, as scenarios vary considerably in granularity and it is not trivial that the type of script knowledge we model can capture all kinds of event structures, even in narrative texts. In order to provide a rough estimate of coverage for the currently existing script material, we carried out a simple annotation study on the recently published ROC-stories database Mostafazadeh et al. (2016) that consists of about 50,000 short narrative texts (see Section 2.3).

For our annotation study, we merged the available datasets containing crowdsourced ESD collections (i.e., OMICS, SMILE, and DeScript), excluding two extremely general scenarios (GO OUTSIDE, CHILDHOOD), which gives us a total of 226 different scenarios (see Section 2.3 for descriptions of SMILE and OMICS corpora).

We randomly selected 500 of the ROC-stories and asked annotators to determine for each story which scenario (if any) was centrally addressed and which scenarios were just referred to or partially addressed with at least one event mention, and to label them with 'C" and "P", respectively. See an example story with its annotation in Fig. 4.17.

Each story was annotated by three students of computational linguistics. To facilitate annotation, the stories were presented alongside ten scenarios whose ESDs showed strongest lexical overlap with the story (calculated as tf-idf). However, annotators were expected to consider the full scenario list. The three annotations were merged using majority vote. I inspected and adjudicated those cases without a clear majority vote containing one single C assignment.

26.4% of the stories were judged to centrally refer to one of the scenarios[4]. Although this percentage cannot be directly translated to coverage values, it indicates that the extraction method presented in this dissertation has the potential to provide a script knowledge

[4]While we take the judgment about the C class to be quite reliable (24.8% qualified by majority vote, only 1.6 % were added via adjudication), there was considerable confusion about the P label. So we decided not to use the P labeling at all.

resource with reasonable costs, which can substantially contribute to the task of text understanding.

4.7 Summary

We have presented a semi-supervised clustering approach to induce script structure from crowdsourced descriptions of event sequences by grouping event descriptions into paraphrase sets (representing event types) and inducing their temporal order. This approach exploits semantic and positional similarity and allows for flexible event order, thus overcoming the rigidity of previous approaches. We incorporate crowdsourced alignments as prior knowledge and show that exploiting a small number of alignments results in a substantial improvement in cluster quality over state-of-the-art models and provides an appropriate basis for the induction of temporal order. On a paraphrase task, this approach outperforms all previous proposals, while still performing very well on the task of temporal order prediction.

We also show a coverage study to demonstrate the scalability of this approach. A study on the ROC-stories suggests that a model of script knowledge created with this method can cover a large fraction of event structures occurring in topically unrestricted narrative text, thus demonstrating the scalability of this approach.

Part III

Scenario detection and classification for narrative texts

Chapter 5

Identifying everyday scenarios in narrative texts

Consider the following text fragment about `eating in a restaurant` from an online blog post:

Example 5.1. *(. . .) we drove to Sham Shui Po and looked for a place to eat. (. . .) One of the restaurants was fully seated [so we] chose another. We had 4 dishes—Cow tripe stir fried with shallots, ginger and chili. 1000-year-old-egg with watercress and omelet. Then another kind of tripe and egg—all crispy on the top and soft on the inside. Finally calamari stir fried with rock salt and chili. Washed down with beers and tea at the end. (. . .)*

The text in Example 5.1 obviously talks about a restaurant visit, but it omits many events that are involved while `eating in a restaurant`, such as *finding a table*, *sitting down*, *ordering food* etc., as well as participants such as *the waiter*, *the menu*,*the bill*. A human reader of the story will naturally assume that all these ingredients have their place in the reported event, based on their common-sense knowledge, although the text leaves them completely implicit. For text understanding systems that lack appropriate common-sense knowledge, the implicitness however poses a non-trivial challenge.

Writing and understanding of narrative texts makes particular use of *script knowledge* (Schank and Abelson, 1977). Given a specific scenario, the associated script knowledge

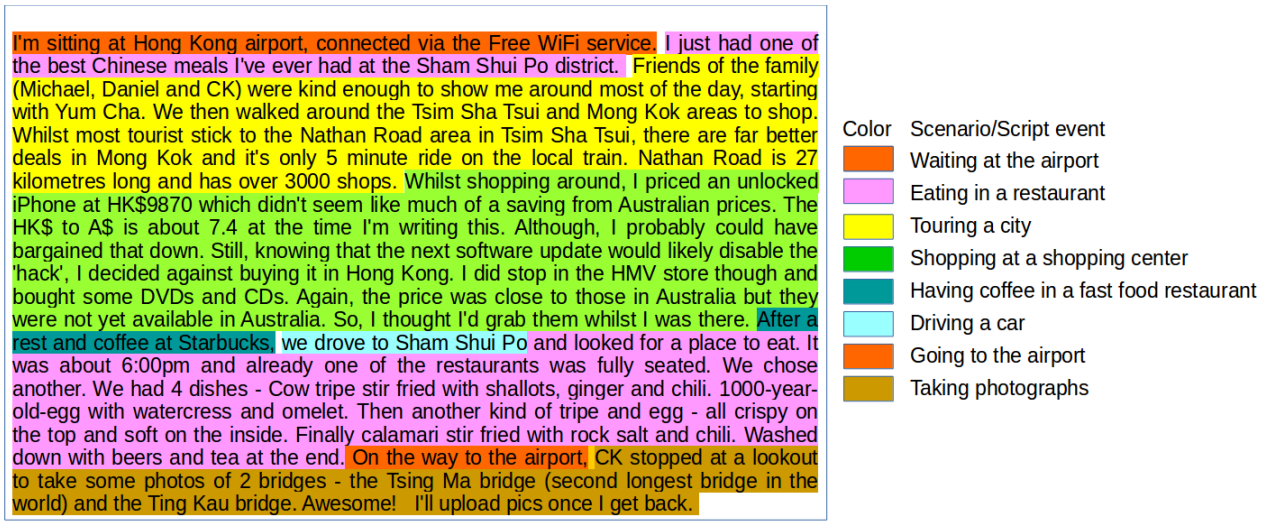


Figure 5.1: Example text[1] showing multi-script instantiation

enables us to infer omitted events that happen before and after an explicitly mentioned event, as well as its associated participants. In other words, this knowledge can help us obtain more complete text representations, as required for many language comprehension tasks.

So far, we have looked at the acquisition of script knowledge, in particular event sequence descriptions (ESDs), via crowdsourcing (Chapter 3). In Chapter 3, we described how we crowdsourced EDS for 40 scenarios to create the DeScript corpus. We further crowdsourced alignments between event descriptions that are particularly difficult for the induction algorithm to determine which event-type they belong to. In Chapter 4, we described how we used the alignments in a semi-supervised clustering model to induce the script-structure from the ESDs. Script acquisition and induction are first necessary steps towards high quality script modeling.

Script acquisition and induction enable script parsing and are a core part for script-based natural language understanding. There has been some work on script parsing (Ostermann et al., 2017, 2018c), i.e., associating texts with script structure given a specific scenario. However, they worked on simplistic texts where the script being referenced in the text was known, thus no scenario detection was required.

Naturalistic texts pose greater challenges. Consider the blog post in Figure 5.1. There are several scenarios mentioned in the text. This makes scenario detection difficult for two

[1] text extracted from Personal Stories Spinn3r corpus (Gordon et al., 2009)

main reasons. First, text passages referring to one scenario do not necessarily need to be contiguous i.e. the scenario could be referred to in different parts of the same text passage, so the scenario label can occur several times in the text. For instance, the writer in Figure 5.1 talks about having *eaten the best Chinese food* at the beginning of the text, goes on to mention other activities and elaborates on what they ate towards the end of the text. Second, one segment can be associated with more than one scenario. The referred scenarios share script events and participants. For such cases, the text segment is annotated with all relevant scenario labels. For instance, in Figure 5.1, while `touring the city`, the person is also `shopping`. The mentioned events belong to both scenarios at the same time. It is plausible to annotate the text segment with both labels.

When human hearers or readers come across an expression that evokes a particular script, they try to map verbs or clauses in the subsequent text to script events, until they face lexical material that is clearly unrelated to the script and may evoke a different scenario. Scenario identification, scenario segmentation, and script parsing are subtasks of story comprehension, which ideally work in close mutual interaction. However, no previous work exist on determining which scenarios are referred to in a text or text segment (see Section 5.1). To the best of our knowledge, this is the first dataset of narrative texts which have annotations at sentence level according to the scripts they instantiate.

In this Chapter, we describe a study on identification and labeling of text segments with the specific scenarios they instantiate. We define the task of scenario identification and introduce a benchmark dataset of annotated narrative texts, with segments labeled according to the scripts they instantiate (Section 5.2). As a result, the labeled scenario-specific text fragments provide themselves as additional linguistic resources from where script-knowledge can be acquired.[2]

5.1 Motivation and Background

A major line of research has focused on identifying specific events across documents, for example, as part of the Topic Detection and Tracking (TDT) initiative (Allan et al., 1998; Allan, 2012). The main subject of the TDT intiative are instances of world events such

[2]The corpus is publicly available for scientific research purposes at this `http://www.sfb1102.uni-saarland.de/?page_id=2582`.

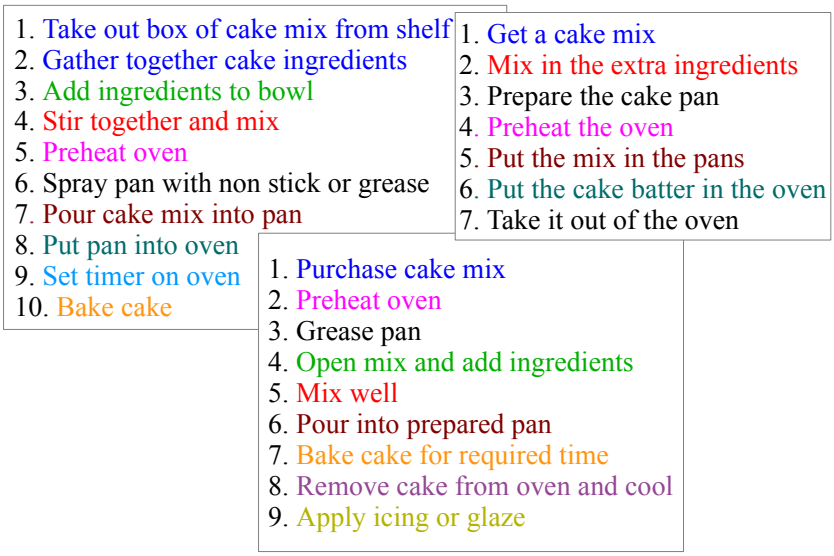
as *Cuban Riots in Panama*. In contrast, everyday scenarios and associated event types, as dealt with in this dissertation, have so far only been the subject of individual research efforts focusing either on acquiring script knowledge, constructing story corpora, or script-related downstream tasks. Below we describe significant previous work in these areas in more detail.

Script knowledge. Different lines of research attempt to acquire script knowledge. One line focuses on the collection of scenario-specific script knowledge in form of event sequence descriptions (ESDs) via crowdsourcing, (Singh et al., 2002; Gupta and Kochenderfer, 2004; Li et al., 2012; Raisig et al., 2009; Regneri et al., 2010) and our work DeScript (see Chapter 3). The top part of Table 5.1 summarizes various script knowledge-bases. While datasets like OMICS seem large, they only cover mundane scenarios that take place indoors (e.g. `open door, switch off lights`). On the other hand, our DeScript corpus covers a wide range of everyday activities. Another line of research tries to leverage existing large text corpora (typically news) to induce script-like knowledge about the topics represented in these corpora (Chambers and Jurafsky, 2008, 2009; Pichotta and Mooney, 2014) (see Chapter 2 for an overview).

The work described in this Chapter lies in between both lines of research and may help to connect them: we take an extended set of specific scenarios as a starting point and attempt to identify instances of those scenarios in a large-scale collection of narrative texts.

Textual resources. Previous work created script-related resources by crowdsourcing stories that instantiate script knowledge of specific scenarios. For example, Modi et al. (2016) and Ostermann et al. (2018a, 2019) asked crowd-workers to write stories that include mundane aspects of scripts "as if explaining to a child". Figure 5.2 contrasts ESDs (Figure 5.2a) from crowdsourced stories that instantiate script-knowledge (Figure 5.2b). The collected datasets, *InScript* and *MCScript* respectively, are useful as training instances of narrative texts that refer to scripts. However, the texts are kind of unnatural and atypical because of their explicitness and the requirement to workers to tell a story that is related to one single scenario only. Gordon and Swanson (2009) employed statistical text classification in order to collect narrative texts about personal stories. The *Spinn3r*[3] dataset (Burton et al.,

[3]http://www.icwsm.org/data/



(a) DeScript ESDs

It was my mother 's birthday last weekend and I wanted to bake her a cake . First , I looked up a recipes on the Internet . I found a chocolate cake recipe that looked very good . I was missing a few ingredients , so I made a quick run to the grocery store . Once I had what I needed , I got to work . First , I got out all of the baking pans that I would need to bake the cake . Then I got out all the ingredients and let them sit at room temperature for a bit . After prepping the pans , I preheated the oven and stirred the dry ingredients together . Then I added the butter , sugar , eggs , and flour . After everything was mixed well , I put the mixture in the pan and then put the pan in the oven . After 40 minutes , the cake was done . I let it cool and then added white frosting . My mom loved the cake .

(b) MCScript story

Figure 5.2: Example DeScript ESDs and MCScript story for `baking a cake` scenario.

2009) contains about 1.5 Million stories. *Spinn3r* has been used to extract script information (Rahimtoroghi et al., 2016, see below). The work described in this Chapter uses the Spinn3r personal stories corpus as a source for data collection and annotation. The bottom part of Table 5.1 summarizes various script-related resources. The large data sets come with no scenarios labels and the crowdsourced data sets only have scenario labels at story level. Our work provides a more fine grained scenario labeling at segment level.

Script-related tasks. Several tasks have been proposed that require or test computational models of script knowledge. For example, Kasch and Oates (2010) and Rahimtoroghi et al. (2016) propose and evaluate a method that automatically creates event schemas, extracted from scenario-specific texts. Ostermann et al. (2017) attempt to identify and label men-

tions of events from specific scenarios in corresponding texts. Finally, Ostermann et al. (2018b) present an end-to-end evaluation framework that assesses the performance of machine comprehension models using script knowledge. Scenario detection is a prerequisite for tackling such tasks, because the application of script knowledge requires awareness of the scenario a text segment is about.

Scenario collections	Scenarios	Total ESDs
SMILE (Regneri et al., 2010)	22	386
Cooking (Regneri, 2013)	53	2500
OMICS (Indoor) (Singh et al., 2002)	175	9044
Raisig et al. (2009)	30	450
Li et al. (2012)	9	500
DeScript (Chapter 3)	40	4000

Story Corpora	Scenarios	Total stories	Classes	Segs.
Mostafazadeh et al. (2016)	✗	~50000	✗	✗
Gordon and Swanson (2009)	✗	~1.5M	✗	✗
Modi et al. (2016)	10	1000	✓	✗
Ostermann et al. (2018a, 2019)	200	4000	✓	✗
Rahimtoroghi et al. (2016)	2	660	✓	✗
This Chapter	181	504	✓	✓

Table 5.1: Top part shows scenario collections and number of associated event sequence descriptions (ESDs). Bottom part lists story corpora together with the number of stories and different scenarios covered. The last two columns indicate whether the stories are classified and segmented, respectively.

5.2 Task and Data

We define scenario detection as the task of identifying segments of a text that are about a specific scenario and classifying these segments accordingly. For the purpose of this task, we view a segment as a consecutive part of text that consists of one or more sentences. Each segment can be assigned none, one, or multiple labels.

Scenario labels. As a set of target labels, we collected scenarios from all scenario lists available in the literature (see Table 5.1). During revision, we discarded scenarios that are too vague and general (e.g. `childhood`) or atomic (e.g. `switch on/off lights`), admit-

ting only reasonably structured activities. Based on a sample annotation of Spinn3r stories, we further added 58 new scenarios, e.g. `attending a court hearing`, `going skiing`, to increase coverage. We deliberately included narrowly related scenarios that stand in the relation of specialization (e.g. `going shopping` and `shopping for clothes`, or in a subscript relation (`flying in an airplane` and `checking in at the airport`). These cases are challenging to annotators because segments may refer to different scenarios at the same time.

Although our scenario list is incomplete, it is representative for the structural problems that can occur during annotation. We have scenarios that have varying degrees of complexity and cover a wide range of everyday activities. Figure 5.3 indicates the range of everyday activities that are covered by our scenario collection. The complete list contains 200 scenarios[4] and is provided in Appendix A.

For purposes of annotation, we grouped the scenarios into 14 classes. We used an automatic clustering as a basis to identify the 14 classes (see Section 6.2). The different classes are indicated by scenario names. The classes cover various activities including:

- Leisure activities: we distinguish between indoor and outdoor leisure activities. Indoor:`playing a movie`, `playing a board game`, `listening to music`, e.t.c. Outdoor: `going to the sauna`, `eating in a restaurant`, `going bowling`, `going to the theater`, e.t.c.
- Cleaning and house chores: `doing laundry`, `ironing laundry`, `washing the dishes`, `cleaning up a flat`, e.t.c.
- Cooking: `baking a cake`, `cooking pasta`, `making coffee`, `making scrambled eggs`, e.t.c.
- Sports: `playing tennis`, `going swimming`, `going bowling`, `playing golf`, e.t.c.
- Gardening: `planting a tree`, `mowing the lawn`, `planting flowers`, e.t.c.

[4]The scenario collection was jointly extended together with the authors of MCScript (Ostermann et al., 2018a, 2019). The same set was used in building MCScript 2.0 (Ostermann et al., 2019). which we make use of in this work.

- Using public transport: `going on a train, flying in an airplane, riding on a bus, taking the underground`, e.t.c.
- Personal grooming: `taking a bath, taking a shower, washing one's hair, getting a haircut`, e.t.c.
- Making transactions: we distinguish between indoor and outdoor transactions. Indoor: `paying with a credit card, ordering pizza, making a dinner reservation`, e.t.c; Outdoor: going grocery shopping, buying a house, fueling a car, e.t.c
- Do-It-Yourself: `sewing a button, changing batteries in an alarm clock, repairing a flat bicycle tire`, e.t.c.
- Traveling: `driving a car, making a camping trip, going to vacation`, e.t.c.
- Renovation: `renovating a room, layering flooring in a room, painting a wall`, e.t.c.
- Animals: `feeding a cat, training a dog, adopting a pet`, e.t.c.
- Children: `putting a child to bed, taking care of children, going to the playground`, e.t.c.
- Other outdoor activities: `visiting a doctor, attending a court hearing, taking a driving lesson`, e.t.c.

5.2.1 Dataset and Annotation

Dataset. As a benchmark dataset, we annotated 504 texts from the Spinn3r corpus. To make sure that our dataset contains a sufficient number of relevant sentences, i.e., sentences that refer to scenarios from our collection, we selected texts that have a high affinity to at least one of these scenarios. We approximate this affinity using a logistic regression model fitted to texts from MCScript, based on LDA topics Blei et al. (2003) as features to represent a document.

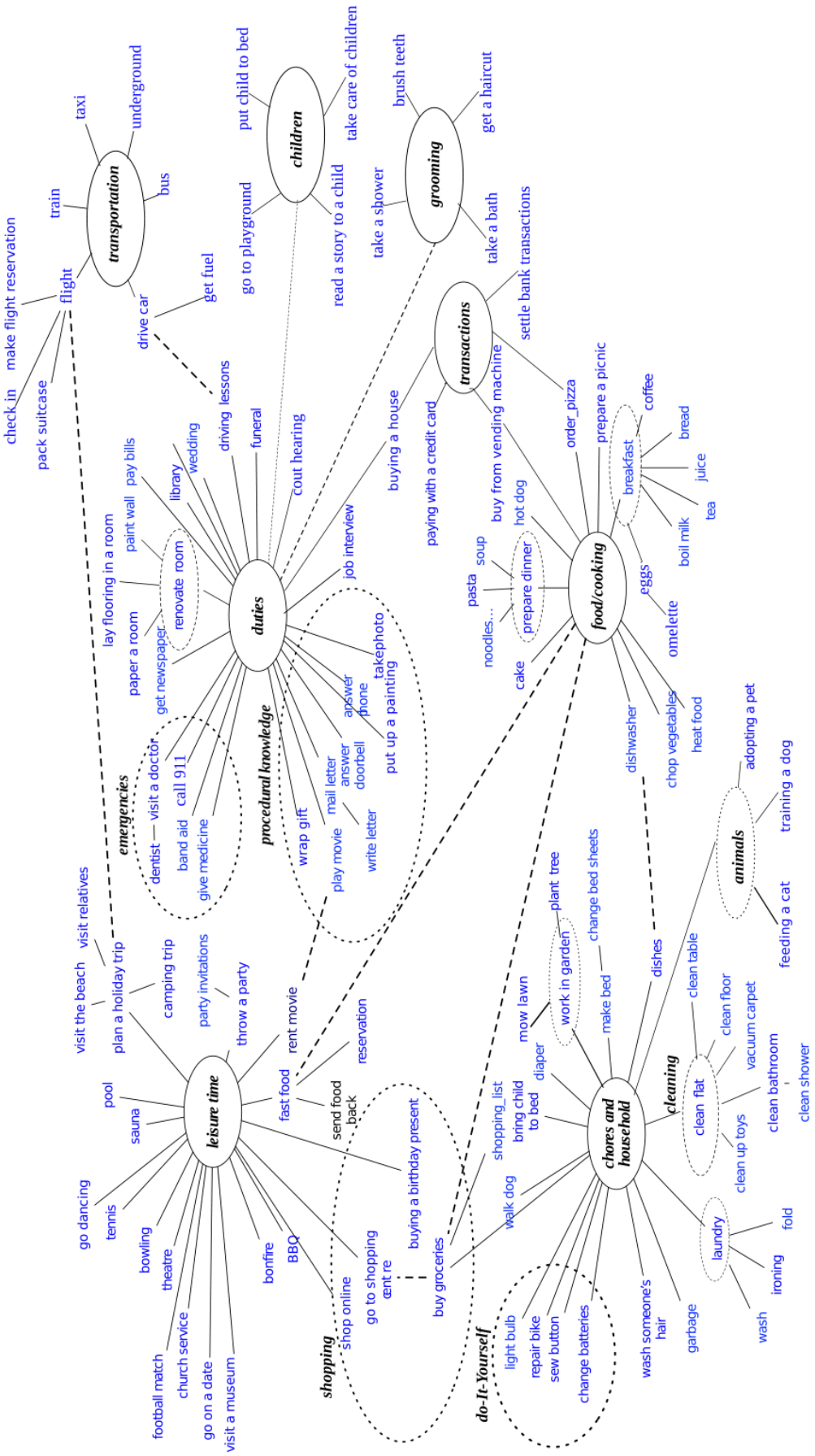


Figure 5.3: Figure illustrating the different classes covering various everyday activities.

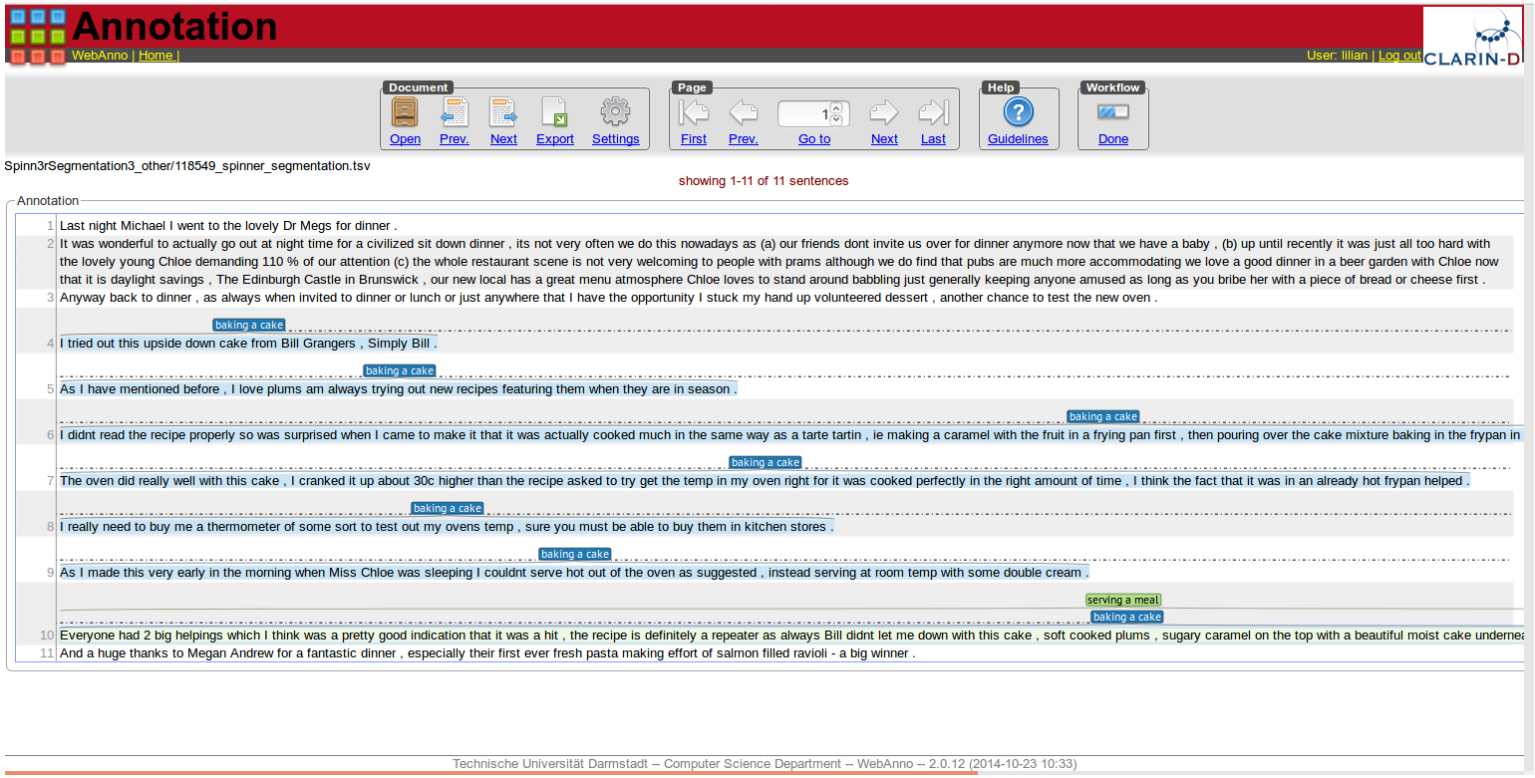


Figure 5.4: Example text annotation using Webanno page

Annotation. We follow standard methodology for natural language annotation (Pustejovsky and Stubbs, 2012). Each text is independently annotated by two annotators, student assistants, who use an agreed upon set of guidelines that is built iteratively together with the annotators . For each text, the students had to identify segments referring to a scenario from the scenario list, and assign scenario labels. If a segment refers to more than one script, they were allowed to assign multiple labels.

We worked with a total of four student assistants and used the Webanno[5] annotation tool (de Castilho et al., 2016). Figure 5.4 shows an example annotation for a give text. Lines 4 through 10 are annotated with `baking a cake` scenario. Line 10 is annotated with two scenario labels `baking a cake` and `serving a meal`[6].

Guidelines. We developed a set of more detailed guidelines for handling different issues related to the segmentation and classification, which is described in Appendix B. A major challenge when annotating segments is deciding when to count a sentence as referring to a particular scenario. For the task addressed here, we consider a segment only if it explicitly realizes aspects of script knowledge that go beyond an evoking expression (i.e., more than one event and participant need to be explicitly realized). Example 5.2 below shows a text segment with minimal scenario information for `going grocery shopping` with two events mentioned. In Example 5.3, only the evoking expression is mentioned, hence this example is not annotated.

Example 5.2. ✓*`going grocery shopping`*

*...We also **stopped at a small shop** near the hotel to **get some sandwiches** for dinner...*

Example 5.3. ✗*`paying for gas`*

*... A customer was heading for the store to **pay for gas** or whatever,...*

Scenario identification in text is also challenging in that people (writers) typically refer to several scenarios in a given text passage (annotation results will show that more than one scenario are referred to in texts (see Section 5.2.2)). The referenced scenarios can occur in different sections of the text and the text passages referring to the same scenario are not necessarily contiguous (see Figure 5.1). Another challenge is that segments can be associated with more than one scenario

[5]https://webanno.github.io/webanno/

[6]The labeling occurs several times for each sentence but that is just the Webanno presentation

5.2.2 Statistics

The annotators labeled 504 documents, consisting of 10,754 sentences. On average, the annotated documents are 35.74 sentences long. A scenario label could be either one of our 200 scenarios or "None" to capture sentences that do not refer to any of our scenarios.

Annotators	2	3	4
1	0.57 (*0.65*)	0.63 (***0.72***)	**0.64** (*0.70*)
2		0.62 (*0.71*)	0.61 (*0.70*)
3			0.62 (*0.71*)

Table 5.2: Kappa (*and raw*) agreement between pairs of annotators on sentence-level scenario labels

Agreement. To measure agreement, we looked at sentence-wise label assignments for each double-annotated text. We counted agreement if the same set of scenario labels is assigned to a sentence by both annotators. As an indication of chance-corrected agreement, we computed Kappa scores Cohen (1960). A kappa of 1 means that both annotators provided identical scenario label(s). When calculating raw agreements, we only counted whether there is (at least one) same scenario label assigned by both annotators. Table 5.2 shows the Kappa and raw (*in italics*) agreements for each pair of annotators. On average, the Kappa score was *0.61* ranging from 0.57 to 0.64. The average raw agreement score was *0.70* ranging from 0.65 to 0.72. The Kappa value indicates relatively consistent annotations across annotators even though the task was challenging.

We used fuzzy matching to calculate agreement in span between segments that overlap by at least one token. Table 5.3 shows pairwise % agreement scores between annotators. On average, the annotators achieve 67% agreement on segment spans. This shows considerable segment overlap when both annotators agreed that a particular scenario exists.

5.2.3 Adjudication and Gold Standard

In this section, we discuss differences in annotations and describe how we handle them in the adjudication in order to create the gold standard.

Figure 5.5 shows to what extent the annotators agreed in the scenario labels. The *None* cases accounted for 32% of the sentences. Although we selected stories with high affinity

Annotators	2	3	4
1	**78.8**	70.6	*59.3*
2		66.0	64.2
3			67.0

Table 5.3: Relative % agreement on segment spans between annotated segments that overlap by at least one token and are assigned the same scenario label

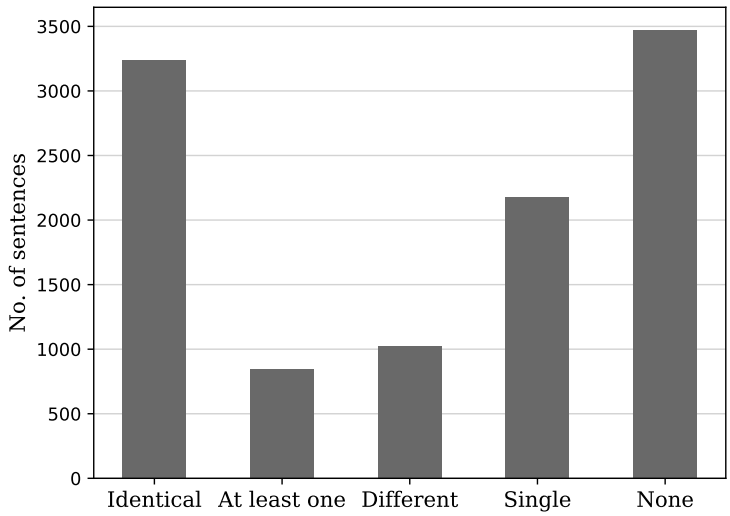


Figure 5.5: Absolute counts on sentence-level annotations that involve the same (Identical), overlapping (At least one) or disagreed (Different) labels; also shown are the number of sentences that received a label by only one annotator (Single) or no label at all (None).

to our scenarios, other scenarios (not in our scenario list) may still occur in the stories. Sentences referring to other scenarios were annotated as *None* cases. The *None* label was also used to label sentences that described topics related to but not directly part of the script being referenced. For instance, sentences not part of the narration, but of a different discourse mode (e.g. argumentation, report) or sentences where no specific script events are mentioned[7]. About 20% of the sentences had *Single* annotations where only one annotator indicated that there was a scenario reference. Single annotations were common for scenarios with flexible internal structure (e.g. `taking care of children` could potentially entail `feeding the child` or `changing the diaper`) scenarios. 47% of the sentences were assigned some scenario label(s) by both annotators (*Identical, At least one, Different*). Less than 10% of the sentences had *Different* scenario labels for the case where both

[7]See examples in Appendix B.

annotators assigned scenario labels to a sentence. This was common for scenarios that are closely related (e.g. `going to the shopping center, going shopping`) or scenarios in a sub-scenario relation (e.g. `flying in a plane, checking in at the airport`) that share script events and participants. 30% of the sentences had *Identical* scenario labels while in about 8% of the sentences, the both annotators agreed on *At least one* scenario label.

The annotation task is challenging, and so are gold standard creation and adjudication. We combined *automatic merging* and *manual adjudication* (by the main author of the paper) as two steps of gold-standard creation, to minimize manual post-processing of the dataset.

We automatically merged annotated segments that have identical scenario labels and overlap by at least one token, thus maximizing segment length. Consider the two annotations shown in Example 5.4. One annotator labeled the whole text as `growing vegetables`, the other one identified the two bold-face sequences as `growing vegetables` instances, and left the middle part out. The result of the merger is the maximal `growing vegetables` chain, i.e., the full text. Taking the maximal chain ensures that all relevant information is included, although the annotators may not have agreed on what is script-relevant.

Example 5.4. *growing vegetables*

The tomato seedlings Mitch planted in the compost box have done really well and we noticed flowers on them today. Hopefully we will get a good. *It has rained and rained here for the past month so that is doing the garden heaps of good. We bought some organic herbs seedlings recently and now have some thyme, parsley, oregano and mint growing in the garden.****We also planted some lettuce and a grape vine. We harvested our first crop of sweet potatoes a week or so ago (. . .)***

Since segment overlap is handled automatically, so manual adjudication must only care about label disagreement: the two main cases are (1) a segment has been labeled by only one annotator and (2) a segment has been assigned different labels by its two annotators. In case (1), the adjudicator had to take a binary decision to accept the labeled segment, or to discard it. In case (2), the adjudicator had three options: to decide for one of the labels or to accept both of them. The adjudication guidelines were deliberately designed in a way that the adjudicator could not easily overrule the double-annotations. The segmentation could not be changed, and only the labels provided by the annotators were available for labeling.

Scenario	#docs	#sents.	#segs.
eat in a restaurant	21	387	22
go on vacation	16	325	17
go shopping	34	276	35
take care of children	15	190	19
review movies	8	184	8
shop for clothes	11	182	12
work in the garden	13	179	17
prepare dinner	14	155	17
play a board game	8	129	12
attend a wedding	9	125	9
...			
taking a bath	3	34	6
borrow book from library	3	33	3
mow the lawn	3	33	3
drive a car	9	32	11
change a baby diaper	3	32	3
make omelette	3	32	3
play music in church	2	32	3
take medicine	5	31	6
get a haircut	3	30	3
heat food in a microwave	3	30	4
...			
replace a garbage bag	1	3	2
unclog the toilet	1	3	1
wash a cut	1	3	1
apply band aid	2	2	2
change batteries in alarm	1	2	1
clean a kitchen	1	2	1
feed the fish	1	2	1
set an alarm	1	2	1
get ready for bed	1	1	1
set the dining table	1	1	1

Table 5.4: Distribution of scenario labels over documents (docs), sentences (sents) and segments (segs); the top and bottom parts show the ten most and least frequent labels, respectively. The middle part shows scenario labels that appear at an average frequency.

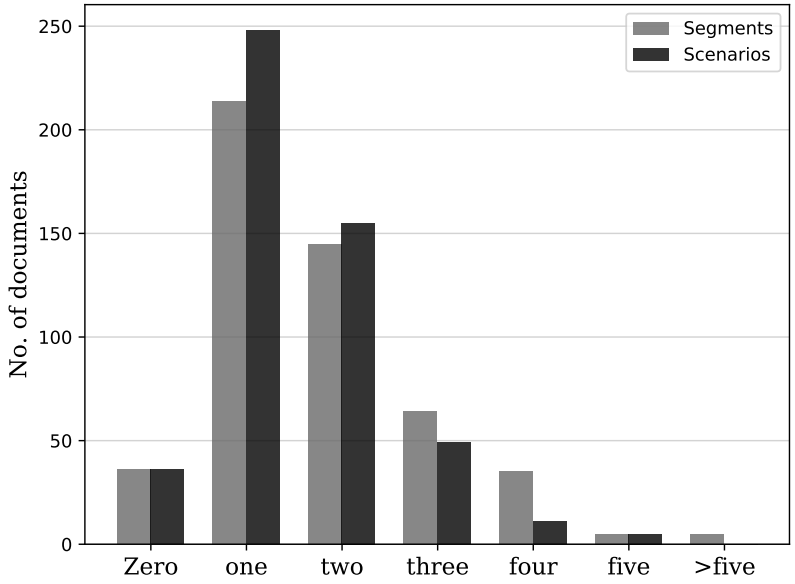


Figure 5.6: Segment and scenario distribution per text

The adjudication guidelines are described in Appendix C.

Gold standard. The annotation process resulted in 2070 single segment annotations. 69% of the single segment annotations were automatically merged to create gold segments. The remaining segments were adjudicated, and relevant segments were added to the gold standard. Our final dataset consists of 7152 sentences (contained in 895 segments) with gold scenario labels. From the 7152 gold sentences, 1038 (15%) sentences have more than one scenario label. 181 scenarios (out of 200) occur as gold labels in our dataset, 179 of which are referred to in at least 2 sentences. Table 5.4 shows example scenarios and the distribution of scenario labels: the number of documents that refer to the given scenario, the number of gold sentences and segments referring to the given scenario, and the average segment length (in sentences) per scenario. 16 scenarios are referred to in more than 100 gold sentences, 105 scenarios in at least 20 gold sentences, 60 scenarios in less than 20 gold sentences. Figure 5.6 shows the distribution of segments and scenario references per text in the gold standard. On average, there are 1.8 segments per text and 44% of the texts refer to at least two scenarios. The segments are on average 9.4 sentences long (standard deviation of 8.47, median of 7). The high standard deviation indicates that segments vary a lot in length, ranging between 1 to about 50 sentences.

In order to have a high level picture of the script verbs used in the gold segments, we extracted all verb lemmas and calculated script relevance based on *tf.idf*. *tf.idf* scores indicate

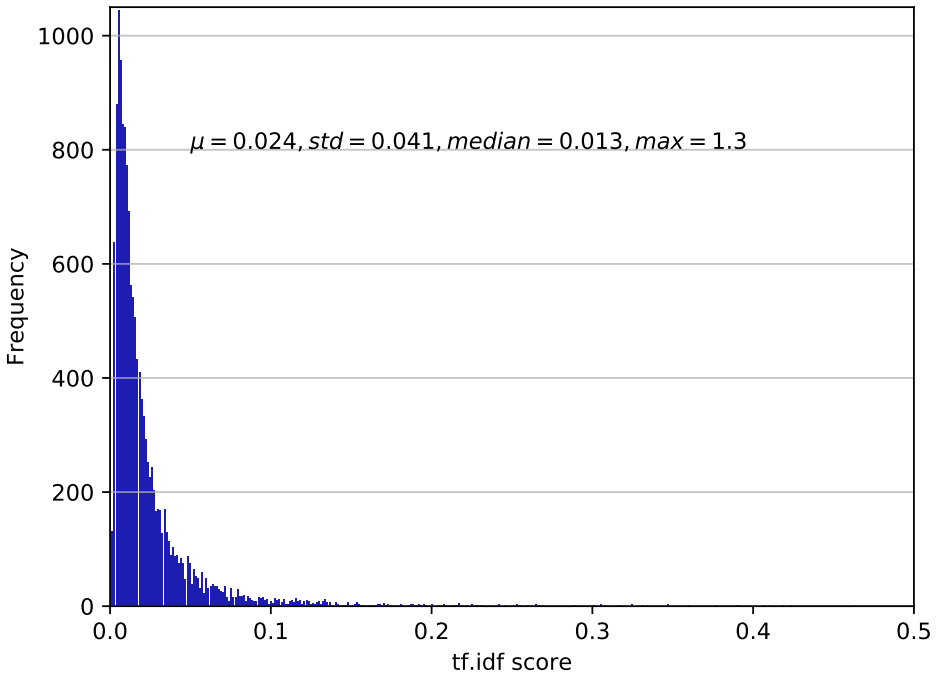


Figure 5.7: Script relevance of words measured using *tf.idf*

how relevant a word for the scenario. Figure 5.7 shows a histogram plotting script relevance of verbs as calculated using *tf.idf*. We can see that the histogram is right-skewed. Most verb mentions are not necessarily relevant to the script being referenced in the segments. As a qualitative analysis, Table 5.5 shows the top 10 words with highest *tf.idf* score for sample scenarios. As can be seen, most top verbs are script relevant. Some general words (*in italics*) with no specific relation to the scenarios also end up in the top. Irrelevant verbs are rare to find in the top most.

Scenarios	**Verb lemmas**									
taking care of children	babysit	wake	play	sleep	feed	watch	nap	cry	bathe	write
going shopping	buy	try	shop	wear	pick	fit	*wouldn*	win	*want*	wander
growing vegetables	plant	grow	harvest	rain	mix	water	twirl	twine	transplant	till
eating in a restaurant	order	eat	tell	serve	seat	pay	cook	boil	write	*weren*
renovating a room	paint	rip	prime	move	gloss	cover	update	scrape	sand	retile
looking for a flat	rent	live	find	search	provide	meet	face	ensure	determine	update
cleaning up a flat	clean	scrub	wipe	move	vacuum	mop	wash	smell	sweep	dust
planning a holiday trip	book	plan	visit	drive	compare	validate	stay	research	redeem	*poke*
adopting a pet	adopt	purr	kill	declaw	temper	submit	specialize	*seem*	rescue	receive
making a bonfire	spark	ignite	pile	light	engulf	expect	*dream*	*conclude*	*communicate*	catch

Table 5.5: Top 10 verb lemmas (Ordered according to decreasing *tf.idf* scores)

5.3 Summary

In this Chapter, we introduced the task of scenario identification in narrative texts and curated a benchmark dataset for automatic scenario segmentation and identification. The dataset is the first of its kind. The analysis of the annotation shows that one can identify scenario references in text with reasonable reliability. A qualitative analysis on the gold segments also indicates that a high number of script-relevant verbs are mentioned in the texts.

Chapter 6

Automatic detection of everyday scenarios in narrative texts

In Chapter 5, we introduced and defined the task of scenario identification and curated a benchmark dataset of annotated narrative texts, with segments labeled according to the scripts they instantiate.

In this Chapter, we describe first steps towards the automatic segmentation and detection of everyday scenarios in narrative texts. We propose a benchmark model for the automatic detection and classification of text segments that instantiate script knowledge in Section 6.1. We present a two-stage model that combines established methods from topic segmentation and text classification (Section 6.1). We show that the proposed model achieves promising results but also reveals some of the difficulties underlying the task of scenario detection (Section 6.2).

6.1 Benchmark model

In this section, we present a model for scenario identification, which is simple in several respects: we propose a two-step model consisting of a segmentation and a classification component. For segmentation, we assume that a change in scenario focus can be modeled by a shift in lexical cohesion. We identify segments that might be related to specific

scripts or scenarios via topic segmentation, assuming that scenarios can be approximated as distributions over topics. After segmentation, a supervised classifier component is used to predict the scenario label(s) for each of the found segment. Our results show that the script segmentation problem can be solved in principle, and we propose our model as a benchmark model for future work.

Segmentation. The first component of our benchmark model reimplements a state-of-art unsupervised method for topic segmentation, called TopicTiling (Riedl and Biemann, 2012). TopicTiling (TT) uses latent topics inferred by a Latent Derichlet Allocation (LDA, Blei et al. (2003)) model to identify segments (i.e., sets of consecutive sentences) referring to similar topics.[1] The TT segmenter outputs topic boundaries between sentences where there are topic shifts. Boundaries are computed based on coherence scores. Coherence scores close to 1 indicate significant topic similarity while values close to 0 indicate minimal topic similarity. A window parameter is used to determine the block size i.e. the number of sentences to the left and right that should be considered when calculating coherence scores. To discover segment boundaries, all local minima in the coherence scores are identified using a depth score (Hearst, 1994). A threshold $\mu - \sigma/x$ is used to estimate the number of segments, where μ is the mean and σ is the standard deviation of the depth scores, and x is a weight parameter for setting the threshold.[2] Segment boundaries are placed at position greater than the threshold.

Classification. We view the scenario classification subtask as a supervised multi-label classification problem. Specifically, we implement a multilayer perceptron classifier in Keras Chollet et al. (2015) with multiple layers: an input layer with 100 neurons and ReLU activation, followed by an intermediate layer with dropout (0.2), and finally an output layer with sigmoid activations. We optimize a cross-entropy loss using adam. Because multiple labels can be assigned to one segment, we train several one-vs-all classifiers, resulting in one classifier per scenario.

We also experimented with different features to represent text segments: term frequencies weighted by inverted document frequency (*tf.idf*, Salton and McGill (1986))[3] and topic

[1] We used the Gensim Rehurek and Sojka (2010) implementation of LDA.

[2] We experimentally set x to 0.1 using our held out development set.

[3] We use SciKit learn Pedregosa et al. (2011) to build *tf.idf* representations

features derived from LDA (see above), and we tried to work with word embeddings. We found the performance with *tf.idf* features to be the best.

6.2 Experiments

The experiments and results presented in this section are based on our annotated dataset for scenario detection described in Chapter 5.

6.2.1 Experimental setting

Preprocessing and model details. We represent each input to our model as a sequence of lemmatized content words, in particular nouns and verbs (including verb particles). This is achieved by preprocessing each text using Stanford CoreNLP (Chen and Manning, 2014).

Segmentation. Since the segmentation model is unsupervised, we can use all data from both MCScript and the Spinn3r personal stories corpora to build the LDA model. As input to the TopicTiling segmentor, each sentence is represented by a vector in which each component represents the (weight of a) topic from the LDA model (i.e. the value of the i^{th} component is the normalized weight of the words in the sentence whose most relevant topic is the i^{th} topic). For the segmentation model, we tune the number of topics (200) and the window size (2) based on an artificial development dataset, created by merging segments from multiple documents from MCScript.

Classification. We train the scenario classification model on the scenario labels provided in MCScript (one per text). For training and hyperparameter selection, we split MCScript dataset (see Section 5.1) into a training and development set, as indicated in Table 6.1. We additionally make use of 18 documents from our scenario detection data (Section 5.2) to tune a classification threshold. The remaining 486 documents are held out exclusively for testing (see Table 6.1). Since we train separate classifiers for each scenario (one-vs-all classifiers), we get a probability distribution of how likely a sentence refers to a scenario. We use entropy to measure the degree of scenario content in the sentences. Sentences with entropy values higher than the threshold are considered as not referencing any scenario (None cases), while sentences with lower entropy values reference some scenario.

Dataset	#train	#dev	#test
MCScript	3492	408	-
Spinn3r (gold)	-	18	486

Table 6.1: Datasets (number of sentences) used in the experiments

model	Precision	Recall	F_1-score
sent_maj	0.08	0.05	0.06
sent_*tf.idf*	0.24	0.28	0.26
random_*tf.idf*	0.32	0.45	0.37
TT_*tf.idf*	0.36	**0.54**	**0.43**
TT_*tf.idf* (gold)	0.54	0.54	0.54

Table 6.2: Results for the scenario detection task

Baselines. We experiment with three baselines: As a lower bound for the classification task, we compare our model against the baseline *sent_maj*, which assigns the majority label to all sentences. To assess the utility of segmentation, we compare against two baselines that use our proposed classifier but not the segmentation component: the baseline *sent_tf.idf* treats each sentence as a separate segment and *random_tf.idf* splits each document into random segments.

Evaluation. We evaluate scenario detection performance at the sentence level using micro-average precision, recall and F_1-score. We consider the top 1 predicted scenario for sentences with only one gold label, and top n scenarios for sentences with n gold labels. For sentences with multiple scenario labels, we take into account partial matches and count each label proportionally. Assuming the gold labels are `washing ones hair` and `taking a bath`, and the classifier predicts `taking a bath` and `getting ready for bed`. `Taking a bath` is correctly predicted and accounts for 0.5 true positive (TP) while `washing ones hair` is incorrectly missed, thus accounts for 0.5 false negative (FN). `Getting ready for bed` is incorrectly predicted and accounts for 1 false positive (FP).

6.2.2 Results

We present the micro-averaged results for scenario detection in Table 6.2. The *sent_maj* baseline achieves a F_1-score of only 6%, as the majority class forms only a small part of

the dataset (4.7%). Our TT model with *tf.idf* features surpasses both baselines that perform segmentation only naively (26% F_1) or randomly (37% F_1). This result shows that scenario detection works best when using predicted segments that are informative and topically consistent.

We estimated an upper bound for the classifier by taking into account the predicted segments from the segmentation step, but during evaluation, only considered those sentences with gold scenario labels (TT_*tf.idf* (Gold)), while ignoring the sentences with "None" label. We see an improvement in precision (54%), showing that the classifier correctly predicts the right scenario label for sentences with gold labels while also including other sentences that may be in topic but not directly referencing a given scenario.

feature	top1	top2	top3	top4	top5
*tm*_Spinn3r+McScript	0.65	0.82	0.88	0.91	0.93
*tm*_McScript	0.59	0.77	0.84	0.88	0.89
*tm*_combined	0.72	0.88	0.94	0.95	0.97
*tf.idf*_Spinn3r+McScript	0.83	*0.97*	0.98	0.99	0.99
*word_emb*_fastText	*0.86*	0.95	0.97	0.98	0.98

Table 6.3: Classifier performance (Accuracy, top5 predictions) on MCScript dev test (topic (*tm*), *tf.idf* and word embeddings (*word_emb*) features).

model	Precision	Recall	F_1-score
TT_*tm*	0.18	0.30	0.22
TT_*word_emb*_fastText	0.23	0.40	0.30
TT_*tf.idf*	**0.36**	**0.54**	**0.43**

Table 6.4: Comparing different features and classifiers for the scenario detection task on gold segments (topic (*tm*), *tf.idf* and word embeddings (*word_emb*) features).

Table 6.3 gives the accuracy results of the classifier as tested on the MCScript dev set using different features and classifiers. *tm*_Spinn3r+McScript is based on the topic model build using both Spinn3r and MCScript corpus, *tm*_McScript is based on the topic model build on only MCScript dataset while *tm*_combined combines both topic features. *tf.idf* features are obtained from the combined corpus (Spinn3r+McScript). The first 4 models use our classifier described in Section 6.1 while *word_emb*_fastText uses fastText library (Joulin et al., 2016) for classification. We can see that *tf.idf* and *word_emb* features are quite superior to topic features with the fastText model having the best top1 result (0.86). Although

the fastText model performed well on the MCScript dev set, we found the performance with *tf.idf* features to be the best when applied to Spinn3r documents (see Table 6.4).

To estimate the performance of the TT segmentor individually, we run TT on an artificial development set, created by merging segments from different scenarios from MCScript. We evaluate the performance of TT by using a standard topic segmentation evaluation metric, P_k (Beeferman et al., 1999). P_k metric express the probability of segmentation error, thus lower values indicate better performance. It uses a fixed window that slides across the text. Figure 6.1 visualizes how P_k metric is calculated by comparing the reference segmentation to an hypothesized segmentation. Windows (a) and (d) are acceptable (count as 1) as both reference and hypothesized boundaries lie or do not lie within the window. Windows (b) and (c) are not accountable (count as 0) as (b) is false negative and (c) is false positive. The final P_k score is got by summing up the scores for each window and dividing by the number of windows. We computed the average performance over several runs. TT attains P_k of 0.28. The low segmentation error suggest that TT segmentor does a good job in predicting the boundaries between scenarios in text.

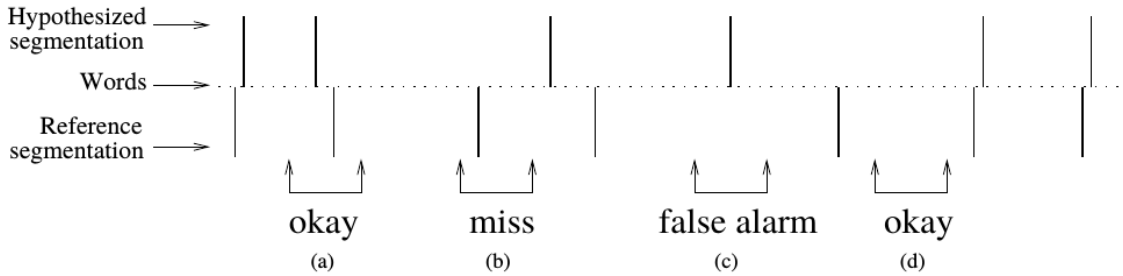


Figure 6.1: Visualizing P_k segment evaluation metric (Beeferman et al., 1999)

6.2.3 Discussion

Even for a purpose-built model, scenario detection is a difficult task. This is partly to be expected as the task requires the assignment of one (or more) of 200 possible scenario labels, some of which are hard to distinguish. Many errors are due to misclassifications between scenarios that share script events as well as participants, that are usually mentioned in the same text: for example, `sending food back in a restaurant` requires and involves participants from `eating in a restaurant`. Table 6.5 shows the 10 most frequent misclassifications by our best model *TT_tf.idf*. These errors account for 16% of all incorrect

True label	Predicted label	No of sents.	PMI
go_vacation	visit_sights	92	3.96
eat_restaurant	food_back	67	4.26
work_garden	grow_vegetables	57	4.45
attend_wedding	prepare_wedding	48	4.12
eat_restaurant	dinner_reservation	39	4.26
throw_party	go_party	36	4.09
shop_online	order_ on_phone	35	3.73
work_garden	plant_a_tree	33	4.81
shop_clothes	check_store_open	33	0.00
play_video_games	learn_board_game	32	0.00

Table 6.5: The 10 most common types of classification errors by our approach TT_*tf.idf*

label assignments (200 *by* 200 matrix). The 100 most frequent misclassifications account for 63% of all incorrect label assignments. In a quantitative analysis, we calculate the commonalities between scenarios in terms of the pointwise mutual information (PMI) between words in the associated stories. On one side, we observe that the scenario-wise recall and F_1-scores of our classifier are negatively correlated with PMI scores (Pearson correlation of -0.33 and -0.17, respectively). These correlations confirm a greater difficulty in distinguishing between scenarios that are highly related to other scenarios.

Scenario	No. of sents	Precision	Recall	F_1-score
going to the dentist	47	0.90	0.96	0.93
having a barbecue	43	0.92	0.88	0.90
going to the sauna	28	0.80	0.89	0.84
making soup	60	0.81	0.87	0.84
baking a cake	69	0.71	0.97	0.82
going skiing	42	0.78	0.83	0.80
attending a court hearing	66	0.71	0.92	0.80
cleaning the floor	6	1.00	0.67	0.80
taking a taxi	27	0.74	0.85	0.79
attending a church service	60	0.70	0.92	0.79

Table 6.6: Top 10 scenario-wise results using our approach TT_*tf.idf*

On the other side, we observe that scenario-wise precision and F_1-score are positively correlated with the number of gold sentences annotated with the respective scenario label (Pearson correlation of 0.50 and 0.20, respectively). As one would expect, our approach seems to perform better on scenarios that appear at higher frequency. Table 6.6 shows the 10 scenarios for which our approach achieves the best results. 96 scenarios have F-Scores

above 40%, 17 scenarios have F-Scores between 30% and 40% while 68 scenarios have F-Scores below 30%.

Scenario approximation using topics. We performed an analyses to qualitatively examine how far topic distributions actually approximate scenarios. We computed a LDA topic model using only the MCScript dataset. We created *scenario-topics* by looking at all the prevalent topics in documents from a given scenario. Table 6.7 shows the top 10 words for each scenario extracted from the *scenario-topics*. As can be seen, the topics capture some of the most relevant words for different scenarios.

order pizza	do laundry	work in garden	have barbecue	change light bulb
pizza	clothes	tree	invite	light
order	dryer	plant	guest	bulb
delivery	laundry	hole	grill	switch
decide	washer	water	friend	shade
place	wash	grow	everyone	lightbulb
deliver	dry	garden	beer	screw
tip	white	dig	barbecue	remove
phone	detergent	dirt	food	turn
number	start	seed	serve	fixture
minute	washing	soil	season	socket

Table 6.7: Example top 10 scenario-words

Similarly, we conducted an analysis to determine the extent to which our set of scenarios are related. We measured scenario similarity by taking into account the proportion of scenario-topics shared between scenarios. We automatically build clusters of related scenarios using spectral clustering. Figure 6.2 shows some example scenario clusters derived using *scenario-topic* features. We can observe meaningful clusters e.g. `cleaning dishes` cluster, `gardening` cluster, `shopping` cluster, `taking care of children` cluster, among others. This analysis supports the use of latent topics as an approximation of scenarios. We used these clusters as a basis for grouping the scenarios into 14 classes for the purpose of annotation (see Section 5.2).

6.3 Summary

In this chapter, we proposed a benchmark model that automatically segments and identifies text fragments referring to a given scenario. While our model achieves promising first re-

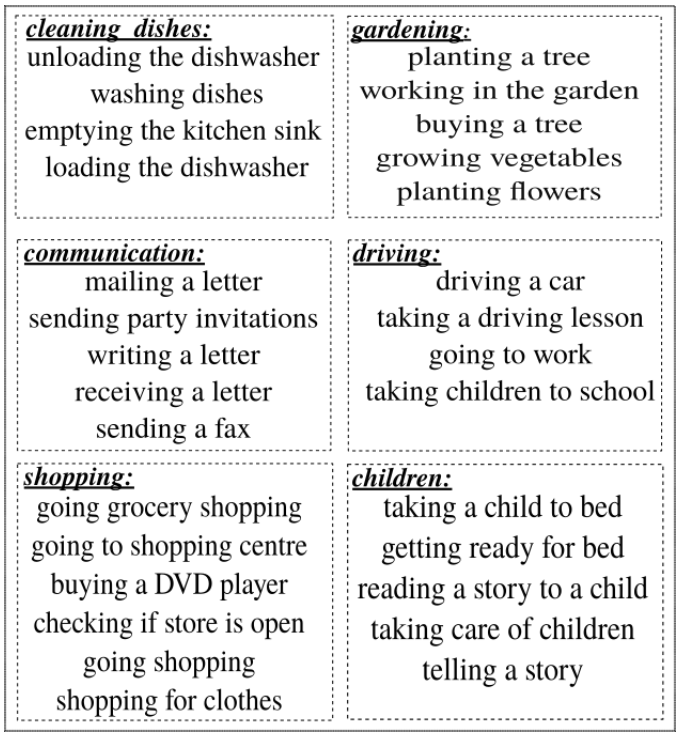


Figure 6.2: Example automatically extracted scenario clusters

sults, it also revealed some of the difficulties in detecting script references. Script detection is an important first step for large-scale data driven script induction for tasks that require the application of script knowledge. We are hopeful that our data and model will form a useful basis for future work.

Chapter 7

Conclusion and Outlook

In this chapter, we summarize the main contributions of this dissertation (Section 7.1) and provide an outlook around script-related tasks that can benefit from incorporating script-knowledge and identification of scenarios in texts (Section 7.2).

7.1 Thesis summary

This dissertation has taken measures to provide sound empirical basis for high-quality script modeling. In **Chapters 1** and **2**, we motivated Scripts as a rich way of representing knowledge about everyday stereotypical activities and with examples, showed that Script knowledge guides expectations in text understanding and makes missing events and referents in a discourse accessible. In order to build script models for natural language understanding (NLU), we first need to induce the script structure from corpora. Script induction forms a basis for script parsing i.e. associating text with script structure given a scenario. Second, we need to perform script identification in text which is a prerequisite for script parsing and script-based NLU. In the following section, we summarize our main contributions in three key areas: script-knowledge acquisition, script induction and scenario identification in text.

Script-knowledge acquisition. In order to provide quality data for high-quality script models, this dissertation extends existing script corpora in two different ways as described in **Chapter 3**. First, we crowdsourced a corpus of 40 scenarios with 100 event sequence descriptions (ESDs) each, thus going beyond the size of previous script collections. Our collected corpus covers a wide range of scenarios ranging from simpler ones to ones that show interesting variation with regard to their granularity, to the events described, and to different verbalizations of the same event within a scenario. Second, we enriched the corpus with partial alignments information on difficult event descriptions, done by human annotators. We further built gold event-clusters by having full alignments by experts for 10 different scenarios (50 event sequence descriptions per scenario), grouped into labeled paraphrase sets, to be used in the evaluation of semi-supervised script induction from event sequence descriptions (**Chapter 4**).

Inducing script-knowledge. In **Chapter 4**, we presented a semi-supervised clustering approach to induce script structure from crowdsourced descriptions of event sequences by grouping event descriptions into paraphrase sets (representing event types) and inducing their temporal order. Our proposed approach exploits semantic and positional similarity and allows for flexible event order, thus overcoming the rigidity of previous approaches. We incorporated crowdsourced alignments as prior knowledge and showed that exploiting a small number of alignments results in a substantial improvement in cluster quality over state-of-the-art models and provides an appropriate basis for the induction of temporal order. We addressed two main tasks that robust script models need to handle: event paraphrasing and event ordering. On the paraphrase task, our approach substantially outperforms all previous proposals, while still performing very well on the task of temporal order prediction.

One major challenge of bottom-up script modeling is the issue of coverage. We conducted a coverage study to demonstrate the scalability of our approach. A study on the ROC-stories (Mostafazadeh et al., 2016) suggested that existing repositories of script knowledge cover already a large fraction of event structures occurring in topically unrestricted narrative texts, thus demonstrating the scalability of our approach.

Script identification in texts. We introduced the task of scenario detection and described first steps towards identification and labeling of text segments with the specific scenarios they instantiate (**Chapter 5**). Texts typically consist of different passages that refer to different scenarios. When human hearers or readers come across an expression that evokes a particular script, they try to map verbs or clauses in the subsequent text to script events, until they face lexical material that is clearly unrelated to the script and may evoke a different scenario. Thus, script detection is an important first step for large-scale data driven script induction for tasks that require the application of script knowledge. In **Chapter 5**, we describe our curated benchmark dataset of annotated narrative texts, with segments labeled according to the scripts they instantiate, for automatic scenario segmentation and identification. The analysis of the annotation shows that one can identify scenario references in text with reasonable reliability. As a result, the labeled scenario-specific text fragments provide themselves as additional linguistic resources from where script-knowledge can be acquired. To the best of our knowledge, our dataset is the first of its kind. In **Chapter 6**, we further proposed a benchmark model to automatically segment and identify text fragments referring to a given scenario. While our model achieved promising results, it also opens up a perspective for wider training material for script parsing and script-based natural language understanding.

7.2 Outlook

In this dissertation, we have motivated the importance of script-knowledge for natural language understanding and looked at the major areas that enable systems to utilize script-knowledge for NLP. In this section, we discuss possible areas of research around tasks that can benefit from incorporating script-knowledge and identification of scenarios in text. We expect that script knowledge will enable future work in many applications including but not limited to script parsing, semantic role labeling, coreference resolution, machine comprehension among others. Below we highlight some possible ares of research.

Script parsing. Script parsing is the task of mapping events and objects in texts to their respective script events and participants. First work on script parsing has been done by Kampmann et al. (2015) and Ostermann et al. (2017). Naturalistic narrative texts

typically refer to more than one scenario and scenario identification in a prerequisite to broad coverage script parsing. We expect that our work on scenario detection in texts and enables future work on automatic script parsing involving more complex texts that refer to multiple scenarios.

Semantic role labeling. Semantic role labeling (SRL) is the task of identifying the semantic relations between a predicate and its associated arguments. Previous work view SRL as a sentence-internal task and do not take into account context beyond the sentence being analyzed. In reality, semantic arguments can interact at discourse level beyond sentence level. Notable previous work on SRL that try to resolve implicit arguments at discourse level include (Ruppenhofer et al., 2010; Gerber and Chai, 2012). With this in mind, script-knowledge acquired in this dissertation can help resolve predicates and participants at discourse level and improve SRL. Likewise, Roth and Lapata (2015) showed that additional context information is useful for SRL. No script information has been applied so far to SRL. It would be attractive to use methods for scenario identification in text proposed in this dissertation to provide additional context information to improve SRL.

Coreference resolution. Script knowledge has been shown to be useful for predicting upcoming discourse referents (Modi et al., 2017), which is a closely related work to coreference resolution. Coreference resolution systems can benefit from incorporating script knowledge to untangle event and participant mentions. Script-knowledge resources proposed in this dissertation can be used to improve models for coreference resolution.

Image captioning. Automatic image captioning is the task of accurately describing a complex scene in an image with natural language expressions (Chen et al., 2015). Given a restaurant scene, it would not be useful to plainly mention the people in the image (e.g. person sitting or standing) but rather provide more information about the scene (e.g. waiter taking an order). Script-knowledge resources proposed in this dissertation can be used to enrich image captioning models and provide deeper understanding of complex scenes.

Story generation. Script knowledge has been used in building plot graphs for story generation (Li et al., 2013), in particular for building domain specific knowledge graphs. Zhai et al. (2019) showed the applicability of flexible temporal script graphs proposed in this dissertation (Chapter 4) for globally coherent story generation. They worked on crowdsourced simplistic texts referring to only single scenarios (Modi et al., 2016). It would be interesting to explore scenario identification proposed in this dissertation together with script parsing for story generation using more naturalistic texts.

To the best of our knowledge, the kind of resources proposed in the dissertation have not been utilized as detailed above and we expect the outcome of this work will substantially improve models for coreference resolution, machine comprehension, story generation, automatic image captioning, among others.

List of Figures

List of Tables

Bibliography

Omri Abend, Shay B Cohen, and Mark Steedman. Lexical event ordering with an edge-factored model. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1161–1171, 2015.

James Allan. *Topic detection and tracking: event-based information organization*, volume 12. Springer Science & Business Media, 2012.

James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study final report. 1998.

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12 (4):461–486, 2008. ISSN 1573-7659.

Shmuel Asafi and Daniel Cohen-Or. Constraints as features. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 1634–1641, 2013.

Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, 1998.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *Proceedings of the 17th International Conference on Computational Linguistics -*

Volume 1, COLING '98, pages 86–90, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

Avron Barr and Edward A Feigenbaum. Frames and scripts. In *The Handbook of Artificial Intelligence*, volume 3, pages 216–222. Addison-Wesley, California, 1981.

Jeff Barr and Luis-Felipe Cabrera. AI gets a brain. *ACM Queue*, 4:24–29, 2006.

Doug Beeferman, Adam Berger, and John Lafferty. Statistical models for text segmentation. *Machine learning*, 34(1-3):177–210, 1999.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Jelke Bloem, Michaela Regneri, and Stefan Thater. Robust processing of noisy web-collected data. In *KONVENS*, pages 189–193, 2012.

Antoine Bosselut, Jianfu Chen, David Warren, Hannaneh Hajishirzi, and Yejin Choi. Learning prototypical event structure from photo albums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1769–1779, 2016.

Lorne Bozinoff. A script theoretic approach to informatin processing: an energy conservation application. *in NA - Advances in Consumer Research Volume 09, eds. Andrew Mitchell, Ann Abor, MI : Association for Consumer Research*, pages 481–486, 1982.

Daren C. Brabham. Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence*, 14(1):75–90, 2008.

Tom J. Brown. Schemata in consumer research: a connectionist approach. *in NA - Advances in Consumer Research Volume 19, eds. John F. Sherry, Jr. and Brian Sternthal, Provo, UT : Association for Consumer Research.*, pages 787–794, 1992.

Steven Burrows, Martin Potthast, and Benno Stein. Paraphrase acquisition via crowdsourcing and machine learning. *ACM Trans. Intell. Syst. Technol.*, 4:43:1–43:21, July 2013.

Kevin Burton, Akshay Java, and Ian Soboroff. The ICWSM 2009 Spinn3r Dataset. In *Third Annual Conference on Weblogs and Social Media (ICWSM 2009)*, San Jose, CA, May 2009. AAAI.

Nathanael Chambers and Daniel Jurafsky. Unsupervised learning of narrative event chains. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 789–797, 2008.

Nathanael Chambers and Daniel Jurafsky. Unsupervised learning of narrative schemas and their participants. In *Proceedings of 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 602–610, Suntec, Singapore, 2009.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

David L Chen and William B Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 190–200. Association for Computational Linguistics, 2011.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

Cuong Xuan Chu, Niket Tandon, and Gerhard Weikum. Distilling task knowledge from how-to communities. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 805–814, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

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

Richard Edward Cullingford. *SAM: In Inside Computer Understanding: Five Programs Plus Miniatures*. Hillsdale, New Jersey: Lawrence Erlbaum Associates., 1981.

Carmen Dalli. Scripts for children's lives: What do parents and early childhood teachers contribute to children's understanding of events in their lives., 09 1991.

Ian Davidson and Sugato Basu. A survey of clustering with instance level. *Constraints*, 1: 2, 2007.

Ian Davidson, Kiri L. Wagstaff, and Sugato Basu. *Knowledge Discovery in Databases: PKDD 2006: 10th European Conference on Principles and Practice of Knowledge Discovery in Databases Berlin, Germany, September 18-22, 2006 Proceedings*, chapter Measuring Constraint-Set Utility for Partitional Clustering Algorithms, pages 115–126. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006a. ISBN 978-3-540-46048-0.

Ian Davidson, Kiri L Wagstaff, and Sugato Basu. Measuring constraint-set utility for partitional clustering algorithms. In *Knowledge Discovery in Databases: PKDD 2006*, pages 115–126. Springer, 2006b.

Ernest Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

Richard Eckart de Castilho, Eva Mujdricza-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, 2016.

Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

Samuel Fernando and Mark Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, 2008.

J.L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

Lea Frermann, Ivan Titov, and Manfred Pinkal. A hierarchical Bayesian model for unsupervised induction of script knowledge. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 49–57, Gothenburg, Sweden, 2014.

Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *Science Direct*, 315(5814):972–976, 2007.

Victoria Fromkin and Robert Rodman. *An introduction to language*. Harcourt Brace College Publishers, 1998.

Matthew Gerber and Joyce Y Chai. Semantic role labeling of implicit arguments for nominal predicates. *Computational Linguistics*, 38(4):755–798, 2012.

Inmar E Givoni and Brendan J Frey. Semi-supervised Affinity Propagation with instance-level constraints. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 161–168, 2009.

Andrew Gordon and Reid Swanson. Identifying personal stories in millions of weblog entries. In *International Conference on Weblogs and Social Media, Data Challenge Workshop, May 20, San Jose, CA*, 2009.

Andrew S. Gordon. Browsing image collections with representations of common-sense activities. *Journal of the American Society for Information Science and Technology.*, 52: 925, 2001.

Andrew S. Gordon. Mining commonsense knowledge from personal stories in internet weblogs. In *Proceedings of the First Workshop on Automated Knowledge Base Construction*, Grenoble, France, May 2010.

Jonathan Gordon and Benjamin Van Durme. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 25–30, New York, NY, USA, 2013. ACM.

Jonathan Gordon, Benjamin Van Durme, and Lenhart K. Schubert. Weblogs as a source for extracting general world knowledge. In *Proceedings of the 5th International Con-*

ference on Knowledge Capture (K-CAP 2009), September 1-4, 2009, Redondo Beach, California, USA, pages 185–186, 2009.

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

Rakesh Gupta and Mykel J. Kochenderfer. Common sense data acquisition for indoor mobile robots. In *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI'04, pages 605–610. AAAI Press, 2004.

Rakesh Gupta and Vasco Calais Pedro. Knowledge representation and bayesian inference for response to situations. In *AAAI 2005 Workshop on Link Analysis*, 2005.

C. Havasi, R. Speer, and J. Alonso. Conceptnet 3: a flexible, multilingual semantic network for common sense knowledge. In *Recent Advances in Natural Language Processing*, Borovets, Bulgaria, September 2007.

Marti A Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 9–16. Association for Computational Linguistics, 1994.

John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65, 2001.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. Skip N-grams and Ranking Functions for Predicting Script Events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344, 2012.

Mark F John. The story gestalt: A model of knowledge-intensive processes in text comprehension. *Cognitive Science*, 16(2):271–306, 1992.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Alexander Kampmann, Stefan Thater, and Manfred Pinkal. A case-study of automatic participant labeling. In *GSCL*, pages 97–105, 2015.

Niels Kasch and Tim Oates. Mining script-like structures from the web. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 34–42, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 307–314, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-873-7.

Lars Kunze, Moritz Tenorth, and Michael Beetz. Putting people's common sense into knowledge bases of household robots. In Rüdiger Dillmann, Jürgen Beyerer, Uwe D. Hanebeck, and Tanja Schultz, editors, *KI 2010: Advances in Artificial Intelligence*, pages 151–159, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

Yaniv Leviathan and Yossi Matias. Google duplex: An AI system for accomplishing real-world tasks over the phone, May 8 2018. URL `https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html`.

Boyang Li. *Learning Knowledge to Support Domain-Independent Narrative Intelligence*. PhD thesis, Georgia Institute of Technology, 2015.

Boyang Li, Stephen Lee-Urban, D. Scott Appling, and Mark O. Riedl. Crowdsourcing narrative intelligence. volume vol. 2. Advances in Cognitive Systems, 2012.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. Story generation with crowdsourced plot graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

Dekang Lin. An information-theoretic definition of similarity. In *In Proceedings of the 15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann, 1998.

Binghui Liu, Xiaotong Shen, and Wei Pan. Semi-supervised spectral clustering with application to detect population stratification. *Frontiers in Genetics*, 2013.

Hugo Liu and Push Singh. Makebelieve: Using commonsense knowledge to generate stories. In *Eighteenth National Conference on Artificial Intelligence*, pages 957–958, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

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

Dongcai Lu, Feng Wu, and Xiaoping Chen. Understanding user instructions by utilizing open knowledge for service robots. *CoRR*, abs/1606.02877, 2016.

M Manshadi, R Swanson, and AS Gordon. Learning a probabilistic model of event sequences from internet weblog stories. FLAIRS Conference, 2008.

Philip M. McCarthy and Scott Jarvis. MTLD, vocd-D, and HD-D: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior Research Methods*, 42(2):381–392, 2010.

Risto Miikkulainen. Discern: A distributed neural network model of script processing and memory. In *Proceedings of Third Twente Workshop on Language Technology, Twente, Netherlands. Computer Science Department, University of Twente*, 1993.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Proceedings of Workshop at ICLR, 2013.

Ashutosh Modi and Ivan Titov. Inducing neural models of script knowledge. In *Proceedings of CoNLL-2014*, volume 14, pages 49–57, 2014.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. Inscript: Narrative texts annotated with script information. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 16)*, Portorož, Slovenia, 2016.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. Modelling semantic expectation: Using script knowledge for referent prediction. *Transactions of the Association for Computational Linguistics*, 5:31–44, 2017. ISSN 2307-387X.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for

deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics.

Erik T. Mueller. Thoughttreasure: A natural language/commonsense platform (online)., 1998. URL `http://alumni.media.mit.edu/~mueller/papers/tt.html`.

Erik T. Mueller. A database and lexicon of scripts for thoughttreasure. *CoRR*, cs.AI/0003004, 1999.

Erik T Mueller. Understanding script-based stories using commonsense reasoning. *Cognitive Systems Research*, 5(4):307–340, 2004.

Boaz Nadler, Stéphane Lafon, Ronald R. Coifman, and Ioannis G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *in Advances in Neural Information Processing Systems 18*, pages 955–962. MIT Press, 2005.

J. Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, and Thomas G. Dietterich. Learning scripts as hidden markov models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1565–1571. AAAI Press, 2014.

Simon Ostermann, Michael Roth, Stefan Thater, and Manfred Pinkal. Aligning script events with narrative texts. *Proceedings of *SEM 2017*, 2017.

Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. MCScript: A Novel Dataset for Assessing Machine Comprehension Using Script Knowledge. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, 2018a.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. SemEval-2018 Task 11: Machine Comprehension using Commonsense Knowledge. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA, 2018b.

Simon Ostermann, Hannah Seitz, Stefan Thater, and Manfred Pinkal. Mapping Texts to Scripts: An Entailment Study. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018c. European Language Resources Association (ELRA).

Simon Ostermann, Michael Roth, and Manfred Pinkal. MCScript2.0: A Machine Comprehension Corpus Focused on Script Events and Participants. *Proceedings of the 8th Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, 2019.

Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithm und Complexity.* Englewood Cliffs, NJ: Prentice-Hall, 1982.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

CA Petri. Communication with automata; rome air development center. *Research and Technology Division*, page 1, 1966.

Karl Pichotta and Raymond J. Mooney. Statistical Script Learning with Multi-Argument Events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, pages 220–229, Gothenburg, Sweden, April 2014.

James Pustejovsky and Amber Stubbs. *Natural Language Annotation for Machine Learning - a Guide to Corpus-Building for Applications*. O'Reilly, 2012. ISBN 978-1-449-30666-3.

Elahe Rahimtoroghi, Ernesto Hernandez, and Marilyn Walker. Learning fine-grained knowledge about contingent relations between everyday events. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 350–359, 2016.

Altaf Rahman and Vincent Ng. Narrowing the modeling gap: a cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research*, 40:469–521, 2011.

Susanne Raisig, Tinka Welke, Herbert Hagendorf, and Elke Van der Meer. Insights into knowledge representation: The influence of amodal and perceptual variables on event knowledge retrieval from memory. *Cognitive Science*, 33(7):1252–1266, 2009.

Lisa F Rau, Paul S Jacobs, and Uri Zernik. Information extraction and text summarization using linguistic knowledge acquisition. *Information Processing & Management*, 25(4): 419–428, 1989.

Michaela Regneri. *Event Structures in Knowledge, Pictures and Text*. PhD thesis, Saarland University, 2013.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988, Uppsala, Sweden, 2010.

Radim Rehurek and Petr Sojka. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010.

Martin Riedl and Chris Biemann. Topictiling: a text segmentation algorithm based on lda. In *Proceedings of ACL 2012 Student Research Workshop*, pages 37–42. Association for Computational Linguistics, 2012.

Marcus Rohrbach, Michaela Regneri, Mykhaylo Andriluka, Sikandar Amin, Manfred Pinkal, and Bernt Schiele. Script data for attribute-based recognition of composite activities. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 144–157, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

Michael Roth and Mirella Lapata. Context-aware frame-semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:449–460, 2015.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987. ISSN 0377-0427.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686, 01 2015.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. Semeval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 45–50. Association for Computational Linguistics, 2010.

Jihee Ryu, Yuchul Jung, Kyung-min Kim, and Sung Hyon Myaeng. Automatic extraction of human activity knowledge from method-describing web articles. In *Proceedings of the 1st Workshop on Automated Knowledge Base Construction*, page 16, 2010.

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

Roger C. Schank. Tell me a story: A new look at real and artificial intelligence. *Simon & Schuster*, 1991.

Roger C. Schank. *Dynamic Memory Revisited.* Cambridge University Press, 2nd Edition. New York, 1999.

Roger C. Schank and Robert P. Abelson. Scripts, plans, goals and understanding, an inquiry into human knowledge structures. *Hillsdale: Lawrence Erlbaum Associates,*, 3(2):211 – 217, 1977.

Roger C. Schank and Christopher K. Riesbeck. *Inside Computer Understanding: Five Programs Plus Miniatures*. Hillsdale, NJ: Erlbaum., 1981.

Lenhart Schubert. Can we derive general world knowledge from texts? In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 94–97, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

Lenhart Schubert and Matthew Tong. Extracting and evaluating general world knowledge from the brown corpus. In *HLT/NAACL 2003 Workshop on Text Meaning, Edmonton, Alberta, Canada*, 05 2003.

Alan. Searleman and Douglas J. Herrmann. *Memory from a broader perspective*. McGraw-Hill,, New York :, 1994.

Push Singh and William Williams. Lifenet: A propositional model of ordinary human activity. 01 2003.

Push Singh, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. Open mind common sense: Knowledge acquisition from the general public. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, pages 1223–1237. Springer, Berlin / Heidelberg, Germany, 2002.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

Niket Tandon, Gerard de Melo, Fabian M. Suchanek, and Gerhard Weikum. Webchild: harvesting and organizing commonsense knowledge from the web. In *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*, pages 523–532, 2014.

Niket Tandon, Gerard de Melo, Abir De, and Gerhard Weikum. Knowlywood: Mining activity knowledge from hollywood narratives. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 223–232. ACM, 2015.

Niket Tandon, Gerard de Melo, and Gerhard Weikum. Webchild 2.0: Fine-grained commonsense knowledge distillation. *Proceedings of ACL 2017, System Demonstrations*, pages 115–120, 2017.

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

Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 1103–1110, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2.

W. Y. Wang, D. Bohus, E. Kamar, and E. Horvitz. Crowdsourcing the acquisition of natural language corpora: Methods and observations. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 73–78, Dec 2012.

Lilian D. A. Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. Descript: A crowdsourced corpus for the acquisition of high-quality script knowledge. In *The International Conference on Language Resources and Evaluation*, Portorož, Slovenia, 2016.

Lilian D. A. Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. Inducing script structure from crowdsourced event descriptions via semi-supervised clustering. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, Valencia, Spain, 2017.

Lilian D. A. Wanzare, Michael Roth, and Manfred Pinkal. Detecting everyday scenarios in narrative texts. In *Proceeding of the Second Storytelling workshop@ACL*, Florence, Italy, 2019.

Fangzhou Zhai, Vera Demberg, Pavel Shkadzko, Wei Shi, and Asad Sayeed. A hybrid model for globally coherent story generation. In *In Proceedings of the 2nd workshop of Storytelling (StoryNLP@ACL2019), Florence, Italy.*, 2019.

Appendices

A List of scenarios used in Part III (Scenario Identification)

	scenario	# docs	# sents.		scenario	# docs	# sents.
1	eating in a restaurant	21	387	101	receiving a letter	5	27
2	going on vacation	16	325	102	taking a shower	4	27
3	going shopping	34	276	103	taking a taxi	4	27
4	taking care of children	15	190	104	going to the playground	3	25
5	reviewing movies	8	184	105	taking a photograph	5	25
6	shopping for clothes	11	182	106	going on a date	3	24
7	working in the garden	13	179	107	making a bonfire	2	23
8	preparing dinner	14	155	108	renting a movie	3	23
9	playing a board game	8	129	109	buying a house	2	22
10	attend a wedding ceremony	9	125	110	designing t-shirts	2	22
11	playing video games	6	124	111	doing online banking	3	22
12	throwing a party	10	123	112	planting flowers	4	22
13	eat in a fast food restaurant	9	113	113	taking out the garbage	4	22
14	adopting a pet	7	111	114	brushing teeth	3	21
15	taking a child to bed	9	108	115	changing bed sheets	3	21
16	shopping online	7	102	116	going bowling	2	21
17	going on a bike tour	6	93	117	going for a walk	4	21
18	playing tennis	5	91	118	making coffee	2	21
19	renovating a room	9	87	119	serving a drink	5	20
20	growing vegetables	7	82	120	taking children to school	3	20
21	listening to music	8	81	121	taking the underground	2	20
22	sewing clothes	6	79	122	feeding a cat	4	19
23	training a dog	3	79	123	going to a party	5	19
24	moving into a new flat	8	78	124	ironing laundry	2	19
25	answering the phone	11	75	125	making tea	3	18
26	going to a concert	5	74	126	sending a fax	3	18
27	looking for a job	5	74	127	sending party invitations	3	18
28	visiting relatives	12	73	128	planting a tree	3	17
29	checking in at an airport	5	71	129	setting up presentation equipment	2	17
30	making a camping trip	5	71	130	visiting a museum	2	17
31	painting a wall	8	71	131	calling 911	2	16
32	planning a holiday trip	12	71	132	changing a light bulb	3	16

	scenario	# docs	# sents.		scenario	# docs	# sents.
33	baking a cake	3	69	133	making toasted bread	1	16
34	going to the gym	6	69	134	playing a song	2	16
35	attending a court hearing	3	66	135	washing clothes	3	16
36	going to the theater	6	66	136	putting up a painting	2	15
37	going to a pub	4	65	137	serving a meal	5	15
38	playing football	3	65	138	washing dishes	3	15
39	going to a funeral	5	64	139	cooking pasta	2	14
40	visiting a doctor	7	64	140	moving furniture	4	14
41	paying with a credit card	6	63	141	put a poster on the wall	2	13
42	settling bank transactions	5	63	142	cleaning up toys	1	12
43	paying bills	6	62	143	preparing a picnic	2	12
44	taking a swimming class	3	62	144	repairing a bicycle	2	12
45	looking for a flat	6	61	145	cooking meat	4	11
46	attending a church service	3	60	146	drying clothes	3	11
47	making soup	3	60	147	give a medicine to someone	3	11
48	flying in a plane	5	57	148	feeding an infant	4	10
49	going grocery shopping	13	57	149	telling a story	2	10
50	walking a dog	5	57	150	unloading the dishwasher	1	10
51	going to the swimming pool	5	56	151	putting away groceries	3	9
52	preparing a wedding	3	56	152	deciding on a movie	1	7
53	writing a letter	5	54	153	going to a shopping centre	1	7
54	buy from a vending machine	3	53	154	loading the dishwasher	2	7
55	attending a job interview	3	52	155	making a bed	1	7
56	visiting sights	9	52	156	making a dinner reservation	1	7
57	attending a football match	4	51	157	making scrambled eggs	1	7
58	cleaning up a flat	6	51	158	playing piano	2	7
59	washing ones hair	6	49	159	wrapping a gift	1	7
60	writing an exam	5	49	160	chopping vegetables	3	6
61	watching a tennis match	3	48	161	cleaning the floor	1	6
62	going to the dentist	3	47	162	getting the newspaper	1	6
63	making a sandwich	4	47	163	making fresh orange juice	1	6
64	playing golf	3	47	164	checking if a store is open	2	5
65	taking a driving lesson	2	44	165	heating food on kitchen gas	1	4
66	going fishing	4	43	166	locking up the house	2	4
67	having a barbecue	4	43	167	cleaning the bathroom	2	3
68	riding on a bus	6	43	168	mailing a letter	1	3
69	going on a train	4	42	169	making a hot dog	1	3

	scenario	# docs	# sents.		scenario	# docs	# sents.
70	going skiing	2	42	170	playing a movie	1	3
71	packing a suitcase	5	42	171	remove and replace garbage bag	1	3
72	vacuuming the carpet	3	41	172	taking copies	2	3
73	order something on the phone	6	40	173	unclogging the toilet	1	3
74	ordering a pizza	3	39	174	washing a cut	1	3
75	going to work	3	38	175	applying band aid	2	2
76	doing laundry	4	37	176	change batteries in an alarm clock	1	2
77	cooking fish	3	36	177	cleaning a kitchen	1	2
78	learning a board game	1	36	178	feeding the fish	1	2
79	fueling a car	3	35	179	setting an alarm	1	2
80	going dancing	3	35	180	getting ready for bed	1	1
81	laying flooring in a room	4	35	181	setting the dining table	1	1
82	making breakfast	2	35	182	change batteries in a camera	0	0
83	paying for gas	3	34	183	buying a tree	0	0
84	taking a bath	3	34	184	papering a room	0	0
85	visiting the beach	4	34	185	cutting your own hair	0	0
86	borrow a book from the library	3	33	186	watering indoor plants	0	0
87	mowing the lawn	3	33	187	organize a board game evening	0	0
88	changing a baby diaper	3	32	188	cleaning the shower	0	0
89	driving a car	9	32	189	canceling a party	0	0
90	making omelette	3	32	190	cooking rice	0	0
91	play music in church	2	32	191	buying a DVD player	0	0
92	taking medicine	5	31	192	folding clothes	0	0
93	getting a haircut	3	30	193	buying a birthday present	0	0
94	heating food in a microwave	3	30	194	Answering the doorbell	0	0
95	making a mixed salad	3	30	195	cleaning the table	0	0
96	going jogging	2	28	196	boiling milk	0	0
97	going to the sauna	3	28	197	sewing a button	0	0
98	paying taxes	2	28	198	reading a story to a child	0	0
99	sending food back	2	28	199	making a shopping list	0	0
100	making a flight reservation	2	27	200	emptying the kitchen sink	0	0

B Annotation guidelines

You are presented with several stories. Read each story carefully. You are required to highlight segments in the text where any of our scenarios is realized.

1. A segment can be a clause, a sentence, several sentences or any combination of sentences and clauses.

2. Usually segments will cover different parts of the text and be labeled with one scenario label each.

3. A text passage is highlighted as realizing a given scenario only if several scenario elements are addressed or referred to in the text, more than just the evoking expression but some more material e.g at least one event and a participant in that scenario is referred to in the text. (see examples (B.1 to B.5)).

4. A text passage referring to one scenario does not necessarily need to be contiguous i.e. the scenario could be referred to in different parts of the same text passage, so the scenario label can occur several times in the text. If the text passages are adjacent, mark the whole span as one segment. (see examples (B.6 to B.10))

5. One passage of text can be associated with more than one scenario label.

    - A passage of text associated with two or more related scenarios i.e. scenario that often coincide or occur together. (see example B.11).
    - A shorter passage of text referring to a given scenario is nested in a longer passage of text referring to a more general scenario. The nested text passage is therefore associated with both the general and specific scenarios. (see example B.12).

6. For a given text passage, if you do not find a full match from the scenario list, but a scenario that is related and similar in structure, you may annotate it. (see example B.13).

Rules of thumb for annotation

1. Do not annotate if no progress to events is made i.e. the text just mentions the scenario but no clear script events are addressed.

   Example B.1. *short text with event and participants addressed*

   ✓ `feeding a child`

   ... Chloe loves to stand around babbling just generally keeping anyone amused as long as you bribe her with a piece of bread or cheese first.

   ✓ `going grocery shopping`

   ... but first stopped at a local shop to pick up some cheaper beer . We also stopped at a small shop near the hotel to get some sandwiches for dinner .

   Example B.2. *scenario is just mentioned*

   ✗ `cooking pasta` *And a huge thanks to Megan & Andrew for a fantastic dinner, especially their first ever fresh pasta making effort of salmon filled ravioli - a big winner.*

   ✗ `riding on a bus,` ✗ `flying in a plane`

   and then catch a bus down to Dublin for my 9:30AM flight the next morning.

   We decide to stop at at Bob Evan's on the way home and feed the children.

   Example B.3. *scenario is implied but no events are addressed*

   ✗ `answering the phone`

   one of the citizens nodded and started talking on her cell phone. Several of the others were also on cell phones

   ✗ `taking a photograph`

   Here are some before and after shots of Brandon . The first 3 were all taken this past May . I just took this one a few minutes ago.

   Example B.4. *different discourse mode that is not narration e.g. information, argumentative, no specific events are mentioned*

   ✗ `writing a letter`

A long time ago, years before the Internet, I used to write to people from other countries. This people I met through a program called Pen Pal. I would send them my mail address, name, languages I could talk and preferences about my pen pals. Then I would receive a list of names and address and I could start sending them letters. ...

2. When a segment refers to more than one scenario, either related scenarios or scenarios where one is more general than the other, if there is only a weak reference to one of the scenarios, then annotate the text with the scenario having a stronger or more plausible reference.

   Example B.5. *one scenario is weakly referenced*
   ✓ *visiting a doctor,* ✗ *taking medicine*
   taking medicine is weakly referenced
   Now another week passes and I get a phone call and am told that the tests showed i had strep so i go in the next day and see the doc and he says that i don 't have strep . ugh what the hell . This time though they actually give me some antibiotic to help with a few different urinary track infections and other things while doing another blood test and urnine test on me .

   ✓ *taking a shower,* ✗ *washing ones hair*
   washing ones hair is weakly referenced
   I stand under the pressure of the shower , the water hitting my back in fierce beats . I stand and dip my hand back , exposing my delicate throat and neck . My hair gets soaked and detangles in the water as it flows through my hair , every bead of water putting back the moisture which day to day life rids my hair of . I run my hands through my hair shaking out the water as I bring my head back down to look down towards my feet . *The white marble base of the shower shines back at me from below . My feet covered in water , the water working its way up to my ankles but it never gets there . I find the soap and rub my body all over*

3. Sometimes there is a piece of text intervening two instances (or the same instance) of a scenario, that is not directly part of the scenario that is currently being talked about. We call this a separator. Leave out the separator if it is long or talks about something not related to the scenario being addressed. The separator can be included

if it is short, argumentative or a comment, or somehow relates to the scenario being addressed. When there are multiple adjacent instances of a scenario, annotate them as a single unit.

Example B.6. *two mentions of a scenario annotated as one segment*

✓ `writing a letter`

I asked him about a month ago to write a letter of recommendation for me to help me get a library gig. After bugging him on and off for the past month, as mentioned above, he wrote me about a paragraph. I was sort of pissed as it was quite generic and short.

I asked for advice, put it off myself for a week and finally wrote the letter of recommendation myself. I had both Evan and Adj. take a look at it- and they both liked my version.

Example B.7. *a separator referring to topic related to the current scenario is included*

✓ `writing an exam`

The Basic Science Exam (practice board exam) that took place on Friday April 18 was interesting to say the least. We had 4 hours to complete 200 questions, which will be the approximate time frame for the boards as well. I was completing questions at a good pace for the first 1/3 of the exam, slowed during the second 1/3 and had to rush myself during the last 20 or so questions to complete the exam in time.

✓ separator: Starting in May, I am going to start timing myself when I do practice questions so I can get use to pacing. There was a lot of information that was familiar to me on the exam (which is definitely a good thing) but it also showed me that I have a LOT of reviewing to do.

Monday April 21 was the written exam for ECM. This exam was surprisingly challenging. For me, the most difficult part were reading and interpreting the EKGs. I felt like once I looked at them, everything I knew just fell out of my brain. Fortunately, it was a pass/fail exam and I passed.

Example B.8. *a long separator is excluded*

✓ `going to the beach`

Today , on the very last day of summer vacation , we finally made it to the beach . Oh , it 's not that we hadn 't been to a beach before . We were on a Lake Michigan beach just last weekend . And we 've stuck our toes in the water at AJ 's and my lake a couple of times . But today , we actually planned to go . We wore our bathing suits and everything . We went with AJ 's friend D , his brother and his mom .

✗ separator: D and AJ became friends their very first year of preschool when they were two . They live in the next town over and we don 't see them as often as we would like . It 's not so much the distance , which isn 't far at all , but that the school and athletic schedules are constantly conflicting . But for the first time , they are both going back to school on the same day . So we decided to celebrate the end of summer together .

✓ *going to the beach*

It nearly looked too cold to go this morning ' the temperature didn 't reach 60 until after 9 :00. The lake water was chilly , too cool for me , but the kids didn 't mind . They splashed and shrieked with laughter and dug in the sand and pointed at the boat that looked like a hot dog and climbed onto the raft and jumped off and had races and splashed some more . D 's mom and I sat in the sun and talked about nothing in particular and waved off seagulls .

Example B.9. *a short separator is included*

✓ *throwing a party*

... My wife planned a surprise party for me at my place in the evening - I was told that we 'd go out and that I was supposed to meet her at Dhobi Ghaut exchange at 7 .

✓ separator: But I was getting bored in the office around 5 and thought I 'd go home - when I came home , I surprised her !

She was busy blowing balloons , decorating , etc with her friend . I guess I ruined it for her . But the fun part started here - She invited my sister and my cousin ...

✓ *visiting sights*

Before getting to the museum we swung by Notre Dame which was very beautiful . I

tried taking some pictures inside Notre Dame but I dont think they turned out particularly well . After Notre Dame , Paul decided to show us the Crypte Archeologioue . ✓ separator: This is apparently French for parking garage there are some excellent pictures on Flickr of our trip there .

Also on the way to the museum we swung by Saint Chapelle which is another church . We didnt go inside this one because we hadnt bought a museum pass yet but we plan to return later on in the trip

4. Similarly to intervening text (separator), there may be text before or after that is a motivation, pre or post condition for the applications of the script currently being referred to. Leave out the text if it is long. The text can be included if it is short, or relates to the scenario being addressed.

   Example B.10. *the first one or two sentences introduce the topic*

   ✓ `getting a haircut`

   I AM , however , upset at the woman who cut his hair recently . *He had an appointment with my stylist (the one he normally goes to) but I FORGOT about it because I kept thinking that it was a different day than it was . When I called to reschedule , she couldn 't get him in until OCTOBER (?!?!?!) ...*

   ✓ `baking a cake`

   I tried out this upside down cake from Bill Grangers , Simply Bill . As I have mentioned before , I love plums am always trying out new recipes featuring them when they are in season . *I didnt read the recipe properly so was surprised when I came to make it that it was actually cooked much in the same way as a tarte tartin , ie making a caramel with the fruit in a frying pan first , then pouring over the cake mixture baking in the frypan in the oven before turning out onto a serving plate , the difference being that it was a cake mixture not pastry ....*

5. If a text passage refers to several related scenarios, e.g. "renovating a room" and "painting a wall", "laying flooring in a room", "papering a room"; or "working in the garden" and "growing vegetables", annotate all the related scenarios.

   Example B.11. *segment referring to related scenarios*

   ✓ `growing vegetables,` ✓ `working in the garden`

The tomato seedlings Mitch planted in the compost box have done really well and we noticed flowers on them today. Hopefully we will get a good crop. It has rained and rained here for the past month so that is doing the garden heaps of good. We bought some organic herbs seedlings recently and now have some thyme, parsley, oregano and mint growing in the garden. We also planted some lettuce and a grape vine. ...

6. If part of a longer text passage refers to a scenario that is more specific than the scenario currently being talked about, annotate the nested text passage with all referred scenarios.

   Example B.12. *nested segment*

   ✓ `preparing dinner`

   I can remember the recipe, it's pretty adaptable and you can add or substitute the vegetables as you see fit!! One Pot Chicken Casserole 750g chicken thigh meat, cut into big cubes olive oil for frying 1

   ✓ `preparing dinner,` ✓ `chopping vegetables`

   large onion, chopped 3 potatoes, waxy is best 3 carrots 4 stalks of celery, chopped 2 cups of chicken stock 2 zucchini, sliced large handful of beans 300 ml cream 1 or 2 tablespoons of wholegrain mustard salt and pepper parsley, chopped

   ##42 The potatoes and carrots need to be cut into chunks,. I used chat potatoes which are smaller and cut them in half, but I would probably cut a normal potato into quarters. Heat the oil; in a large pan and then fry the chicken in batches until it is well browned...

7. If you do not find a full match for a text segment in the scenario list, but a scenario that is related and similar in its structure, you may annotate it.

   Example B.13. *topic similarity*

   - *Same structure in scenario – e.g. going fishing for leisure or for work, share the same core events in going fishing*
   - *Baking something with flour (baking a cake, baking Blondies,)*

C Adjudication guidelines

1. **Unrelated scenario.**

   The mentioned scenario is not referred to (unrelated) in the text segment.

   Example C.1. ✗*settling bank transactions*

   *So I climbed back up the college hill to return my student loan papers . I asked the financial aid adviser about the "Driver's License Number " since I don 't drive and only have a nondriver 's license . She said to put it on the paper anyway and they couldn 't say anything about it because it 's not a law that you have to have a driver 's license (not her words exactly but kinda there). I have one more paper to fill out - online since it 'd be faster - and then I 'm good . I went to the computer lab to check my email and my adviser replied to my packet ... I got burned again . *sigh* When I sent it to her it was 15 pages so when I decided to print it out I wondered why it kept printing and only looking at the document I noticed that she changed the font and size so I had 30 freaking pages ! What is she , blind ? My last packet was short and she didn 't do that . *sigh* So I filled up the printer with more paper due to that incident and I decided that next time I 'm going to check it before printing it , and may even omit the double spacing so it 's not so many pages .*

   Example C.2. ✗*taking copies*

   **sigh* When I sent it to her it was 15 pages so when I decided to print it out I wondered why it kept printing and only looking at the document I noticed that she changed the font and size so I had 30 freaking pages !*

   Example C.3. ✗*buying a house*

   My mother first mentioned the idea of moving out here from MN the very first time she held the Deuce , just a few days after they were born . Since then the two of them have been busily planning their west coast migration and last week they finally arrived .

2. **Event mentions.**

   Only the evoking event is mentioned.

Example C.4. ✗*sending party invitations*

*So , the thought of having to do up a guest list , **send out** invitations , hold my breath and cross my fingers waiting to find out if enough people would show up to my wedding was something I just wanted to avoid .*

Example C.5. ✗*taking a photograph*

I took his picture !

Example C.6. ✗*serving a meal*

*I **served** these with a few spoonfuls of blueberries on the side , and just a splash of heavy cream over the fruit .*

Only a single script event is mentioned. More than one script event and participant should be mentioned in the text.

Example C.7. ✗*going shopping*

*I **bought** a table ! A real one ! With MATCHING chairs !*

Example C.8. ✗*moving furniture*

*I bought a table ! A real one ! With MATCHING chairs ! Rob was kind enough to **truck** it home for me .*

Example C.9. ✗*making a dinner reservation*

*i was quite surprise that we manage to **book** a table , just 1 day before .*

Example C.10. ✗*buying from a vending machine* *I have recently enjoyed the convenience of the movie rental machines at the grocery stores , only $1 per movie and my girls can **pick it out** while I get my groceries .*

Example C.11. ✓*repairing a bicycle*

After getting home , I broke out the big box o 'bike parts , in which I was almost positive I had a seat clamp or two . I did indeed have them , and one had a bolt to fit my bike . After about ten minutes of work , all was well .

Example C.12. ✓*ordering something on the phone*

I called , the owner happened to answer . I explained the situation , and she said , "Of COURSE will get you all fixed up ! Look for me at just after 7 ." I gave her the tire specs , and sure enough , there she was this morning .

Example C.13. ✓*shopping online*

Once a month it 's my job to do the weekly grocery shop . I had spent an hour yesterday finding all the items and ordering them .

3. **Implied scenario.**
   The scenario is implied but there are no script-events mentioned in the text.

Example C.14. ✗*answering the phone*

*Things were going so well , then **we got a call from** Hollywood video , 'We noticed you haven 't been in for a while and you have a free rental so come on in .'*

Example C.15. ✗*receiving a letter*

*Again , I didn 't hear from her for a while , and when she wrote to me again , she said she had a baby (but no baby daddy) and she sent me pics of her and her cute little daughter . I used to **get something from her** at least once a month , sometimes more , but it came to a point (obviously with her being a single mom) that the amount of letters decreased , until it stopped altogether .*

Example C.16. ✗*riding on a bus*

*Well , as I 'm sweeping the driveway this morning , the school buses were **picking up kids** in his neighborhood to take to school . But what I noticed was that this one bus driver stopped in front of several homes , some just two doors down from each other , and blew the horn for the kids to come out to board the bus . Say what ? You mean to tell me that the kids are not standing on the corner to catch the bus anymore ? I remember when I was a kid , we had to walk a block or two in the rain , sleet or snow to catch the bus . Hell , we were like the mailman , we walked , whether we wanted to or not . Sometimes our parents dropped us off at the bus stop , but that 's neither here nor there .*

Some scenario cover a broad range of topics - taking care of children, planning a holiday trip (booking hotel, bus, flight, checking schedules and prices), telling a story, reviewing movies - difficult to find rnouh structure to understand it.

Example C.17. ✗ *telling a story*

Capt . Ford started telling a story about how his arm got infected after sticking his arm in a live well full of spot . It would seem that he had a small cut on him arm and the water he stuck his arm into was contaminated with vibrio bacteria . He shortly there after became very sick and when he finally made it to the doctor and they did a test on him which at that time they found the bacteria . He explained that if he waited any longer he would have lost his arm to the infection that ensued from the vibrio bacteria .
After the initial shock , my friend told the cab driver about the lost $50 ,000 because of the keno runners mistake .

Example C.18. ✗ *reviewing movies*

After the talk , I came back here and watched the Grey 's season premiere - so happy George is still on the show . And that they saved the deer . And that George loves Izzie back even though he 's married to Callie ... just . love .

4. **Overlapping scenarios.**
   The text is referring to overlapping scenarios. One scenario more central/specific than the other, the less central / general scenario is somehow referred to e.g. (taking a driving lesson ⇔ driving a car), (visiting a museum/sights ⇔ going on a holiday/-vacation), (planting flowers⇔planting a tree). The more central one is annotated.

   Example C.19. ✗*going shopping (*✓*going grocery shopping)*

   Today I went grocery shopping . I discovered these frozen waffles that you heat in the oven and are ready in a few minutes

   Example C.20. ✗*going on vacation/going on a holiday trip (*✓*visiting a museum)*

   My high school buddies and I spent a day touring Las Vegas . We ventured into the Titanic Exhibit at the Tropicana Hotel . We were not allow to take any photos inside

the exhibit . However , we did take a photo by the house photographer in front of a green screen . She then superimposed us in front of the grand staircase of the Titanic . By the way , this is a real set . They did a great job replicating the grand staircase . The tour was relatively short because we read every entry and looked at every artifact carefully . And still , the whole tour took less than 1 hour even though we spent 15 min at the gift shop . Fortunately , we got half price tickets at the Tuscany Suites Hotel where we stayed last night . I believe the exhibit will be moving to the Luxor Hotel next month . All in all , it was facinating looking at the artifacts that are almost 100 years old and how well it was restored from the bottom of the ocean . Definitely a must see but leave the kids home as it is a somber exhibit .

Example C.21. ✓*planting a tree (planting a flower)*

With Ostara (March 20) I had planted an acorn in a flowerpot . It grew pretty good and had leaves in no time . Unfortunately the cats discovered the leaves tasted good ... Nothing left but a pitiful stem ... I wasn 't prepared to give up though . Ron thought I was being silly . I watered the stem and talked to it , even gave it reiki . It helped ! Last week I noticed a little green sprout and today it looks like this ! Ha !

5. **Related scenarios.**
   The scenario referred to in the text is related to one of our scenarios. If the scenarios are related and similar in structure, you may annotate it.

   Example C.22. ✗*sewing clothes (knitting)*

   I decided to bridge the morass of not having the right project immediately at hand by starting another sweater . I 've been knitting mainly small projects for a long time now , but for some reason , the long term projects are very appealing right now . I decided to start the Tangled Yoke Cardigan despite the fact that nearly everyone on the planet has made one at some point and despite the fact that I generally avoid The Popular Project of the Moment just out of sheer cussedness . I like the cardigan . I have yarn that will work that has been in my stash for far too long . I need cardigans . Ergo , I will knit this sweater . And while I was marinating on the idea of starting this project , I went into gathering mode , and wound all the yarn into balls . I photocopied the pattern and put it into a plastic sleeve .

Example C.23. ✓*putting up a painting (putting up a pictures)*

After that we decided to put my panda collection to work - we hung a very adorable panda border and started hanging pictures with black picture frames and red or while mats , as well as my panda plate collection (most of the pictures and plates no one knew I had).

Example C.24. ✓*playing music in church (playing music at a concert)*

When we arrived at the mall , I felt a bit of nervousness , especially when I saw the audience , waiting for the program to begin . My mom , sister , some of our relatives and friends were there to support us . When it 's about our turn to perform , the band said a little prayer , reminding each other that our performance is not about winning , it 's all about HIM , to worship HIM and share to others our God -given talent . When were already up on the stage , I said to myself , 'this is for you Lord , this is for You '. For the first time in my life , I played on the stage and it was like a dream come true for me . It was an amazing feeling to play for the Lord and at the same do something I really enjoy .

6. **Other narrative structures.**

   The text is describing something that would have happened but never happened. There are no actual script events.- not narrative - If there is enough structure i.e. the script events are mentioned, and not just in topic ,we leave it in.

   Example C.25. ✓*going to the theater*

   *My husband and I were all **set to go and see** Bottle Shock when he was on holidays a few weeks ago , but although it was said to have opened , we couldn 't find it playing anywhere here . We did find it this week , it 's in limited release , there are two shows , one an afternoon show and the evening show starts at 6 :40pm. We won 't be going this week , it interferes with our holiday schedule of late dinners and not being organized enough to be out of the house for a 6 :40pm show .*

   Example C.26. ✗*sending a fax*

   *I filled it out Saturday , and it talked about wanting a **fax** of my DD214 (army stuff) and a transcript of my grades . Yesterday I asked about getting a transcript , and*

*they mentioned that if I got the official transcript and tried to **fax** it I 'd get 'VOID" all over the copy , and it might not do any good . So I called up the agency to find out whether they needed an official transcript , or if it 'd be okay if I just fax a print out of what courses I 'd taken ...and they didn 't seem to think I should be faxing a transcript at all , because they just require those documents when they sit down for Phase I or something .*

7. **Partly correct segments.**
   The segmentation is partly correct. The segment is wide. Some sentences do not refer to the given scenario.

   Example C.27. ✓`playing music in church`

   *The 2006 Gospel Revolution festival sponsored by the Philippine Children 's Fund of America was held last Saturday , November 11 , 2006 at the Robinson 's Starmills Pampanga . G -Rev is a song writing competition (gospel music) among bands from different local churches in Luzon . After series of screenings , 14 bands made it to the finals including our local church 's band , Bringers of Faith . 'Twas a memorable event for me . A week before , my brother asked me if I could play rhythm because they need to add musicality to the composition and with no second thoughts I said 'yes'. **We practiced at our church every night (after my work) and I really enjoyed every single moment of it . I loved it . Though my fingers ached and stiffen because it 's been a while since I played the guitar , it 's all worthwhile .***

   Example C.28. ✓`deciding on a movie`

   I was at my local Zellers , picking up my Coldwater Liquid Tide which is on sale (this topic should actually be a separate post) when I wandered into the DVD section to see what I might find . The lady in the section told me they had some new releases on at a good price , so I had a look . I bought these four movies , all comedies , which I like , because if the movie gets interrupted by someone coming in , the phone ringing or the dog barking , it 's easy enough to sit back down and continue watching . Comedies are great in our house for those lazy rainy /snowy afternoons , where we want to watch something from this century but we don 't want to have to drive all the way (about one minute from my house) to the video store . We want to be lazy , we

want to laugh and at a bargain price ($5.97 each), they were cheaper than renting . ***So here are my movies picks , based on my either remembering them from seeing previews or knowing who is in them . Actually , I think the biggest selling point was the price .***