Event Structures in Knowledge, Pictures and Text

Dissertation zur Erlangung des
akademischen Grades
eines Doktors der Philosophie
der Philosophischen Fakultäten der
Universität des Saarlandes

vorgelegt von Michaela Regneri
aus Sankt Ingbert

Dekan der Philosophischen Fakultät II:	Prof. Dr. Roland Marti
Berichterstatter:	Prof. Dr. Manfred Pinkal
	Prof. Dr. Alexander Koller
Tag der letzten Prüfungsleistung:	04. Dezember 2013

Abstract

This thesis proposes new techniques for mining scripts. Scripts are essential pieces of common sense knowledge that contain information about everyday scenarios (like GOING TO A RESTAURANT), namely the events that usually happen in a scenario (*entering, sitting down, reading the menu...*), their typical order (*ordering* happens before *eating*), and the participants of these events (*customer*, *waiter*, *food*...).

Because many conventionalized scenarios are shared common sense knowledge and thus are usually not described in standard texts, we propose to elicit sequential descriptions of typical scenario instances via crowdsourcing over the internet. This approach overcomes the implicitness problem and, at the same time, is scalable to large data collections.

To generalize over the input data, we need to mine event and participant paraphrases from the textual sequences. For this task we make use of the structural commonalities in the collected sequential descriptions, which yields much more accurate paraphrases than approaches that do not take structural constraints into account.

We further apply the algorithm we developed for event paraphrasing to parallel standard texts for extracting sentential paraphrases and paraphrase fragments. In this case we consider the discourse structure in a text as a sequential event structure. As for event paraphrasing, the structure-aware paraphrasing approach clearly outperforms systems that do not consider discourse structure.

As a multimodal application, we develop a new resource in which textual event descriptions are grounded in videos, which enables new investigations on action description semantics and a more accurate modeling of event description similarities. This grounding approach also opens up new possibilities for applying the computed script knowledge for automated event recognition in videos.

Kurzzusammenfassung

Die vorliegende Dissertation schlägt neue Techniken zur Berechnung von Skripten vor. Skripte sind essentielle Teile des Allgemeinwissens, die Informationen über alltägliche Szenarien (wie IM RESTAURANT ESSEN) enthalten, nämlich die Ereignisse, die typischerweise in einem Szenario vorkommen (*eintreten, sich setzen, die Karte lesen...*), deren typische zeitliche Abfolge (man *bestellt* bevor man *isst*), und die Teilnehmer der Ereignisse (*ein Gast, der Kellner, das Essen,...*).

Da viele konventionalisierte Szenarien implizit geteiltes Allgemeinwissen sind und üblicherweise nicht detailliert in Texten beschrieben werden, schlagen wir vor, Beschreibungen von typischen Szenario-Instanzen durch sog. "Crowd-sourcing" über das Internet zu sammeln. Dieser Ansatz löst das Implizitheits-Problem und lässt sich gleichzeitig zu großen Daten-Sammlungen hochskalieren.

Um über die Eingabe-Daten zu generalisieren, müssen wir in den Text-Sequenzen Paraphrasen für Ereignisse und Teilnehmer finden. Hierfür nutzen wir die strukturellen Gemeinsamkeiten dieser Sequenzen, was viel präzisere Paraphrasen-Information ergibt als Standard-Ansätze, die strukturelle Einschränkungen nicht beachten.

Die Techniken, die wir für die Ereignis-Paraphrasierung entwickelt haben, wenden wir auch auf parallele Standard-Texte an, um Paraphrasen auf Satz-Ebene sowie Paraphrasen-Fragmente zu extrahieren. Hier betrachten wir die Diskurs-Struktur eines Textes als sequentielle Ereignis-Struktur. Auch hier liefert der strukturell informierte Ansatz klar bessere Ergebnisse als herkömmliche Systeme, die Diskurs-Struktur nicht in die Berechnung mit einbeziehen.

Als multimodale Anwendung entwickeln wir eine neue Ressource, in der Text-Beschreibungen von Ereignissen mittels zeitlicher Synchronisierung in Videos verankert sind. Dies ermöglicht neue Ansätze für die Erforschung der Semantik von Ereignisbeschreibungen, und erlaubt außerdem die Modellierung treffenderer Ereignis-Ähnlichkeiten. Dieser Schritt der visuellen Verankerung von Text in Videos eröffnet auch neue Möglichkeiten für die Anwendung des berechneten Skript-Wissen bei der automatischen Ereigniserkennung in Videos.

Ausführliche Zusammenfassung

Die vorliegende Dissertation beschäftigt sich mit dem automatischen Lernen sogenannter *Skripte*. Skripte sind prototypische Ereignisabläufe alltäglicher Situationen, wie ein Einkauf im Supermarkt oder das Füttern eines Hundes. Konkret umfassen sie die einzelnen Ereignisse eines Szenarios, deren typische zeitliche Abfolge, und teilnehmende Personen und Objekte.

Es gilt als gesichert, dass Skripte beim Menschen wichtige kognitive Funktionen haben, weil sie helfen, große Mengen alltägliches Wissen zu organisieren und effizient anzuwenden. Am deutlichsten wird die Rolle des Skriptwissens im Zusammenspiel mit dem menschlichen Gedächtnis: eine häufiges Phänomen im Alltag ist z.B. dass man, wenn man ein Haus verlässt, sich nicht mehr erinnern kann, ob man den Herd ausgeschaltet hat (und aus Sicherheitsgründen noch einmal nachsieht). In der überiwegenden Zahl der Fälle hat man den Koch-Prozess allerdings vollständig und korrekt beendet und den Herd somit ausgeschaltet. Dies veranschaulicht die beiden Haupt-Funktionsweisen von Skripten: Bei oft eingeübten Handlungen ist es so, dass sie sich erstens in ihrer Ausführung dem Bewusstsein entziehen (man erinnert sich nicht daran), aber man zweitens meistens die Handlung richtig ausführt.

Skriptwissen beeinflusst auch viele Aspekte des Sprachgebrauchs. Da Menschen mit ähnlichen kulturellen Hintergründen große Überschneidungen in ihrem Skriptwissen haben, können viele Äußerungen stark abgekürzt werden, ohne die Verständlichkeit der übermittelten Information zu beeinträchtigen. In einem Gespräch wäre z.B. auf die Aussage *"Ich gehe einkaufen"* die Rückfrage *"Hast Du genug Geld dabei?"* problemlos verständlich, weil die Tatsache, dass ein Einkauf einen Bezahlvorgang beinhaltet für den man Geld braucht, implizites geteiltes Skriptwissen ist. Ein weiteres Beispiel sind etwa Straßenschilder mit Warnhinweisen wie *"Enge Brücke"*, die mit dem stark abgekürzten Text bei den vorbeifahrenden Lesern eine sehr schnelle und komplexe Reaktion erzielen. Würden sämtliche Vorsichtsmaßnahmen (wie langsam fahren, den Gegenverkehr beachten, ggf. anhalten / ausweichen) explizit auf dem Schild ausgeführt, wäre die Kommunikation für eine zeitkritische Umgebung wie im Straßenverkehr viel zu ineffizient. Durch erlernte und durch Erfahrung sukzessive erweiterte Skripte kann ein Fahrer solch einen knappen Warnhinweis jedoch problemlos verarbeiten.

Während Menschen Skriptwissen permanent unterbewusst benutzen, sind

Skripte als Wissensbasis für computergestützte Systeme nur unzureichend entwickelt. Insbesondere vor dem Hintergrund psychologischer Studien über Skriptverarbeitung beim Menschen sind sich die meisten Forscher einig darüber, dass automatischer Zugriff auf Skripte essentiell wäre, um vollständiges Sprachverstehen zu gewährleisten. Erste Forschungen über Skripte in der Künstlichen Intelligenz gabe es auch schon seit Einführung des Skript-Begriffs in den 70er Jahren, allerdings ist der automatische Erwerb und die Anwendung von Skriptwissen seit dem ein ungelöstes Problem. Die ersten Programme, die Skriptwissen verwendeten um Frage-Antwort-Aufgaben ("Question Answering") zu bearbeiten, bezogen ihr Wissen von manuell konstruierten, kleinen Datensätzen. Skriptwissen von Hand zusammenzutragen ist jedoch kein skalierbarer Ansatz, zumal man sowohl eine große Menge an Szenarien abdecken möchte, als auch möglichst viele Varianten jedes einzelnen Szenarios.

Mit den heutigen Methoden des maschinellen Lernens und riesigen Datenquellen (im Extremfall dem Internet) sind zwar viele Aspekte des Allgemeinwissens automatisch erlernbar, jedoch bleiben Skripte in Texten fast immer implizit. Da jeder Schreiber annehmen kann, dass der Leser mit ihm die relevanten Skripte teilt, werden beispielsweise in Texten über Restaurantbesuche keine selbstverständlichen Ereignisse wie das Lesen der Karte oder das Essen der Mahlzeit beschrieben. Ebendiese Art von Ereignissen sind es jedoch, aus denen sich Skripte zusammensetzen. Dieses *Implizitheitsproblem* wird komplementiert von einem *Abdeckungsproblem*, das sich in zwei Facetten spiegelt: Erstens ist es nicht klar, wie viele Szenarien von einer adäquaten Skript-Datenbank abgedeckt werden müssten. Zweitens ist auch unbekannt, wie viele Varianten eines Szenarios gelernt werden müssen, um ein Modell zu erstellen, das sich in zufriedenstellendem Maße der Realität annähert.

Da Skripte aufgrund des Implizitheits-Problems nicht aus Texten erlenbar sind und manuelles Beschreiben von Skriptwissen nicht sinnvoll skaliert, schlagen wir *Crowdsourcing* als Lösung vor. Crowdsourcing bedeutet, dass eine informationstechnische Aufgabenstellung mit "Schwarmintelligenz" über das Internet von einer großen Menge von Leuten bearbeitet wird. Da bekannteste und wahrscheinlich größte Ergebnis eines Crowdsourcing-Projektes ist die Online-Enzyklopädie Wikipedia, allerdings kann Crowdsourcing nachgewiesenermaßen auch sehr effizient für fokussiertere und insbesondere computerlinguistische Aufgabenstellungen sein. Die von uns genutzte kommerzielle Platform hierfür ist Amazon Mechanical Turk, wo Tasks an eine riesige Gruppe von Leuten vergeben und so in kurzer Zeit und verhältnismäßig kosteneffizient erledigt werden. Mit dieser Methode erzeugen wir mehrere skript-beschreibende Korpora aus sogenannten Ereignis-Sequenz-Beschreibungen. Der Beginn einer Ereignis-Sequenz-Beschreibung für das Szenario "im Supermarkt einkaufen" wäre z.B. *"1. Einkaufszettel machen, 2. zum Supermarkf fahren, 3. Einkaufswagen holen 4. ..."* usw. Für jedes Szenario sammeln wir eine Menge solcher Beschreibungen, die stichpunktartig typische Handlungsabläufe für das jeweilige Szenario ausformulieren. Während in einer ersten Studie die teilweise unsauberen Daten von Hand mit Hinblick auf Rechtschreibung und Konformität mit der Aufgabenstellung aufbereitet wurden, entwickeln wir auch neue unüberwachte Algorithmen für automatische Rechtschreibekontrolle und Anaphern-Resolution auf den verrauschten Daten. Die so erworbenen und vor-verarbeiteten Texte dienen dann als Grundlage für die Berechnung kompakter Skript-Repräsentationen.

Um aus mehreren Ereignis-Sequenz-Beschreibungen eine Skript-Repräsentation zu berechnen, müssen in den unterschiedlichen Text-Sequenzen Paraphrasen für Ereignisse und Teilnehmer gefunden werden. Diese Aufgabe ist vor allem deshalb schwierig, weil die semantische Ähnlichkeit der Ereignisbeschreibungen nur bedingt Auskunft über die Ähnlichkeit der beschriebenen Ereignisse im entsprechenden Szenario gibt: im Einkaufs-Szenario bedeutet z.B. *"Produkte aussuchen"* und *"eine Auswahl treffen"* das gleiche, während die beiden Phrasen keine hohe semantische Ähnlichkeit haben, und außerhalb des Szenario-Kontextes auch keine Paraphrasen wären. Gleichzeitig überlappen sich viele Ereignisbeschreibungen sehr stark obwohl sie unterschiedliche Ereignisse repräsentieren, was durch den gemeinsamen Skript-Kontext gegeben ist: *"einen Einkaufswagen nehmen"* und *"den Einkaufswagen ausräumen"* hat z.B. eine deutliche höhere Wort-Überlappung als *"Produkte aussuchen"* und *"eine Auswahl treffen"*, referiert aber zu zwei völlig unterschiedlichen Ereignissen, die an entgegengesetzten Zeitpunkten im Einkaufs-Prozess auftauchen.

Um trotz dieser Problematik szenario-spezifische Paraphrasen zu berechnen, machen wir uns die strukturelle Parallelität der Ereignis-Sequenz-Beschreibungen zu nutze: Wir können erwarten, dass in Beschreibungen des selben Szenarios gleiche Ereignisse immer in ähnlichen (temporalen) Positionen auftreten. Während z.B. *"einen Einkaufswagen nehmen"* tendenziell immer im ersten Drittel des Einkaufs-Prozesse zu erwarten wäre, sollte *"den Einkaufswagen ausräumen"* erst gegen Ende des Prozesses auftauchen. Um solche strukturellen Informationen auszunutzen, benutzen wir multiple Sequenz-Alignierung (MSA). MSA ist ein Verfahren aus der Bioinformatik, das Elemente in zwei Sequenzen aufeinander abbildet, wenn die summe der Element-Ähnlichkeit und der Ähnlichkeit der Element-Kontexte ausreichend hoch ist. Im unserem Fall sind die einzelnen Elemente Ereignisbeschreibungen, deren Element-Ähnlichkeiten mit einem von uns eigens dafür entworfenen semantischen Ähnlichkeitsmaß bestimmt wird. Gleichzeitig werden die strukturellen Ähnlichkeiten inhärent im Algorithmus mitberechnet und mit den elementaren textuellen Ähnlichkeiten kombiniert. Mit diesem neuen Paraphrasierungs-Algorithmus können wir Ereignisparaphrasen mit hoher Präzision berechnen und erzielen dabei deutlich bessere Ergebnisse als mit konventionellen Methoden, die lediglich semantische Ähnlichkeit beachten.

Ein zu den Ereignis-Paraphrasen analoges Phänomen findet sich auch bei Teilnehmer-Beschreibungen, wie z.B. *Hundefutter* und *Fleisch*, die im Szenario des Hund-Fütterns beide als synonym für Futter gebraucht werden können. Hier tritt insbesondere das Problem szenario-spezifischer Metonymien auf: *Hundefutter* und *Dose* können z.B. im Kontext von *öffnen* oder *aus dem Schrank nehmen* synonym gebraucht werden. Auch auf dieser Granularitätsebene können wir uns die strukturellen Eigenschaften der Skript-Daten zunutze machen: Wir bewerten grundsätzlich zwei Teilnehmer-Beschreibungen als synonym, wenn sie in der Summe eine hohe semantische Ähnlichkeit aufweisen und häufig in Beschreibungen für das gleiche Ereignis auftreten. Gleichzeitig gibt es eine Dispräferenz zwei Teilnehmerbeschreibungen als synonym anzusehen, wenn sie in unterschiedlichen Ereignisbeschreibungen der gleichen Sequenz auftauchen können. Um die Information über Ereignis-Strukturen zu gewinnen, benutzen wir die im vorherigen Schritt erzeugten Alignierungen von Ereignis-Sequenz-Beschreibungen. Auch hier zeigt sich eine signifikante Verbesserung gegenüber konventionellen Methoden, die keine strukturelle Information für die Paraphrasierung hinzuziehen.

Eine weitere Herausforderung im Zusammenhang mit Skripten ist deren Anwendung: Eine generische textuelle Anwendung wäre von vornherein nur mit vollständiger Szenarien-Abdeckung möglich. Weiterhin zeigt sich hier auch eine weitere Facette des Implizitheits-Problems: Da Skripte in Texten selten ausformuliert sondern häufig nur von einem einzigen Ausdruck evoziert und dann vorausgesetzt werden, ist die Verankerung von Skript-Repräsentationen in tatsächlichen Texten ein weiteres ungelöstes Forschungsproblem. Da in absehbarer Zeit keine Skript-Datenbank mit vollständiger (oder auch nur ausreichend breiter) Abdeckung erzeugt werden kann, können wir Skript-Repräsentationen vorerst nicht in vorhandene textverarbeitende Algorithmen einspeisen und so deren Nutzen zu quantifizieren.

Anstelle einer text-bezogenen Anwendung stellen wir die Skripte in einen größeren interdisziplinären Zusammenhang und erforschen das Zusammenspiel der geschrieben Skript-Daten mit Algorithmen zur automatischen Ereigniserkennung in Videos. Diese multimodale Anwendung ist sowohl von der computerlinguistischen Perspektive als auch von Seiten der automatischen Videoverarbeitung her höchst interessant: Einerseits zeigen wir in einem Pilot-Experiment, wie man Skript-Daten benutzen kann um automatisch Ereignisse in Videos zu Erkennen. Das Prinzip basiert hier auf der Idee, dass die Identifikation szenario-typischer Objekte und Handlungen automatisch auf das Szenario schließen lässt. So kann man z.B. bei automatischer Identifikation einer Gurke und eines Gurkenschälers davon ausgehen, dass im weiteren zeitlichen Verlauf eine Gurke geschält und wahrscheinlich auch geschnitten wird. Zusammen mit weiteren Identifkationen von Handlungen wie *ein Gemüse reiben* kann dann auf das Szenario *"Gurkensalat zubereiten"* geschlossen werden. Da entsprechende Textdaten als Trainingsgrundlage mit deutlich weniger Aufwand gesammelt werden können als Videos, bietet dieser Ansatz großes Potential für die Skalierung von Algorithmen zur automatischen Ereignis-Erkennung in bewegten Bildern.

Aus computerlinguistischer Sicht bietet die multimodale Verankerung von Ereignisbeschreibungen in Videos eine neue Perspektive auf Semantik von Ereignisbeschreibungen. Hierfür erstellen wir das TACoS-Korpus, in dem Videos und deren Beschreibungen aufeinander abgebildet und mit Hilfe von Zeit-Stempeln synchronisiert werden. Dieses Korpus bietet die Grundlage für eine neue Berechnung von Ähnlichkeiten zwischen Ereignissen bzw. deren Beschreibungen, da auch visuelle Merkmale aus den Videos für die Berechnung der Ähnlichkeiten herangezogen werden können. Die Intuition hier schließt sich an das Paraphrasierungs-Problem an: Wie bereits ausgeführt ist die lexikalische Ähnlichkeit zweier Ereignisbeschreibungen nicht immer ausreichend um die Relation der zugrunde liegenden Ereignisse zu bestimmen. Beim Hinzuziehen von Video-Daten kann man jedoch auf die visuelle Informations-Ebene zugreifen, die bei gewöhnlichen text-basierten Ähnlichkeitsmaßen unzugänglich bleibt. Wir können zeigen, dass schon sehr einfache multimodale Modelle für Ereignis-Ähnlichkeit sowohl rein text-basierte Ansätze als auch rein visuelle Maße deutlich übertreffen.

In einer letzten, text-bezogenen Anwendung abstrahieren wir von skriptbasierten Ereignisstrukturen weg und wenden den Paraphrasierungs-Algorithmus, den wir für Ereignis-Sequenz-Beschreibungen entwickelt haben, auf Standard-Texte an. Grundlage hierfür sind unterschiedliche Zusammenfassungen für die gleichen Folgen einer Fernseh-Serie. Da hier die Handlung der Serien-Folge die zeitliche Abfolge der beschriebenen Ereignisse vorgibt, verhalten sich diese Zusammenfassungen sehr ähnlich zu Ereignis-Sequenz-Beschreibungen. Mit multipler Sequenz-Alignierung und einem generischen,

klassischen Ähnlichkeitsmaß können wir aus diesen parallelen Korpora Paraphrasen auf Satz-Ebene extrahieren. Wir zeigen, dass der Alignierungs-Algorithmus auch hier hohe Präzision erzielt, und die strukturelle Diskurs-Information einen wichtigen Beitrag zum Erwerb von Paraphrasen mit hoher lexikalischer Varianz leistet. Darüber hinaus demonstrieren wir wie die Verarbeitung der Paraphrasen auf Satzebene zu kürzeren sogenannten Paraphrasen-Fragmenten indirekt stark positiv von der strukturellen Komponente beeinflusst wird. Wir weiten damit die klassische distributionelle Hypothese aus, die üblicherweise für Wörter und Phrasen zur Ermittlung von Bedeutungsähnlichkeiten herangezogen wird: In unserer Anwendung zeigen wir, dass die Ähnlichkeit gemeinsamer Kontexte auch für die Paraphrasierung von Sätzen im Diskurs-Kontext als wichtige Komponente gesehen werden muss.

Acknowledgements

During the years in which I worked on this thesis, I received invaluable support of several people:

My first thanks go to my advisor Manfred Pinkal, who supported me constantly while letting me carry out the research I was interested in. He taught me incredibly much about keeping the bigger picture in sight, about writing, about teaching, and simply about doing research. He also took a lot of time to read earlier versions of this thesis, and some of its chapters multiple times; his constructive and precise criticism showed me how to turn a collection of research nuggets into an actual thesis.

I'm equally grateful to my second supervisor Alexander Koller. He developed and discussed with me many research ideas in this thesis, ranging from abstract concepts to concrete technical problems. His critical views and alternative points of view in general helped my research innumerable times, and his detailed comments on many of the papers I have written both taught me a lot and made those papers much better.

Im also indebted to several helpful colleagues that supported my work on diverse parts of this thesis: Stefan Thater always took time to discuss roughly everything in this thesis, and he also gave me much practical support with data for basically each experiment involving semantic similarity. Ines Rehbein re-trained a parser that afterwards could cope with my messy data. Josef Ruppenhofer worked on concepts and annotation for participant mining with me. Dustin Smith from MIT sent me the OMICS corpus in a form that I could actually process, which increased the size of my input data by a factor of 5. Asad Sayeed & Prashant Rao helped with the annotation of the action similarity corpus for multimodal models.

Special thanks go to Alexis Palmer: she provided me with invaluable technical support for all Mechanical Turk experiments, she annotated the action similarity data set, she proof-read nearly every paper I published, and she provided detailed comments on previous versions of this thesis.

Of course I'm grateful to all the people who co-authored research papers with me. Apart from those mentioned above, I additionally want to thank Marcus Rohrbach for his kind and extensive support with vision data and infrastructure, and Rui Wang for resuming the work on paraphrasing, and for being a so incredibly relaxed and patient friend.

A big thank you goes to all the students who worked with me in the past years (as research & teaching assistants, interns, or writing their theses) - in chronological / alphabetical order: Nils Bendfeldt, Uwe Boltz, Jonas Sunde, David Przybilla, Marc Schmit, Carolin Shihadeh, Jonathan Poitz, Simon Ostermann, Max Paulus, Carolyn Ladda, Jelke Bloem, Noushin Fadaei, Antonia Scheidel, Torsten Jachmann, the SMILE annotation crew (Oscar Braun, Katharina Dellos, Erik Hahn, Anne-Julia Hoffmann, Evi Kigaia, Lennart Schmeling, Marc Schulder, Hannah Seitz and Philipp Wettmann), Dominikus Wetzel and Nicholas Merritt. All of you helped me (and also this thesis) a lot, I learned a lot, and I had a lot of fun. I hope at least one of these points is true for you, too.

I received very important support from my colleagues (and friends) Alexis Palmer, Andrea Horbach, Annemarie Friedrich and Diana Steffen, whom I had interesting research discussions with, but who especially made my workdays so much more fun.

Heartfelt thanks go to my parents and my brother, who helped in many indispensable ways. I don't know what I would do without their caring support, their constant kind encouragement, their willingness to listen to my complaints and all the fun gatherings we had.

My final deepest thanks go to Thomas, for being there, for giving me the opportunity to discuss my work when I wanted to, and for distracting me from work when I needed it, for all the comfort, and for always helping me with an innumerable number of smaller and bigger things, especially during the final phase of this thesis. I think a huge part of this thesis was actually his work, simply because he kept me going during all those years. Thanks so much for everything.

Contents

Chapter 1

Introduction

It is often said that humans are creatures of habit. While this saying sometimes carries a negative connotation related to weak will or laziness, the proverb actually depicts the essence of our brains' incredible efficiency: while conducting activities that we repeat often (like driving to work or cooking something), we can easily get them done without mistakes, while occupying our minds with something completely different.

The structures in our minds that store information about courses of actions in frequently trained routines are often called *scripts* (Schank & Abelson, 1977). Scripts and how we apply them affect our whole life, every day. This knowledge lets us execute habitual actions correctly and spares our consciousness the effort to think about how or whether we did it; it also affects our expectations and our perception of everyday situations; and it is pervasive in communication, enabling us to utter and interpret very short and highly ambiguous statements in a situation-specific, unambiguous manner.

Because of this habitual event knowledge, we can listen to audiobooks while driving, or do anything else that leaves one hand free. And if this wasn't the case, living would be extremely tedious: If we had to spend our entire ride to work actively telling ourselves things like "Stop at the red light - look over your shoulder before you turn - go slowly on crossroads" and so on, driving would not only be as exhausting as it was in the very first driving lesson, but we probably also would forget to signal every other turn.

Such internalized actions often even evade our consciousness completely: If somebody turns at the doorstep because he or she is not sure whether they turned the stove off, they will mostly find it properly shut down. Because the habit of turning the stove off after removing a pot is so tightly anchored in our brains (at least those of us who cook regularly), this task can completely be driven by our subconscious minds, which means that if nothing out of the ordinary happens, a) we always do it and b) we can never remember whether or not we did it.

Apart from tasks that we daily do by ourselves, scripts also cover more complex scenarios like GOING SHOPPING or EATING IN A RESTAURANT. Do you remember the details of the last time you went grocery shopping? It probably was not that long ago. What items did you buy and in which order did you select them? Most people cannot remember such things exactly; they can, at best, derive the answer from a mixture of memory and their default way of going shopping - by imagining in which order they would pass through the store if they went shopping again. This is the same mechanism that makes us forget whether we turned the stove off.

Scripts not only help us to *do* things without effort or conscious awareness and memory, but also directly affect how we speak and talk, and allow us to *communicate* much more efficiently, too: If your spouse says "I'll meet some colleagues for dinner!", you can kindly offer him or her "I'll get your wallet from the kitchen!" without anybody even mentioning restaurants, food prices, paying and giving someone money - your partner will understand, because he shares a lot of script knowledge with you, also about the RESTAURANT script. At the same time, you could (theoretically) say "I'll get your wallet from the kitchen!" after you announced to go shopping by yourself, and probably get a completely different reaction - but again, the both of you will understand your intentions without any explanation, because you both also share the script for GOING SHOPPING.

As indispensable as scripts are for humans, so are they difficult and under-researched when it comes to computers. If we ever want to construct software that is capable of basic human-like reasoning, like simply understanding what a human implies by saying *'I'll get your wallet from the kitchen!"*, we need to bring script knowledge to machines. While there were some small-scale attempts to make such event structures accessible for reasoning in machines, the past 40 years of research did not find a solution for learning script knowledge, not to mention any possibility to actually apply scripts to computational tasks.

This thesis develops new techniques for the automatic acquisition and application of scripts, with a particular focus on language-based acquisition methods and multimodal applications. In this first chapter, we will first explain how standardized event-knowledge is pervasive in language, and why it is thus essential for full-scale natural language processing. The following sections explain what makes script acquisition and application so challenging, and then give a first overview of our approach to script learning and multimodal script application.

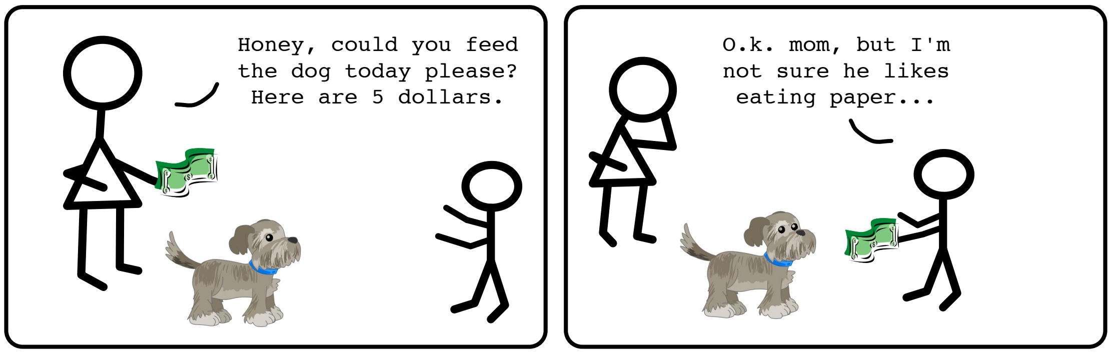

Honey, could you feed the dog today please? Here are 5 dollars.
O.k. mom, but I'm not sure he likes eating paper...

Figure 1.1: An illustration of (failed) script knowledge application.

1.1 Scripts and Language

Shared event knowledge about everyday scenarios is pervasive in everyday conversation, and even necessary for efficient communication. The importance of script knowledge and the way it influences language is best illustrated with an example that shows *failed* script-based reasoning in action - like the comic in Figure 1.1.

The utterance in panel one is natural and understandable, because we can easily infer a connection between the two sentences. Upon reading the text, the reader will assume that the money will either be used to buy the dog food, or as a reward for the kid that is meant to take over the dog-feeding duty, or maybe both. Everyone understands that the mother expects her child to know what the money is meant for, given that the usual procedures for going shopping, taking over household chores and feeding a dog (with proper food) is trivial information that everybody of a certain age and background can be expected to know. The punchline works by introducing a clash between the mother's assumption (which the reader shares) and the strange fact that the child does not fulfill those expectations (or at least pretends not to do so): he or she seemingly does not know how to feed a dog and what to use the money for, but rather utters the absurd assumption that the money should be used as dog food.

In terms of knowledge structures, two scripts are evoked here: the script for FEEDING A DOG, and the script for ASKING SOMEONE A FAVOR. It's not even important to know exactly how the money will be used - in both possible scenarios, it is clear that the money will be an essential part of the process (either for paying for the food, or for rewarding the kid). Scripts can be evoked either explicitly or implicitly: Saying *"Could you feed the dog, please?"* brings up the scenario with a direct mention; the process of asking for this favor indirectly puts the mother and the child into the relevant situation, and thus also evokes the respective script. In both cases, any elaboration about the purpose of the money or the detailed list of things expected from the kid seemingly becomes unnecessary.

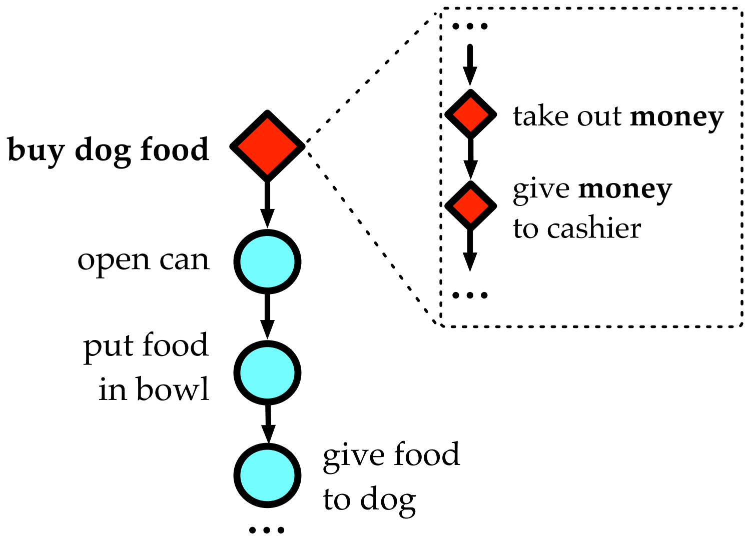


Figure 1.2: Script knowledge triggered by the scenario in Figure 1.1.

If the child in the comic knew about those scripts, he or she could infer which event involves the money, and which does not. The utterance in the last panel breaks the script for FEEDING A DOG by inserting a wrong object into a event, more concretely by exchanging the *dog food* with the *money* in the event *give food to dog*. Similar script-based mechanisms of punchlines actually form the basis for one of the most prominent linguistic theories about jokes and humorous effects in language (Raskin, 1985).

Elements of a script

According to most definitions, a script contains several bits of information about a given scenario (e.g. FEEDING A DOG):

- the **events** and their typical *temporal order* (when feeding a dog, one needs to *get dog food* first, then *open the can*, then *put the food into a bowl*, and so on)
- the **participants** of the scenario, which comprise all persons, objects or other entities involved (*dog*, *dog owner*, *food*, *can opener*,...)
- **causal relationships** and other constraints on the events (*the can needs to be opened to get the actual dog food*)

The joke in Figure 1.1 illustrates what communication could look like if such knowledge was not shared between users: Instead of inferring that dog food is the only thing one should use to feed the dog, and using the money to obtain the food, the kid's assertion skips any script-based inference and combines the instruction of *feeding the dog* with the *five dollars* as a means for feeding the pet. If one does not know which of the events related to feeding a dog actually involves the money, this might be a valid interpretation; even if one knows that a dog is usually not fed with money, information regarding how to actually use the money is still missing.

The kid's seemingly absurd inference is blocked for normal readers by many connected pieces of common sense knowledge. One of them is of course that dogs should not be fed with money, for various reasons - but the actual process of inferring what to do is commonsense-based script knowledge as pictured in Figure 1.2:

- To feed a dog, one has to get dog food first.
- One can acquire dog food by buying it at a supermarket.
- Buying something at the supermarket involves paying for the goods.
- Paying involves the transfer of money, possibly in cash terms.

Scripts and efficient communication

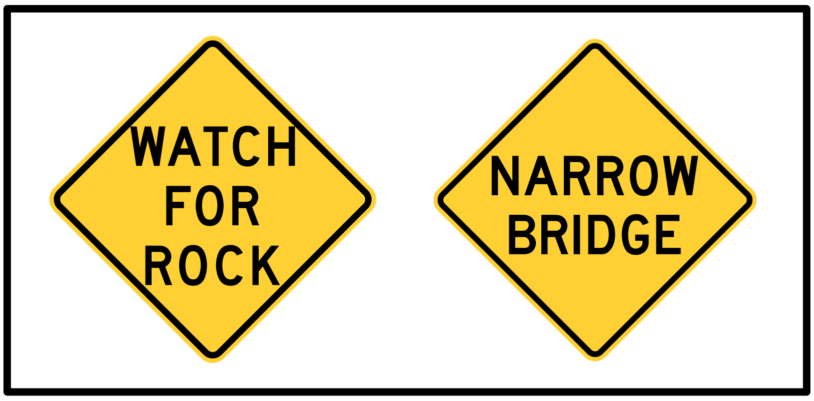
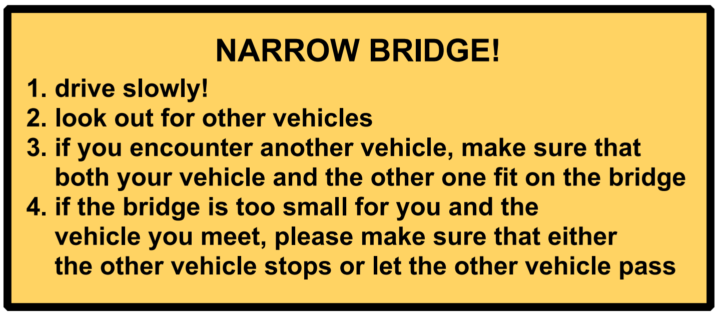


Figure 1.3: Two road signs illustrating text abbreviation licensed by script knowledge.

Script knowledge allows us to dramatically shorten conversations, like in the (misconceived) second sentence of the first comic panel. This is not only relevant for everyday conversation, but even more so for situations where extremely short instructions need to be processed quickly, like cautions on road signs: warning signs stating *"NARROW BRIDGE"* or *"WATCH FOR ROCK"* (cf. Figure 1.3) would not work at all if a driver could not within a second infer how to behave in a scenario of driving on a narrow bridge, or how and where falling rocks could cause serious damage, and why it does not make sense to just stare out of the window to watch the rocks. If such inferences by drivers could not be taken for granted, such signs would have to contain detailed advice, as in Figure 1.4:



NARROW BRIDGE!

1. drive slowly!
2. look out for other vehicles
3. if you encounter another vehicle, make sure that both your vehicle and the other one fit on the bridge
4. if the bridge is too small for you and the vehicle you meet, please make sure that either the other vehicle stops or let the other vehicle pass

Figure 1.4: A road sign without script knowledge?

Obviously, this warning message is very inefficient and impractical. Signs that abbreviate such long explanations with just two or three words still fulfill their communicative purpose, because drivers share script knowledge and can access this knowledge within a minimal amount of time. Before such information becomes habitual, drivers are taught in driving lessons, collect some experience, and sometimes use other common sense bits about the world: by evoking a generic script like DRIVING IN DANGEROUS ENVIRONMENTS, combined with some reasoning about vehicle sizes, gravity and the stability of metal, one can omit anything else related to the script and safely put drastically condensed warnings on signs. Such a warning automatically evokes the relevant scripts, and we can assume that the reader will infer the more detailed precautions. Additionally, we can count on drivers to do this inference very quickly, without having to revisit theories of gravity or driving lessons in their minds. This can only be managed because we have efficient event knowledge structures which guide us through such situations.

The mere fact that many aspects of everyday communication would fail spectacularly if humans did not share similar scripts and could not process them so quickly leads to the conclusion that any computer system that wants to understand, maybe take part in, or just analyze human-produced communication needs to have access to script knowledge. However, the current state computational understanding would lead to unintended jokes rather than to successful application of scripts, because there is no way for a computer to generically learn those pieces of common sense knowledge that humans acquire, apply and extend on a daily basis, with incredibly little effort.

In the following two sections, we will discuss the main challenges for learning and applying scripts in more detail (Section 1.3 & 1.4), and then outline the solutions we propose in this thesis (Section 1.5).

1.2 Scripts and Natural Language Processing

Although humans apparently use script knowledge all the time, it is not obvious at all in which tasks a machine could apply it. Up to now, there is no application that uses full-fledged scripts – simply because there is no sufficiently large resource that provides such knowledge. Even the few script-based systems that are actually implemented (Cullingford, 1981; DeJong, 1982; Gordon & Swanson, 2009) are not particularly good examples for the application of scripts – they either do not use actual script representations, or they do not use genuine texts, or both.

However, there have always been showcases for applications that could profit from script knowledge - if only a sufficient amount of suitable script data was available.

Question Answering

Question answering (QA, see Kolomiyets & Moens (2011) for an overview) is the archetype application for scripts: shortly after the script concept was introduced in the late seventies, the first proof-of-concept QA systems were implemented. Systems like SAM (Cullingford, 1981) were capable of some basic inference and could answer questions on (artificial) texts about restaurants or earthquakes: Given a short text about a restaurant visit, a user could ask the system a question like *Who brought the food to the patron?*, and the system would answer *probably the waiter* – no matter whether this event was mentioned in the text or not (see Chapter 2 for more details).

Current question answering systems have access to much more data and much more computing power, and they yield impressive results for factoid questions like *who-*, *where-* and *when-*questions. For such tasks, QA systems do not need to exploit deep script knowledge: to extract an answer for a question like *"Who wrote 'Ulysses'?"*, it is probably not very important to know that authors typically write books, or how the whole publication process works.

However, processing and answering more complicated *why*-questions would actually require deeper reasoning with event relations and causality, which most QA systems cannot do. Consider the following example for a *why*-question with its potential answer:

> *Why did Bradley Manning go to jail?*
> *"He was charged with a number of offenses, including communicating national defense information to an unauthorized source and aiding the enemy"*[1]

The information that a charge (in particular the given one) typically results in arrest,

[1]http://en.wikipedia.org/wiki/Bradley_Manning

which means being put into jail, is script information (about a CRIMINAL PROCESS). Hajishirzi & Mueller (2012) recently showed an attempt to incorporate some shallow event-related knowledge into a question answering system, but with regard to generic *why*-questions, such "information sources have not as yet developed sufficiently to be exploited in a QA system" (Verberne *et al.*, 2010).

Other potential applications in language processing

There are more textual applications that have often been imagined as being able to profit from script knowledge, but which have never actually been implemented in this way. Such use cases include planning and reasoning about success or failure of a plan (Schank & Abelson, 1977), and anaphora resolution (McTear, 1987). Gardent & Striegnitz (2001) later also showed how to generate referring expressions with script knowledge, but did not actually evaluate this approach - due to the lack of script data.

In the early years of script research, there was neither a sufficient amount of script data nor a sufficient amount of text data that could have profited from this kind of common sense reasoning. While we still do not have a comprehensive script resource, the internet brought a tremendous change concerning textual data for script application: internet blogs and boards offer a large source for opinionated texts where commonsense-based reasoning is in fact much more important than for e.g. newspaper articles. In particular, the very short texts in microblogs often provide good examples where script-based reasoning may be necessary.

The automated processing of microblog texts is a very new and rapidly growing research area, in particular for marketing-relevant topics like sentiment analysis (Pak & Paroubek, 2010), user categorization (Pennacchiotti & Popescu, 2011; Burger *et al.*, 2011) or even stock market predictions (Ruiz *et al.*, 2012). Because microblogs contain highly condensed information, script knowledge is often required to abbreviate the messages as far as possible, and thus could be highly useful to analyze them.

Consider for example the following short texts ("tweets"), crawled from Twitter:[2]

(a) *Just made coffee but forgot to put in water.*
(b) *Made coffee only to realise I don't have any milk.*
(c) *Just drank a big mug of coffee and forgot to have decaf. #bedtime #wideawake*

Many natural language applications would need script knowledge to process those sentences properly:

[2] `http://twitter.com`

- *Textual Entailment* systems should be able to infer whether the writer actually had coffee: in example (a), there is no coffee, and in example (b), there might be coffee, but it can't be prepared in the writer's preferred way. To do this inference, the system needs to know that the step of putting water in a coffee machine is essential for making coffee, while milk is only an optional supplement.

- *Sentiment analyzers* or mood detectors need to do a similar kind of inference to determine the writer's current mood. For this task it is necessary to distinguish the case in which there is no coffee yet, but can be made soon (a), or there is no "good" coffee and it's not available without much effort (b), or there is good coffee, but it had unexpected and unintended consequences (c).

- *Coreference resolution* needs to resolve the bridging anaphora for *the water* as a participant in the coffee making process.

- *Semantic role labeling* might stumble over the ellipsis in the first sentence, because the water is not put into the coffee, but rather in the (textually omitted) coffee machine.

To process tweets like this on a large scale, one would need a huge database that exhaustively covers many scenarios, and a mechanism to map the scripts to actual texts. Previous approaches to script mining did succeed in creating such a database, because they could not overcome the major script learning challenges.

1.3 The Challenge of Learning Script Knowledge

The script knowledge we need for further application are representations for many different scenarios, similar to the example in Figure 1.5: This graph displays a (simplified) script for the scenario FEEDING A DOG. The script contains the script participants (indicated with icons) and their possible linguistic realizations (in the last row), events that the participants are involved in (again with different verbalizations), and constraints between the events. Any two events can have a default sequential ordering (indicated with arrows from the earlier to the later event), or they can be alternatives to achieve the same goal (like the first two events that make the food available), or they can both be necessary, but happen in arbitrary order (like *filling the water bowl* and *trashing the empty food can*).

Acquiring a comprehensive knowledge base of such structures is challenging in several respects:

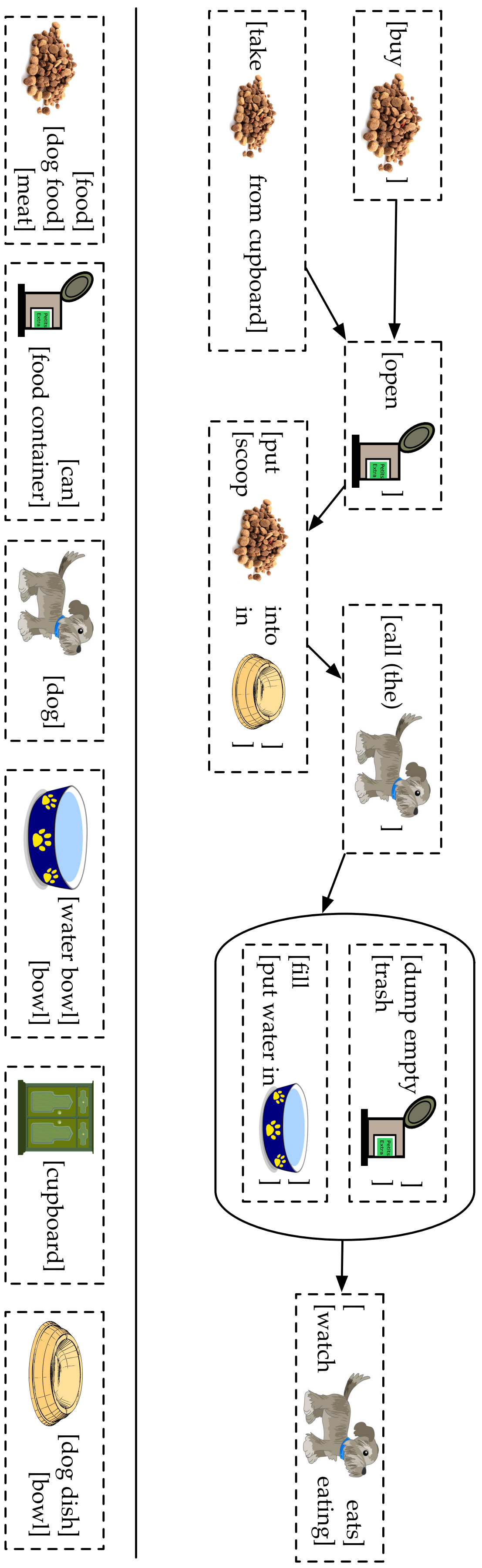


Figure 1.5: A script for FEEDING A DOG.

- **Scaling script data collection** is the most challenging part. If we needed only 50 scenarios and one description of each, one could simply write down a script collection by hand (as in the early script-based systems). However, it is not clear how many scenarios there are at all, but there are probably very many. Further, each scenario bears some degree of structural variance and has many possible linguistic realizations. To capture as many valid scenario variants as possible, one needs to collect many examples as input data. Because we cannot solve these *coverage problem(s)* by creating a script database manually, we have to find a different input source to leverage - which is then again hindered by a subsequential problem:

- **Finding appropriate input data** is difficult, due to what we call the *problem of implicitness*: as we have explained earlier, humans take script knowledge for granted. They can abbreviate it in any conversation, and thus they also hardly ever write it down. Even the biggest data sources (like the internet) do not contain all detailed facets of everyday common sense actions (which often even happen without our conscious interaction). Humans spell such things only out if they have to - e.g. if a kid would ask precisely what he or she has to do for feeding the dog.

- **Generalization over the input data** is the last major challenge: if we in fact found appropriate input data to compute scripts from, we still need to compute concise script representations. The challenge here lies in the fact that there are many ways to verbalize what happens in a scenario (which is also the reason why we want to enrich script representations with many linguistic variants). When it comes to *mining* scripts from data, those variants bear an inherent *paraphrasing problem*: How do we map different realizations of events and participants to their abstract representation in the script?

The remainder of this section discusses these challenges in more detail, and sketches our proposal for possible solutions.

Collecting input data: the problems of implicitness and coverage

Acquiring source data for mining scripts is challenging in two ways: we need a large amounts data, and even small amounts of data are hard to find. To begin with, the task of comprehensively collecting script knowledge is ill-defined: It is not clear how many relevant scenarios there are, and what they are. There is no exhaustive list of everyday tasks that everybody knows as a human. This means that we do not know what it would take to collect a comprehensive script collection in the first place, and there is no specific set of scenarios to start with, either.

A second coverage-related problem concerns the variance within a script: It is not sufficient to just find one story describing a particular scenario in order to derive a complete representation for it – we rather need many valid examples for possible event courses of a scenario. When FEEDING A DOG e.g., one can have different alternatives for certain events (*buy the dog food* or just *get it from the cupboard*), different event orders (*fill the water bowl* either before or after *trashing the can*), and many verbalization variants for all events and participants. For new scenarios, it is not clear beforehand how many examples would be needed to fully grasp all variants of a script. Even though any computational model is just an approximation of reality, one should at least be able to estimate how well the collected data represents the important scenario variants.

Up to now, there is only a tentative answer to the question of how many scenarios a script database needs to cover, and how many variants we need per scenario: As many as possible.

Previous approaches (Mueller, 2000; Schank & Abelson, 1977; Gordon, 2001, cf. Section 2.2) did not find a satisfactory solution for creating a comprehensive script data collection: most approaches tried to assemble knowledge bases manually, but this obviously does not scale up for a comprehensive scenario collection. Further, one sequence by one annotator only displays one (possibly incomplete) way to execute a script, and we want to capture reasonably many scenario variants. Previous databases also did not cover verbalization variants, which are necessary to map scripts to actual texts.

To overcome similar scalability problems, many types of world knowledge have successfully be acquired by mining large amounts of text: Several approaches learn diverse semantic relations and ontologies from big corpora (Buitelaar *et al.*, 2008, among others), and rapidly growing online-resources like Wikipedia[3] are successfully used for gathering detailed information about persons, locations or other named entities (Suchanek *et al.*, 2007).

When trying to mine script knowledge from text, one faces the *problem of implicitness*: the more internalized a script is, the lower is the likelihood of finding it spelled out in text. Texts that could be used for script learning should contain *prototypical sequences of events* that are temporally ordered and attributed to a specific scenario. Children's journals, for instance, could be a possible text source for such sequences, like the following (made-up) example for FEEDING A DOG:

> *Today I had to feed the dog. First I bought new dog food with the money mom gave me. At home, I immediately took out a can opener and a spoon. then I opened the food can and put the food into the dog's bowl. I called the dog and he was very hungry.*

[3]`http://www.wikipedia.com`

Texts like this are rather uncommon, probably even for children's writings. We rarely find any prototypical episodes in which nothing out of the ordinary happens. Even if children's diaries might be an appropriate data source, they are mostly inaccessible, and not available on a scale that would allow for effective knowledge mining.

Collecting input data: our crowdsourcing approach

We propose a method for data collection that overcomes the problem of implicitness and, at the same time, maintains scalability: We will use *crowdsourcing* for data collection.

1. buy dog food	1. take food from cupboard
2. open can	2. open food container
3. put meat into bowl	3. scoop food in dog dish
4. call the dog	4. call dog
5. trash can	5. put water in bowl
6. fill water bowl	6. dump empty food container
7. watch dog eating	7. dog eats

Figure 1.6: Two exemplary event sequence descriptions for FEEDING A DOG.

Crowdsourcing is a currently trending and growing technique for linguistic data acquisition. By asking many people over the internet, one can target all kinds of common sense scenarios, and still get a reasonable coverage over different scenarios and variants within a reasonable amount of time. This technique allows us to ask thousands of people directly for the information we want, so we can circumvent the problem of implicitness and still maintain a scalable approach. As a crowdsourcing platform, we use Amazon Mechanical Turk, and ask people what they do when they experience a certain scenario. The two sequences in Figure 1.6 are two example answers to a question like "HOW DO YOU FEED A DOG?". (We restrict this running example to two sequences for simplicity. The actual script learning architecture is designed for much larger datasets.)

Generalizing over input data: the paraphrase problem

Mining global representations from textual scenario descriptions is unfortunately not straight-forward. Scripts display constraints between abstract *event types*, and we only have concrete *event verbalizations* in our source data. To find out how the (abstract) events in our scenarios relate to one another, we need to find out which event descriptions in the source sequences actually refer to the same event, and which of them describe different ones: we need to solve the **paraphrase problem**.

The paraphrase problem is shared by all algorithms that aim to learn common sense knowledge: Language is vague and ambiguous, thus there are many different ways of verbalizing the same thing. This is already challenging for far less complex learning tasks, e.g. acquiring basic semantic relations like meronyms: if, for instance, an algorithm aims to learn that *wheels* are parts of *cars*, it might have to cope with texts that only contain information on *"wheels of automobiles"* or *"wheels of Ferraris"*, and thus should know that Ferraris are, in fact, cars.

This challenge grows with the size of the unit of interest: while it is already non-trivial for words, it is even harder for sentences, because there can be infinitely many different sentences expressing the same fact. Script knowledge acquisition is at the upper end of the scale for challenging paraphrasing problems: We are considering not just pairs of words or pairs of sentences, but rather *groups of sequences of sentences* that describe the same scenario. If we have multiple descriptions of FEEDING A DOG, we need to find out which sentences of different descriptions match each other, in order to generalize over those sequences and recognize the scenario's variances.

1. buy dog food	1. take food from cupboard
2. open can	2. open food container
3. put meat into bowl	3. scoop food in dog dish
4. call the dog	4. call dog
5. trash can	5. put water in bowl
6. fill water bowl	6. dump empty food container
7. watch dog eating	7. dog eats

Figure 1.7: Event paraphrases in the sequences of Figure 1.6.

Take another look at the example in Figure 1.6. Most events have different realizations in the two texts, e.g. *put meat into bowl* and *scoop food in dog dish* both describe the event of transferring the food into some container. In Figure 1.7, we repeat the same sequences, indicating the event paraphrases with colors. Finding those paraphrases is challenging, because the semantic similarity of the event descriptions does not necessarily indicate the similarity of the underlying events: different event realizations within the same scenario share a very narrow vocabulary, simply because they all turn around the same scenario participants, like *put water in bowl* and *put meat into bowl*. Inversely, actual paraphrases in different sequences are not necessarily highly similar: For example, *open can* and *trash can* overlap more than *trash can* and *dump empty food container*. It is often the case that descriptions of different events, written by the same author, show more surface similarity than paraphrases in different descriptions from different writers.

Generalizing over input data: our structure-aware paraphrasing approach

It is hard to recover event paraphrases from any similarity information we could derive from the event realizations by themselves. However, we can make use of the inherent *structural* similarities of event sequence descriptions: If two people describe the same scenario, they probably assume similar temporal event orderings. We can thus incorporate this information in the paraphrasing process and take two event descriptions as similar if they are semantically similar, *and* if they appear in a similar position in the sequence. In practice, we use a sequence alignment algorithm from bioinformatics for matching event descriptions according to these semantic and structural criteria. The result are then groups of event descriptions that refer to the same event in the script (like the event boxes in Figure 1.5). Using those as events, we can then compute concise script representations.

1. buy dog food	1. take food from cupboard
2. open can	2. open food container
3. put meat into bowl	3. scoop food in dog dish
4. call the dog	4. call dog
5. trash can	5. put water in bowl
6. fill water bowl	6. dump empty food container
7. watch dog eating	7. dog eats

Figure 1.8: Participant paraphrases in the sequences of Figure 1.6.

Scripts also contain participants, which we want to include into our script representations. In order to do so, we need to solve the paraphrase problem for participants, too: like for event descriptions, annotators use different names to refer to the same participant (like *bowl* and *dog dish*), and descriptions of the same participant can have rather low surface similarity. Building on the structure-aware event paraphrasing approach, we will present a new algorithm that extracts participant paraphrases that also takes sequential structure into account (cf. Figure 1.8). In addition to semantic similarity, this algorithm incorporates two structural heuristics. First, we re-use the event paraphrase information: if two participant descriptions occur in two event paraphrases, they are likely to be paraphrases themselves. Second, the same annotator usually refers to the same participant with the same name, so two different participant descriptions in the same sequence most likely refer to *different* participants. The outcome of this step are participant representations consisting of equivalent participant descriptions (like the boxes in the bottom line of Figure 1.5).

1.4 Scripts and new Applications

This thesis presents the first scalable script-mining architecture. We additionally show two more advanced applications in different disciplines: First, we ground script knowledge in video data and explore the advantages for semantics of event descriptions as well as action recognition in videos. Second, we focus on text processing and show that the insights we gained for paraphrasing in script data also bring tremendous advantages for extracting generic paraphrases from parallel texts.

Multimodal application of script knowledge

We have already reviewed possible applications of script knowledge in natural language processing. Text processing is by far not the only computational discipline that can take advantage of script knowledge, though: previous research has already shown that script-like knowledge is very useful for sorting and retrieving images (Gordon, 2001). We will bring this connection of scripts and vision to a new level, moving away from static pictures and grounding script knowledge in video data.

There are multiple aspects that make the connection of scripts and vision interesting: in computer vision applications, the problem of implicitness does not exist, because there are no possibilities for any shortcuts with implied events. Talking or writing about single steps of a standardized procedure seems unnecessary and boring, but the actual execution of those single steps is still necessary: recording somebody making coffee necessarily involves watching each single step on the way.

From the other perspective, textual data is a very interesting supplement for computer vision approaches (Rohrbach *et al.*, 2010), because video data is very hard and tedious to collect and to annotate. Enhancing visual models with textual data can help to collect data for many scenarios in a tiny fraction of the time it would take to create recordings for the same amount of scenarios.

When it comes to applications, grounding scripts in videos is also attractive for both computational semantics and computer vision: viewed from the computer vision perspective, script knowledge can help to *predict* events in a scenario (cf. Figure 1.9). If an algorithm has recognized a carrot, a knife, a cutting board, it can use scripts as prior knowledge to infer that most likely there will be a *cut*-action (with *carrot* and *knife* as participating objects) soon. For computational semantics, the grounding of textual script data in visual features offers new possibilities to assess event semantics: we already explained that the surface similarity of event descriptions often does not correspond with the surface similarity of the underlying actions. It is intriguing to ascertain whether visual features are a better way to score event similarities, while textual descriptions for

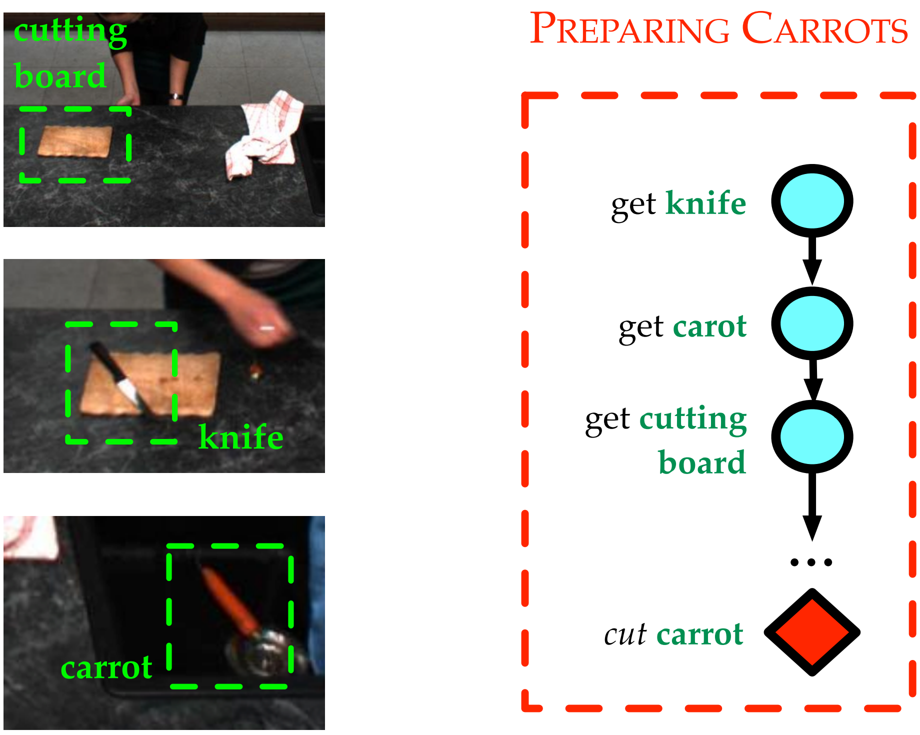


Figure 1.9: Action recognition in videos with scripts.

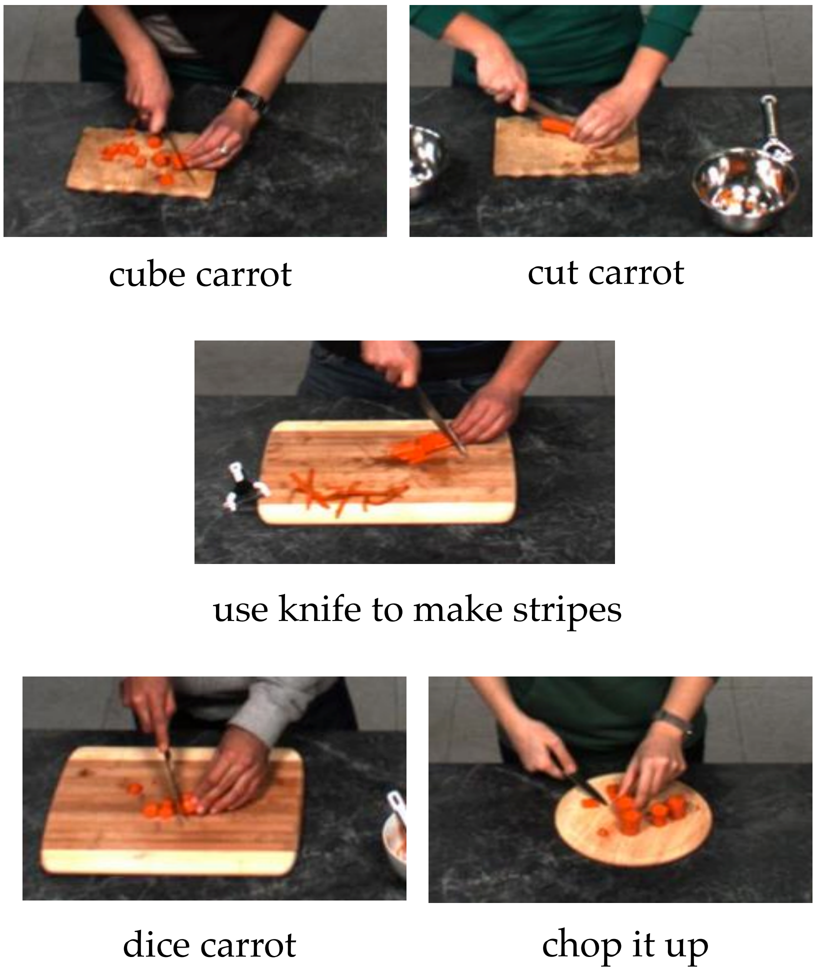


Figure 1.10: Event description paraphrasing with video data.

the same event can vary a lot. The hypothesis here is that e.g. the process of turning a carrot into bite-sized pieces always looks very similar in a video, while descriptions like *cut carrot*, *use knife to make stripes* and *chop it up* are highly dissimilar from a linguistic perspective (cf Figure 1.10).

Paraphrasing and event structures in text

We have argued that the paraphrasing problem for script events is nearly impossible to solve for algorithms that only consider the surface similarity of sentences. To achieve better results, we propose to consider the parallel sequential structure in event sequence descriptions and apply a combined measure of structural and semantic similarity.

While we have introduced this paraphrasing technique as a means to an end for script representation mining, the algorithm itself has more applications. Large paraphrase collections have many applications in natural language processing, and in consequence, many algorithm have been proposed to mine such paraphrases automatically (Lin & Pantel, 2001; Szpektor *et al.*, 2004; Dolan *et al.*, 2004; Ganitkevitch *et al.*, 2013). Monolingual parallel corpora in particular have evolved as a useful resource for extracting such paraphrases (Barzilay & McKeown, 2001; Bannard & Callison-Burch, 2005; Zhao *et al.*, 2008).

However, most of those approaches neglect structural similarities in parallel texts, which misses out on important opportunities: Just like the sequential structure in event sequence descriptions can hint on similarity of event descriptions, the similarities of event structures in parallel texts can help to identify sentences with equivalent meaning. We will show how to port our paraphrasing technique to standard texts and thus yield much more precise paraphrases than approaches that rely on semantic similarity alone and ignore the text structure.

1.5 Plan of the Thesis

This thesis covers several aspects related to script learning and application, organized in three larger parts:

Part 1: Script learning

The first half of this thesis is concerned with **Script Mining: Chapter 2** summarizes previous work on scripts and script processing. In **Chapter 3**, we detail how we acquire the basic input data using *crowdsourcing*, showing how this technique overcomes the *problem of implicitness* while still being a reasonable choice considering the *coverage problem*.

Chapter 4 describes how we mine concise script representations out of the acquired textual input, focusing on the central task of computing event paraphrases. The paraphrasing algorithm heavily relies on script-inherent structural constraints and applies them using sequence alignment. To enable more fine-grained reasoning, we also exploit structural script information for a new algorithm to compute paraphrases for script participants (**Chapter 5**). **Chapter 6** shows several preprocessing techniques that guarantee domain-independence for the script mining system.

Part 2: Applications

We show two potential applications for script data itself and for the algorithms we developed on the way: **Chapter 7** shows a new approach to grounding event descriptions in video data. We build a corpus of videos and temporally aligned textual descriptions of the events that happen in those videos. We use this corpus to assess semantic similarity between the event descriptions, using a combination of visual and textual features. Thus we provide a new resource for semantic processing of event descriptions, and show how visual information contributes to the semantics of event descriptions.

Chapter 8 shows how the algorithm we used to compute event paraphrases can be applied to standard texts for extracting generic sentential paraphrases. Our script-inspired solution to the *paraphrase problem* offers a new perspective on text mining as well. While the texts we are looking at do not contain actual script data, they establish similar structural properties and thus fit the algorithms we designed for scripts. The outcome is a new technique for paraphrase extraction as well as a collection of paraphrases and paraphrase fragments.

Part 3: Outlook

We conclude this thesis with a survey on ongoing work related to script processing (**Chapter 9**), showing recent advances concerning data acquisition as well as current work on applications of scripts for visual processing.

Chapter 10 then provides a summary and highlights some interesting directions for future work.

1.6 Contributions of this Thesis

The main contributions of this thesis cover various aspects of script-related processing and event semantics:

1. The application of crowdsourcing to collect source data for script mining, and a corpus with sequences of event descriptions for various scenarios (Chapter 3); joint work with Alexander Koller and Manfred Pinkal (Regneri *et al.*, 2010).
2. The introduction of structure-aware paraphrasing algorithms for event descriptions within a script (Chapter 4) and script participants (Chapter 5); joint work with Alexander Koller, Manfred Pinkal and Josef Ruppenhofer (Regneri *et al.*, 2010, 2011).
3. A multimodal corpus of videos and their timed descriptions, and the first account for action description semantics using textual and visual features at the same time, outperforming each of the single modes taken by itself (Chapter 7); joint work with Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele and Manfred Pinkal (Regneri *et al.*, 2013)
4. The application of the sequence alignment architecture built for script data to standard texts, as the first discourse-aware paraphrasing algorithm for sentential paraphrase extraction from parallel corpora (Chapter 8); joint work with Rui Wang (Regneri & Wang, 2012).

Relevant Publications

The following publications report on parts of the research described in this thesis:

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

Regneri, Michaela, Koller, Alexander, Ruppenhofer, Josef, & Pinkal, Manfred. 2011. Learning Script Participants from Unlabeled Data. *In: Proceedings of RANLP 2011.*

Regneri, Michaela, & Wang, Rui. 2012. Using Discourse Information for Paraphrase Extraction. *In: Proceedings of EMNLP-CoNLL 2012.*

Regneri, Michaela, Rohrbach, Marcus, Wetzel, Dominikus, Thater, Stefan, Schiele, Bernt, & Pinkal, Manfred. 2013. Grounding Action Descriptions in Videos. *Transactions of the Association for Computational Linguistics (TACL), Issue 1.*

Additionally, some preliminary results of ongoing work (cf. Chapter 9) were published in the following articles:

BLOEM, JELKE, REGNERI, MICHAELA & THATER, STEFAN. 2012. Robust processing of noisy web-collected data. *In: Proceedings of KONVENS 2012*.

ROHRBACH, MARCUS, REGNERI, MICHAELA, ANDRILUKA, MICHA, AMIN, SIKANDAR, PINKAL, MANFRED, & SCHIELE, BERNT. 2012. Script Data for Attribute-based Recognition of Composite Activities. *In: Proceedings of ECCV 2012*.

Chapter 2

Background on Scripts and Script Processing

Scripts and script-like knowledge are recognized as essential elements of human common sense knowledge. They came up as interdisciplinary concept in the early seventies, since which time they have been investigated by both AI scholars and cognitive scientists. Minsky (1974) set a starting point for both lines of research, introducing *frames* as a generic concept for cognitive processing: frames are everyday situations (like *being in a kitchen* or *going to a party*) that activate certain associated pieces of knowledge and guide a human through a situation. The associated knowledge is partially about standard behavior within the frame (e.g. preparing food), expectations about events one can encounter (e.g. someone else entering the kitchen to get coffee) or strategies to deal with violated expectations (e.g. if there is no coffee left, one should prepare new coffee).

Schank & Abelson (1977) specified frames and developed the idea of scripts: according to their original definition, a script describes a scenario, using typical events in their typical temporal order, the events' key players, the whole process' goal, and causal connections between the individual events. Scripts were designed to model knowledge structures in human brains, comprehensively showing how we think our way through everyday situations, as well as adapting our minds to new scenarios. As script knowledge was shown to be pervasive in human behavior, reasoning and language, people working on computational intelligence wanted to make this knowledge accessible for computers in order to enable human-like reasoning in machines.

As far as script representations in human minds are concerned, psychologists found reasonable evidence that scripts constitute essential organizational structures for common sense knowledge, and that human minds process everyday events by recalling regularities they learned from earlier, similar episodes (Graesser *et al.*, 1979, among others).

There is evidence that children learn scripts from an early point in time, and that their ability to relate new episodes of a certain scenario to their memorized scripts increases as their cognitive abilities increase with age. (Adams & Worden, 1986)

The process of script acquisition and application in machines are less well understood. Shortly after the first script theories arose, several researches developed small sample programs capable of script-based reasoning – Schank & Riesbeck (1981) provide a survey of those. Those programs were all developed in the late seventies, when computing power was still several evolutionary steps behind the current state of the art: there was no internet, no large data resources, no way to get more than 100 mhz CPU or more than 1mb of RAM on one machine, and not even desktop computers. From this background, there was neither the possibility to develop machine learning algorithms, nor any chance to apply them to large amounts of data. Early systems thus had to take a very simple approach to script processing: their authors entered 2 to 5 scripts manually into their machines and then did reasoning based on made-up texts that contained exclusively events from the pre-defined script events. The hope at that time was that if one could manage to simply enter a representation for each single script into the machine – no matter how – all script-related problems could be solved. The core problem of getting this data was postponed to some indefinite later point in time, anticipating better machinery or enough manpower to become available over time.

The next 40 years brought tremendous progress for data-driven research in general. The available computing power and the amounts of available data from the internet grew (and still grow) exponentially. However, one old question about script data acquisition remains unanswered: While we actually do have the machinery to do elaborate data-driven research, we still don't know from which data we could learn scripts. As we already argued, a lot of script information is simply not elaborated in text (cf. Chapter 1). Further, we don't know which scripts to look for in the first place.

This chapter reports on previous work on scripts, summarizing some literature on cognitive script research and detailing previous script-related approaches in artificial intelligence. We first review some details about scripts in general as well as from the psychological point of view (Section 2.1), drawing a more precise picture of the knowledge we want to acquire. Then we introduce previous approaches to automatic script processing (Section 2.2), followed by a short analysis of those systems' capabilities to approach the main challenges related to script acquisition (Section 2.3).

2.1 Scripts and Human Script Processing

It is widely agreed that systems targeted at full language understanding need access to script knowledge, and this assumption is intuitively clear and easy to attest with countless examples. However, it is not clear at all which knowledge parts essentially belong to a script representation, and in particular, which of them could help natural language processing at all: there is no wide-coverage system that uses script-based inference, simply because the data for such a system is not available. Without any application-driven guidance, a natural step is to investigate which kind of script knowledge humans use, and try to make this knowledge available through automatic learning.

First definition of scripts

Schank & Abelson (1977) introduced scripts as a framework for knowledge organization in humans that was intentionally designed to be applicable for machines, too. According to their definition, a script describes a certain *scenario* (or *scene*), like EATING IN A RESTAURANT or FEEDING A DOG, and has the following components:

- A **scenario**, the underlying type of situation or story for the script (e.g. FEEDING A DOG); scenarios are mostly everyday life events.
- **Goals** tied to the script, each one defining an intended outcome of the scenario (e.g. FEEDING A DOG has the goal of preventing the dog from starving)
- Stereotypical **events** and their expected **temporal order** (*get dog food* $\rightarrow$ *open the can* $\rightarrow$ *put the food into a bowl*).
- The **participants** of the scenario, which comprise all persons, objects or other entities involved (*dog*, *dog owner*, *food*, *can opener*,...)
- **Causal relationships** or causal chains of the events, defined by pre- and post-conditions of each single event (e.g. *open can* changes the world state such that the previously closed can is open, and an open can is a precondition for *put the food into a bowl*).

Once such a script is internalized, no complicated inference is required to execute it - as long as nothing extraordinary happens. As a consequence, people can make breakfast or feed dogs or drive cars without any planning efforts and thus little cognitive load, because they just do such activities according to their script-manifested habits.

Schank distinguished scripts from specific *plans* which need to be constructed to achieve higher-level *goals*. A script can be part of such a plan, for instance **getting to work on**

time and **preventing the dog from starving** on the same day (= 2 *goals)* requires (a) to BUY DOG FOOD on the day before and (b) to FEED THE DOG before one leaves for work (= a plan featuring two scripts).

Human script knowledge: some basics

Humans acquire numerous scripts by dealing with everyday scenarios over time. Practically this means that if we entered our first restaurant, this would be a new experience for which we would built a new script in our mind. During the next few restaurant visits, we would notice that the core elements are always the same (there are waiters, we read the menu, we order, the waiter brings the food...). This would then, after we gained a little experience with eating out, constitute a script representation for EATING IN A RESTAURANT, stored in our minds and recalled whenever we visit any restaurant (that we have or haven't been to before). Waiters and ordering won't surprise us, and we won't need to think long about what to do next.

If, after several encounters of the same stereotypical restaurant script, we would find something not recorded in our internal representation for the RESTAURANT scenario, our script would still not change dramatically: depending on the importance of the new event, we would regard it as an exception, or add a new so-called *track* to the script, storing an alternative script as a derived variant of the original (like *eating in a fast food restaurant* vs. *eating in a diner*). Psychologists have shown that children in fact develop a preference for keeping certain scripts as a knowledge backbone and distinguish them from extraordinary, less frequent exceptions. The ability to differentiate between exceptions and recurring script variants increases with age (Adams & Worden, 1986).

Humans script knowledge: more recent evidence

The initial theory of scripts was later revised in several respects.

Concerning the knowledge pieces within a script, recent evidence shows that causality does not play a crucial role for scripts. This holds for the overall script goal as well as for single tasks: firstly, humans acquire parts of a script equally well regardless of whether they contribute to the scripts' overall goal (Seitz & Watanabe, 2009), so the overall plan does not seem to be too important for learning a script in the first place. Second, humans are not necessarily aware of the ordering constraints they have learned with a script (Swallow & Zacks, 2008), even though they apply these constraints. Any causal connections that may restrict the actual event ordering are thus not always relevant for script processing.

As far as script acquisition is concerned, scripts are internalized as dynamic processes

(Schank, 1999) rather than strictly constrained structures, and they adapt to every single new experience. This means that we don't internalize *the* one script for going to a restaurant, but rather grow a more complex mechanism that stores expectations about new restaurant visits and "awaits" potential updates. According to this theory, each new encountered episode of a scenario bears the potential to teach us new things which we then add to our script representations: as long as an episode of a scenario (like EATING IN A RESTAURANT) complies with our internal scripts (i.e. everything happens as expected), we won't note anything special about this episode. But if there are unexpected events (e.g. on a holiday trip abroad, a waiter reacts with surprise on being tipped, because it's unusual there), they will influence our representation of the respective scripts (*tip the waiter* becomes an optional event that is subject to the country one is in). Script learning is thus an adaptive process, capable of changing with new input.

Humans even seem to acquire meta-knowledge about the expected degree of variance within episodes of a scenario. For highly conventionalized scenarios (e.g. SWITCHING THE LIGHT ON), we do not expect any notable variation. However, if one encountered very diverse episodes for the same scenario from the beginning on (like different ways to SET UP THE DINING TABLE depending on meal, guests etc.), one would not even memorize a prototypically ordered event structure in the first place, but rather anticipate even more variation to come up (Hudson *et al.*, 1992). Note that the experiments carried out here were mostly simple lab experiments with children and building-bricks - nothing with any plan or purpose obvious to the trial subject.

Essentially, we, as humans, acquire script knowledge over time, and not because we make logical plans, but because we get used to some events happening after one another. We also develop expectations towards the degree of variation of a script, as a piece of meta-knowledge. The more script instances we see, the more variance can come into our internal representations. The specific degree of variance differs according the nature of the scenario, and the number of script episodes one lives through. Most trivially this means that our knowledge becomes more complex and more accurate as we learn more by gaining more experience. Each unexpected experience teaches us new things.

2.2 Script Processing in Machines

This section reviews previous computational approaches to script knowledge collection and application. We start with two early systems that use manually created script knowledge bases, which were then applied in proof-of-concept applications for text understanding. We also show other manually created script data collections which all differ in size, objective and application. In a last part, we show two more recent corpus-based approaches to script learning, that create knowledge bases on a much larger scale.

Early script-based reasoning

Shortly after the introduction of the script concept (cf. Section 2.1), the first script-based systems were presented. We introduce the two most well-known systems (*SAM* and *FRUMP*) in more detail.

```
script:   restaurant
roles:    customer, waitress, chef, cashier
reason:  to get food so as to go up in pleasure and go down in hunger

scene 1: entering
          PTRANS   self into restaurant
          ATTEND   eyes to where empty tables are
          MBUILD   where to sit
          PTRANS   self to table
          MOVE     sit down
scene 2: ordering
          ATRANS   receive menu
          MTRANS   read menu
          MBUILD   decide what self wants
          MTRANS   order to waitress
scene 3: eating
          ATRANS   receive food
          INGEST   food
scene 4: exiting
          MTRANS   ask for check
          ATRANS   receive check
          ATRANS   tip to waitress
          PTRANS   self to cashier
          ATRANS   money to cashier
          PTRANS   self out of restaurant
```

Figure 2.1: The RESTAURANT script in the SAM system, with the customer as an agent.

SAM

SAM (Cullingford, 1981, *Script Applier Mechanism*) directly implements the theory of Schank & Abelson (1977). Figure 2.1 shows a representation of the RESTAURANT script (from the perspective of the customer) used in SAM. The representation defines the scenario (named `script`, here RESTAURANT), the participants (called `roles`), and goals that can be fulfilled by executing the script. The actual content consists of four so-called scenes, which are groups of events that share a setting and participants: They each have different events in a certain order, classified according to Schank's Conceptual Dependency Theory (Schank *et al.*, 1974; Lytinen, 1992, CD Theroy). CD Theroy defines a hierarchy of action types that categorizes all possible actions. Classification in this

PTRANS	The local transfer of an object.
ATRANS	The transfer of ownership, possession or control of an object.
MTRANS	The transfer of mental information (i.e. communication).
MOVE	Movement of a body part of the agent.
INGEST	Ingesting food, fluids or air.
MBUILD	The construction of a thought or new information by the agent.
ATTEND	Focussing attention on something.

Figure 2.2: Event types according to Conceptual Dependency Theory.

hierarchy allows for more advanced (automated and human) reasoning on the action's outcomes, preferences, and the changes it makes to the world. We show a short reference for the CD Theroy classifications in Figure 2.2.

Causal connections are not explicitly noted in the script representation: the events are temporally ordered, and the completion of each action in a scene is taken as a precondition for the next scene to happen.

`Input text:`

John went to a restaurant. The hostess seated John. The hostess gave John a menu. The waiter came to the table. John ordered lobster. John was served quickly. John left a large tip. John left the restaurant.

`Questions & SAM's answers:`

User: What did John eat?
SAM: `LOBSTER.`
User: Who gave John the menu?
SAM: `THE HOSTESS.`
User: Who gave John the lobster?
SAM: `PROBABLY THE WAITER.`
User: Who paid the check?
SAM: `PROBABLY JOHN.`

Figure 2.3: Script-based question answering task with the SAM software.

The inference component in SAM solves question answering tasks like the one in Figure 2.3 (cf. Schank & Abelson (1975)): The system is capable of detailed inference within its scope of knowledge. It identifies the script to apply (EATING IN A RESTAURANT) and it infers implicit information according to the script content (John ate lobster, because that's what he ordered). It even gives script-informed guesses about non-inferable information (the waiter served John, thus he is supposedly the one who brought the lobster). In its inference abilities, this system is still to be beaten. Unfortunately, it was designed only as a proof of concept application, because the required knowledge is manually inserted, and thus cannot scale beyond a certain (rather low) threshold.

Schank & Riesbeck (1981) collected results from SAM and four other comparable text understanding programs that more or less operate in a similar way, but on different scenarios and with slightly different internal structures. Like SAM, all of them rely on hand-coded knowledge bases. To the best of our knowledge, there is no script-based inference system that exceeds the capabilities of SAM or any of the other pioneering text understanding systems.

FRUMP

```
$DEMONSTRATION
 Predicted Events:
   1: The demonstrators arrive at the demonstration location.
   2: The demonstrators march.
   3: Police arrive on the scene.
   4: The demonstrators communicate with the target of the demonstration.
   5: The demonstrators attack the target of the demonstration.
   6: The demonstrators attack the police.
   7: The police attack the demonstrators.
   8: The police arrest the demonstrators.
```

Figure 2.4: A *sketchy script* for the DEMONSTRATION scenario used in the FRUMP system.

Another early and famous script-based system was designed by DeJong (1982). His FRUMP (*Fast Reading Understanding and Memory Program*) summarization system focuses on covering diverse scenarios. The main idea was to gain coverage by replacing the detailed common sense scripts encoded for SAM with a representation that focuses on essential events of typical newspaper stories. DeJong invented *sketchy scripts* for this purpose, and built a knowledge base of them with scenarios like PUBLIC DEMONSTRATION, EARTH QUAKES or CAR ACCIDENTS. Each scenario was annotated with the events considered important and expected in reports on new episodes of the a scenario.

In contrast to SAM, FRUMP was designed for general newspaper texts, and thus does not contain common sense scenarios like FEEDING A DOG.

Figure 2.4 shows a sketchy script for the DEMONSTRATION scenario. The events are temporally ordered, the roles remain implicit but can be resolved due to the consequent naming (e.g. *demonstrators* are always called *demonstrators*). Sketchy scripts are the basis for automatic summarization of newspaper articles by determining which sketchy script to use, finding the predicted events in the text and summarizing them to one sentence.

FRUMP was designed as a stand-alone component for summarization. It was also successfully integrated into an Information Extraction system as a script-based summarization component (Rau *et al.*, 1989).

Other manually coded script knowledge bases

During the 90s and early 2000s, there were very few developments around script learning. However, script-related research never completely subsided. There were some notable attempts to collect script-like knowledge and make it available for semantic processing or more targeted applications like image sorting. Similar to the earliest attempts at script processing, those knowledge bases were manually coded. They differ from each other in size, target application and depth of the knowledge applied. In the following, we summarize the most important examples.

ThoughtTreasure

ThoughtTreasure (Mueller, 2000) is a manually encoded common sense knowledge base that contains – among other types of facts – 94 scripts for different scenarios. The scenarios contain plain common sense knowledge (e.g. GO FOR A HAIRCUT, GROCERY SHOPPING) as well as less common events (US SPACE SHUTTLE FLIGHT, MEIOSIS).

Figure 2.5 shows example scripts from ThoughtTreasure that describe the scenario GROCERY SHOPPING and the related scenarios PAY-TO and PAY-CASH (the entry is slightly shortened for readability).

The script representations provide linguistic information in the form of possible verbal realizations, like *go shopping* and *shopping* for GROCERY SHOPPING (`V` and `N` stand for the part of speech, `z` stands for 'English language'). Each script contains references to different types of participants and goals, as well as pre- and post-conditions for the whole scenario.

ThoughtTreasure arranges scripts in a deep hierarchy, not only listing their associated events but also showing hyponymy and meronymy relations between scenarios: e.g. PAY is an event within the GROCERY SHOPPING script (meronymy), and it has several possible instantiations, e.g. PAY BY CHECK, PAY BY CARD and PAY IN CASH (hyponymy, marked with `ako`). Those instances then have their own script information (roles, goals and events) associated with them.

Mueller (1998) shows how the different components in ThoughtTreasure contribute to different natural language processing tasks, e.g. for automated storytelling and text understanding.

Gordon's *common sense Activities*

Gordon (2001) crafted a large database of 768 *common sense activities* in order to create a search index for images. The database was manually created by an expert annotator.

```
grocery-shop grocery_shopping-Nz go-Vz go-Vz go-Vz shop-Vz
 [ako grocery-shop necessary-shop]
 [used-for shopping-cart grocery-shop]
 [used-for shopping-bag grocery-shop]
 [used-for shopping-basket grocery-shop]

 [role01-of grocery-shop shopper]
 [role02-of grocery-shop food-store]
 [role03-of grocery-shop checkout-person]
 [role04-of grocery-shop food]

 [goal-of grocery-shop [s-hunger shopper]]
 [goal-of grocery-shop [s-profit food-store]]

 [entry-condition-of grocery-shop [low-on-food shopper]]
 [result-of grocery-shop [stocked-up-on-food shopper]]
 [related-concept-of grocery-shop expensive]

 [event01-of grocery-shop [write-shopping-list shopper]]
 [event02-of grocery-shop [select-from shopper na food]]
 [event03-of grocery-shop [put-in shopper food shopping-cart]]
 [event04-of grocery-shop [goto event02-of]]
 [event05-of grocery-shop [pay-to shopper checkout-person
                                           financial-instrument]]
 [event06-of grocery-shop [return-home shopper]]
 [event07-of grocery-shop [put-away-in shopper food pantry]]
 [event07-of grocery-shop [put-away-in shopper food refrigerator]]


pay-to payer-Nz pay-Vz pay-Vz
 [ako pay-by-check pay-to]
 [ako pay-by-card pay-to]
 [ako pay-cash pay-to]


pay-cash pay-Vz [ako pay-cash pay-to]
 [role01-of pay-cash buyer]
 [role02-of pay-cash seller]
 [role03-of pay-cash currency]
 [role04-of pay-cash wallet]

 [result-of pay-cash [holding seller currency]]

 [event01-of pay-cash [take-from buyer wallet currency]]
 [event02-of pay-cash [hand-to buyer seller currency]]
 [event02-of pay-cash [collect-from seller buyer currency]]
```

Figure 2.5: Three scripts from ThoughtTreasure, describing the scenario GROCERY SHOPPING and its daughters PAY and PAY WITH CASH.

Being abducted by space aliens	Building a snowman
:Events	:Events
Experiments	Children playing in snow
Interplanetary voyages	
Kidnappings	
Surgery	
:Places	:Places
Forests	Parks
Unidentified flying objects	
:Things	:Things
Electronic apparatus & appliances	Carrots
Extraterrestrial life	Coal
	Gloves
	Snow
	Snowballs
	Snowmen
:Misc	:Misc
Night	Winter
Tall tales	

Figure 2.6: common sense activities for Alien Abduction and Building a Snowman

The scenario catalogue is based on the pictures: the annotator assigned one or more scenarios to each image, and then entered the scenario with its relevant elements into the database (if it was not present there yet).

Figure 2.6 shows two example activities from this common sense activity set.[1] The full set of activities comprises a wide range of topics reaching from standardized common sense scenarios (like building a snowman) to fantasy-story content (e.g. being abducted by space aliens). The representations contain *events* (without temporal ordering), *places* and *things* associated with the scenario. *Things* roughly correspond to participants, but usually don't include the protagonists (like children or space aliens). Gordon also notes some free associations with the scenario that don't fit any other category. Labelled as miscellaneous (*Misc*), they likely serve to establish commonsense-based (but script-irrelevant) connections in the image index (e.g. time or season information, like that alien abductions usually happen at night).

Because the targeted system for image indexing had no linguistic focus at all, the database itself does not give any information about linguistic realizations of the events.

[1] http://people.ict.usc.edu/~gordon/downloads/Activities.txt

FrameNet

Taking up the original frame idea in which scripts have their roots, FrameNet (Baker *et al.*, 1998) was designed as a semantic database with frames and their associated *semantic roles*. FrameNet offers a comprehensive set of frames with different degrees of complexity (like buy, cause motion or get a job).

Commerce Buy:	These are words describing a basic commercial transaction involving a Buyer and a Seller exchanging Money and Goods, taking the perspective of the Buyer. The words vary individually in the patterns of frame element realization they allow. For example, the typical pattern for the verb buy: *Buyer* buys *Goods* from *Seller* for *Money*.
Core FEs:	*Buyer, Goods*
Non-Core FEs:	*Charges, Court, Jury, Judge, Defendant, Offense, Prosecution, Defense, Suspect, an Authority*
Inherits from:	***Getting***
Is Inherited by:	***Renting***
Perspective on:	***Commerce Goods Transfer***
Is Used by:	***Importing, Shopping***
Lexical units:	*buy.v, buyer.n, purchase.n, purchase.v, purchaser.n*

Figure 2.7: Definition of the FrameNet frame commerce buy.

Figure 2.7 shows a FrameNet entry for the Commerce Buy frame. A textual definition is followed by *frame elements* (FEs), which are either required in all realizations of the frame (*core*) or can be omitted (*non-core*). In the case of buying something, the process must minimally name a buyer and the good that is acquired. Additionally, there can be a seller, money, a manner of buying, and other related frame elements

To anchor frames in texts, each frame defines several *lexical units* that evoke the frame. A sentence is taken as a textual instance of a frame if it contains one of its lexical units.

FrameNet arranges the frames in a type hierarchy, and provides semantic relations between frames: *Inheritance* means that a frame inherits frame elements from a more general one (e.g. Getting is less specific than Buying), or can be re-used as a parent that passes its element to a more specific frame (Renting is a special case of Buying). If a frame is another *Perspective on* a second frame, this roughly corresponds to verbal passive-active alternations: Commerce Buy in describes the same scenario as Commerce goods transfer, but focussing on the buyer's perspective; Commerce Sell is another frame with *perspective on* Commerce Goods Transfer, but with focus on the seller.

FrameNet also contains special frame relations in so-called *scenario frames*, which basically encode script knowledge: In a scenario frame, *subframes* denote partial events necessary for executing the parent frame. One example for a scenario frame is the frame Criminal Process, which is (in an abbreviated form) displayed in Figure 2.8:

Criminal Process:	A Suspect is arrested by an Authority on certain Charges, then is arraigned as a Defendant. If at any time the Defendant pleads guilty, then the Defendant is sentenced, otherwise the Defendant first goes to trial. If the Verdict after the trial is guilty, then the Defendant is sentenced. In the end, the Defendant is either released or is given a Sentence by a Judge at the sentencing.
Core FEs:	*Charges, Court, Jury, Judge, Defendant, Offense, Prosecution, Defense, Suspect, an Authority*
Subframes:	***Arraignment, Arrest, Sentencing, Trial***

Figure 2.8: Definition of the scenario frame criminal process.

In terms of a script , the parent frame denotes a scenario, the subframes are the events, and the frame elements correspond to participants. The temporal order between subframes is marked with frame relations: the Trial frame stands in a *precedes* relation to sentencing, and in a *is preceded by* relation with arraignment.

One of the applicational goals underlying FrameNet was the support of diverse text understanding tasks (Fillmore & Baker, 2001). In connection with automatic semantic role labelers (see Palmer *et al.* (2010) for an overview), FrameNet was indeed successfully applied to textual entailment (Ben Aharon *et al.*, 2010), generating paraphrases (Coyne & Rambow, 2009; Ellsworth & Janin, 2007) and question answering (Narayanan & Harabagiu, 2004; Shen & Lapata, 2007). Like Minsky already noted (Minsky, 1974), frame application is not restricted to language, and, in fact, FrameNet has been used for the multimodal application of text-to-scene generation (Coyne *et al.*, 2010).

For general-purpose applications, FrameNet has emerged as a very versatile and precise resource, which exhaustively covers frames and their lexical realizations. FrameNet also comprises script information, but scenario frames are very rare in FrameNet, and they are not systematically marked and thus hard to retrieve at all. We view the FrameNet approach as complementary to script processing in general, whereas FrameNet's applicability for script-related tasks remains to be explored in future work.

Automated script learning

Manual encoding of script knowledge hardly scales up to sufficiently large knowledge bases. Such expert-annotated resources are restricted to what the annotators can recall, and it is tedious to create such complex data structures. With the availability of modern machine learning methods and large corpora to train them with, the idea of learning scripts automatically offers itself. The following details three approaches that learn script knowledge from texts: *GENESIS*, *narrative schemas* and *event n-grams*.

GENESIS

The *GENESIS* system (Mooney, 1990) is an early approach to learning and applying script-like knowledge. The system uses structures that are very similar to scripts (called *schemas*) for text-based reasoning in order to answer questions about short narratives, similar to *SAM*. The main difference to the first script-based systems is that *GENESIS* can in principle acquire new schemas automatically. The system stores event sequences from seen texts, and turns them into potential schemas for understanding new texts by replacing descriptions of participants with variables. The core idea is compelling: *GENESIS* can automatically determine which schemas are worth storing, and it can analyze new texts with deep reasoning by applying the schemas it has learned. However, the texts that *GENESIS* works with are not very natural, but rather seem as artificial as the sample texts from the very first script-based question answering systems. In this respect, *GENESIS* replaced the manual annotation of scripts with the manual creation of sample texts, which is unfortunately not sufficient for scaling up script data collection.

Narrative schemas

Chambers & Jurafsky (2008b, 2009) exploit coreference chains and co-occurrence frequency of verbs in text corpora to extract *narrative schemas* describing sequences of events and their participants.[2] Because this approach is fully unsupervised, its coverage is in principle unlimited. Each schema provides a family of verbs and arguments related by the same narrative context. Roughly speaking, verbs end up together in a schema if they tend to share the same arguments when mentioned in the same text.

Figure 2.9 shows an example for a narrative schema. The first score gives an overall confidence value for the output (the schema we show here is the second best schema of size 6). The events are represented by single verbs, and their order reflects the order in which the mining algorithm added them to the schema. Events have scores that indicate

[2] See `http://cs.stanford.edu/people/nc/schemas` for the data.

```
score=16.873066
  Events: sell buy borrow buy_back repurchase own
  Scores: 6.798 5.997 5.981 5.826 4.783 4.361

  [sell-s buy-s borrow-s buy_back-s repurchase-s own-s]
   company  8.056   investor  7.350    trader  5.834   corp       5.742
   enron    5.647   bank      5.481    inc     5.460   government 5.432
   sellers  5.313   itt       5.299    family  5.299   gm         5.289

  [sell-o buy-o borrow-o buy_back-o repurchase-o own-o]
   share   8.817  stock    7.575  stocks   7.193   bond   7.020
   company 7.013  security 6.821  team     6.676   house  6.644
   funds   6.632  land     6.563  property 6.563   dollar 6.562
```

Figure 2.9: A narrative schema with 6 events & most strongly associated participants.

their association strength with the events that were added to the chain before.

The schema's fillers correspond to script participants. They are specific to either subject or object slots in the chain, here abbreviated with `s`(ubject) and `o`(bject). In the example, the fillers in the first set only occur in subject slots, and the ones in the second set only in object positions (we only noted the first 12 for each slot). In some of the schemas, fillers can also alternate between the slots (like `[ask-s answer-o]`). Each filler is labelled with a confidence value, indicating its association strength with all verbs in the schema in its slot (subject or object). Fillers are determined independently from any filler in the other slot; their association scores take only the verbs and their grammatical role into account.

Chambers & Jurafsky provide a temporal classification of the schema events, computed by an automated temporal classifier (Chambers & Jurafsky, 2008a; Chambers *et al.*, 2007). Apart from this temporal classification, the relations between the verbs remain underspecified: Two verbs of a schema might describe the same, different or contradictory events. The aim here is not to collect data describing predetermined activities, but rather to establish verb groups that share an (unknown) underlying scenario.

In their format and the scenario coverage, narrative schemas are similar to the "sketchy scripts" used in FRUMP (DeJong, 1982): both of them target scenarios present in newspapers, and while the "sketchy scripts" actually do have scenarios assigned, FRUMP's inference component does not use those scenario assignments. Consequently, the schemas can probably, like FRUMP, serve for similar applications to summarization and information extraction, but on a much larger scale.

Overall, narrative schemas scale up much more easily than any manual script collection, which enables their use for general text processing in the first place. Chambers & Jurafsky themselves apply the schemas to predicting missing events in a larger context. The approach can in principle cover arbitrary scenarios, as long as they are actually elaborated in text – however, the scenarios themselves are not determined. For two different schemas, it's not clear whether they describe similar scenarios or even the same one, or whether they are completely distinct.

The permissive high-coverage algorithm needs no supervision and thus could scale up easily, but consequently also results in a lot of unfiltered noise: The events within a scenario are only loosely connected, and the events have only vague relationships to their participants. This is already evident from the example: The highest-ranking slot-fillers are *company* (subject) and *share* (object), but instantiating the schema with those two participants does not result in a meaningful event chain, because shares can't be borrowed, and the shares that a company typically sells (their own) are not the same ones they buy (if they buy any at all).

Further, there is no clear assignment of temporal ordering. While the associated temporal classifier assigns ordering constraints for pairs of events, the events named in the schema do not even have a natural sequential order: *buy back* and *repurchase* are synonyms, the order of *sell* and *buy* depends on the context, and *borrow* does not fit into the chain at all. The noise in the schema collection is the price one has to pay for such robust unsupervised data collection.

As it stands, the resource does not contain the type of common sense knowledge we are after, but rather higher-level scenarios like TRADING events. This is mainly due to the problem of implicitness: more elementary scenarios simply cannot be learned from text.

The purpose of narrative schemas is not to represent full-fledged script representations including ordering constraints and precise event-participant-relations. They rather provide a wide-coverage collection of scenario-based clusters that contain related event and participant descriptions. The result can still be highly useful, especially because it easily scales up to many scenarios. For actual script-based application, the schemas would have to be extended with linguistic variants for events and participants, and they should provide a clearer assignment of relations between events as well as more accurate event-participant associations.

Event n-grams

Manshadi *et al.* (2008) present another corpus-based learning approach: They built an n-gram model for events, based on sentences found in a large corpus. Similarly to Chambers and Jurafsky, they also successively generate event sequences of verb-complement tuples. Under this approach, the best follow-up of an intermediate sequence is the verb-complement tuple that has the highest association value with the previous events.

Swanson & Gordon (2008) exploit a related model from the same source dataset in an automated storytelling system.

There are some essential similarities between narrative schemas and event n-grams: both approaches operate on corpus data and have a high potential to solve the coverage problem. At the same time, they are both tied to scripts that are actually elaborated in text, and to the level of detail that is usually verbalized. Neither approach represents variances of a scenario in its model, but both approaches bear the potential to extend their models in such a respect and include sufficient amounts of data to cover more than one variant for the same scenario.

Apart from their commonalities, the two systems differ in three core points: first, event associations for n-grams are based on plain co-occurrence rather than coreference chains. However, the data source for the event n-grams are personal blogs that often contain stories around one specific protagonist, thus Gordon & Swanson (2009) assume that there is some participant-based coherence in the n-grams. Second, event n-grams do not model participants. Events are treated as opaque structures, including the verb complements. Lastly, the n-gram model treats textual order as temporal precedence, whereas Chambers & Jurafsky offer a temporal ordering based on a classifier.

A performance comparison between the two approaches is not possible, because they were applied to completely different domains and completely different tasks.

2.3 Previous Systems and Script Learning Challenges

As we have argued earlier (cf. Section 2.1), a script mining system should ideally learn representations for known scenarios, consisting of typical events in this scenario, their typical order, and participants. It would also be desirable to capture as many variants of a scenario as possible, and, optimally, represent those variants in a concise way along with lexical variants that allow for anchoring the script representation in text.

Measured by those goals, both manual and automatic approaches suffer from at least one coverage problem, as well as the problem of implicitness:

Covering many scenarios is easier to achieve for machine learning approaches, because manual script encoding efforts are tedious and time-consuming. However, automatic approaches have at the same time the **problem of implicitness**: very conventionalized common sense scenarios are typically not realized in standard text, because writers can assume that each reader knows things like that one needs to open a can for feeding a dog, and that dog food can also come in bags. So while corpus-based learning enables the creation of much larger databases than manual expert-encoding, neither of the approaches can lead to a sufficiently big database of common sense scripts.

Covering many variants of a scenario is a two-fold problem: On the one hand, one needs data that supports many possible sequential and lexical variants of a scenario, which is a problem for manually created resources. On the other hand, the final representation of a script needs to be expressive enough to actually represent different variants. This has not been attempted for any of the automatic learning approaches, and only partially tried by the early proof-of-concept systems.

Chapter 3

Corpora of Written Script Data

To compute script representations, we first of all need appropriate input data. The data we are looking for are descriptions of prototypical event courses for a scenario, from which we will later compute more general script representations.

We collect such data via *crowdsourcing*, which means that we distribute our task to a huge number of people over the internet. This technique is particularly suitable for acquiring script data, for several reasons:

First of all, the task does not require any expert knowledge, but rather commonsense knowledge about simple everyday activities, which the largest part of internet users can describe. Crowdsourcing also allows us to address the problem of implicitness adequately, because we do not depend on "natural" standard texts but can ask precise questions about the information we want.

Last but not least, our dataset will associate each entry with its scenario. This comes at the cost of a little supervision, because we need to pre-determine the scenarios we want to collect data for. However, corpus-based approaches that extract event sequences from texts deliver no information on the association of event sequences with scenarios at all, and it is not clear whether they could easily be extended to accomplish this.

This chapter is structured as follows:we first describe how to collect script data with Mechanical Turk in general, using scenarios as stimuli and yielding multiple event sequence descriptions for each scenario (Section 3.1).

The next three sections report on our first corpus collection, which contains a broad range of commonsense scenarios and is the basis for all our script mining experiments. Section 3.2 explains some details on the scenario selection for this experiment and shows some examples from the resulting data. Section 3.3 outlines how we clean the raw corpus and how we build new preprocessing systems that fit the idiosyncratic language of the event descriptions. A statistical analysis (Section 3.4) completes the description of the first corpus and gives some insights on the corpus size, the vocabulary, and similarities within and between scenarios.

Section 3.5 describes our second data collection which covers one specific domain (the cooking domain) in depth. At last we give a short description of the OMICS corpus, a data collection from MIT which is very similar to our corpora (3.6).

Parts of this chapter describe work published by Regneri *et al.* (2010).

3.1 Collecting Script Data with Mechanical Turk

Mechanical Turk[1] (MTurk) is a crowdsourcing platform provided by Amazon. The platform has been used for a variety of language processing related tasks recently, and can even replace expert-annotations for some tasks (Snow *et al.*, 2008).

In Mturk, any registered requester can upload tasks, and MTurk manages distribution of tasks to its registered workers. If, as in our case, the same task needs to be done by several different people, MTurk will guarantee that the requested number of assignments is completed by different workers (or at least: different MTurk accounts).

One assignment for one worker is called a *HIT (Human Intelligence Task)*. For each HIT, the requester decides how much money a worker will earn for completing it. After the worker is finished, it is up to the requester to either accept or reject the completed assignment; in case of rejection, the worker will not receive any money; in case of acceptance, Amazon transfers the money to the worker. The service fee for Amazon is 10% of the amount paid to the workers (10%).

In a event data collection for a scenario like EATING IN A FAST FOOD RESTAURANT, we ask the annotators what they usually do in such a scenario. We use two question templates, *What do you do when you...?* and *How do you...?*, which are randomly assigned to the scenarios and completed with the scenario titles (e.g. *What do you do when you eat at a fast food restaurant?*). The annotation interface shown on MTurk is a plain form with distinct fields for event descriptions (16 fields in our case). The annotators had to enter a minimum number of events (4 in our case) per scenario, making up a concise description of a typical event course for the given scenarios. We call the short texts collected in that way *event sequence descriptions* (ESDs).

We also present a short annotation guideline and the beginning of an exemplary ESD for a scenario which is not part of the annotation set. The guidelines advise the annotators to list the events in sequential order.

[1]`http://www.mutrk.com`

In order to assess how hard this task is and how difficult certain scenario descriptions are, we give the participants the option to skip a scenario if they feel unable to enter events for it. In order to still earn the full HIT compensation, they have to indicate why the scenario is impossible for them to describe. In the form, they have to answer a multiple choice question, offering the choice between four options to earn the money:

1. *"I described the scenario in the form"* is selected by default and means that the annotator needs to provide at least the minimum number of events for the ESD (in our case: 4).

2. *"I don't know how this works"* allowes the annotator to skip the scenario, indicating that he or she is simply not familiar with the activity in question.

3. *"This has no meaningful subevents"* is meant for cases in which the granularity of the scenario is too high or too low to give a prototypical, ordered sequence of at least 4 events.

4. A third option for skipping requires the worker to explain the reasons in a comment field. The field can also be used to explain one of the other skipping options in more detail.

We admit only workers that had at least one third of their overall HITs accepted, but put no other restrictions on the participants. One HIT, i.e. providing one ESD for one scenario, pays $0.09 upon successful completion.

With this experimental setup, we collect two corpora: the first one was designed to cover a broad range of common-sense scenarios. We provide a very detailed analysis of this corpus (Section 3.2 - 3.4), because all follow-up experiments on paraphrasing and script extraction use this corpus as input data (cf. Chapter 4 & 5).

In a second corpus collection, we aim to thoroughly cover a narrow domain. This second data collection results in a larger corpus containing fine-grained scenarios from the kitchen domain (Section 3.5).

3.2 Assembling a Corpus of Common Sense Scripts

In our first corpus collection, we create a corpus that contains a balanced selection of commonsense scenarios. In this section, we describe how we selected the scenarios for this corpus and shows some examples from data we obtained.

eating in a restaurant	eating in a fast food restaurant	sending food back (in a restaurant)
taking a bus	checking in at an airport	flying in an airplane
taking a train	taking a driving lesson	fixing a flat tire on a bike
going shopping	paying (after buying sth.)	paying with a credit card
ironing something	taking a shower	putting a poster on the wall
cleaning up a flat	creating a homepage	taking copies (with a copy machine)
going for a haircut	a wedding ceremony	childhood
	making scrambled eggs	

Figure 3.1: The 22 Scenarios for which we collected data on MTurk.

Scenario Selection

We aimed to create a corpus which is as representative as possible. We manually chose scenarios, considering the following objectives:

- We targeted *common sense* scenarios, i.e. scenarios that are usually shared knowledge between humans with the same cultural background. We excluded scenarios that require expert knowledge (e.g. PROGRAMMING A MICROCONTROLLER) or cover very long, standardized processes (like BUSINESS ACQUISITION).
- Within this scope, we chose scenarios of different complexity. Some require more specific experience (CREATING A HOMEPAGE) and some are presumably easy to describe (TAKING A SHOWER). Furthermore, the scenarios differ in the number of sub-events (PAYING WITH A CREDIT CARD VS. FIXING A FLAT TIRE) and how conventionalized type and order of their sub-events are (TAKING A BUS VS. CHILDHOOD).
- We intentionally included some scenario pairs that have certain *semantic relations*:
  - *Event hypernyms;* PAYING WITH CREDIT CARD is a hyponym of PAYING, and EATING AT A FAST FOOD RESTAURANT is a more specific version of EATING AT A RESTAURANT.
  - *Event holonyms:* PAY is a part of several other scenarios (e.g., GOING SHOPPING OR EATING AT A RESTAURANT), and CHECKING IN AT AN AIRPORT is a part of FLYING IN AN AIRPLANE
  - *Event sister-terms:* Some scenarios are hyponyms of the same more general process, e.g. TAKING A BUS, TAKING A TRAIN and FLYING WITH AN AIRPLANE are all ways to use public transportation.

Figure 3.1 shows the final set of 22 scenarios that we used for the first study. One main objective was to learn which kinds of scenarios were particularly easy or hard to describe, which lead to particularly similar or dissimilar results, and whether the data sets of related scenarios are comparable to each other.

HOW DO YOU EAT AT A FAST FOOD RESTAURANT?

1. walk into restaurant
2. find the end of the line
3. stand in line
4. look at menu board
5. decide on food and drink
6. tell cashier your order
7. listen to cashier repeat order
8. listen for total price
9. swipe credit card in scanner
10. put up credit card
11. take receipt
12. look at order number
13. take your cup
14. stand off to the side
15. wait for number to be called
16. get your drink

---

1. look at menu
2. decide what you want
3. order at counter
4. pay at counter
5. receive food at counter
6. take food to table
7. eat food

---

1. walk to the counter
2. place an order
3. pay the bill
4. wait for the ordered food
5. get the food
6. move to a table
7. eat food
8. exit the place

---

1. enter restaurant
2. go to counter
3. make selection
4. place order
5. pay for food
6. pick up order
7. pick up condiments
8. go to table
9. consume food
10. clear tray

Figure 3.2: Four ESDs describing the scenario of EATING AT A FAST FOOD RESTAURANT.

Results

On Mechanical Turk, we created 25 HIT assignments per scenario, which means that 25 different people got the chance to enter an ESD for a scenario. Annotators could enter data for all 22 scenarios, but submit only once per scenario. In this first experiment, we encouraged the participants to use "bullet point style" for event descriptions, which we pushed further by restricting the length of each event description to 250 characters. We hoped that these style requirements would lead to short and focussed event descriptions, which are fairly uncomplicated to analyze later on. We did not precisely define what "bullet point style" means, but rather gave examples like "*1. open door, 2. get into car, ...*" for the example scenario DRIVING SOMEWHERE BY CAR.

In total, we collected 493 ESDs across the 22 scenarios. The annotators used the possibility to skip a form 57 times; the most frequent explanation for this was that they didn't know how a certain scenario worked (40 times). The scenario with the highest proportion of skipped forms was CREATING A HOMEPAGE (6 Hits, i.e. 25%), whereas MAKING SCRAMBLED EGGS was the only one in which nobody skipped an assignment.

Figure 3.2 shows four of the ESDs we collected for EATING IN A FAST-FOOD RESTAURANT. The example shows that descriptions of the same scenario can differ in various aspects:

- *Starting point:* Without further specification, is not clear where or when to begin with EATING IN A FAST FOOD RESTAURANT. The event sequence might start at somebody's home, or in the restaurant, or anywhere in between. As a consequence, some people start with *walk into restaurant*, and others assume that they just came inside (*walk to the counter*) or are already standing at the counter (*look at menu*).

- *Endpoint:* In the same way the scenario's starting point is underspecified, there is also no fixed point at which it ends. Some people finish with *exit the place* (regardless whether they mentioned entering it in the beginning), some people stop after the last action inside the restaurant (*clear tray*). Probably influenced by the wording of the question (*How do you eat...?*), some people immediately stop after *eat food*, but even that does not seem to be obligatory (cf. the first sequence).
- *Granularity:* Script structures are recursive: the events within a script are scenarios of their own, which have event sequences associated with them. When writing up ESDs, people thus have the choice to insert a high-level event like *pay* or insert a whole ESD that makes this process explicit (like steps 8–11 in the first scenario).
- *Wording:* Naturally, people have different ways of realizing event descriptions in natural language. *decide on food and drink*, *decide what you want* and *make selection* all refer to the same step in the scenario, but they vary largely in their wording.

We show some statistics on commonalities and mismatches of ESDs within the same scenario in Section 3.4.

3.3 Preprocessing

The texts we obtained were rather noisy, which might partly be due to the fact that we opposed no restrictions on the MTurk participants. (Admitting only native speakers as participants or managing the task accessibility with an entrance test could probably help to filter out people with insufficient language competence.)

Initially, we manually filtered unusable instances and corrected spelling mistakes. For preprocessing, we later developed an automated spell-checking algorithm for the script data. Additionally, we built a parsing model capable of processing the idiosyncratic bullet-point language style.

Manual corrections

For the first experiments, we manually corrected the data for orthography and grammar mistakes, with the objective of creating a particularly clean corpus. Some of the ESDs were completely discarded, mainly for one or a combination of the following reasons:

How do you Iron Something?

1. First i open the bedshit
2. neetly in the floor
3. then take a Iron box
4. then iron box blugged in power
5. i take a what dress is iron
6. Iron box is adjusted in what
7. type of clothes, ex: cotton
8. Heat is normaly i start iron
9. i do dresses full side iron
10. then finish the iron power is
11. switch off & removed.
12. Iron the dresses is put the
13. anchor, when want me wearning.

How do you create a homepage?

1. Decide the Web Page
2. Go to tools on the Screen
3. it will show you, which one is presently as a home page.
4. Cancel it.
5. Print your Favourite site
6. click ok. and click apply.
7. Now your Fevorite home page is ready to use.
8. you can do it setting path too
9. i explained one of the easy method to make home page.

What happens during a driving lesson?

1. excited
2. fun
3. happy
4. first time
5. open car door
6. sit in drive seat
7. get a feel of the car
8. instructor beside you
9. gives instructions
10. follow them
11. learn how to drive
12. time consuming
13. get a license

How do you take a shower?

1. nozzle sprays
2. requires water transportation
3. hygenic
4. practical and versatile choice
5. shower curtains

How do you send food back (in a restaurant) ?

1. I MEET MY OLD FRIENDS
2. WE ARE EATING CHAPPATHI
3. THIS RESTURENT WAS LUXARY TYPE
4. SPENT MONEY
5. IN A RESTURENT I SPENT MONEY
6. RESTURENT BOY IS A GOOD MAN

Figure 3.3: Some examples for sequences we discarded manually.

- *Incomprehensible language:* Sequences that were not correctable or not even understandable were discarded.
- *Misunderstanding of the scenario:* Sometimes the workers mistook one scenario for another, e.g. EATING IN A FAST FOOD RESTAURANT for EATING IN A RESTAURANT.
- *Misunderstanding of the task:* Some annotators either did not understand the task or intentionally tried to defraud by entering arbitrary words. The results are not in sequential order, or do not even contain event descriptions but rather associative sets of nouns, adjectives and verbs.

Overall we filtered around 15% of the ESDs out. Figure 3.3 shows some self-explanatory examples that we discarded for various reasons.

Scenario-specific spelling correction

Because we are working towards large-scale data collection, we automatized the preprocessing as far as possible. Spelling mistakes are the most important noise factor in our corpus. At the same time, the crowdsourced data often contains colloquial terms or domain-specific words that we want to have in our dataset, but which would not be recognized as correct by standard spell checkers. We developed a spell-checking algorithm that can capture scenario-specific terminology without the need of external resources.

The algorithm was developed in the context a larger follow-up study to our first experiment: we collected a corpus of ESDs for several cooking scenarios (cf. Section 3.6). The results were published by Bloem *et al.* (2012).

In a first attempt to preprocess the event sequence descriptions, we tried to use a standard spelling correction system (GNU Aspell2[2]) out of the box. Manual inspections reveal a quite unsatisfying precision of the results: in a manually inspected a set of of 162 automatically corrected instances, Aspell has a precision of only 43%. This is due to two shortcomings in the spell-checker: at first, many domain-specific or colloquial words are not in Aspell's dictionary. Further, Aspell does not rank its suggestions for spelling corrections; the order in which suggestions are returned is solely based on the edit distance to the misspelled token, and suggestions with equal distance are randomly ordered. These shortcomings often leads to implausible edits, sometimes "correcting" words which were not misspelled at all, or replacing misspelled words with something that is inadequate in the context.

Because we have for each scenario many texts that describe it, correct words tend to occur in more than one event sequence descriptions. Leveraging the redundancy in the parallel data helps us to improve the spell checker's performance with respect to the actual identification of misspelled words as well as the choice between different correction suggestions.

Objective one is to gain more precision for the identification of misspellings: we indirectly expand Aspell's dictionary with scenario-specific vocabulary. If a word occurs in at least n other sequences with the same spelling, our system accepts it as correct. (In our experiment, $n = 3$ turned out to be a reliable value.)

As a second enhancement, we improve the *correction process* by guiding the selection of corrected words. We verify their plausibility by taking the first suggested word that occurs in at least one other sequence. This makes sure that e.g. the misspelling *(credit) carf*, is corrected to *card*, whereas Aspell might return *care* or *calf* as (randomly chosen) first suggestion.

To evaluate our modifications, we compare our system's performance to a baseline of standard Aspell, always picking the first correction suggestion. The gold standard consists of all corrections made for all ESDs from 10 scenarios. Our system has no features that can change standard Aspell's recall, so we have not evaluated recall performance. Fig. 3.3 shows a comparison of both methods using various measures of precision, based

[2]Available from `http://aspell.net/`

System	Prec.	FPs	True Prec.	Sem. Prec.	corrections
Aspell	0.43	0.28	0.57	0.58	162
Enhanced Aspell	**0.58**	0.29	**0.79**	**0.76**	150

Figure 3.4: Evaluation of the baseline and improved spell-checker.

on manual judgement of the corrections made by the different Aspell versions.

Since Aspell's spelling error detection is not perfect, some of the detected errors were not actually errors, as shown in the "False Positives" (FPs) column. For this reason, we also included "true precision" (True Prec.), which is calculated only over actual spelling errors. Another measure interesting with respect to later processing is "semantic precision" (sem prec.), a more loosely defined precision measure in which a correction that results in any inflection of the desired lemma is considered correct, ignoring grammaticality. The last column shows the overall number of errors corrected (reduced by 12 after the dictionary expansion). Overall, we gain 15% precision, and even 22% by considering genuine misspellings only. If we relax the measure and take every semantically correct (and thus processable) lemma form as correct, we gain an overall precision of 18%.

Similar work on spelling correction has been done by Schierle *et al.* (2008), who trained their system on a corpus from the same domain as their source text to gain context. We take this idea a step further by training on the noisy (but redundant) dataset itself, which makes definition and expansion of the domain superfluous.

Scenario-specific parsing[3]

For several follow-up analyses, we need syntactic information for our event descriptions (cf. Chapter 5 & 6). Parsing the descriptions is a challenge, because the data is written in telegraphic style (cf. Figure 3.2), thus each description is a minimalistic imperative sentence. The subject (typically the protagonist) is frequently left implicit, and determiners seem to be left out arbitrarily (cf. *walk into restaurant* vs. *walk to the counter*).

Similarly to the spelling-correction approach, we first tried to use a pre-trained standard parsing model, which is trained on newspaper texts. In all our experiments, we use the Stanford parser (Klein & Manning, 2003). To evaluate the standard model's performance on our data, we manually labelled a set of 100 example event descriptions with phrase structure trees. Used out of the box, the Stanford parser finds the correct parse tree in 59% of the cases. The most frequent and most serious error is misclassification of the phrase-initial verb (like *walk* or *look*) as a noun, which often leads to subsequent errors

[3]For all experiments related to parser model adaption, we had invaluable support by Ines Rehbein.

in the rest of the event description parse.

Our available dataset of event descriptions is much too small to serve as a training corpus of its own. To achieve better parsing accuracy, we combine and modify existing resources to build a refined parser model. We included three standard corpora in our training corpus, namely the Penn Treebank (Marcus *et al.*, 1993), the ATIS corpus (Dahl *et al.*, 1994) and the Brown corpus (Francis & Kucera, 1979). To adapt the model to our corpus' bullet-point style language, we additionally put modified versions of ATIS and Brown into the training data. The modification consists of pruning the standard parse trees such that they look like our minimalistic imperatives: we deleted all subjects in the sentences and all the determiners. Re-training the parser's model on the such adapted training data raises the rate of correctly computed trees to 72%. All the following experiments that use syntactic information employ parses from the Stanford parser under this adapted model (if not indicated otherwise).

3.4 Corpus Statistics

In this section we report on our corpus' size, its contents and their variance. Apart from purely quantitative benchmarks, we also want to give some intuition on similarity and variance with respect to the scenario-based sub-corpora. We consider two types of variance: the first one examines the homogeneity of a scenario, measured by the (dis-)similarity of ESDs in the respective sub-corpus.

The second aspect concerns (dis-)similarities between different scenarios, which shows whether the sequences of two different scenarios are different on the surface level, and to which degree sequences of related scenarios are similar.

Corpus size and vocabulary

Table 3.1 shows some general statistics for the corpus, grouped by scenario (scenario names are abbreviated for readability). We count sequences, event descriptions and words for each scenario and for the whole corpus. An *event description* is one element of a sequence (a sentence). *Content words* are words that were tagged as nouns, adjectives or verbs by the parser. (For word types, the *overall* sums are not the sum of the scenario rows, because a word type can occur in more than one scenario.)

Overall, we have a corpus of 386 event sequence descriptions, composed of 6,998 event descriptions. As the numbers show, the sub-corpora for the different scenarios vary greatly in size, with MAKING SCRAMBLED EGGS as the largest one and PAYING WITH A CREDIT CARD as the one with the fewest sequences. Interestingly, descriptions of the

Scenario	ESDs	event descriptions	word tokens	word types	content word types
restaurant	19	396	609	164	122
fast food	15	268	425	127	*96*
returning food	15	178	349	136	*91*
scrambled eggs	**24**	**510**	**1047**	**274**	**193**
flying	19	442	724	213	166
airport check in	19	332	651	204	149
taking bus	21	366	650	190	144
taking train	14	248	428	134	109
going shopping	20	360	619	181	133
paying	19	238	424	139	*98*
credit card	*6*	*68*	*133*	*62*	*42*
haircut	**23**	**476**	**826**	**261**	**199**
wedding	18	388	701	243	**207**
taking shower	21	**474**	**759**	157	112
fixing tire	19	304	583	164	117
cleaning up	17	312	495	209	165
driving lesson	16	298	502	211	165
ironing	20	360	696	185	133
creating hp	13	*192*	359	171	126
taking copies	20	342	684	230	154
poster	**23**	332	697	227	167
childhood	*5*	*114*	*190*	*102*	*83*
average	18	318	571	181	135
overall	386	6,998	12,551	1,662	1,586

Table 3.1: Quantitative corpus statistics grouped by scenario. Particularly high values are marked in **bold**, minima in *italics*.

WEDDING scenario contain the widest variety of content words, even though the scenario has far fewer descriptions (and thus fewer words) than the SCRAMBLED EGGS scenario. A possible explanation could be that events within the wedding ceremony are mostly complex and involve at least two people, so probably most function words were left out in order to fit more content words into the character-restricted form fields.

TYPE	COUNT	TYPE	COUNT	TYPE	COUNT	TYPE	COUNT
get	238	take	90	find	74	tire	64
put	140	food	89	pay	73	leave	64
go	130	hair	85	is	72	ticket	57
wait	129	bus	77	turn	72	order	57
check	105	iron	75	eggs	68	remove	56

Table 3.2: The 20 most frequent content words in the corpus.

Table 3.2 shows the 20 most frequent content words of the whole corpus, along with their absolute frequencies. While some of these words do occur in all scenarios (e.g. the most frequent five), some others are restricted to a few particular scenarios (*hair, bus,ticket*) or even only a single one (*iron*, *eggs*, *tire*). Appendix Table A.1 shows the ten most frequent content words for each scenario separately.

Scenario homogeneity and variance

A number of features can indicate how homogeneous the ESD set of a scenario is, and thus help to estimate how hard it will be to compute script representations from those sequences: highly similar ESDs are much easier to match to each other than sequences with higher variance. In a later step, we evaluate which of these aspects have the highest influence on the performance of our script mining algorithm (cf. Chapter 4).

Measures

Table 3.3 shows different statistics for ESD similarity within the same scenario:

- *ESDs* repeats the number of ESDs per scenario, for reference.
- *Events per ESD* is the average number of event descriptions per ESD. Longer sequences tend to be more diverse and complex. The subsequent column shows the standard deviation, indicating the range of sequence lengths within a scenario.
- *Words Per Event* is the average word count per event description (we consider punctuation marks and spaces as word separators). We assume that longer sentences

are harder to process. We also provide the number of *content words* per event.

- *Tokens per type* is the number of token realizations of one type within the scenario's ESDs, averaged over all types. A small value here indicates a high degree of wording variance (because there are more different types).

- *ESD token overlap*, *ESD type overlap* and *ESD synset overlap* are the proportions of words or concepts (counted as tokens, types or WordNet (Fellbaum, 1998) synsets) shared by each pair of sequences within the scenario. We compute the overlap for two ESDs s_1 and s_2 with the Dice coefficient (with *words* denoting either tokens, types or synsets):

$$overlap(s_1,s_2) = \frac{2 * |words(s_1) \cap words(s_2)|}{|words(s_1)| + |words(s_2)|}$$

For synsets, we consider all possible synsets for each word in the ESD. In all three cases, the results are averaged over all ESD pairs in the same scenario.

- *ESD Reordering* is a heuristic to estimate how strictly sequentially ordered the events in a scenario are. The sequence-based algorithm we use for paraphrasing (cf. Chapter 4) assumes that the input sequences are strictly ordered, i.e. that there is no case in which any two events have a different relative order in different sequences. In our data, this constraint is often violated, e.g. for MAKING SCRAMBLED EGGS, you can use salt and pepper at any point in time. We try to quantify such potential reorderings by counting pairs of head verbs from event descriptions that can occur in arbitrary sequential order.

Let C be a scenario, composed of ESDs $\{s_1,s_2,...,s_n\}$. Any ESD s_i consists of ordered event descriptions $\{e_{i,1},e_{i,2},...e_{i,m}\}$. For each event description $e_{i,k}$, let $v_{i,k}$ be the head verb of $e_{i,k}$. C thus has an associated set of ordered verb pairs $pred\_pairs(C) = \{(v_{i,j},v_{i,k}) | \exists s_i \;\; s.t. \;\; e_{i,j} \in s_i \wedge e_{i,k} \in s_i \wedge j < k\}$.
The potential for reordering scenario C is then computed as follows:

$$reordering(C) = \frac{|\{(v_1,v_2)|(v_1,v_2) \in pred\_pairs(C) \wedge (v_2,v_1) \in pred\_pairs(C)\}|}{|pred\_pairs(C)|}$$

We divide the number of head verb pairs that occur in both possible orders by the overall number of ordered verb pairs. Because we're actually interested in concepts rather than concrete realizations, we consider two verbs as equal if they share a WordNet synset.

The numbers indicate which proportion of head verb pairs can be switched, i.e. out of all verb pairs in a scenario that occur in the same ESD, how many of them occur in both possible orders. This is of course only an approximation of the actual event reordering. On the one hand, it misses cases in which equivalent event descriptions have different head verbs, and on the other hand, it mistakenly classifies repetitions of head verbs in a sequence as reordering. Empirical assessment of the results shows that our approximative measure differs from the actual event reordering by a constant factor, which is very similar for all scenarios. In consequence, this measure is sufficient for comparing scenarios with each other.

Word counts and lexical variance

The results in Table 3.3 show several interesting differences between the scenarios. FLYING, TAKING A SHOWER and CHILDHOOD have the longest sequences on average. Those scenarios either have many distinct and well-known steps (FLYING and SHOWER), or have no particularly clear definition, or have a long duration (both true for CHILDHOOD). In contrast, PAYING WITH A CREDIT CARD is a rather short process.

The ESDs of PUTTING A POSTER ON THE WALL contain the longest event descriptions, followed by MAKING SCRAMBLED EGGS and TAKING COPIES. This mirrors the complexity of the scenarios' events: In all three cases, diverse objects (e.g. tools or ingredients) have to be used together in the same event, while in scenarios like EATING IN A RESTAURANT mostly one person (the patron or the waiter) performs one action with at most one object (food, menu, table).

As far as vocabulary variance is concerned, the descriptions for "easier" commonsense scenarios use a more concise lexicon: EATING IN A RESTAURANT or EATING IN A FAST FOOD RESTAURANT as well as TAKING A SHOWER show the smallest variance in their vocabulary, measured by their token / type ratio as well as by their sequence-based overlap . On the other end of the scale, the descriptions of CREATING A HOMEPAGE show a very diverse lexicon. Most of the ESDs for this scenario revealed that their authors are obviously non-experts who would use different online tools to create such a page, rather than following a standardized coding process.

The corpus for CHILDHOOD behaves somewhat idiosyncratically: while there are only a few tokens per type, sequence-based overlap is above the average. One reason for this might be that there are comparably few repetitions of words within a sequence (thus fewer tokens per type), but different annotators use similar vocabulary. However, the sub-corpus for this scenario is too small to draw any certain conclusions.

Reordering

The results indicating event *reordering* vary between very few switchable events (TAKING A TRAIN, 0.03) to almost a fifth of event pairs that can occur in arbitrary order (CHECK-

Scenario	ESDs	events per ESD	events std. deviation	words per event	content words per event	tokens per type	ESD token overlap	ESD type overlap	ESD synset overlap	ESD reordering
restaurant	19	10.42	3.48	*3.08*	2.34	**3.80**	0.23	**0.36**	0.37	0.10
fast food	15	8.93	2.69	3.17	*2.20*	3.07	**0.31**	**0.35**	**0.41**	0.08
returning food	15	*5.93*	*1.01*	3.92	2.73	2.67	0.2	0.27	0.28	*0.05*
scrambled eggs	**24**	10.63	2.99	**4.11**	**2.82**	**3.72**	0.25	0.32	0.33	0.14
flying	19	**11.63**	3.74	3.28	2.35	3.13	0.27	0.33	0.38	0.10
airport check in	19	8.74	**3.83**	3.92	2.63	2.93	0.21	0.28	0.38	**0.18**
taking bus	21	8.71	3.34	3.55	2.46	3.13	**0.30**	0.33	**0.41**	0.08
taking train	14	8.86	3.5	3.45	2.47	2.81	**0.30**	**0.35**	**0.41**	*0.03*
going shopping	20	9.00	3.23	3.44	2.38	3.23	0.23	0.28	0.33	0.14
paying	19	6.26	*1.13*	3.56	2.62	3.18	0.2	0.25	0.30	0.12
credit card	*6*	*5.67*	1.67	3.91	2.71	*2.19*	0.27	0.31	0.35	*0.04*
haircut	**23**	10.35	**3.85**	3.47	2.58	3.09	0.19	0.25	0.34	0.12
wedding	18	10.78	**3.71**	3.61	**2.80**	2.62	0.19	0.27	0.29	0.14
taking shower	21	**11.29**	3.08	3.2	*2.23*	**4.71**	**0.30**	**0.40**	**0.50**	**0.15**
fixing tire	19	8.00	2.61	3.84	2.72	3.53	0.24	0.29	0.33	0.09
cleaning up	17	9.18	3.52	3.17	2.41	2.28	*0.16*	*0.22*	0.27	0.13
driving lesson	16	9.31	3.68	3.37	2.51	2.27	*0.18*	*0.22*	0.32	0.10
ironing	20	9.00	3.45	3.87	2.63	3.56	0.26	0.28	0.34	0.10
creating hp	13	7.38	2.06	3.74	2.75	*2.10*	*0.14*	*0.19*	*0.17*	0.06
taking copies	20	8.55	2.84	**4.00**	**2.89**	3.21	0.24	0.29	0.33	0.10
poster	**23**	7.22	3.1	**4.20**	2.73	2.71	0.22	0.24	*0.23*	0.06
childhood	*5*	**11.40**	2.27	3.33	2.44	*1.67*	0.28	0.3	0.44	**0.15**
average	17.55	8.97	2.94	3.6	2.56	2.98	0.23	0.29	0.34	0.10

Table 3.3: Sequence-based corpus statistics for all scenarios. Particularly high values are marked in **bold**, minima in *italics*.

IN AT AIRPORT, 0.18). As one would expect, longer sequences lead to more reordering possibilities. (correlation between those metrics, measured as Spearman's ρ, is 0.63). The other measures do not correlate as clearly with reordering – not even number or sequences the word overlap.

In general, reordering is often a local phenomenon of two close events; e.g. some people seem to *turn on the water* before they *enter the shower*, other people do it the other way around. However, we can expect both events to occur close together. The same holds for *stirring* and *putting eggs into the pan*: some people stir their scrambled eggs in a bowl, others save the dish and stir them directly in the pan.

Other phenomena can cause less predictable reorderings: Some scenarios are generally vaguely defined (CHILDHOOD). Other scenarios contain repeated events: When TAKING A SHOWER e.g., one can *wash* or *rinse* repeatedly. Similar constellations occur in CLEAN UP, where several events can be repeated. Some events can even occur at any arbitrary point in time (either just once or with a later repetition), like *rinsing* in TAKING A SHOWER, but also *add salt and pepper* in MAKING SCRAMBLED EGGS.

Similarities between scenarios

Apart from the variance within scenarios, which directly affect our processing pipeline, we are also interested in the comparison of different scenarios. One objective here is to ascertain whether related scenarios (e.g. TAKE A BUS and TAKE A TRAIN) have similar event sequence descriptions, considering lexical and structural aspects.

For a crude lexical analysis, we compute word type overlap for all scenario pairs. In a more fine-grained experiment, we apply the ESD-based similarity measures we used to compute scenario homogeneity, and thus get an impression of how well two scenarios can blend: we mix the ESDs of two scenarios at a time, and then show how much variance we find in these artificial scenario fusions. The results also indicate how easily the scenario of an ESD could be identified solely from analyzing its lexical content: ESDs mixed from two different scenarios should theoretically look less homogenous than the sequences of one single scenario.

Lexical overlap of scenarios

As a basic similarity measure, we compute pairwise word type overlap (using the Dice coefficient) over the word type sets of the scenarios. Table 3.4 shows the ten scenario pairs with the highest overlap, along with the 10 least similar ones for reference. Most results come out as intuitively expected. Finding an average type overlap of 27%, we identify several scenario pairs which are much more similar than that: half of the vocab-

Scenario 1	*Scenario 2*	*Type Overlap*
FAST FOOD RESTAURANT	RESTAURANT	0.50
AIRPORT CHECK-IN	FLYING WITH AN AIRPLANE	0.48
PAYING	PAYING WITH CREDIT CARD	0.47
TAKING A TRAIN	TAKING A BUS	0.47
TAKING A TRAIN	FLYING WITH AN AIRPLANE	0.46
GOING SHOPPING	PAYING	0.44
GOING SHOPPING	FAST FOOD RESTAURANT	0.44
TAKING A BUS	FLYING WITH AN AIRPLANE	0.42
PAYING	FAST FOOD RESTAURANT	0.41
TAKING A BUS	AIRPORT CHECK-IN	0.40
Average (over all scenario pairs)		0.27
CLEANING UP	PAYING WITH CREDIT CARD	0.18
MAKING SCRAMBLED EGGS	CREATING A HOMEPAGE	0.18
CREATING A HOMEPAGE	CLEANING UP	0.17
MAKING SCRAMBLED EGGS	RETURNING FOOD	0.17
WEDDING	CHILDHOOD	0.16
WEDDING	CLEANING UP	0.16
MAKING SCRAMBLED EGGS	CREATING A HOMEPAGE	0.16
RETURNING FOOD	CLEANING UP	0.16
CHILDHOOD	PUTTING A POSTER ON THE WALL	0.15
MAKING SCRAMBLED EGGS	PAYING WITH CREDIT CARD	0.14

Table 3.4: The 10 scenario pairs with the highest vocabulary overlap, and the 10 with the lowest one.

ulary used to describe EATING IN A RESTAURANT is also used for the FAST FOOD RESTAURANT descriptions. The other scenario hyponym pair – PAYING VS. PAYING WITH A CREDIT CARD – reaches a comparable overlap of 44%, as do the three sister scenarios TAKE A BUS, TAKE A TRAIN and FLY WITH AN AIRPLANE (42 – 46%). Further, scenarios in a part-of relation seem to be particularly similar, e.g. PAYING compared with GOING SHOPPING or EATING IN A FAST FOOD RESTAURANT (41 – 47%). (Table A.2 in the appendix shows the statistics for all scenario pairs.)

Blending Sequences of Different Scenarios

The lexical overlap of two scenarios unsurprisingly shows that related scenarios are described with the same words. We want to take a closer look at the sequences within the scenarios and directly compare their similarities. We mix ESDs of two different scenarios at a time and check whether the outcome looks comparably homogeneous to the textual descriptions of just one scenario. We compute two statistics also used in the scenario-based analysis: ESD-based type overlap, and predicate reordering within the blended scenarios. We can compare the results directly to the values we computed for ESDs from the same scenario (given in Table 3.3). For reordering, we also compute an expected value as reference, which is the average reordering of the two mixed scenarios. This expected value is in most cases identical with the computed value (on average, they differ by 0.01), so higher deviations from it can identify interesting outliers.

Table 3.5 shows the scenario pairs with the most similar ESDs. The table notes type overlap and mixed reordering, along with the expected reordering in brackets. (We provide the complete results in the Appendix, shown in Table A.3 for ESD-based type overlap and in Table A.4 for mixed reordering.) Recall that the intra-scenario ESD type overlap (cf. Table 3.3) was 0.29 on average, ranging between 0.19 and 0.40. Most scenario pairs differ much more than that: the average type overlap for ESDs of different scenarios is only 0.02. However, some sequences share a lot of vocabulary, even though they are not from the same scenario. While the previous experiment already showed high lexical overlap between these scenario pairs, the ESD-based overlap indicates that the blended scenarios are comparably homogenous as if all the ESDs were originally from the same scenario.

This looks most interesting in combination with the structural similarity measure: ESDs of two different scenarios are structurally similar if they "blend well", i.e. if mixing their ESDs does not introduce more reorderings.

The two lexically most similar scenarios are the two restaurant settings (FAST FOOD RESTAURANT and RESTAURANT). However, mixing their sequences also introduces reorderings: The proportion of switchable predicates is (at 4%) slightly higher than ex-

Scenario 1	*Scenario 2*	*ESD Overlap*	*reordering (+ expected)*
FAST FOOD RESTAURANT	RESTAURANT	0.31	0.14 (0.10)
AIRPORT CHECK-IN	FLYING IN AN AIRPLANE	0.27	0.16 (0.13)
PAYING	PAY WITH CREDIT CARD	0.27	0.12 (0.11)
TAKING A TRAIN	TAKING A BUS	0.25	0.08 (0.08)
TAKING A TRAIN	FLYING IN AN AIRPLANE	0.23	0.11 (0.09)
Average over scenario pairs		0.02	0.12 (0.11)
Average within one scenario		0.29	0.10

Table 3.5: Five exemplary scenarios with high lexical and / or structural similarity.

pected. This is intuitively explainable: one does roughly the same things in both restaurant types, but the events differ in their order. In a restaurant, you sit down first, and then you order, and you eat first and pay afterwards. In a fast food restaurant, you have to order and pay before you sit down and eat.

The two public transport scenarios TAKING A BUS and TAKING A TRAIN are less easy to distinguish: they have a fairly high lexical overlap, and blending them introduces no additional word reordering. This is also intuitively clear: one does similar things in the same order in both scenarios, just the type of vehicle is different.

Further research on larger corpora is necessary to determine whether or how sequences from different scenarios can be distinguished. This question is particularly relevant for unsupervised approaches that collect sequences without associating them to scenarios.

3.5 A Domain-Specific Corpus of Cooking ESDs

In a second data collection, we built a corpus with the objective of reaching high and dense coverage for one specific domain. For this experiment, we again used the Mechanical Turk setup we have described in Section 3.1.

For this domain-specific corpus, we chose to explore the cooking domain in more detail. Scenarios around kitchen tasks are largely parts of common sense knowledge, and there are many detailed scenario collections available which we can take as stimuli. Concretely, we adopted a list of scenarios by mining the tutorials on the webpage "Jamie's Home Cooking Skills".[4] The 53 tasks in this list are instructions for processing ingredients or using certain kitchen tools, like CUTTING AN ONION or USING A SPEED PEELER.

[4]`http://www.jamieshomecookingskills.com`

CHOPPING A CUCUMBER

1. grab a sharp knife
2. take out the cucumber and place it on a chopping board
3. take the knife in one hand
4. stead the cucumber with the other hand
5. make small slices at one side of the cucumber
6. proceed to cut until at the end

1. wash the cucumber in water
2. cut off the ends of the cucumber
3. hold the cucumber length wise
4. using a sharp knife, peel the cucumber
5. peel the cucumber in strips, lengthwise
6. lay the cucumber on it's side
7. carefully cut the cucumber into 1/4 inch slices

1. first, you will peel the cucumber. get the peeler from the drawer.
2. hold the cucumber firmly on one end.
3. put the blade of the peeler on the middle of the cucumber.
4. press down slightly, and push the peeler away from you along the skin of the cucumber. always push away from your body, not toward your body.
5. the first strip you peel is the hardest. after you get the first strip off, the rest of them are easier.
6. go all around the cucumber taking the skin off one end.
7. now turn the cucumber around and peel the other end.
8. put the peeled cucumber on a cutting board.
9. get a sharp knife and, being very careful, slice the cucumber into circles about as thick as a pencil.
10. cut each circle into four equal pieces.
11. now the cucumbers are ready to put in the salad.

USING A BOX GRATER

1. place boxgratedr over plate or bowl
2. open item to be grated like cheese
3. choose the side of the grater you want to use.
4. big holes means big shreds
5. push item to be grated against side of grater at top
6. with firm pressure rub food down side of grater
7. move item to top again and repeat
8. carefully lift grated from plate

1. decide what to grate.
2. put grater on a smooth, level surface.
3. carefully move food across grater blades.
4. when done, remove food from inside of the grater.
5. wash the grater, again being careful. the blades are sharp!

1. first, make sure you have a flat clean work surface like a counter or a table. don't try to use a grater on a squishy surface like a chair cushion.
2. when you use your box grater, shreds are going to come out, so make sure you have a cutting board or piece of wax paper something to grate over..
3. turn the grater so the side you want to grate on is facing your dominant hand.
4. (if you are right handed, your right hand is your dominant hand, left handed it's your left)
5. hold the grater down firmly with your non dominant hand and hold the object you want to grate in your dominant hand.
6. rub the object up and down against the grater to shred it, change the angle occasionally for easier grating.
7. be careful of your fingers, if you grate them into your food, you will ruin it!!!

Figure 3.5: 6 cooking ESDs for CHOPPING A CUCUMBER and USING A BOX GRATER.

In terms of corpus size, we scaled up the data collection. We gathered 50 ESDs for each of the 53 scenarios (compared to 25 ESDs for 22 scenarios in the first experiment). Figure 3.5 shows 6 of the resulting "cooking ESDs". In principle the texts are similar to the ESDs from our first experiment, but with longer event descriptions, more variance in the description and sequence length. Many people also included non-sequential events, in particular warnings and special advice for using kitchen equipment. All those peculiarities are most likely due to the cooking domain and the fact that questions like HOW DO YOU CUT AN ONION triggered a recipe-like writing style in many participants.

We did some minor modifications to the Mechanical Turk interface, adapting it to the domain and to the experience from our first experiment: story-wise, we told the subjects to put themselves in the position of an immobile elderly lady whose grandson helps her in the kitchen, but does not know the slightest bit about cooking. The workers should enter instructions they'd give the grandson in order to complete the cooking tasks. In order to achieve better language quality, we restricted the workers to participants from the U.S.. Further, we neither encouraged bullet-point style nor set any restrictions on the number of characters in a text field this time. Next to the priming by the kitchen domain, this also contributed to the more verbose writing style.

The restriction to U.S. workers indeed helped to reduce the number of completely unusable ESDs, but the texts still contain a lot of spelling mistakes (evident from the example sequences). Because this corpus is much larger then our first ESD collection, manual preprocessing was not an option, so we developed the automated spelling algorithm on this data (cf. Section 3.3). In order to apply our script mining algorithms to the kitchen corpus, we also developed new algorithms for coreference resolution and clause splitting for this corpus (cf. Chapter 6); neither of these steps were necessary for the shorter bullet-point style event descriptions from the first data collection.

In a multimodal application, we use the cooking ESDs as a basis for some first experiments on automated event-recognition in videos (cf. Section 9.4).

3.6 The OMICS corpus

The *Open Mind Indoor Common Sense* (OMICS) corpus,[5] created at MIT media lab, is very similar to our data collection. Within the Open Mind Project (Singh *et al.*, 2002), there is an ongoing effort to develop the OMICS corpus via crowdsourcing. It contains about 170 indoor scenarios, each annotated with several *stories* (which look like our event sequence descriptions). The corpus has never been officially released, and we learned about it when it was referred to by Smith & Arnold (2009). Smith & Arnold actually

[5] `http://openmind.hri-us.com/`

show a case study in which they compute script representations out of some OMICS stories. While we use a different approach to compute the scripts themselves, we use the OMICS data as additional input for our experiments (cf. Chapter 4 and 5).

Chapter 4

Mining Script Representations

Out of the event sequence descriptions we collected for several scenarios (as described in the previous chapter), we want to compute concise script representations that display events with their lexical variants as well as temporal constraints learned from the input.

Computing *event paraphrases* is the necessary first step for mining script representations from ESDs: we want to represent an event as a set of equivalent descriptions from the input texts. We thus need to identify all equivalence sets of event descriptions within ESDs for the same scenario.

While there are a wealth of methods for paraphrasing from standard texts, we face several challenges particular to mining event paraphrases from ESDs: First of all, our crowdsourced data contains idiosyncratic language with incomplete sentences where subjects are missing and determiners used only sporadically. Standard NLP approaches for computing semantic similarity thus need to be very robust to process these descriptions with any result at all, and we must expect them to deliver less reliable results than for standard texts.

Further we face what we earlier called the *paraphrase problem* (cf. Section 1.3): Some event paraphrases are simply not very similar semantically, and many of them are paraphrases only with respect to the tiny domain of their scenario. Take the ESDs from the FAST FOOD scenario in Figure 3.2 (repeated in Figure 4.1) as an example: The paraphrased realizations for the event in which the actor decides what he or she wants to order are *decide on food and drink* , *decide what you want* and *make selection*. In general, *decide on food and drink* does not have the same meaning as *decide what you want*, but in this very narrow FAST FOOD domain, the two expressions are fully interchangeable. The wording of the three event descriptions has very little lexical overlap. In particular, the support verb construction (*make decision*) in the last ESD does not overlap at all with the other two phrases (*decide on food and drink, decide what you want)*; they do not even contain

HOW DO YOU EAT AT A FAST FOOD RESTAURANT?

1. walk into restaurant
2. find the end of the line
3. stand in line
4. look at menu board
5. decide on food and drink
6. tell cashier your order
7. listen to cashier repeat order
8. listen for total price
9. swipe credit card in scanner
10. put up credit card
11. take receipt
12. look at order number
13. take your cup
14. stand off to the side
15. wait for number to be called
16. get your drink

---

1. look at menu
2. decide what you want
3. order at counter
4. pay at counter
5. receive food at counter
6. take food to table
7. eat food

---

1. walk to the counter
2. place an order
3. pay the bill
4. wait for the ordered food
5. get the food
6. move to a table
7. eat food
8. exit the place

---

1. enter restaurant
2. go to counter
3. make selection
4. place order
5. pay for food
6. pick up order
7. pick up condiments
8. go to table
9. consume food
10. clear tray

Figure 4.1: Four ESDs describing the scenario of EATING AT A FAST FOOD RESTAURANT.

synonyms, but still they are perfect paraphrases in this context.

Event paraphrasing is thus a big challenge. On the other hand, we can take advantage of the inherent structural similarities between ESDs of the same scenario: The sequential context of an event description is a key feature we leverage for finding its paraphrases. Consider again the three descriptions verbalizing the meal choice: They all occur immediately before the *ordering* event occurs, and they are all in the end of the first third of the sequence. Also, two of them are directly preceded by the *look at menu* event, and two of them non-directly by *entering the restaurant*.

We will show that the sequential structure constitutes an invaluable information source for this paraphrasing task, and that our system can use it to match paraphrases with low semantic similarity but highly similar discourse contexts. Inversely, the system is capable of distinguishing very similar events that occur on opposite ends of the discourse, like *enter restaurant* and *leave restaurant*.

After we have computed event paraphrases, we can mine a graph-based script representation that uses sets of equivalent textual realizations to represent events, and comprises partial temporal constraints between these events. The concise graph representation also allows for some post-processing that filters noise introduced by the paraphrasing algorithm.

In this chapter, we first give an informal explanation of how we interpret paraphrasing as sequence alignment (Section 4.1). Then we show in detail how we use Multiple Sequence Alignment (MSA) to compute event paraphrases (Section 4.2), using a robust similarity measure (described in Section 4.3). The resulting paraphrase information enables the mining of temporal script graphs (Section 4.4). Finally, we show the evaluation results, along with some examples (Section 4.5). We conclude the chapter with a short summary of related work (Section 4.6)

This chapter presents work published by Regneri *et al.* (2010).

4.1 Paraphrasing as Sequence Alignment

Sequence Alignment is the task of finding the most direct way to re-write one sequence into another by adding and deleting elements. The most famous application is the computation of Levenshtein distance: The Levenshtein distance between two words is just the number of insertions and deletions one has to make to rewrite one word into the other one.

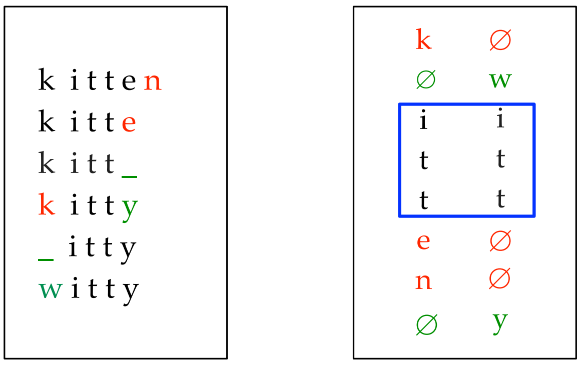


Figure 4.2: Two ways to display the rewriting of *kitten* into *witty*.

Figure 4.2 shows two ways to display an alignment of the words *kitten* and *witty*; words are understood as sequences of letters. The left picture shows a possible process of rewriting *kitten* into *witty* in detail: We delete *k,n* and *e* (marked in red) and add *w* and *y* (green). The single steps can be taken in arbitrary order, and the Levensthein distance would be the overall number of edits (5).

The right part of the figure shows a more concise representation of the re-writing process, picturing it as actual sequence alignment. (We will use this kind of representations in all following sequence alignment diagrams.) Insertions and deletions are signaled by so-called *gaps* (marked with ∅). A gap in the source word (*kitten)* stands for an insertion, and a gap in the target word (*witty*) represents a deletion. The best alignment is always the alignment with the fewest gaps, thus symbolizing the re-writing process with the fewest edit operations. The exact placing of gaps in an optimal alignment (as depicted in the right graph) can vary to some degree: the first two lines of the alignment could simply be switched, which would result in slightly different gap positions, but still in an optimal alignment. Note that the alignment is symmetric; re-writing *witty* into *kitten* would result in an equivalent alignment, with gaps in the same places. The Levenshtein distance here is just the number of gaps.

The part of the alignment that is interesting for paraphrasing is the *matching* part of the two sequences, marked with the blue box in the example. This is the bit of information shared by both sequences, which remains unchanged in the rewriting process.

go to restaurant	∅
∅	enter restaurant
go to counter	∅
read menu	∅
make choice	decide what to eat
wait in line	∅
order food	make order
pay	pay for food
∅	find free table
sit down	∅
eat food	enjoy meal

Figure 4.3: Sequence alignment of two ESDs, paraphrases are indicated with colors.

In order to find event paraphrases, we align ESDs as sequences of event descriptions, and find the parts that are retained when we try to re-write one sequence into a second one. Figure 4.3 shows an example, with colors marking the matched event descriptions. The main difference between aligning ESDs and character-wise word alignment is that we want to match sequence elements that are not completely identical. For letter-based re-writings, it is clear that only identical letters can be matched. For event descriptions, we need to match phrases that are somewhat similar (to varying degrees), but almost never identical. We thus combine sequence alignment with a semantic similarity measure for finding good matches.

When we compute paraphrases from event sequence descriptions, we need to align more than two sequences for a scenario. We thus employ *Multiple Sequence Alignment*, which generalizes binary sequence alignment to arbitrary many input sequences. The formal details of standard sequence alignment and MSA are provided in the following section.

4.2 Multiple Sequence Alignment

Sequence Alignment and Multiple Sequence Alignment (MSA) are well-researched in bioinformatics, where they are typically used to find corresponding elements in proteins or DNA (Durbin *et al.*, 1998).

A binary sequence alignment algorithm takes as its input some *sequences* $s_1, \ldots, s_n \in \Sigma^*$ over some alphabet Σ, along with a *score function* $s_m : \Sigma \times \Sigma \to \mathbb{R}$ for substitutions (or, in our case, matchings) and *gap costs* $c_{gap} \in \mathbb{R}$ for insertions and deletions. In bioinformatics, the elements of Σ are nucleotides, and the aligned sequences are DNA sequences; in our case, Σ contains the individual event descriptions, and the sequences are the ESDs.

A *Multiple Sequence Alignment* A is then a matrix as in Table 4.1: The i-th column of A is the sequence s_i, possibly with some gaps ("$\emptyset$") interspersed between the symbols of s_i, such that each row contains at least one non-gap. If a row contains two non-gaps, we take these symbols to be *aligned*; aligning a non-gap with a gap can be thought of as an insertion or deletion.

Each sequence alignment A can be assigned a *score* $s(A)$ in the following way:

$$s(A) = c_{gap} \cdot \Sigma_{\emptyset} + \sum_{i=1}^{n} \sum_{\substack{j=1,\\ a_{ji} \neq \emptyset}}^{m} \sum_{\substack{k=j+1,\\ a_{ki} \neq \emptyset}}^{m} s_m(a_{ji}, a_{ki})$$

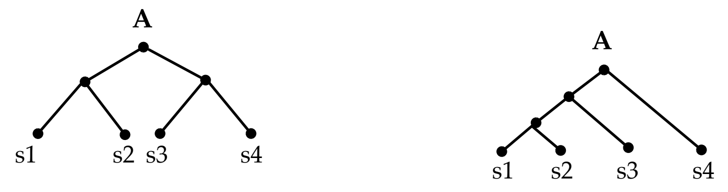
where $\Sigma_{\emptyset}$ is the number of gaps in A, n is the number of rows and m the number of sequences. In other words, we sum up the alignment score for any two symbols from Σ that are aligned with each other, and add the gap cost (= negative scores) for each gap. The Needleman-Wunsch-Algorithm computes the optimal pairwise alignments (i.e. $n = 2$) in polynomial time (Needleman & Wunsch, 1970). In our case, the optimal alignment is the alignment A with the highest score $s(A)$.

Alignment Trees

We use a generalization of the Needleman-Wunsch algorithm for n sequences to compute the MSA. Finding an optimal solution for this problem is NP-complete. In general, the best MSA is approximated by aligning two sequences first, considering the result as a single sequence whose elements are pairs, and repeating this process until all sequences are incorporated in the MSA (Higgins & Sharp, 1988). The order in which the sequences are added can be seen as a binary tree with the elementary sequences as its leaves and the final alignment as root node, so-called *alignment trees*. We experimented with two paradigms for computing such a tree, one based on content similarity of the sequences, and one purely based on the sequences' length:

row	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	$\mathbf{s}_4$
1	$\oslash$	walk into restaurant	$\oslash$	enter restaurant
2	$\oslash$	$\oslash$	walk to the counter	go to counter
3	$\oslash$	find the end of the line	$\oslash$	$\oslash$
4	$\oslash$	stand in line	$\oslash$	$\oslash$
5	look at menu	look at menu board	$\oslash$	$\oslash$
6	decide what you want	decide on food and drink	$\oslash$	make selection
7	order at counter	tell cashier your order	place an order	place order
8	$\oslash$	listen to cashier repeat order	$\oslash$	$\oslash$
9	pay at counter	$\oslash$	pay the bill	pay for food
10	$\oslash$	listen for total price	$\oslash$	$\oslash$
11	$\oslash$	swipe credit card in scanner	$\oslash$	$\oslash$
12	$\oslash$	put up credit card	$\oslash$	$\oslash$
13	$\oslash$	take receipt	$\oslash$	$\oslash$
14	$\oslash$	look at order number	$\oslash$	$\oslash$
15	$\oslash$	take your cup	$\oslash$	$\oslash$
16	$\oslash$	stand off to the side	$\oslash$	$\oslash$
17	$\oslash$	wait for number to be called	wait for the ordered food	$\oslash$
18	receive food at counter	get your drink	get the food	pick up order
19	$\oslash$	$\oslash$	$\oslash$	pick up condiments
20	take food to table	$\oslash$	move to a table	go to table
21	eat food	$\oslash$	eat food	consume food
22	$\oslash$	$\oslash$	$\oslash$	clear tray
22	$\oslash$	$\oslash$	exit the place	$\oslash$

Table 4.1: A MSA of four event sequence descriptions.



a) Symmetric alignment tree, result of Bag of Words aligner

b) Left-recursive alignment tree, result of the Length aligner

Figure 4.4: Two exemplary alignment trees.

Bag of Words aligner: One model groups the two most similar sequences together first, using word overlap to measure similarity. The resulting pair of aligned sequences is then considered as a single sequence. The final alignment tree often looks similar to the one in Figure 4.4 (a), in which pairs of elementary sequences are aligned with each other before they are grouped together with other aligned pairs. The actual tree structure however depends on the wording of the different ESDs; theoretically, any binary tree structure is possible.

Length aligner: The second strategy for creating alignment trees orders sequences according to their length and builds up the tree by starting with the longest and the second-longest sequences, successively adding the next longest one (as shown in Figure 4.4 (b)). The intuition behind this strategy is that the longest sequence tends to be the most detailed one, and elements of shorter sequences will mostly have a matching element in the longer one. In consequence, the number of gaps is kept small because mostly deletions have to be performed, and hardly any insertions.

While both techniques clearly perform better than random alignment, they virtually never differ in their performance. For the remainder of this work, we use the Bag of Words aligner as default system.

4.3 Semantic Similarity

In order to apply MSA to the problem of aligning ESDs, we choose Σ to be the set of all individual event descriptions in a given scenario. The scoring function s_m thus needs to compare event descriptions, which is a much more complex task than comparing nucleotides (like in bioinformatics) or letters (for computing Levenshtein distance). In contrast to DNA matching, we are dealing with an infinite set of possible descriptions which we can't know beforehand, so we cannot simply create a table with substitution costs for all DNA building bricks. What we of course intuitively want is to prefer the alignment of two phrases if they are semantically similar, i.e. we should give a lower score to aligning *exit* with *eat* than we would to aligning *exit* with *leave*. Thus we take a measure of semantic (dis)similarity as the score function s_m.

Problems with standard similarity measures

We experimented with several standard similarity measures of varying complexity. Simple word-edit distance and word overlap were too strict and matched only a small part of the event paraphrases we wanted to find. In our final evaluation (Section 4.5), we use word-edit distance to compute one of the baselines.

A standard vector-space model (Thater *et al.*, 2011) also resulted in recall problems, but for different reasons. The training data for this model (and such models in general) mostly contains standard newspaper text, and applying these models displays another facet of the implicitness problem: we simply do not find many newspaper texts that describe events at a fast food restaurant or how to make coffee. In particular, the corpus-based vectors cannot capture scenario-specific similarities, like the exchangeability of *food* and *order* within the RESTAURANT scenarios. Numerically, the vector-space models scored about 15-20% worse than the best performing similarity measures we tried (measured in f-score, cf. Section 4.5), solely due to recall problems. (Section 9.2 describes our ongoing efforts to improve this performance by creating scenario-specific training corpora for vector-space models.)

In general, WordNet-based measures performed best for our purposes: starting out from simple sense overlap, we also tested standard WordNet similarity scores (Lin, 1998) by either pairing each word in a phrase with its most similar counterpart, or summing up the scores for all word pairs. All of these measures share one main problem: they count all similar word pairs with an equal weight. As a consequence, *get food* and *eat food* are scored as being just as similar as *wait for food* and *wait for order*. Intuitively, the head verb of an event description contributes the most to the phrase's semantics, but such heuristics can't be captured by standard WordNet measures.

A robust WordNet-based similarity measure

Using WordNet as a basis, we incorporate some shallow syntactic information to assign different weights to the different constituents of the event description. With our adapted dependency parser (cf. Section 3.3), we identify the head verb as *pred*, the subject (if any) as *subj* and all noun phrases (NPs). We record all non-complex NPs as an unordered list

turn	*turn on*	*add*	*remove*	*go*	*go to*	*stand*
walk	*get*	*take*	*be*	*have*	*put*	*place*	*make*

Table 4.2: Verbs with a lower semantic contribution, and a lower α in *sim*.

of objects ($[obj_1, obj_2, ..., obj_n]$), including NPs which occur inside prepositional phrases. This helps us to do a reliable shallow matching of support verbs and complex noun phrases, e.g. when comparing *take food to table* with *take tray with food*. (The version of the similarity measure we noted in (Regneri *et al.*, 2010) used a more shallow, WordNet-based heuristic to approximate the syntactic information, because the adapted parsing model was not yet available at that point in time. It turned out that exchanging this heuristic for a parser eventually made no difference for the performance of the system.) In addition to using syntactic information, we also optimize the measure so as to fit well into the MSA: in general, the sequence aligner delivers better results if very similar elements have a clearly higher score than dissimilar elements. This is, for example, not given in standard measures ranging between 0 and 1. Instead of such standard numbers, we chose similarity values from a wider scale, depending on the WordNet relationship of the syntactic components. This is represented by the following formula for computing the similarity measure *sim* on the basis of the flattened parse:

$$sim = \alpha \cdot pred + \beta \cdot subj + \gamma \cdot \sum_{i=1}^{n} obj_i$$

where *pred*, *subj*, and *obj* are the similarity values for predicates, subjects and objects respectively, and α, β, γ are weights. If a constituent is not present in one of the phrases to compare, we set its weight to zero and re-distribute it over the other weights (e.g. if the description contains just one verb, we add α and γ to β. We fix the individual similarity scores *pred*, *subj*, and *obj* depending on the WordNet relation between the most similar WordNet senses of the respective lemmas (100 for synonyms, 0 for lemmas without any relation, and intermediate numbers for different kind of WordNet links). Setting the similarity scores in this way allows us to directly influence the numerical differences between different degrees of similarity.

We optimized the values for *pred*, *subj*, and *obj* as well as the weights α, β and γ using a held-out development set of five scenarios. Our experiments confirmed that in most cases, the verb contributes the largest part to the similarity (accordingly, α needs to be higher than the other factors). We achieved improved accuracy by distinguishing a class of verbs that contribute little to the meaning of the phrase (i.e., support verbs, verbs of movement, and the verb "get"), and assigning them a separate, lower α. Table 4.2 shows the full list of lemmas for which we lowered α.

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

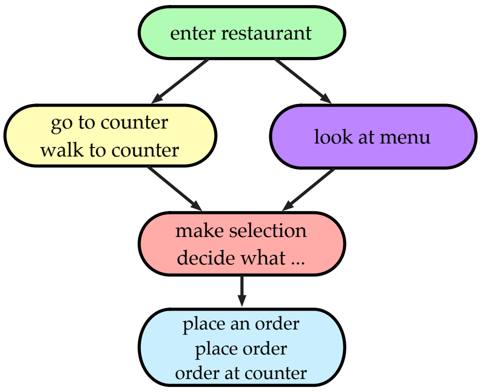


Figure 4.5: A multiple sequence alignment and its corresponding temporal script graph.

4.4 Building Temporal Script Graphs

After combining the MSA algorithm with the semantic similarity measure, we can identify equivalent elements of event sequence descriptions as event paraphrases. Those paraphrases enable us to abstract away from the input sequences and built a global script representation for a scenario, in our case a *temporal script graph* (TSG). The TSG includes sets of event paraphrases as nodes representing events, and the sequential ordering from the input sequences as temporal constraints, mirrored in the graph's edges.

Temporal script graphs

A temporal script graph is a directed, acyclic graph, and just as in the other graph-based representations, its nodes are event description sets, and the edges represent temporal constraints. If two events occur in the TSG, and there is a directed path between them, the TSG states that the source event typically happens before the target event. It does, however, not state which events are obligatory, or how many events should be in a sequence. The edges also do not imply *immediate* temporal precedence, but rather a partial ordering over the represented events. For any two events in the TSG, the existence of a path between them means the source event of the path usually happens before the target event happens, with possible other intervening events.

We build the temporal script graph using the matrix of aligned sequences. Figure 4.5 shows a temporal script graph (right) and the MSA it is built from (left). At first we construct an initial graph which has one node for each row of the MSA (indicated with colors in the picture). We interpret each node of the graph as representing a single event in the script, and the paraphrases that are collected in the node as different possible realizations of this event. We then add an edge (u,v) to the graph iff (1) $u \neq v$, (2) there was at least one ESD in the original data in which some phrase in u directly preceded some phrase in v, and (3) if a single ESD contains a phrase from u and from v, the phrase from u directly precedes the one from v. In terms of the MSA, this means that if a phrase from u comes from the same column as a phrase from v, they are either in adjacent rows, or there are only gaps between them.

This initial graph contains roughly the same information as the MSA it is computed from: each node pictures the paraphrasing information from the MSA table, and each edge means that some event description in the source node directly preceded an event description in the target node in the original input ESDs.

Postprocessing of the graph

Because the initial graph is a direct representation of the table, it also retains all the noise introduced during the paraphrasing process. We automatically post-process the graph in a second step to simplify it and eliminate some of the noise from the sequence alignment. The goal here is to get rid of nodes that should either not be there in the first place (e.g. if only one person referred to the specific event), or nodes that duplicate events, because all their event descriptions should have been merged into another node. The last case mostly occurs if an event has two different, non-synonymous head verbs in its lexical variants, like *order food* and *place order*. Initially, the node containing *order food* might just contain descriptions with *order* as head verb, and the cluster with *place order* could end up in a different, but closely connected node.

As a first step, we prune spurious nodes which contain only one event description, thus removing event instances with insufficient support from our data.

We then process the graph by merging nodes whose event descriptions should have been grouped in the first place, but were missed during the sequence alignment. When doing this, we want to maintain high precision for both the paraphrase information and the temporal constraints in the graph. We thus need to make sure that we a) merge only nodes that contain equivalent event descriptions and b) don't introduce invalid temporal constraints, in particular cycles. We merge two nodes if they satisfy the following semantic and structural constraints:

The **semantic constraints** check whether the event descriptions of the merged node would be sufficiently consistent according to the similarity measure from Section 4.3. To check whether we can merge two nodes u and v, we use an unsupervised clustering algorithm (Flake *et al.*, 2004) to first cluster the event descriptions in u and v separately. Then we combine the event descriptions from u and v and cluster the resulting set. If the union has more clusters than either u or v, we assume the nodes to be too dissimilar for merging.

The **structural constraints** depend on the graph structure. We only merge two nodes *u* and *v* if their event descriptions come from different sequences and one of the following conditions holds:

- *u* and *v* have the same parent;
- *u* has only one parent, *v* is its only child;
- *v* has only one child and is the only child of *u*;
- all children of *u* (except for *v*) are also children of *v*.

These structural constraints prevent the merging algorithm from introducing new temporal relations that are not supported by the input ESDs.

An exemple temporal script graph

The output of this post-processing step is then the final temporal script graph, picturing events and temporal constraints of a scenario. An excerpt of the graph we obtain for our running example (FAST FOOD RESTAURANT) is shown in Figure 4.6. One node created by the merging step was the top left one, which combines one original node containing *walk into restaurant* and another with *go to restaurant*. The graph mostly groups phrases together into event nodes quite well, although there are some exceptions, such as the *collect utensils* node. The temporal information in the graph is pretty accurate, apart from paths through nodes that show bad paraphrase accuracy (e.g. you can't pay twice, as the connection between the *pay* node and the node around *collect utensils* suggests).

Perhaps most importantly, our MSA-based algorithm generally manages to abstract away from surface-level linguistic similarity: Using the sequential structure, the system correctly combines *make selection* and *decide on food and drink* into the same node, although they are not similarly worded at all. In the reverse perspective, it can keep similar phrases like *wait in line* and *wait for my order* apart because they are always separated by other events (like *order* and *pay*).

4.5 Evaluation

For our final evaluation, we optimized the gap scores for the MSA (cf. Section 4.2) and the parameters for the semantic similarity measure (cf. Section 4.3) on a held-out set of five scenarios (SEND FOOD BACK IN A RESTAURANT and TAKE A SHOWER from our own corpus, and ANSWER THE DOORBELL, DO LAUNDRY and USE A MICROWAVE from OMICS).

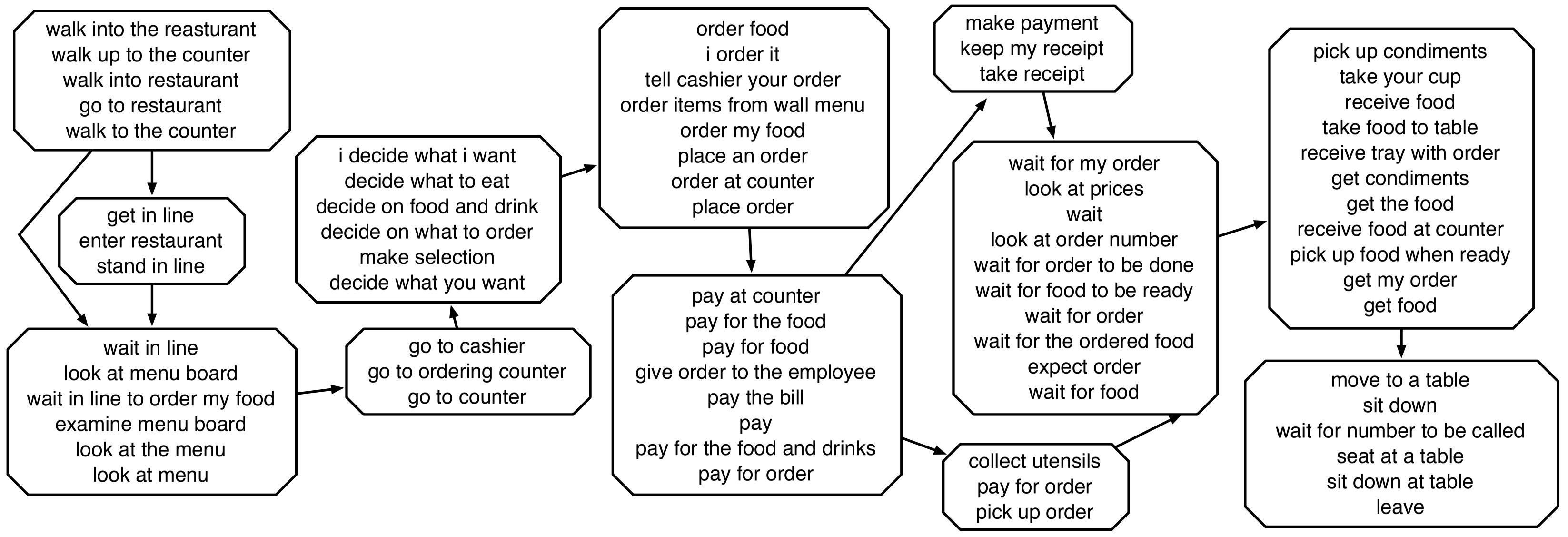


Figure 4.6: An extract from the graph computed for EATING IN A FAST FOOD RESTAURANT.

Evaluating the performance of our script mining approach is not straight-forward: an end-to-end evaluation is not possible at all, because there are no systems that actually use common sense script knowledge (cf. Section 1.3). There is also no gold standard reference for script structures that we could compare our temporal script graphs to, because all script databases contain just event sequence descriptions rather than abstract representations. We thus have no direct possibility to show the effect or correctness of our graphs as complete structures.

Instead we set up intrinsic evaluations for the two core aspects of our system: paraphrase computation and derivation of sequential order. We create different evaluation scenarios for both, the *paraphrase task* to demonstrate how well our system recognizes descriptions of the same event, and the *happens-before task* to judge the resulting temporal constraints over the event descriptions. In a nutshell, we sample pairs of event descriptions from the same scenario, and then create a gold standard with human judgements about both paraphrasing and temporal relations of the event descriptions. We then evaluate the information in the temporal script graphs against this gold standard annotation.

Gold Standard

For the gold standard, we select ten scenarios which were never used for development purposes. Five of the scenarios are taken from the corpus described in Chapter 3, the other five from the OMICS corpus (cf. Section 3.6). The *stories* ($\approx$ scenarios) in OMICS strongly resemble our ESD collections, but are restricted to “indoor activities" and typically contain more sequences per scenario than our corpus (around 40 per scenario).

For each scenario, we create two different evaluation sets, a *paraphrase set* and a *happens-before set*. For the *paraphrase set*, we sample 30 random pairs of event descriptions which the system classified as paraphrases and 30 other completely randomly selected pairs. The *happens-before set* consists of 30 pairs classified as *happens-before*, 30 randomly selected pairs and additionally all 60 pairs in reverse order. We added the pairs in reversed order to check whether the raters really prefer one direction or whether they accept both and were biased by the order of presentation.

We presented each pair to 5 non-experts, all US residents, via Mechanical Turk. For the *paraphrase set*, an exemplary question we asked the rater looks as follows (instantiating the scenario and the two descriptions, of course):

> Imagine two people, both telling a story about SCENARIO. Could the first one say $event_2$ to describe the same part of the story that the second one describes with $event_1$?

For the *happens-before task*, the question template was the following:

> Imagine somebody telling a story about SCENARIO in which the events *event*$_1$ and *event*$_2$ occur. Would *event*$_1$ normally happen before *event*$_2$?

In both cases, the annotators had to answer either "yes" or "no", with the possibility to add further comments in a text field. We constructed a gold standard by a majority decision of the raters. An expert rater adjudicated all pairs with a 3:2 vote ratio.

Baselines and Upper Bound

Because there are no comparable script mining systems, we cannot compare our approach to previous work. To show a meaningful comparison, we implemented two informed baselines that incorporate either the structural MSA component or our semantic similarity measure, either combined with a less sophisticated similarity measure or with a structure-unaware clustering component. Additionally, we show the inter-annotator agreement from our gold standard annotation experiment as an upper bound for both paraphrase computation and derivation of sequential order.

Clustering Baseline: We use the same unsupervised clustering algorithm that we apply for node merging (Flake *et al.*, 2004), and feed it all event descriptions of a scenario. We first create a similarity graph with one node per event description. Each pair of nodes is connected with a weighted edge; the weight reflects the semantic similarity of the nodes' event descriptions as described in Section 4.3. To include all trivial input information from the source sequences, we do not allow for edges between nodes containing two descriptions occurring together in one ESD. The underlying assumption here is that two different event descriptions of the same ESD always represent distinct events.

The clustering algorithm uses a parameter which influences the cluster granularity, without determining the exact number of clusters beforehand. We optimize this parameter automatically for each scenario by picking the value that yields the optimal result with respect to density and distance of the clusters: the elements of each cluster have to be as similar as possible to each other, and as dissimilar as possible to the elements of all other clusters.

The clustering baseline considers two phrases as paraphrases if they are in the same cluster. It claims a happens-before relation between phrases e and f if some phrase in e's cluster precedes some phrase in f's cluster in the original ESDs. This baseline demonstrates the contribution of MSA, compared to a more standard approach that does not consider the ESD's sequential structure.

Levenshtein Baseline: This system follows the same steps as our system, but using Levenshtein distance as the measure of semantic similarity for MSA and for node merging. This lets us measure the contribution of the more fine-grained similarity function. We compute Levenshtein distance as the character-wise edit distance on the phrases, divided by the phrases' character length so as to get comparable values for shorter and longer phrases. The gap costs for MSA with Levenshtein were optimized on our development set so as to produce the best possible alignment.

Upper bound: We also computed a human-performance upper bound. Because no single annotator rated all pairs of ESDs, we constructed a "virtual annotator" as a point of comparison, by randomly selecting one of the human annotations for each pair.

Results

We calculated precision, recall, and f-score for our system, the baselines, and the upper bound as follows, with all_{system} being the number of pairs labelled as *paraphrase* or *happens-before*, all_{gold} as the respective number of pairs in the gold standard and *correct* as the number of pairs labeled correctly by the system.

$$precision = \frac{correct}{all_{system}} \qquad recall = \frac{correct}{all_{gold}} \qquad f\text{-}score = \frac{2 * precision * recall}{precision + recall}$$

Table 4.3 and 4.4 show the results of our system and the reference values; Table 4.3 describes the *happens-before task* and Table 4.4 the *paraphrase task*. The upper half of the tables describes the results from our own corpus, the remainder refers to OMICS data. The columns labelled *sys* contain the results of our system, $base_{cl}$ describes the clustering baseline and $base_{lev}$ the Levenshtein baseline. The f-score for the upper bound is in the column *upper*. For the f-score values, we calculated the significance for the difference between our system and the baselines as well as the upper bound, using a resampling test (Edgington, 1986). The values marked with • differ from our system significantly at a level of $p \leq 0.01$, ◦ marks a level of $p \leq 0.1$. The remaining values are not significant with $p \leq 0.1$. (For the average values, no significance is calculated because this does not make sense for scenario-wise evaluation.)

Happens-before task: In most cases, and on average, our system is superior to both baselines. Where a baseline system performs better than ours, the differences are not significant. In four cases, our system does not differ significantly from the upper bound.

Scenario	Precision			Recall			F-Score			
	sys	base$_{cl}$	base$_{lev}$	*sys*	base$_{cl}$	base$_{lev}$	*sys*	base$_{cl}$	base$_{lev}$	upper
PAY WITH CREDIT CARD	0.86	0.49	0.65	0.84	0.74	0.45	**0.85**	● 0.59	● 0.53	0.92
EAT IN RESTAURANT	0.78	0.48	0.68	0.84	0.98	0.75	**0.81**	● 0.64	0.71	● 0.95
IRON CLOTHES I	0.78	0.54	0.75	0.72	0.95	0.53	**0.75**	0.69	● 0.62	● 0.92
COOK SCRAMBLED EGGS	0.67	0.54	0.55	0.64	0.98	0.69	0.66	***0.70***	0.61	● 0.88
TAKE A BUS	0.80	0.49	0.68	0.80	1.00	0.37	**0.80**	● 0.66	● 0.48	● 0.96
ANSWER THE PHONE	0.83	0.48	0.79	0.86	1.00	0.96	0.84	● 0.64	***0.87***	0.90
BUY FROM VENDING MACHINE	0.84	0.51	0.69	0.85	0.90	0.75	**0.84**	● 0.66	○ 0.71	0.83
IRON CLOTHES II	0.78	0.48	0.75	0.80	0.96	0.66	**0.79**	● 0.64	0.70	0.84
MAKE COFFEE	0.70	0.55	0.50	0.78	1.00	0.55	**0.74**	0.71	○ 0.53	○ 0.83
MAKE OMELETTE	0.70	0.55	0.79	0.83	0.93	0.82	0.76	○ 0.69	***0.81***	● 0.92
AVERAGE	0.77	0.51	0.68	0.80	0.95	0.65	**0.78**	0.66	0.66	0.90

Table 4.3: Evaluation results for the *happens-before task*; *sys* = our system, *base$_{cl}$* = clustering baseline, *base$_{lev}$* = Levenshtein baseline, *upper* = upper bound; significance of difference to *sys*: ● : $p \leq 0.01$, ○ : $p \leq 0.1$

Scenario	Precision			Recall			F-Score			
	sys	base$_{cl}$	base$_{lev}$	*sys*	base$_{cl}$	base$_{lev}$	*sys*	base$_{cl}$	base$_{lev}$	upper
PAY WITH CREDIT CARD	0.52	0.43	0.50	0.84	0.89	0.11	**0.64**	0.58	● 0.17	0.60
EAT IN RESTAURANT	0.70	0.42	0.75	0.88	1.00	0.25	**0.78**	● 0.59	● 0.38	● 0.92
IRON CLOTHES I	0.52	0.32	1.00	0.94	1.00	0.12	**0.67**	● 0.48	● 0.21	● 0.82
COOK SCRAMBLED EGGS	0.58	0.34	0.50	0.86	0.95	0.10	**0.69**	● 0.50	● 0.16	● 0.91
TAKE A BUS	0.65	0.42	0.40	0.87	1.00	0.09	**0.74**	● 0.59	● 0.14	● 0.88
ANSWER THE PHONE	0.93	0.45	0.70	0.85	1.00	0.21	**0.89**	● 0.71	● 0.33	0.79
BUY FROM VENDING MACHINE	0.59	0.43	0.59	0.83	1.00	0.54	**0.69**	0.60	0.57	0.80
IRON CLOTHES II	0.57	0.30	0.33	0.94	1.00	0.22	**0.71**	● 0.46	● 0.27	0.77
MAKE COFFEE	0.50	0.27	0.56	0.94	1.00	0.31	**0.65**	● 0.42	○ 0.40	● 0.82
MAKE OMELETTE	0.75	0.54	0.67	0.92	0.96	0.23	**0.83**	● 0.69	● 0.34	0.85
AVERAGE	0.63	0.40	0.60	0.89	0.98	0.22	**0.73**	0.56	0.30	0.82

Table 4.4: Evaluation results for the *paraphrasing task*; *sys* = our system, *base$_{cl}$* = clustering baseline, *base$_{lev}$* = Levenshtein baseline, *upper* = upper bound; significance of difference to *sys*: ● : $p \leq 0.01$, ○ : $p \leq 0.1$

Regarding precision, our system outperforms both baselines in all scenarios except one (MAKE OMELETTE).

Paraphrase task: Our system outperforms both baselines clearly, reaching significantly higher f-scores in 17 of 20 cases. Moreover, for five scenarios, the upper bound does not differ significantly from our system. For judging the precision, consider that the test set is slightly biased: Labeling all pairs with the majority category (*no paraphrase*) would result in a precision of 0.64. However, recall and f-score for this trivial lower bound would be 0.

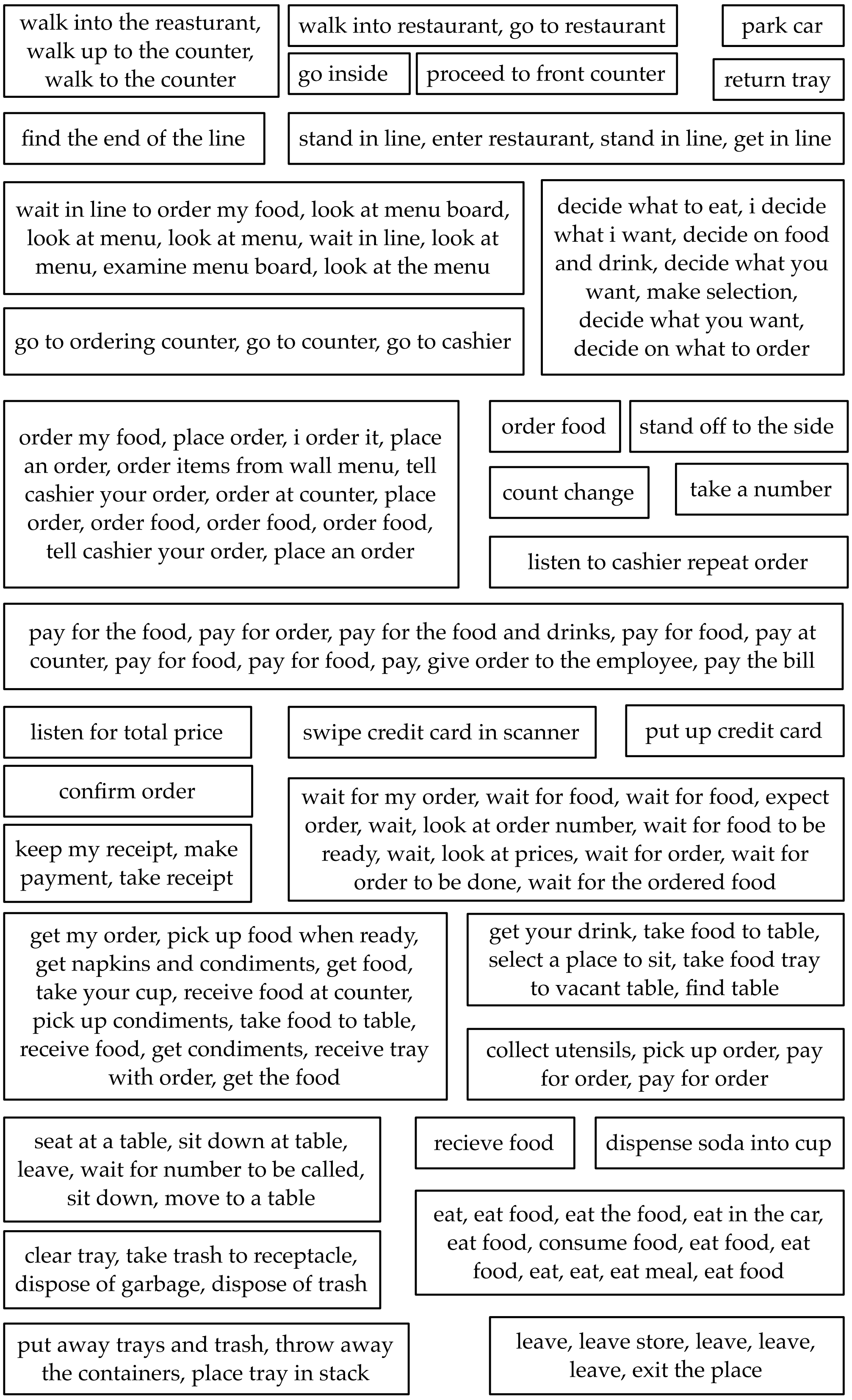
The only scenario in which our system doesn't score very well is BUY FROM A VENDING MACHINE, where the upper bound is not significantly better either. The clustering system, which can't exploit the sequential information from the ESDs, has trouble distinguishing semantically similar phrases (high recall, low precision). The Levenshtein similarity measure, on the other hand, is too restrictive and thus results in comparatively high precisions, but very low recall.

Figure 4.7 shows the paraphrases our system identifies for the FAST FOOD scenario. One phrase group represents one line in the sequence alignment. Some clusters show the remarkable contribution that context similarity provides, aligning paraphrases with very different wording, like *order items from wall menu* and *tell cashier your order*, *decide what to eat* and *make selection*, *clear tray* and *take trash to receptacle* or *pick up food when ready* and *receive tray with order*.

However, there remain some errors that are either due to the similarity measure or related to the non-optimal alignment procedure: *look at prices* and *look at order number* end up with the *waiting* paraphrases, because *look* and *wait* share a Synset in WordNet. *order food* and *receive food* each end up in a singleton group although there are paraphrase sets with string-identical event descriptions. Errors of that kind can happen using non-optimal alignment, especially if the remainder of a sequence is somehow dissimilar compared to other sequences.

Relating scenario complexity & Evaluation Results

In a last evaluation step, we analyze which properties of a scenario and its associated ESDs have the most influence on the final result. To do so, we will use the statistical measures we have introduced earlier to asses a scenario's complexity (cf. Chapter 3). Those measures asses simple quantities (like sequence and sentence length) and also scenario homogeneity with respect to the wording variants and possible event reorderings. Measuring the correlation of those numbers with our final results means investigating which kind of variance in a scenario makes processing most difficult.



walk into the reasturant, walk up to the counter, walk to the counter

walk into restaurant, go to restaurant

park car

go inside

proceed to front counter

return tray

find the end of the line

stand in line, enter restaurant, stand in line, get in line

wait in line to order my food, look at menu board, look at menu, look at menu, wait in line, look at menu, examine menu board, look at the menu

decide what to eat, i decide what i want, decide on food and drink, decide what you want, make selection, decide what you want, decide on what to order

go to ordering counter, go to counter, go to cashier

order my food, place order, i order it, place an order, order items from wall menu, tell cashier your order, order at counter, place order, order food, order food, order food, tell cashier your order, place an order

order food

stand off to the side

count change

take a number

listen to cashier repeat order

pay for the food, pay for order, pay for the food and drinks, pay for food, pay at counter, pay for food, pay for food, pay, give order to the employee, pay the bill

listen for total price

swipe credit card in scanner

put up credit card

confirm order

wait for my order, wait for food, wait for food, expect order, wait, look at order number, wait for food to be ready, wait, look at prices, wait for order, wait for order to be done, wait for the ordered food

keep my receipt, make payment, take receipt

get my order, pick up food when ready, get napkins and condiments, get food, take your cup, receive food at counter, pick up condiments, take food to table, receive food, get condiments, receive tray with order, get the food

get your drink, take food to table, select a place to sit, take food tray to vacant table, find table

collect utensils, pick up order, pay for order, pay for order

seat at a table, sit down at table, leave, wait for number to be called, sit down, move to a table

recieve food

dispense soda into cup

eat, eat food, eat the food, eat in the car, eat food, consume food, eat food, eat food, eat, eat, eat meal, eat food

clear tray, take trash to receptacle, dispose of garbage, dispose of trash

put away trays and trash, throw away the containers, place tray in stack

leave, leave store, leave, leave, leave, exit the place

Figure 4.7: Event paraphrases computed by our final system for EATING IN A FAST FOOD RESTAURANT.

We introduce an additional feature that subsumes both a structural component and our semantic similarity measure. *Sim first / last* computes a score for two ESDs by comparing the two first event descriptions with each other and the two last ones. The numerical score is computed with our similarity measure, and averaged over the two pairs of sequence-initial and -final events. This measure will have higher score if a) a scenario tends to have fixed starting and endpoints and b) the respective event descriptions are similar, according to our measure. In this way, this measure assesses a scenario's homogeneity (as indicated by similarity in ESD start- and endpoints), but its accuracy strongly depends on the performance of the similarity measure.

Concretely, we correlate the f-score of our system for the *paraphrasing task* with different measures for lexical ESD overlap and proportion of predicate pairs without strict sequential order. We did not calculate the correlations with the happens-before task, because the paraphrasing task is more difficult in the first place, and actually subsumes the happens-before task: if we had a perfect paraphrasing algorithm, we could easily derive all temporal constraints from the input data.

Table 4.5 shows the complexity features we introduced in Section 3.4 along with the f-scores of our system. Because it is hard to achieve any significant results with a small sample size (10 scenarios), we also include the five scenarios from our development set to compute correlation of f-scores with scenario complexity. While it would not be adequate to compare our system's performance on its own training data to any baselines, the f-score we reached for those sets can still be used to assess the task's complexity.

The last row of table 4.5 shows the Spearman correlation coefficient (ρ) of the respective complexity feature with the system's f-score. The coefficient is a value between -1 and 1, with 0 meaning that two value distributions do not correlate at all, 1 denoting perfect correlation (of a distribution with itself, e.g.) and -1 a perfect negative correlation (cf. also Chapter 3). On this small dataset, we could not get any significant correlation results. However, the numbers show very interesting tendencies: The best indicator for a scenario with low complexity is the similarity of the first and last events ($\rho = 0.41$). This means that scenarios that have well defined start- and endpoints (e.g. MAKE OMELETTE usually starts in the kitchen, without any ingredients or cutlery) have the general tendency to be easier to match structurally and lexically. A related and also important point is the general sense overlap between ESDs ($\rho = 0.36$). The reason is simply that we can compute more reliable similarity scores for sequences with high sense overlap, because we use a WordNet based similarity measure.

The factor with the highest negative influence on our system's performance is the degree of reordering within the scenario: the negative correlation ($\rho = -0.36$) stands out compared to the other factors. Reordering even seems to complicate processing as much as sense-based similarity of the sequences simplifies it. This is especially interesting

	Scenario	Average Length	Words per Event	Unknown Words	Type Overlap	Token Overlap	Sense Overlap	Sim. First / Last	Reordering	F-Score
ACL-10 Data Set	pay with credit card	5.67	3.91	0.00	0.37	0.33	0.41	140.11	0.04	0.64
	eat at a restaurant	10.42	3.08	0.01	0.37	0.25	0.40	187.63	0.10	0.78
	iron clothes I	9.00	3.87	0.02	0.30	0.28	0.36	110.21	0.10	0.67
	make scrambled eggs	10.63	4.11	0.03	0.33	0.26	0.35	76.82	0.14	0.69
	take a bus	8.71	3.55	0.01	0.35	0.31	0.43	153.10	0.08	0.74
	answer the phone	3.47	3.66	0.03	0.41	0.37	0.45	127.95	0.04	0.89
	buy f vending machine	4.53	3.44	0.04	0.24	0.22	0.25	50.75	0.07	0.69
	iron clothes II	5.14	4.67	0.02	0.36	0.35	0.41	45.35	0.06	0.71
	make coffee	5.00	4.15	0.07	0.36	0.32	0.39	71.22	0.11	0.65
	make omelette	6.16	4.07	0.03	0.30	0.25	0.33	77.68	0.11	0.83
Suppl. Data	send food back	5.93	3.92	0.02	0.29	0.23	0.31	76.13	0.05	0.71
	take a shower	11.29	3.20	0.03	0.41	0.32	0.52	99.45	0.15	0.78
	answer the doorbell	3.59	3.74	0.02	0.39	0.38	0.48	145.14	0.02	0.69
	do laundry	5.69	3.97	0.02	0.32	0.25	0.31	41.26	0.17	0.44
	use a microwave	5.03	3.84	0.06	0.40	0.36	0.40	108.56	0.14	0.75
	average	6.68	3.81	0.03	0.35	0.30	0.39	100.76	0.09	0.71
	Correlation with F-Score	0.11	-0.30	0.02	0.33	0.24	**0.36**	**0.41**	**-0.36**	(1.00)

Table 4.5: Correlation of scenario complexity scores with f-scores from the paraphrasing task.

since the measure underestimates the actual re-ordering, because it approximates event-reordering by counting how many head verb paris occur in both possible orders. The correlation would most likely be much stronger if we had a statistics that assesses the true event-reordering within the sequences. The difficulty with scenarios that have a lot if reordering is actually a price we have to pay for the structural information delivered by multiple sequence alignment, because MSA cannot cope with reordering of any kind.

Surprisingly, the average sequence length does not have any important influence, so the system seems to cope equally well with long and short sequences. However, the event description length bears possible obstacles for processing: Event descriptions with more words have a tendency to co-occur with lower f-scores ($\rho = -0.30$). This is probably due to the fact that the WordNet similarity measure reduces phrases to flat representations of predicates and noun phrases and discards deeper syntactic information, which leads to more noise if there are equivalent nouns in different constellations (e.g. *get your shirt and the iron* vs. *put the iron on the shirt*).

Apart from the trivial conclusion that more similar sets of sequences are easier to process, this analysis points to interesting directions for future work: first, we should explore ways to incorporate structural constraints in a less strict manner than MSA, in order to allow for some treatment of reordering. Chapter 9 sketches our current experiments in that direction. Second, our paraphrasing algorithm can cope surprisingly well with long input sequences, so we could also consider applying it to more complex scenarios, or completely different text types, like the texts we used for paraphrasing in Chapter 8.

4.6 Related Work

There are only a few script learning approaches, and only a small fraction of them have been systematically evaluated. We already discussed the most prominent systems by Chambers & Jurafsky (2008b, 2009) and Manshadi *et al.* (2008) in Chapter 2.

Two approaches that share our main objectives were proposed by Smith & Arnold (2009) and Finlayson (2009). Smith & Arnold (2009) also use the OMICS corpus to compute graph-based representations very similar to ours. In effect, their mining technique and multiple sequence alignment draw on similar ideas. Smith & Arnold give interesting hints on semantic generalization over event descriptions, and they define recursive structures of script representations that allow an event to be a script graph itself. Smith and Arnold present their approach mostly as a feasibility study, in which they outline their theory using a detailed example, but do not apply it to more data. In consequence we cannot compare our results to theirs in terms of precision or recall, because they did not do any numerical evaluation or system tests with more than the example scenarios.

Finlayson (2009) introduces a similar approach in which so-called *story morphologies* are derived from sets of folk stories that share some similar patterns. Such morphologies (or "topoi") are plot archetypes that often recur in folk tales. Finlayson builds a Bayesian model by filtering and merging common parts of related folk tale stories, resulting in one probabilistic model of story events and weighted temporal constraints over them. The theory is applied in an extensive manual experiment that demonstrates the extraction of a morphology from different Shakespeare plays. The proposed modeling account is very interesting with respect to the level of generalization and in its ability to capture probabilities in event structures. Unfortunately, there is no discussion of the actual performance or the degree of supervision that is necessary to actually apply the system.

In a sense, learning event paraphrases also resembles the task of cross-document event coreference resolution, which means to automatically find different textual realizations of the same event (mostly newsworthy stories) in different texts. Algorithms dealing with that problem mostly use standard paraphrasing techniques that do not take any structural constraints into account (Bagga & Baldwin, 1999; Tomadaki & Salway, 2005). Jones & Thompson (2003) describe an approach to identifying different natural language realizations for the same event and considering the temporal structure of a scenario. As in event coreference resolution, they only target paraphrases for one specific event rather then computing synchronized sequences.

Barzilay & Lee (2003) also use MSA for learning paraphrases. Unlike Barzilay and Lee, we do not tackle the general paraphrase problem, but only consider whether two phrases describe the same event in the context of the same script. Furthermore, the atomic units of our alignment process are entire phrases, while in Barzilay and Lee's setting, the atomic units are words.

There is more work on paraphrasing in general, which we review in Section 8.7.

Chapter 5

Mining Script Participants

We have shown how to mine Temporal Script Graphs as concise script representations from different event sequence descriptions (cf. Chapter 4). Taking a more detailed look inside the event descriptions, we also mine script *participants* as another layer of information that can enable more precise inference. As for events, we also find many lexical variants in participant descriptions that refer to the same participant in different ESDs.

Take the scenario of HEATING FOOD IN A MICROWAVE, with the exemplary aligned sequences in Table 5.1: the participants are e.g. the `food`, the `microwave`, the `container` and the `actor`. In different ESDs, we can find different participant descriptions for some of them: While the `food` is always referred to as *food*, the `microwave` appears as *oven* or *microwave*, and the `container` (in which the food is put) can be either a *plate* or a *bowl* or a *dish*. A special case that we discussed earlier (cf. Chapter 3) is the `actor`, which usually remains implicit in this bullet-point style.

Similar to the task for event paraphrasing, finding participant paraphrases is challeng-

	ESD 1	**ESD 2**	**ESD3**
1	put food on plate	put food in bowl	put food on dish
2	open microwave	open door	open oven
3	put plate in	put food inside	place dish in oven
4	set the timer	$\oslash$	$\oslash$
5	close microwave	close door	close
6	$\oslash$	enter time	select desired length
7	press start	push button	$\oslash$
8		...	

Table 5.1: Alignment for the MICROWAVE scenario.

ing, because knowing the micro-domain of a given scenario already resolves many potential ambiguities for the human reader, so allowing semantically very different descriptions to refer to the same participant. Two phenomena are typical for participant realization in event sequence descriptions:

- *Underspecification:* some very ambiguous expressions are disambiguated by the scenario context or current event, and thus words that are barely semantically related can often refer to the same thing. For example, *time* and *desired length* in event 6 (the MICROWAVE scenario) refer to the same kind of setting, but have only a small lexical overlap. This phenomenon is often paired with ellipses or metonymies (like in this case), which leads to references that are easy to resolve for humans, while the lexical items are almost semantically unrelated. Another example is e.g. the use of *coffee* in the scenario MAKE COFFEE, where *coffee* can refer to the package, beans, powder or the fluid - depending on the event context.
- *Metonymies:* We often find metonymies that are typical for the scenario-specific language usage and occur in many distinct events. As for all metonymies, the metonymic expression can in many cases replace the literal source word, but not in every context: Take for example the event in row 3 (Table 5.1), where *food* and *dish* can be used interchangeably; the dish's content is used as metonymic reference for the physical container. We can find the same metonymy in multiple event descriptions, e.g. *put lid on* or *open*, but it's not valid in all contexts (e.g. it's impossible to *stir the bowl*). Other scenario-typical metonymies are e.g. *food / order* in the RESTAURANT context, or *envelope / letter* in descriptions for MALING A LETTER. Metonymy processing in general is a hard problem for natural language processing (Stefanowitsch & Gries, 2006).

Lexical similarity alone seems insufficient to resolve the equivalences. However, we can again make use of structural information, making two empirically validated assumptions: first, participant descriptions occurring in similar positions of equivalent event descriptions are likely to be equivalent. Second, the same annotator typically uses only one lexical type to refer to the same participant, so different participant descriptions that occur within a single ESD are likely to refer to different participants.

As for event paraphrasing, we use a combination of *semantic similarity* and *structural information* for finding equivalent participant descriptions.

In the following, we first explain how we model the problem of participant mining as a set partitioning problem (Section 5.1). We then formulate this problem as an Integer Linear Program that includes semantic similarity and structural information as constraints for the optimization problem (Section 5.2). Finally, we evaluate the system and score the influence of structural and semantic similarity separately by comparing our system to several informed baselines (Section 5.3). A discussion of the results (Section 5.4) and a short survey of related work (Section 5.5) conclude the chapter.

This chapter presents work published by Regneri *et al.* (2011).

5.1 Participants and ESDs

For computing script-specific participants, we start from the alignment tables from the multiple sequence alignment that was used to compute event paraphrases. We then want to find out that *plate*, *bowl* and *dish* of the alignment in Table 5.1 fill the same role in the MICROWAVE scenario. We call a mention of a participant (typically a noun phrase) in some event description a *participant description*.

Learning participants from aligned ESDs is done in two steps: first, we identify candidate participant descriptions in event descriptions. Then, we partition the participant descriptions for each scenario into equivalence classes which we call *participant description sets* (PDSs). The resulting sets correspond to script-specific participants, and their members are possible verbalizations of the respective participants.

We consider all noun phrases (NPs) in our data set as participant descriptions, and thus reduce the task of their identification to the task of syntactic parsing. In order to extract noun phrases, we use an adapted version of the Stanford Parser (Klein & Manning, 2003) that was specifically re-trained for our bullet point style event descriptions (cf. Chapter 3).

As discussed earlier, the bullet-point style phrases mostly do not realize actors, so we add the “implicit protagonist” whenever the subject position in the parse tree is empty.

5.2 Participant Computation as Set Partitioning with ILP

The main task consists in the actual learning of script participants. More specifically, we propose a method that groups participant descriptions occurring in the event sequence descriptions for a given scenario into participant description sets that comprise all different mentions of one participant.

Features for participant paraphrasing

We assume that two token-identical participant descriptions always have the same word sense and represent the same participant, not only in one ESD, but across all event descriptions within a scenario. This extends the common "one sense per discourse" heuristic (Gale *et al.*, 1992) with a "one participant per sense" assumption. There are very few examples for which this results in loss of precision, e.g. in the scenario RETURN FOOD IN A RESTAURANT, there are two participants that are referred to as *dish*, namely the original meal and the replacement that the customer gets upon complaining.

Because examples like this are extremely rare, we can safely take participant description types (PTs) rather than tokens to be basic entities, which drastically reduces the size of the basic entity set.

To compute equivalence of participant types, we exploit structural information given in the alignment tables. Intuitively, we can assume equivalent event descriptions to contain references to the same participants. Taking aligned event descriptions as equivalent, we consider two PTs that occur in aligned event descriptions as more likely to belong to the same participant. In the example of Table 5.1, this supports identification of *time* and *desired length*.

A structural feature that signals non-equivalence is based on the empirically confirmed assumption that one annotator usually uses the same referring expression (or a pronoun) to refer to the same participant. If two different PTs occur in the same event sequence description, they are likely to belong to different PDSs.

We complement these structural indicators with semantic similarity information. As described in the previous chapter, standard corpus-based similarity models trained on newspaper texts do not perform well on our data, and, as for event paraphrasing, we obtained the best results by using WordNet-based features. In the example of Table 5.1, the identification of *bowl* and *dish* is e.g. supported by WordNet hyponymy. We use semantic similarity information in two different ways:

- WordNet synonymy of PTs, as well as synonymy and direct hyponymy of the head of multiword PTs (like *full can* and *full container*) guarantee participant identity
- A WordNet-based similarity score is used as a soft indicator of participant identity

Like for event descriptions, we always compare all possible synsets of two participant descriptions, and evaluate the features of the ones with the closest relationship.

The set problem as linear program

We combine all information sources by modeling the equivalence-class problem as an Integer Linear Program (Wolsey, 1998, *ILP*). An ILP computes an assignment of integer values to a set of variables, maximizing a given *objective function*. Additional linear equations and inequalities can constrain the possible value assignments.

The problem we want to solve is to determine for each pair pt_i and pt_j in the set of PTs $\{\text{pt}_1, \ldots, \text{pt}_n\}$ whether they belong to the same equivalence class. We model this in our ILP by introducing variables x_{ij} which can take the values 0 or 1; if x_{ij} takes the value 1 in a solution of the ILP, this means that the tokens of pt_i and the tokens of pt_j belong to the same participant description set.

Objective function

The objective function combines semantic and structural information as soft indicators for equivalence. The function prefers to classify participant types as equivalent if they are semantically similar and occur in event paraphrases, and avoids grouping dissimilar PTs used by the same annotator in a single event sequence description.

Formally, we require the ILP solver to maximize the value of the following linear term:

$$\sum_{i,j=1, i \neq j}^{n} (\text{sim}(pt_i, pt_j) \cdot \text{struc}(pt_i, pt_j) - \theta) \cdot x_{ij} \tag{5.1}$$

$\text{sim}(i, j)$ stands for the semantic similarity of pt_i and pt_j and is computed as follows:

$$\text{sim}(pt_i, pt_j) = \begin{cases} lin(pt_i, pt_j) + \eta & \text{if } pt_i \text{ and } pt_j \text{are hyponyms} \\ lin(pt_i, pt_j) & \text{otherwise} \end{cases} \tag{5.2}$$

For computing similarity, we use Lin's (WordNet-based) similarity measure (Lin, 1998; Fellbaum, 1998), which performs better than several distributional measures which we have tried. Direct hyponymy is a particularly strong indicator; therefore we add the empirically determined constant η to *sim* in this case.

θ is an empirically optimized cutoff. Every pair with a similarity lower than θ adds a negative score to the objective function when its variable is set to 1. In the final solution, pairs with a similarity score smaller than θ are thus avoided whenever possible.

$\text{struc}(i, j)$ encodes structural information about pt_i and pt_j, i.e. how tokens of pt_i and pt_j are related in the alignment table. Equation 5.3 defines this:

$$\text{struc}(pt_i, pt_j) = \begin{cases} \lambda_{bonus} & \text{if } pt_i \text{ and } pt_j \text{ from same row} \\ \lambda_{penalty} & \text{if } pt_i \text{ and } pt_j \text{ from same column and unrelated} \\ 1 & \textit{otherwise} \end{cases} \tag{5.3}$$

This implements the structural constraints we explained earlier. If pt_i and pt_j are aligned at least once (i.e., their enclosing event descriptions are paraphrase candidates), struc(i, j) takes a constant value λ_{bonus} greater than 1, thus boosting the similarity of pt$_i$ and pt$_j$. If the tokens of pt$_i$ and pt$_j$ occur in the same *column* and the two types have no direct WordNet link (i.e., they are semantically unrelated and used by the same subject in an ESD), struc(pt_i, pt_j) takes a constant value smaller than 1 ($\lambda_{penalty}$) and lowers the similarity score. Both values are empirically optimized.

Hard constraints

Some features indicate not only a tendency to equivalence, but almost exclusively occur with actual paraphrases, in particular within the same scenario. Three of those features refer to semantic similarity (e.g. synonymous terms), two are of purely logical nature (i.e. symmetry and transitivity), and one concerns a special treatment of the (mostly implicit) protagonist.

For semantic features, we use the following as hard indicators for equivalence, adding a constraint $x_{ij} = 1$ for a pair i, j if one of the following conditions holds:

- pt$_i$ and pt$_j$ share a synset in WordNet. In the narrow context of a scenario, two words that share a synonymous meaning are usually not used to refer to other readings, simply to avoid unnecessary ambiguity. This is the reverse side of the fact that even very underspecified terms can be easily resolved by humans in the scenario context. (E.g. in the scenario FEEDING A DOG, nobody will use the word *canine* to refer to a tooth.)
- pt$_i$ and pt$_j$ have the same head, like *laundry machine* and *machine*. This "same-head heuristic" is often applied in coreference resolution and is particularly suitable for simple noun phrases (without any relative clauses) and phrases from a narrow domain (Elsner & Charniak, 2010).
- pt$_i$ and pt$_j$ are both multiword expressions, their modifiers are identical and their heads are either synonyms or hyponyms. This is a modified version of the same head heuristic, in which we allow strongly related (but not synonymous) heads to be seen as equivalent if their modifiers are the same.

We introduce a special rule for the implicit protagonist, which rarely gets an explicit reference, and if so the referent is generally a personal pronoun (*I* or *you*). If pt$_i$ is the implicit protagonist, we set $x_{ij} = 1$ if pt$_j$ is a first or second person pronoun, and $x_{ij} = 0$ otherwise.

Apart from those semantic constraints, we ensure that the ILP groups the participant types into equivalence classes by enforcing symmetry and transitivity. Symmetry is trivially encoded by the following constraint over all i and j:

$$x_{ij} = x_{ji} \tag{5.4}$$

Transitivity can be guaranteed by adding the following constraints for each i, j, k:

$$x_{ij} + x_{jk} - x_{ik} \leq 1 \tag{5.5}$$

This is a standard formulation of transitivity, used e.g. by Finkel & Manning (2008).

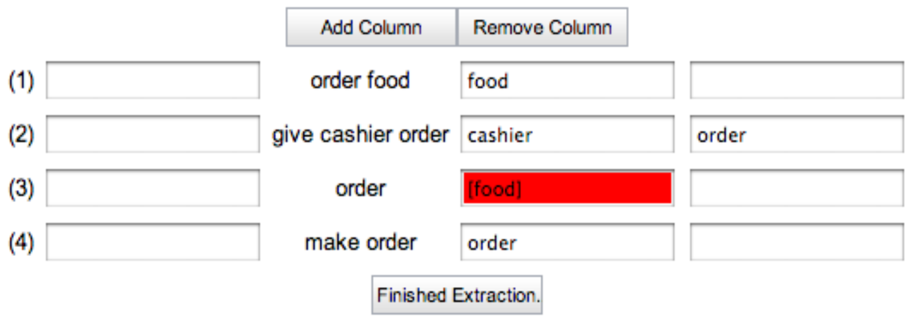
5.3 Evaluation and Results

We evaluate our system against a gold standard of 10 scenarios. On average, one scenario consists of 180 event descriptions, containing 54 participant description types realized in 233 tokens. The scenarios are EAT AT A FAST FOOD RESTAURANT, RETURN FOOD (IN A RESTAURANT), PAY WITH CREDIT CARD, TAKE A SHOWER, FEED A PET DOG, MAKE COFFEE, HEAT SOMETHING IN A MICROWAVE, MAIL A LETTER, BUY SOMETHING FROM A VENDING MACHINE, and DO LAUNDRY. The VENDING MACHINE and LAUNDRY scenarios were used for parameter optimization. The values we determined were $\theta = 5.3, \eta = 0.8, \lambda_{bonus} = 3.4$ and $\lambda_{penalty} = 0.4$. We solve the ILP using LPSolve (Berkelaar *et al.*, 2004).

Gold Standard

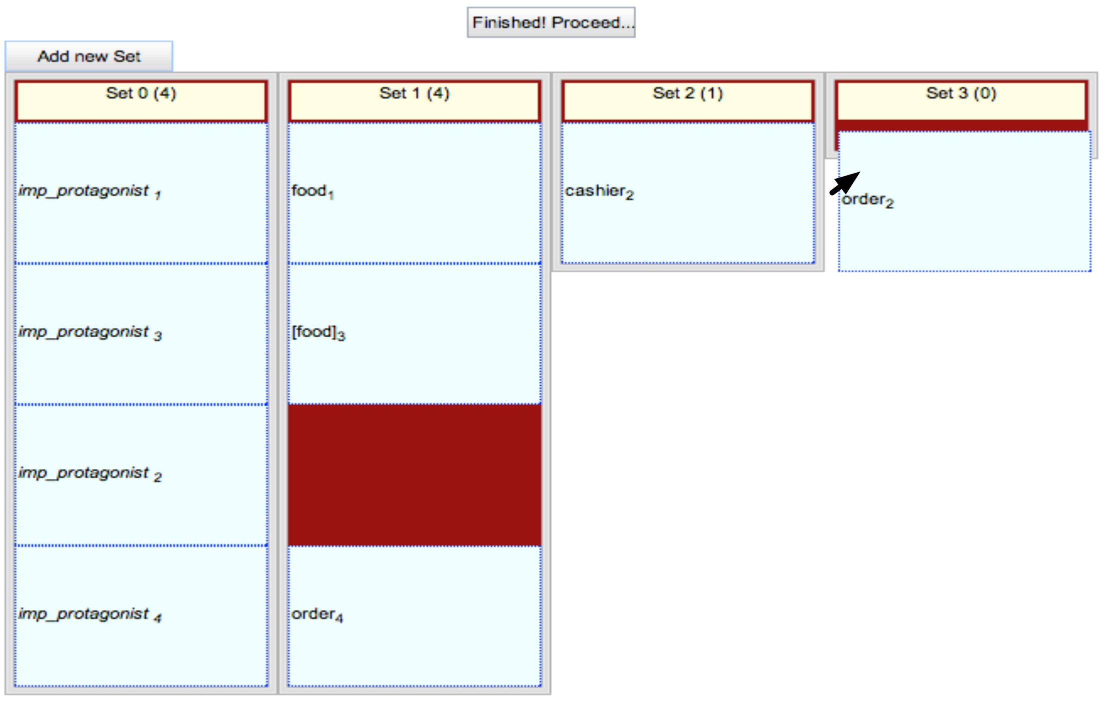
We preprocessed the 10 evaluation scenarios by aligning them with the multiple sequence alignment algorithm introduced in the previous chapter. Two annotators then labeled the 10 aligned scenarios, recording which noun phrases referred to the same participant. Figure 5.1 and 5.2 show the annotation interface we implemented for this coreference annotation task. The labelers were shown, in order, the sets of aligned event descriptions (Figure 5.1, step 1). For instance, for the FAST FOOD scenario, they would first encounter all available alternative descriptions for ordering food. From each aligned description, the annotators extracted the participant-referring NPs, which were then grouped into blocks of coreferent mentions (Figure 5.1, step 2). After all sets of component-event descriptions had been processed, the annotators also manually sorted the previously extracted blocks into equivalence sets (Figure 5.2).

In case of metonymies (like *wait for order* instead of *wait for food*), the coreference was assigned for the literal reading of the noun phrase (*food* and *order* would end up in separate equivalence classes). However, for later re-use, those cases were marked as metonymically paired (this marking is not shown in the screen shots). Implicit participants, including the implicit protagonist, were annotated, too, and marked as null



(1)		order food	food	
(2)		give cashier order	cashier	order
(3)		order	[food]	
(4)		make order	order	

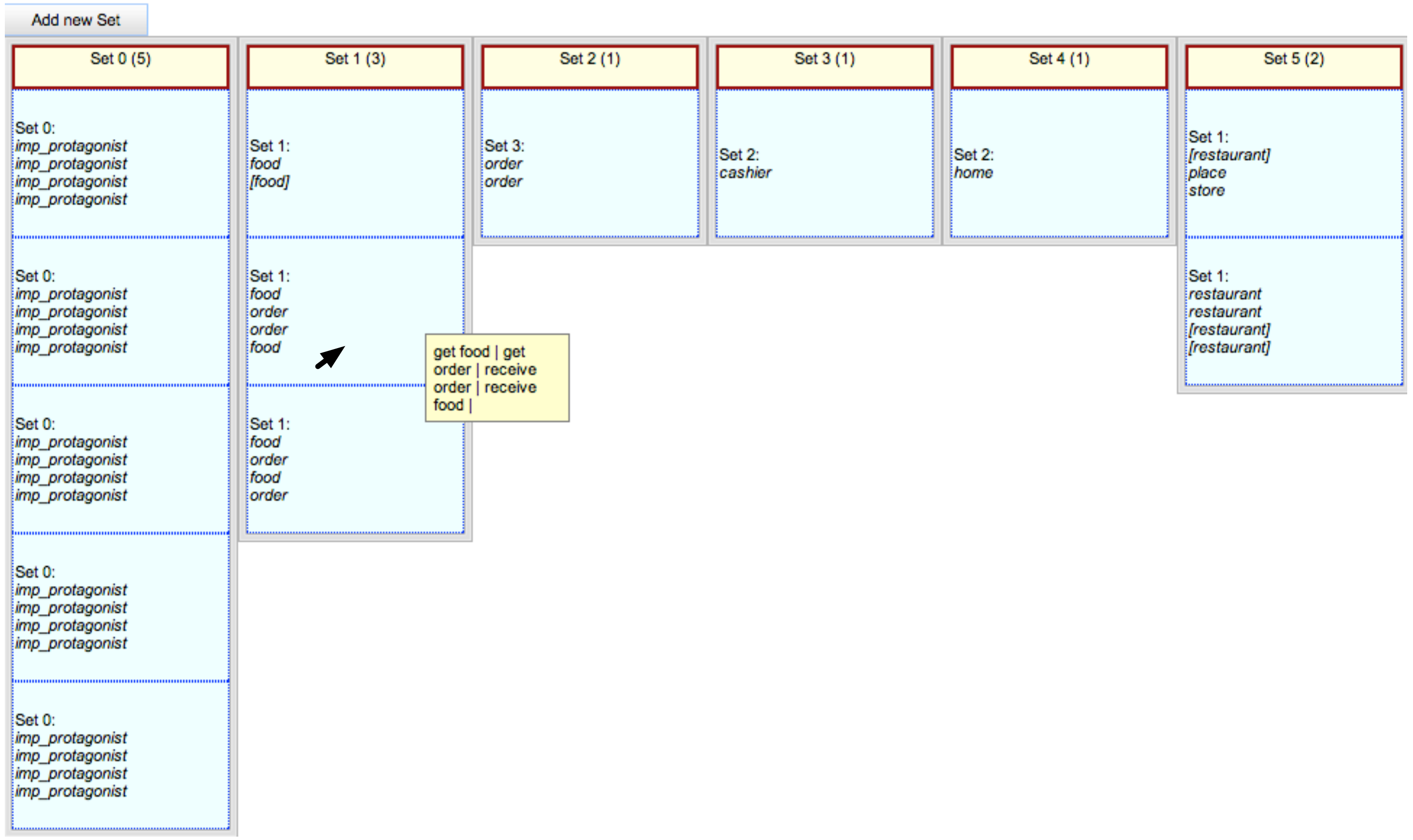
Step 1: participant description extraction (from one set of event paraphrases), to avoid typos, a red background marks tokens that are not part of the respective event description



Set 0 (4)	Set 1 (4)	Set 2 (1)	Set 3 (0)
imp_protagonist $_1$	food$_1$	cashier$_2$	order$_2$
imp_protagonist $_3$	[food]$_3$		
imp_protagonist $_2$			
imp_protagonist $_4$	order$_4$		

Step 2: sort participant descriptions (from one set of event paraphrases) into equivalence sets

Figure 5.1: Annotation interface for participant annotation (step 1 and 2).



Step 3: sort equivalence sets participant descriptions (from one scenario) into larger equivalence sets, tooltips for each set show the original event descriptions that contained the participant descriptions

Figure 5.2: Annotation interface for participant annotation (step 3).

instantiations (e.g. the implicit mention to *food* in *order food*). For the final evaluation, we include missing subjects but do not consider other implicit participants. Each annotator labeled 5 of the scenarios independently, and reviewed the other annotator's work. Difficult cases, mostly related to metonymies, were solved in consultation.

Baseline and Scoring Method

The system sorts participant descriptions into their equivalence classes, thus we evaluate whether the equivalence statements are correct and whether the classes it found are complete. Speaking in terms of participant description sets, we evaluate the purity of each set (whether all items in a set belong there) and the set completeness (whether another set should have been merged into the current one).

Baselines

We compare our system with three baselines: As a naïve baseline (*base*), we group participant descriptions together only if they are string-identical. This is equivalent to just employing the type-abstraction step we took in the full system and ignoring other information sources.

Additionally, we show the influence of the structural information with a more informed baseline (*sem*): we replicate our full system but just use the semantic similarity including all hard constraints, without any structural information from the alignment tables. This is equivalent to setting $\text{struc}(i,j)$ in equation 5.1 always to 1.

In order to show that semantic similarity and the alignment indeed provide contrastive knowledge, we test a third baseline that contains the structural information only (*align*). Here we group all noun phrases i and j together if $\text{struc}(i,j) > 1$ and the pair (i,j) meets all hard constraints.

All parameters for the baselines were optimized separately using the same development scenarios as for the full system.

Scoring Method

Because the equivalence classes we compute are similar to coreference sets, we apply the b^3 evaluation metric for coreference resolution (Bagga & Baldwin, 1998). b^3 defines precision and recall as follows: for every token t in the annotation, take the coreference set C_t it is assigned to. Find the set C_{t+gold} that contains t in the gold standard, and assign *precision*$_t$ and *recall*$_t$:

$$precision_t = \frac{|C_t \cap C_{t+gold}|}{|C_t|} \quad recall_t = \frac{|C_t \cap C_{t+gold}|}{|C_{t+gold}|} \quad f\text{-}score = \frac{2 * precision * recall}{precision + recall}$$

Unlike in coreference resolution, we have the problem that we compare gold standard annotations against tokens extracted from automatic parses. However, the b^3-metric is only applicable if the gold standard and the test data contain the same set of tokens. Thus we use b^3_{sys}, a variant of b^3 introduced by Cai & Strube (2010). b^3_{sys} extends the gold standard and the test set such that both contain the same set of tokens. Roughly speaking, every token that appears in the gold standard but not in the test set is copied to the latter and treated as a singleton set, and vice versa (see Cai and Strube for details).

With the inaccurate parser, noun phrases are often parsed incompletely, missing modifiers or relative clauses. We therefore consider a participant description as equivalent with a gold standard phrase if they have the same head. This relaxed scoring metric evaluates the system realistically by punishing parsing errors only moderately.

Results

Scores

Table 5.2 shows the results for our system and three baselines. *full* marks the complete system, *sem* is the baseline without structural information, *align* uses exclusively structural information and *base* is the naïve string matching baseline. The starred scenarios were used for parameter optimization and excluded from the final average score. (The AVERAGE* row includes those scenarios.) In terms of the average F-Score, we outperform the baselines significantly ($p < 0.05$, paired two-sample t-test on the f-scores for the different scenarios) in all three cases. The system difference to the naïve baseline even reaches a significance level of $p < 0.001$.

While the naïve baseline always gets the best precision results, the *align*-baseline performs best for recall. Because the alignment baseline also incorporates alignment errors, many mistakenly aligned participant types end up as paraphrases, and the transitivity constraint then enforces putting nearly all participant descriptions into the same cluster. This often leads to a simple partition of all PTs into two sets, one containing the protagonist, and one containing everything else.

Our system finds the best tradeoff between precision and recall, gaining 15% recall on average compared to the naïve baseline and losing only about 6% precision. *sem* and the

Scenario	Precision				Recall				F-Score			
	full	sem	align	base	full	sem	align	base	full	sem	align	base
DOING LAUNDRY	0.85	0.76	0.53	0.93	0.75	0.83	0.89	0.57	**0.80**	0.79	0.67	0.70
VENDING MACHINE	0.80	0.74	0.57	0.84	0.78	0.83	0.97	0.62	**0.79**	0.78	0.72	0.72
FAST FOOD	0.82	0.65	0.55	0.87	0.82	0.85	0.84	0.70	**0.82**	0.74	0.66	0.78
RETURNING FOOD	0.80	0.78	0.53	0.88	0.44	0.52	0.63	0.34	0.57	***0.62***	0.57	0.49
MAKING COFFEE	0.85	0.77	0.53	0.92	0.80	0.81	0.98	0.68	**0.82**	0.79	0.68	0.78
FEEDING A DOG	0.81	0.67	0.53	0.90	0.88	0.92	0.94	0.57	**0.84**	0.78	0.68	0.70
HEATING IN MICROWAVE	0.89	0.78	0.55	0.93	0.84	0.84	0.89	0.70	**0.86**	0.81	0.68	0.80
PAYING W. CREDIT CARD	0.90	0.82	0.60	0.94	0.54	0.54	0.64	0.40	**0.67**	0.65	0.62	0.56
MAILING A LETTER	0.92	0.78	0.54	0.96	0.88	0.88	0.93	0.74	**0.90**	0.83	0.68	0.84
TAKING A SHOWER	0.87	0.79	0.57	0.94	0.83	0.83	0.86	0.66	**0.85**	0.81	0.69	0.77
AVERAGE*	0.85	0.75	0.55	0.91	0.75	0.79	0.86	0.60	• **0.79**	• 0.76	0.66	0.71
AVERAGE	0.86	0.76	0.55	0.92	0.75	0.77	0.84	0.60	• **0.79**	0.75	0.66	0.71

Table 5.2: Results for the full system, the system without structural constraints (sem), the system with structural information only (align) and the naive baseline. Participant descriptions with the right head are considered correct. Starred scenarios have been used for parameter optimization, *average** includes those scenarios, the unmarked average doesn't. A black dot (•) means that the difference to the next lower baseline is significant with $p < 0.05$. The difference between full and base is significant at $p < 0.001$.

PRECISION				
np-matching	full	sem	align	base
Gold Tokens	0.92	0.81	0.54	0.97
Matching Head	0.86	0.76	0.55	0.92
Strict Matching	0.82	0.74	0.52	0.91

RECALL				
np-matching	full	sem	align	base
Gold Tokens	0.86	0.88	0.96	0.71
Matching Head	0.75	0.77	0.84	0.60
Strict Matching	0.70	0.71	0.77	0.59

F-SCORE				
np-matching	full	sem	align	base
Gold Tokens	0.89	0.84	0.70	0.81
Matching Head	0.79	0.75	0.66	0.71
Strict Matching	0.74	0.71	0.62	0.71

Table 5.3: Averaged evaluation results for three scoring methods:
Gold Tokens uses gold standard participant descriptions rather than automatic extraction.
Matching Head uses parsing for participant description extraction, and phrases with the right head are considered correct.
Strict Matching requires the whole phrase to match.

naïve baseline differ only moderately. This shows that semantic similarity information alone is not sufficient for distinguishing the different participant descriptions, and that the exploitation of structural information is crucial. However, the structural information by itself is insufficient to distinguish the descriptions: high precision loss makes *align* even worse than the naïve baseline.

Table 5.3 compares the same-head scoring metric described in the previous section *(Matching Head)* against two other approaches of dealing with wrongly recognized NP tokens: *Strict Matching* only accepts two NP tokens as equivalent if they are identical; *Gold Tokens* means that our PDS identification algorithm runs directly on the gold standard tokens. This shows that parsing accuracy has a considerable effect on the overall performance of the system. However, our system robustly outperforms the baselines regardless of the matching approach.

Example Output

Figure 5.3 and 5.4 illustrate our system's behavior showing its output for the MICROWAVE scenario. Each rectangle in Figure 5.3 represents one PDS, which we replace by an icon in the graph in Figure 5.4. (We omit some PDSs in the presentation for readability and clarity.) The participant types in the sets are ordered by frequency, starting with the most frequent. The labels of the sets are manually generated script role labels and were introduced for readability. Note that the structural alignment information allows

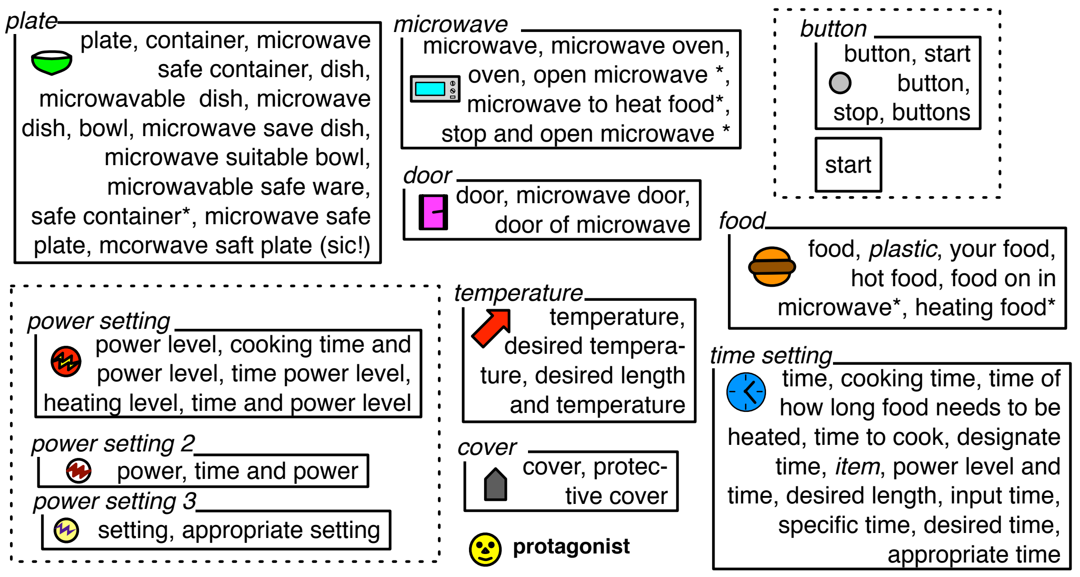


plate
plate, container, microwave safe container, dish, microwavable dish, microwave dish, bowl, microwave save dish, microwave suitable bowl, microwavable safe ware, safe container*, microwave safe plate, mcorwave saft plate (sic!)
microwave
microwave, microwave oven, oven, open microwave *, microwave to heat food*, stop and open microwave *
button
button, start button, stop, buttons
start
door
door, microwave door, door of microwave
food
food, *plastic*, your food, hot food, food on in microwave*, heating food*
power setting
power level, cooking time and power level, time power level, heating level, time and power level
power setting 2
power, time and power
power setting 3
setting, appropriate setting
temperature
temperature, desired temperature, desired length and temperature
cover
cover, protective cover
protagonist
time setting
time, cooking time, time of how long food needs to be heated, time to cook, designate time, *item*, power level and time, desired length, input time, specific time, desired time, appropriate time

Figure 5.3: Example output of participant description sets for the MICROWAVE scenario. Asterisks (∗) indicate parsing errors, *italics* mark genuine clustering mistakes.

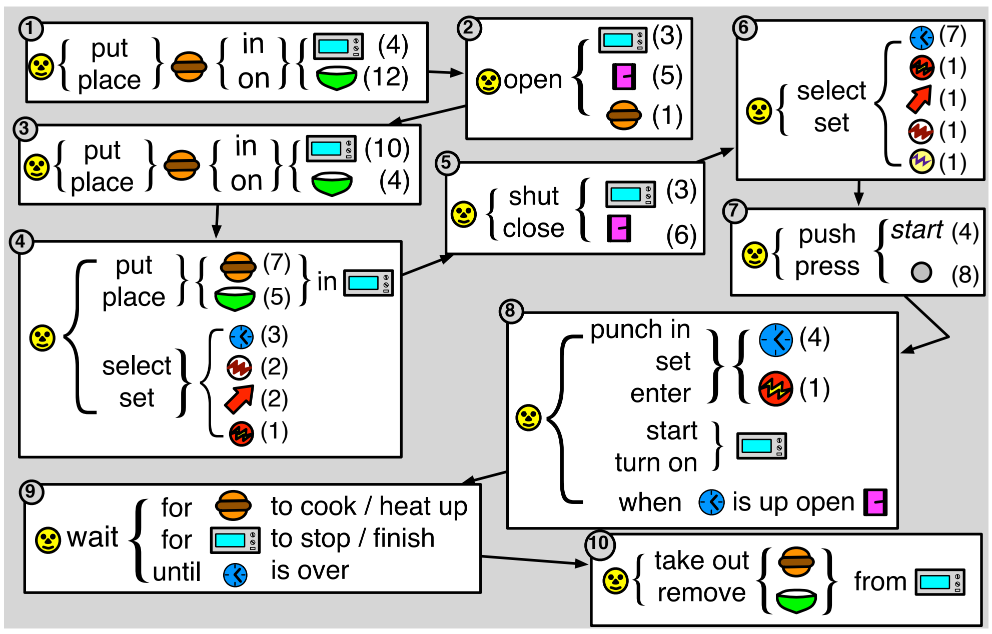


Figure 5.4: Simplified TSG of the MICROWAVE scenario; participant descriptions of the same PDS are collapsed.

us to correctly classify *plate* and *container*, and *stop* and *button*, as equivalent, although they are not particularly similar in WordNet. However, especially for rare terms, our algorithm seems too strict: it does not combine the three *power setting* PDSs. Also, we cannot tell start from stop buttons, which is mainly due to the fact that most people did not distinguish them at all but just called them *button(s)* (some microwaves just have one button). The separate grouping of *start* is also related to parsing errors: *start* was mostly parsed as a verb, even when used as object of *push*.

Figure 5.4 shows an abstract version of the temporal script graph for this scenario, with all NP tokens replaced by icons for their PDSs. Ten of its nodes are shown with their temporal ordering, marked by the edges and additionally with encircled numbers. Alternative PDSs are marked with their absolute frequencies.

As the subject is always left out in the example data, we assume an implicit protagonist in all cases. The figure demonstrates that we can distinguish the participants even though the event alignment has errors.

5.4 Discussion

In our approach, we combine structural and semantic features to compute script participants. As far as the results are concerned, we have a very tough baseline to beat: clustering identical noun phrases has an f-score of over 0.7, and even over 0.8 if we assume perfect parses. That lexemes are often tied to one specific sense is also a fact known from Word Sense Disambiguation, where the most-frequent-sense baseline is extremely hard to beat (see Navigli (2009) for a survey on the task). We successfully beat a comparably "naïve" yet strong baseline by combining the structural and the semantic components, whereas the semantic component by itself does not score much better than the baseline, and the structural component scores lower than the baseline.

Of course the current results could be further improved. A perfect parser would already increase the final score by at least 10% (cf Table 5.3). A more sophisticated measure for semantic similarity, possibly with better awareness of local metonymies, would also help. To apply the approach to a larger dataset, we would also have to do coreference resolution, which is not trivial for event sequence descriptions. (Chapter 6 describes our work towards coreference resolution for our corpus.)

As the evaluation suggests, the task of extracting participants is very similar to coreference resolution, in particular across documents. We therefore believe that in future work, cross-document anaphora resolution could benefit from combining similar types of constraints, e.g. semantic similarity and document structure. On the other hand, we could learn from recent approaches to cross-document coreference resolution that resolve event and entity coreference synchronously (Lee *et al.*, 2012). If we port such a method to event and participant paraphrasing, both tasks would likely profit from the bidirectional information flow.

5.5 Related Work

There is very few little on participant computation for scripts: Chambers and Jurafsky (Chambers & Jurafsky, 2009) extend their narrative chains to *narrative schemas* which include events and their participants, which we summarized in Chapter 2.

Integer Linear Programming as a means to globally combine information from diverse sources has been applied to a variety of problems in NLP (Althaus *et al.*, 2004; Barzilay & Lapata, 2006; Berant *et al.*, 2010). In particular, there are approaches to coreference resolution that use ILP as their core mechanism (Denis & Baldridge, 2007; Finkel & Manning, 2008).

Chapter 6

Domain Generalization and Preprocessing

In the two previous chapters, we have presented new paraphrasing approaches for script events and participants. We applied our systems to a rather peculiar corpus of event sequence descriptions (cf. Chapter 3): The short texts are written in bullet point style, and individual sentences are rather short, focused descriptions of mostly exactly one event. Also, the corpus contains very few discourse-related phenomena like anaphoric references, and very many elliptical phrases.

This special event description language emerged from our experimental setup: we encouraged the telegraphic writing style in the instructions for the annotators, and we also set a length restriction for each event description. If we want to scale up our approach and apply it to other datasets from different data collections, we will most likely encounter a different language style that requires different kinds of pre-processing before we can run our algorithms. One example corpus is the kitchen-related ESD set we collected (cf. Section 3.6). In this dataset, no length restrictions were imposed on the event descriptions. Further the kitchen domain, with scenarios like PREPARING A CUCUMBER, led many annotators to write their texts in a recipe-like narrative style, which resulted in complex sentences with many pronouns. For these more complex texts, we consequently need more preprocessing than for the bullet-point style data.

We have already shown how to implement scenario-specific spelling correction (cf. Section 3.3). In this chapter, we will show how we exploit domain-specific redundancy for pronoun resolution, and how we split complex sentences into single event clauses.

This chapter presents work published by Bloem *et al.* (2012).

6.1 Coreference Resolution

Coreference resolution is a standard preprocessing step for texts. In particular, finding the antecedent for a pronoun is essential for text understanding, but also for computing semantic similarity. Consider for example the following event sequence description that describes how to PREPARING A CUCUMBER:

(1) 1. get a large sharp knife
2. get a cutting board
3. put the cucumber on the board
4. hold the *cucumber* in *your weak hand*
5. chop **it** into slices with your strong hand

To resolve the pronoun *it* in the last event description, it is important to understand that the *weak hand* is not chopped up, but rather the *cucumber*. For paraphrase computation it is also essential to know that this sentence (#5) refers to the `cucumber` as a participant, e.g. in order to match it with a description like *slice the cucumber*.

We tried one of the best performing systems for coreference resolution to resolve the pronouns in our dataset (Raghunathan *et al.*, 2010; Lee *et al.*, 2011). Like most open source NLP systems, this tool is mostly trained on newspaper texts, and is thus not designed for our domain. As a consequence, it resolves only 18% of the pronouns in our dataset correctly.

While the grammatical and domain-specific peculiarities of our dataset obviously add complications for standard pronoun resolution systems, we can make use of the redundancy in the narrow domain to make up for these complications.

Due to the imperative writing style, first person and second person pronouns always refer to the subject performing a task. This leaves as unresolved only third person pronouns, of which possessives are uninteresting for the script learning task – with just one person active in the dataset, there should be no ambiguity attributable to personal possessive pronouns (see also our notes on the implicit protagonist in the previous chapter). There is also a relatively restricted space of possible antecedents due to the short texts.

We thus propose to use a simple model of selectional preferences for coreference resolution. To guide the antecedent selection, we check the head verb of each pronoun (e.g. *chop* for *chop it*), and compute its selectional preferences within the whole kitchen ESD corpus. For the verb *chop*, we would find mostly vegetables (like *cucumber*, *onion*, etc.) as complements, apart from a few other direct objects (*fingers* in fact occurs once as a complement of *chop*, in the event description *"tuck your fingers in so they don't get*

Model	EM	VECTOR	OUR SYSTEM
Accuracy	0.18	0.54	**0.63**

Table 6.1: Evaluation of different selectional preference models for pronoun resolution, on two scenarios (103 pronouns total).

chopped!"). From a list of potential antecedents for a certain pronoun, our system picks the one that has the strongest association with the head verb. In the example, this model would chose *cucumber* over both *hand* and *board* as antecedent for *it*, because *cucumber* is more often encountered as the object of *chop* than either of the other two others.

Note that this approach is unsupervised in that it includes only our raw dataset in its selectional preference model, rather than any external corpus or additional annotations. While the model is computationally rather simple, it takes advantage of our parallel data.

In the concrete implementation, we first preprocess the whole dataset with an adapted POS tagger and parser (as described in Section 3.3). As candidate antecedents, we take the n closest preceding noun phrases before the pronoun within the same sequence (in our case, $n = 3$), plus the main theme from the scenario title (for the scenario PREPARING A CUCUMBER, this theme would be *cucumber*). The selectional preference model for guiding antecedent selection is based on verb associations from in the whole corpus. Association values are computed with the marginal odds ratio measure on a dictionary of all head-subject and head-object relations occurring in our data.

We test our model's performance by comparing it to two baselines: The first is a generic state-of-the-art pronoun resolver based on expectation maximization (Charniak & Elsner, 2009, EM), which we applied out of the box.

As a second baseline, we use a state-of-the-art vector space model (Thater *et al.*, 2011, VECTOR) to compute selectional preferences instead of tuning the preferences on our dataset. For each word or phrase, this model computes a vector representing its meaning, based on a large general corpus. For each candidate antecedent of phrases containing a pronoun like *chop it*, we compute the model's vector for the original phrase in which the pronoun is instantiated with the antecedent (*chop cucumber*, *chop hand*, ...). The candidate vector that is closest to the vector of the original verb, computed as the scalar product, is taken as the preferred antecedent.

We evaluate our system and both of the baselines in terms of accuracy compared to human annotation, manually annotating all anaphoric pronouns in ESDs of two scenarios. The EM system crucially relies on grammatical features to identify anaphoric pronouns, and therefore it often fails to even *identify* pronominal anaphora in our idiosyncratic parse trees (recall 0.262). We do not evaluate recall on any of the other systems, because achieving full recall on third person pronouns is actually trivial. The accuracy results show that our approach clearly outperforms both baselines. With its simple heuristics, it is fully adapted to our domain and is robust towards the idiosyncratic language style. Compared to the vector space model approach we observe a 16% accuracy gain, which shows the advantage of our unsupervised domain-adaptation.

6.2 Event Splitting

When we collected the kitchen corpus, we ran a slightly different experiment than in our very first Mechanical Turk experiment (cf. Chapter 3). First we created a tutorial-like context, telling people that they should imagine giving instructions to a boy who has never worked in a kitchen. The domain itself also led people to write in recipe-like style. Further, we did not impose any length constraints on the event descriptions, so the annotators could enter arbitrarily long texts for each individual event. Consequently, many people did not split their texts into single event descriptions, but rather into temporally ordered steps for following a recipe, and they sometimes filled single event slots with two events that need to happen right after another.

Take the following excerpt from an ESD of the PREPARING PASTA scenario:

(2) 1. fill a pot halfway with water. add a dash of salt and a splash of olive oil
    2. turn the stove onto high and boil the water
    3. ...

As cooking instructions, the text works perfectly, and the division of steps intuitively makes sense: actions that have to happen right after one another are combined in one step. However, in an event sequence description, we actually want sequences containing descriptions of *single* events. As a further preprocessing step, we thus develop a shallow heuristic to split such complex event descriptions. More concretely, we split an event description either if it contains a sentence boundary, or if it is made up of conjugated verb phrases.

In example (2), the first sentence simply needs to be split at the sentence border (the fullstop). In cases like (2.2), two distinct event descriptions are coordinated in the same sentence: here we need to dismiss the conjunction and treat the two conjugated clauses as distinct events. Technically, we parse the sentence and separate complete verbal phrases (VP nodes) that are conjugated with any conjunction (in most cases *and*).

Our heuristic captures cases that are grammatically clearly recognizable. There are also more complex examples that would require deeper logical reasoning, like the following event descriptions (from different scenarios):

(3) a. pay with cash or credit card

b. put away trays and trash

c. repeat step 4 to 6

None of the above examples refers to exactly one event, but it is not clear how they can be processed automatically. Disjunctions can't be represented within a single sequence, events that happen at the same time can be seen as one or two events, and repetition mentions need certainly deeper knowledge or more data to be resolved. We ignored those cases for the time being, given that they are rare and thus have only little influence overall.

Chapter 7

Connecting Scripts to the World

As we have explained before (cf. Section 2.2), scripts are a fundamental part of common sense knowledge which we apply every day. This fact also affects language, because it allows us to rely on script-based implications for efficient communication: *I made cucumber salad* implies by default that the acting person also got cucumbers from somewhere, probably paid for them, washed them, cut them, put them into a bowl, and so on. For the very same reason, it is not necessary to make any of those events more explicit in the normal case; moreover, it would actually violate Gricean Maxims of communication to spell out unnecessary details.

While this implicitness problem makes it hard to apply scripts to standard texts, we can exploit the expectation-driving script mechanism when applying scripts to a modality where implicitness and abbreviations do not exist: we ground event descriptions in video data. In recordings, events that are usually abbreviated in text do not just disappear: even though it is not necessary to *mention* that one has to cut cucumbers in order to make cucumber salad, it is still necessary to *do* it.

Connecting videos and textual event descriptions comes with benefits for both computer vision algorithms and linguistic computation. Viewed from the computer vision perspective, event structures can serve as prior knowledge and help to *predict* events in a scenario: if an algorithm recognizes a cucumber and a grater, it could use script information to infer that most likely the cucumber will be grated, possibly to prepare cucumber salad. Additionally, using text-based knowledge as training data is particularly appealing for computer vision problems because textual training data can be acquired much faster and much more cheaply than videos or video annotations. While this dissertation focusses on language-related aspects, we sketch some first studies on event sequence descriptions as prior knowledge for visual computing in Section 9.4.

From a linguistic perspective, the alignment of textual script representations with visual data offers new possibilities to assess event semantics. So far, we have used textual semantic similarity combined with structural event knowledge to compute meaning and similarity of event descriptions. We have also found that descriptions for the same event show high lexical variance due to underspecification, metonymies or simple wording variations. It is thus intriguing to find out whether the visual features of event descriptions behave similarly, or whether instances of the same event always look very similar, while descriptions for the same event can vary a lot (more).

This chapter describes how we ground action descriptions in video data and explore new approaches to automated meaning computation with multimodal action similarity. For this purpose we introduce the TACoS corpus ; TACoS contains videos of common sense tasks along with temporally synchronized textual descriptions of the events in the video. This data source enables the computation of multimodal similarity models that use both visual and textual features. We also present some first experiments in which we show that the multimodal approach has a much higher correlation with human judgements than either of the two separate modalities

After a general introduction to semantic similarity (Section 7.1), we describe the creation of the TACoS corpus, our new multimodal resource (Section 7.2). For our experiments on action similarity, we create the ASim dataset of event description pairs along with human annotations of their similarity (Section 7.3). This dataset then serves as a basis for experiments in which we try to reproduce the similarity rankings of the ASim gold standard with visual as well as textual similarity measures, both alone and in combination. Section 7.4 describes the similarity measures, and we report their performance on the ASim corpus in Section 7.5.

This chapter contains work published in Regneri *et al.* (2013) and Rohrbach *et al.* (2012b).

7.1 Semantic Similarity and Grounding

The estimation of semantic similarity between words and phrases is a basic task in computational semantics, not only for paraphrasing. Vector-space models of meaning are one standard approach. Following the distributional hypothesis, frequencies of context words are recorded in vectors, and semantic similarity is computed as a proximity measure in the underlying vector space. Such distributional models are attractive because they are conceptually simple, easy to implement and relevant for various NLP tasks (Turney & Pantel, 2010). At the same time, they provide a substantially incomplete picture of word meaning, since they ignore the relation between language and extra-linguistic information, which is constitutive for linguistic meaning. In the last few years, a growing amount of work has been devoted to the task of grounding meaning in visual information, in particular by extending the distributional approach to jointly cover texts and images (Feng & Lapata, 2010; Bruni *et al.*, 2011). The results clearly indicate that visual information improves the quality of distributional models. Bruni *et al.* (2011) show that visual information drawn from images is particularly relevant for concrete common nouns and adjectives.

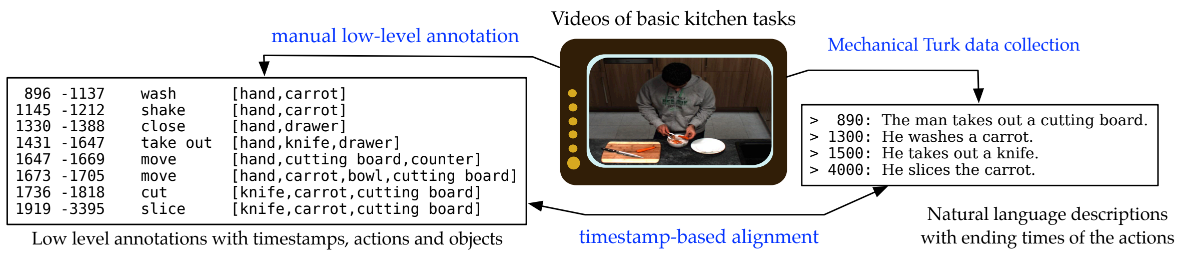
A natural next step is to integrate visual information from *videos* into a semantic model of event and action verbs. Psychological studies have shown the connection between action semantics and videos (Glenberg, 2002; Howell *et al.*, 2005), but to our knowledge, we are the first to provide a suitable data source and to implement such a model.

We offer three new contributions for research on grounded semantic similarity:

- We present the *TACoS corpus*, a **multimodal corpus** containing textual descriptions aligned with high-quality videos. Starting from the video corpus of Rohrbach *et al.* (2012b), which contains high-resolution video recordings of basic cooking tasks, we collected multiple textual descriptions of each video via Mechanical Turk. We also provide an accurate sentence-level alignment of the descriptions with their respective videos. We expect the corpus to be a valuable resource for computational semantics, and moreover helpful for a variety of purposes, including video understanding and generation of text from videos.
- We provide a **gold standard dataset** for the evaluation of similarity models for action verbs and phrases, called *ASim*. The dataset has been designed as analogous to the Usage Similarity dataset of Erk *et al.* (2009) and contains pairs of natural-language action descriptions plus their associated video segments. Each of the pairs is annotated with a similarity score based on several manual annotations.
- We report an experiment on **similarity modeling of action descriptions** based on the video corpus and the gold standard annotation, which demonstrates the impact of scene information from videos. Visual similarity models outperform text-based models; the performance of combined models approaches the upper bound indicated by inter-annotator agreement.

7.2 The TACoS Corpus

We build a new corpus on top of the "MPII Cooking Composite Activities" video corpus (Rohrbach *et al.*, 2012b, *MPII Composites*), which contains videos of different activities in the cooking domain, e.g., *preparing carrots* or *separating eggs*. We extend the existing corpus with multiple textual descriptions collected by crowd-sourcing via Amazon Mechanical Turk. To facilitate the alignment of sentences describing activities with


Videos of basic kitchen tasks
manual low-level annotation
Mechanical Turk data collection
timestamp-based alignment
Low level annotations with timestamps, actions and objects
Natural language descriptions with ending times of the actions

```
 896 -1137    wash       [hand,carrot]
1145 -1212    shake      [hand,carrot]
1330 -1388    close      [hand,drawer]
1431 -1647    take out   [hand,knife,drawer]
1647 -1669    move       [hand,cutting board,counter]
1673 -1705    move       [hand,carrot,bowl,cutting board]
1736 -1818    cut        [knife,carrot,cutting board]
1919 -3395    slice      [knife,carrot,cutting board]
```

```
>  890: The man takes out a cutting board.
> 1300: He washes a carrot.
> 1500: He takes out a knife.
> 4000: He slices the carrot.
```

Figure 7.1: TACoS corpus Overview

their proper video segments, we also obtained approximate timestamps, as described in Sec. 7.2.

MPII Composites comes with timed gold standard annotation of low-level activities and participating objects (e.g. `open [hand,drawer]` or `take out [hand,knife,drawer]`). By adding textual descriptions (e.g., *The person takes a knife from the drawer*) and aligning them on the sentence level with videos and low-level annotations, we provide a rich multimodal resource (cf. Figure 7.1), the “Saarbrücken Corpus of Textually Annotated Cooking Scenes” (TACOS). In particular, the TACoS corpus provides:

- A collection of coherent *textual descriptions for video recordings* of activities of medium complexity, as a basis for empirical discourse-related research, e.g., the selection and granularity of action descriptions in context
- A high-quality *alignment of sentences with video segments*, supporting the grounding of action descriptions in visual information
- *Collections of paraphrases* describing the same scene, which result as a by-product from the text-video alignment and can be useful for text generation from videos (among other things)
- The alignment of activity descriptions with *sequences of low-level activities*, which may be used to study the decomposition of action verbs into basic predicates

We expect that our corpus will encourage and enable future work on various topics in natural language and video processing. In this paper, we will make use of the second aspect only, demonstrating the usefulness of the corpus for the grounding task.

After a more detailed description of the basic video corpus and its annotation, we describe the collection of textual descriptions with MTurk, and finally show the assembly and some benchmarks of the final corpus.

The video corpus

MPII Composites contains 212 high resolution video recordings of 1-23 minutes length (4.5 min. on average). 41 basic cooking tasks (or scenarios) such as CUTTING A CUCUMBER were recorded, each between 4 and 8 times, with different subjects for each recording. The cooking scenarios in the corpus are the same ones that we used for our kitchen ESD corpus (cf. Section 3.6), based on the tutorials from "Jamie's Home Cooking Skills"[1]. The corpus is recorded in a kitchen environment with a total of 22 subjects. Each video shows one scenario executed by an individual subject; each subject performed on average about 5 different tasks.

The dataset is labelled with expert annotations of low-level activity tags: each segment that shows a semantically meaningful cooking related movement pattern is tagged as one low-level activity. The action in the segment must be more complex than single body part movements (such as *move arm up*), and must have the goal of changing the state or location of an object. (In some cases, this goal is not actually achieved, e.g. if a person looks for a knife but cannot find it.)

Each low-level activity tag consists of an activity label (`peel`), a set of associated objects (`carrot`, `drawer`,...), and the associated timeframe (starting and ending points of the activity). There are 60 different activity labels, denoting the main actions in the video segment (e.g. `peel, stir, trash`). Associated objects are the participants of the respective activity, namely tools (e.g. `knife`), patients (`carrot`) and locations (`cutting board`). We provide the coarse-grained role information for *patient*, *location* and *tool* in the corpus data, but we did not use this information in our experiments. The dataset contains a total of 8818 annotated segments, on average 42 per video.

Collecting textual video descriptions

We collected textual descriptions for a subset of the videos in *MPII Composites*, restricting collection to scenarios that involve manipulation of specific cooking ingredients (excluding e.g. USING A BOX GRATER or USE PESTLE AND MORTAR). We also excluded scenarios with fewer than four video recordings in the corpus, leaving 26 tasks to be described. We randomly selected five videos from each scenario, except the three scenarios for which only four videos each are available. This resulted in a total of 127 videos.

For each video, we collected 20 different textual descriptions, resulting in 2540 annotation assignments overall. We published these assignments (HITs) on MTurk, using an adapted version[2] of the annotation tool Vatic (Vondrick *et al.*, 2012). In the MTurk

[1] `http://www.jamieshomecookingskills.com/`

[2] `github.com/marcovzla/vatic/tree/bolt`

setting, we required a HIT approval rate of 75% for the annotators. We opened the assignments to workers in the US only, in order to increase the general language quality of the English annotations. Each task paid 1.20 USD. For quality assurance, we manually inspected randomly selected sample annotations. The total cost of collecting the annotations amounted to 3,353 USD, and we obtained the data within 3.5 weeks.

During the annotation process, the subject saw one video specified with the task title (e.g. How to prepare an onion), and then was asked to enter at least five and at most 15 complete English sentences to describe the events in the video. The annotation instructions contained images of example annotations from a kitchen task that was not part of our actual dataset.

Annotators were encouraged to watch each video several times, skipping backward and forward as they wished. They were also asked to take notes while watching, and to sketch the annotation before entering it into the interface. Once familiarized with the video, subjects did the final annotation by watching the entire video from beginning to end, without the possibility of further non-sequential viewing.

For the actual annotation run, the subjects had to stop the video right after an action was *finished*, and then enter a description for the event that was just completed. We recorded pause onset for each sentence annotation as an approximate ending timestamp of the described action. The annotators resumed the video manually.

We did not define beforehand what an event actually is, but rather showed some examples and instructed the annotators to take everything as an event that they can describe easily within one meaningful sentence. The granularity was further guided by the limitation to a maximum of 15 events. Previous work with similar settings has shown that such loosely guided video segmentation tasks yield high inter-annotator agreements for event segmentations (Zacks *et al.*, 2009).

Putting the TACoS corpus together

Our corpus is a combination of the MTurk data and MPII Composites, created by filtering out inappropriate material and computing a high-quality alignment of sentences and video segments. The alignment is done by matching the approximate timestamps of the MTurk data to the accurate timestamps in MPII Composites.

We discard text instances if people did not time the sentences properly, taking the association of several (or even all) sentences to a single timestamp as an indicator. Whenever we find a timestamp associated with two or more sentences, we discard the whole instance. Overall, we had to filter out 13% of the text instances, which leaves us with 2206 textual video descriptions.

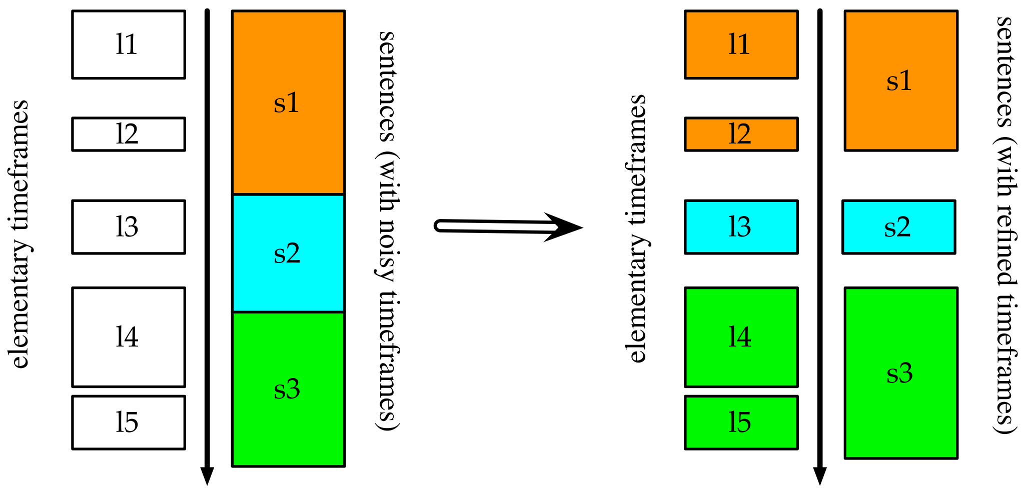


Figure 7.2: Aligning action descriptions with the video.

For the alignment of sentence annotations and video segments, we assign a precise timeframe to each sentence in the following way: we take the timeframes given by the low-level annotation in MPII Composites as a gold standard micro-event segmentation of the video, because they mark all distinct frames that contain activities of interest. We call them *elementary frames*. The sequence of elementary frames is not necessarily continuous, because idle time is not annotated.

The MTurk sentences have end points that constitute a coarse-grained, noisy video segmentation, assuming that each sentence spans the time between the end of the previous sentence and its own ending point. We refine these noisy timeframes to gold frames as shown in Figure 7.2: each elementary frame (*l1-l5*) is mapped to a sentence (*s1-s3*) if its noisy timeframe covers at least half of the elementary frame. We define the final gold sentence frame then as the timespan between the starting point of the first and the ending point of the last elementary frame.

The alignment of descriptions with low-level activities results in a table as given in Figure 7.3. The first five columns display information from MPII Composites (an example picture from the time frame, start and end frame number, low-level activity tags), and the last three columns (*NL Description X*) contain some (**N**atural **L**anguage) descriptions of the videos. Each row corresponds to one segment that has a low-level activity tag, with different columns for low-level actions and for associated participants. Each sentence of a textual description is meant to be aligned with the last of its associated low-level actions.

As a side effect of the temporal alignment, we also obtain multiple paraphrases for each

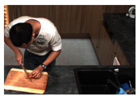
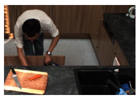
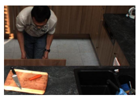
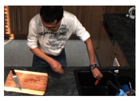
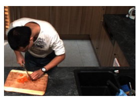
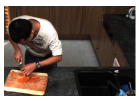
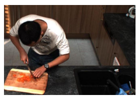
Sample frame	Start	End	Action	Participants	NL Sequence 1	NL Sequence 2	NL Sequence 3
	743	911	wash	hand, carrot	He washed carrot	The person rinses the carrot.	He rinses the carrot from the faucet.
	982	1090	cut	knife, carrot, cutting board	He cut off ends of carrots	The person cuts off the ends of the carrot.	He cuts off the two edges.
	1164	1257	open	hand, drawer			
	1679	1718	close	hand, drawer			He searches for something in the drawer, failed attempt, he throws away the edges in trash.
	1746	1799	trash	hand, carrot		The person searches for the trash can, then throws the ends of the carrot away.	
	1854	2011	wash	hand, carrot			He rinses the carrot again.
	2011	2045	shake	hand, carrot	He washed carrot	The person rinses the carrot again.	He starts chopping the carrot in small pieces.
	2083	2924	slice	knife, carrot, cutting board			
	2924	2959	scratch off	hand, carrot, knife, cutting board			
	3000	3696	slice	knife, carrot, cutting board	He diced carrots		He finished chopping the carrots in small pieces.

Figure 7.3: Excerpt from the corpus for a video on Preparing a Carrot. Example frames and low-level annotation (*Action* and *Participants*) are shown, along with three of the MTurk sequences (*NL Sequence 1-3*).

#	Verbs	Activities	Nouns	Participants
1	*cut*	`move`	*person*	`hand`
2	*take*	`take out`	*knife*	`cutting board`
3	*get*	`cut`	*board*	`knife`
4	*put*	`wash`	*plate*	`counter`
5	*wash*	`take apart`	*drawer*	`drawer`
6	*place*	`add`	*half*	`plate`
7	*rinse*	`shake`	*bowl*	`bowl`
8	*remove*	`screw`	*egg*	`cauliflower`
9	*pan**	`put in`	*orange*	`pan`
10	*peel*	`peel`	*man*	`spice shaker`

Table 7.1: The 10 most frequent verbs, low-level actions, nouns and low-level participants in the TACOS corpus. **pan* is often mis-tagged as a verb in the textual descriptions.

sentence, by considering all sentences with the same associated time frame as equivalent realizations of the same action.

The corpus contains 17,334 action descriptions (tokens), realizing 11,796 different sentences (types). It consists of 146,771 words (tokens), 75,210 of which are content word instances (i.e. nouns, verbs and adjectives). The verb vocabulary comprises 28,292 verb tokens, realizing 435 lemmas. Since verbs occurring in the corpus typically describe actions, we can note that the linguistic variance for the 58 different low-level activities is quite large.

Table 7.1 gives an impression of the action realizations in the corpus, listing the most frequent verbs and nouns from the textual data, along with the most frequent low-level activities and low-level participants. In particular, the most frequent activity (`move`) and the most frequent participant (`hand`) have no counterpart in the list of textual realizations, and the most frequent noun (*person*) does not have a corresponding low-level tag.

The corpus shows how humans vary the granularity of their descriptions, measured in time or number of low-level activities, and it shows how they vary the linguistic realization of the same action. For example, Figure 7.3 contains *dice* and *chop into small pieces* as alternative realizations of the low-level activity sequence `slice - scratch off - slice`. On average, each individual sentence covers 2.7 low-level activities, which indicates a clear difference in granularity between the natural language descriptions and the low-level annotation. 38% of the descriptions correspond to exactly one low-level activity, about a quarter (23%) cover two of them; 16% have 5 or more low-level elements, 2% more than 10.

The descriptions are of varying length (9 words on average), reaching from two-word phrases to detailed descriptions of 65 words. Most sentences are short, consisting of a reference to the person in the video, a participant and an action verb (*The person rinses the carrot*, *He cuts off the two edges*). People often specify an instrument (*from the faucet*), or the resulting state of the action (*chop the carrots in small pieces*). Occasionally, we find more complex constructions (support verbs, coordinations).

As Figure 7.3 indicates, the timestamp-based alignment is pretty accurate; occasional errors occur like *He starts chopping the carrot...* in NL Sequence 3. The data contains some typos and ungrammatical sentences (*He washed carrot*), but for our own experiments, the small number of such errors did not lead to any processing problems.

7.3 The Action Similarity Dataset

In this section, we present a manually annotated standard dataset with action similarity scores, as a basis for the evaluation of visually grounded semantic similarity models. We call it the "Action Similarity Dataset" (ASim) in analogy to the Usage Similarity dataset (USim) of Erk *et al.* (2009) and Erk *et al.* (2012). Similarly to USim, ASim contains a collection of sentence pairs with numerical similarity scores assigned by human annotators. However, we asked the annotators to focus on the similarity of the activities described rather than on assessing semantic similarity in general. We use sentences from the TACoS corpus and record their timestamps. Thus each sentence in the corpus comes with the video segment which it describes (these were not shown to the annotators).

Selecting action description pairs

Random selection of annotated sentences from the corpus would lead to a vast majority of pairs which are completely dissimilar, or in any case difficult to grade (e.g., *He opens the drawer – The person cuts off the ends of the carrot*). We constrained the selection process in two ways: First, we consider only sentences describing activities of manipulating an ingredient. The low-level annotation of the video corpus helps us identify candidate descriptions. We also exclude rare and special activities, ending up with sentences verbalizing the actions `cut`, `slice`, `chop`, `peel`, `take apart`, and `wash`, which occur reasonably frequently, with a wide distribution over different scenarios. Candidate sentence pairs thus are pairs of sentences which each include one of these activities in their timespans. This results in a conceptually more focussed repertoire of descriptions, and at the same time admits full linguistic variation (*wash an apple under the faucet – rinse an apple*, *slice the cucumber – cut the cucumber into slices*).

Second, we require the pairs to share some lexical material, either the head verb or the manipulated object (or both). With "object", we refer to the manipulated ingredient (the main theme from the scenario); we do not require the ingredient term to be the actual grammatical object in the action descriptions, but rather use "object" in its semantic role sense as the entity affected by an action.

In more detail, we composed the ASim dataset from three different subsets:

Same object, different activity: This subset contains pairs describing different types of actions carried out on the same type of object (e.g. *The man washes the carrot. – She dices the carrot.*). Its focus is on the central task of modeling the semantic relation between *actions* (rather than the objects involved in the activity), since the object nouns in the descriptions refer to the same ingredient, and thus the respective video segments show the same type of object, too.

Same object, same activity: Description pairs of this subset are paraphrases in most cases. Sometimes, the head verbs that realize the same activity can differ, which then leads to very similar, but not fully equivalent description pairs. For example, the descriptions realizing `peel [onion]` are almost all equivalent, while the low-level tag `cut [carrot]` can label segments described with *she chops up the carrot*, or *she slices the carrot* or *the carrot is cut in halves*.

Different object, same activity & verb: Description pairs in this subset share head verb and low-level activity, but have different objects (e.g. *The man washes the carrot. – A girl washes an apple under the faucet.*). This dataset enables the exploration of the objects' meaning contribution to the complete action, established by the variation of equivalent actions that are done to different objects.

We assembled 900 action description pairs for annotation: 480 pairs share the object; 240 of which have different activities, and the other 240 pairs share the same activity. We included paraphrases describing the same video segment, but we excluded pairs of identical sentences. 420 additional pairs share their head verb, but have different objects.

Manual annotation

Three native speakers of English were asked to judge the similarity of the action pairs with respect to how they are carried out (rather than how they are described on the surface level). The annotators were explicitly asked to ignore the actor of the action (e.g. whether it is a man or a woman) and to score the similarities of the underlying actions

PART OF GOLD STANDARD	SIM	VAR	ρ
Same object, different activity	2.20	1.15	0.73
Same object, same activity	4.19	1.10	0.73
all with same object	3.20	2.08	0.84
Different object, same activity & verb	3.34	0.48	0.43
complete dataset	3.27	1.33	0.73

Figure 7.4: Average similarity ratings (*Sim*), their variance (*Var*) and annotator agreement (ρ) for ASim.

rather than their verbalizations. For example, their ratings should indicate that *open a can* and *open the recipe book* are very different actions, whereas *putting oil into a pan* and *emptying the olive oil bottle into the bowl* are very similar.

Each subject rated all 900 pairs, with a score from 1 (not similar at all) to 5 (the same or nearly the same). The pairs were shown to them in completely random order, with a different order for each subject.

We compute inter-annotator agreement (and the forthcoming evaluation scores) using Spearman's rank correlation coefficient (ρ), a non-parametric test which is widely used for similar evaluation tasks (Bruni *et al.*, 2011; Erk & McCarthy, 2009). Spearman's ρ evaluates how the single datapoints are ranked relative to each other, but discounts the exact numerical distance between the rankings.

Figure 7.4 shows the average resulting similarity ratings in the different annotation subsets, along with the inter-annotator agreement. The annotators had an average inter-rater agreement of $\rho = 0.73$, with pairwise results of $\rho = 0.77$, 0.72, and 0.69, respectively, which are all highly significant at $p < 0.001$.

As expected, pairs with the same activity and object are rated very similar (4.19) on average, while the similarity of different activities on the same object is the lowest (2.2). For both subsets, inter-annotator agreement is high ($\rho = 0.73$), and even higher for both SAME OBJECT subsets together (0.84).

Pairs with identical head verbs and different objects have a small variance, at 0.48. The inter-annotator agreement on this set is only half as good as for pairs from the SAME OBJECT set. This indicates that similarity assessment for different variants of the same activity is a hard task even for humans.

7.4 Models of Action Similarity

In the following, we demonstrate that visual information contained in videos of the kind provided by the TACoS corpus (Sec. 7.2) substantially contributes to the semantic modeling of action-denoting expressions. In Sec. 7.5, we evaluate several methods for predicting action similarity on the task provided by the ASim dataset. In this section, we describe the models used in that evaluation. We use two different models based on visual information, and in addition two text based distributional models. We employ the latter models to explore the effect of combining linguistic and visual information and to investigate which mode is most suitable for which kinds of similarity.

Text-based models

We use two different models of textual similarity to predict action similarity: a simple word-overlap measure (Jaccard coefficient) and a more refined model based on contextualized vector representations of word meaning (Thater *et al.*, 2011).

Jaccard coefficient: The Jaccard coefficient gives the ratio between the number of (distinct) words common to two input sentences and the total number of (distinct) words in the two sentences. Such simple surface-oriented measures of textual similarity are often used as baselines in related tasks such as recognizing textual entailment (Dagan *et al.*, 2005) and are known to deliver relatively strong results.

Vector model: We use the vector model of Thater *et al.* (2011), which "contextualizes" vector representations for individual words based on the particular sentence context in which the target word occurs. The basic intuition behind this approach is that the words in the syntactic context of the target word in a given input sentence can be used to refine or disambiguate its vector. Intuitively, this allows us to discriminate between different actions that a verb can refer to, based on the different objects of the action.

We first experimented with a version of this vector model which predicts action similarity scores of two input sentences by computing the cosine similarity of the contextualized vectors of the verbs in the two sentences only. We achieved better performance with a variant of this model which computes vectors for the two sentences by summing over the contextualized vectors of all constituent content words.

In the experiments reported below, we only use the second variant. We use the same experimental setup as Thater *et al.* (2011), as well as the parameter settings that are reported to work best in that paper.

Video-based models

We distinguish two approaches to computing the similarity between two video segments. The first approach is unsupervised and extracts a video descriptor before computing similarities between these raw features (Wang *et al.*, 2011). The second (supervised) approach builds upon the first by additionally learning higher level attribute classifiers Rohrbach *et al.* (2012b) on a held out training set. The similarity between two segments is then computed between the classifier responses.

Visual raw feature vectors: We use the state-of-the-art video descriptor "Dense Trajectories" (Wang *et al.*, 2011) which extracts visual video features, namely histograms of oriented gradients, flow, and motion boundary histograms, around densely sampled and tracked points.

This approach is particularly suitable for our dataset as it ignores non-moving parts in the video: we are interested in activities and manipulation of objects, and this type of video feature models only information in activity-related locations ("where something happens"). In previous experiments with videos from MPII Composites, this particular descriptor performed better for automated action recognition than commonly used human pose-based approaches (Rohrbach *et al.*, 2012b). Technically, we use a bag-of-words representation to encode the features in a 16,000 dimensional codebook. We compute the similarity between two encoded features by computing the intersection of the two (normalized) histograms.

Both the features and the codebook are provided in the TACoS corpus data.

Visual classifiers: Visual raw features tend to have several dimensions in the feature space which provide unreliable, noisy values. A supervised approach can help to eliminate noise from the final similarity scores: intermediate level attribute classifiers can learn which feature dimensions are distinctive and thus significantly improve performance over raw features. Rohrbach *et al.* (2012b) show that using such an attribute classifier representation yields far better results for composite activity recognition. The relevant attributes come from the same set of low-level tags of activities and objects that we have in our corpus; the classifier thus learns to model our "interlingua" between visual data and textual video descriptions.

For the experiments reported below we use the same setup as Rohrbach *et al.* (2012b) and use all videos in MPII Composites and MPII Cooking (Rohrbach *et al.*, 2012a), excluding the 127 videos used during evaluation. This results in a 218-dimensional vector of classifier outputs for each video segment. To compute the similarity between two vectors we compute the cosine between them.

	Model	same object	same verb	overall
text	Jaccard	0.28	0.25	0.25
	Textual vectors	0.30	0.25	0.27
	Text combined	0.39	**0.35**	0.36
vision	Visual raw feature vectors	0.53	-0.08	0.35
	Visual classifier	0.60	0.03	0.44
	Vision combined	0.61	-0.04	0.44
mix	*All unsupervised*	0.58	0.32	0.48
	All combined	**0.67**	0.28	**0.55**
	Upper bound	0.84	0.43	0.73

Table 7.2: Evaluation results for textual models, visual models and the multimodal approach in Spearman's ρ. All values > 0.11 are significant at $p < 0.001$.

7.5 Evaluation

We evaluate the different similarity models introduced in Section 7.4 by calculating their correlation with the gold standard similarity annotations of ASim (cf. Section 7.3). For all correlations, we use Spearman's ρ as a measure. We consider the two textual measures (referred to as "Jaccard" and "Textual vectors") and their combination ("Text combined"), as well as the two visual models ("Visual raw feature vectors" and "Visual classifier") and their combination ("Vision combined"). We create a multimodal model from textual and visual features together, in two variants: The first includes all measurements ("All combined"), the second version only the unsupervised components, omitting the visual classifier ("All unsupervised"). For any combination of similarity measures, we simply average their normalized scores (using z-scores).

Table 7.2 shows the scores for all of these measures on the complete ASim dataset (overall), along with the two subparts, where description pairs share either the object (same object) or the head verb (same verb). In addition to the model results, the table also shows the average human inter-annotator agreement as upper bound.

On the complete set, both visual and textual measures have a highly significant correlation with the gold standard, whereas the combination of both leads to the best performance by far (0.55). The results on the same object and same verb subsets shed light on the division of labor between the two information sources. While the textual measures show a comparable performance over the two subsets, there is a dramatic difference in the contribution of visual information. On the same object set, the visual models clearly outperform the textual ones, whereas the visual information has no positive effect on the same verb set. This is clear evidence that the visual model captures genuine action

	Model (Same Object)	*same action*	*different action*
TEXT	Jaccard	0.44	0.14
	Textual vectors	0.42	0.05
	Text combined	**0.52**	0.14
VISION	Visual raw feature vectors	0.21	0.23
	Visual classifier	0.21	**0.45**
	Visual combined	0.26	0.38
MIX	*All unsupervised*	0.49	0.24
	All combined	0.48	0.41
Upper bound		0.73	0.73

Table 7.3: Results for sentences from the same scenario, with either the same or different low-level activity.

similarity rather than the similarity of the participating object.

The numbers shown in Table 7.3 support this hypothesis, showing the two groups in the SAME OBJECT class. For sentences that share the same activity, the textual models seem to be much more suitable than the visual ones. In general, visual models perform better on actions with different activity types, and textual information does better on closely related activities.

Overall, the supervised classifier contributes a good part to the final results. However, the supervision is not strictly necessary to arrive at a significant correlation; the raw visual features alone are sufficient for the main performance gain seen from the integration of visual information.

7.6 Discussion

The TACoS corpus and our grounding approach advance the state of research in several respects. With the TACOS corpus, we provide the first resource that contains both high-resolution recordings and precisely synchronized discourses of sentential event descriptions. In consequence, we can present the first multimodal semantic similarity approach for action descriptions: we have shown that visual and textual features have complementary strengths, and that the combination of both information sources delivers a more accurate semantic model than either of the single modalities taken by itself.

The TACoS corpus builds on an existing collection of high-quality video material, which is restricted to the cooking domain. As a consequence, the corpus covers only a limited inventory of activity types and action verbs. As far as scalability is concerned, it takes only moderate effort to collect corpora containing comparable information: one needs videos of reasonable quality and some sort of alignment with action descriptions. Some data sources even provide such alignments for free, e.g. via subtitles in movies, or descriptions of short video clips that depict just a single action.

The models we applied to our corpus are unsupervised (except for the visual classifier), and thus can be applied without modification to arbitrary domains and action verbs, given that they are about observable activities. There are certainly more refined ways to combine the different modes, as the ones reported by Silberer & Lapata (2012). We leave the optimization of this combination step for future work.

In order to investigate the grounding of given script data, we plan to port the visual information to other event descriptions by paraphrasing. In Chapter 3, we described our data set of event sequence descriptions for several kitchen scenarios. The scenarios in the TACoS corpus are a subset of those in the kitchen ESD corpus, thus we can try to match the event descriptions from our text collections to the grounded action descriptions in the new TACoS corpus (e.g. using the paraphrasing algorithm described in Chapter 4). In that way, we could compute visually grounded script representations.

7.7 Related Work

There are several multimodal approaches to grounding language, in particular in pictures or videos. The Microsoft Video Description Corpus (Chen & Dolan, 2011, MSVD) is the most prominent resource providing videos along with their textual descriptions. It consists of multiple crowd-sourced textual descriptions of short video snippets. The MSVD corpus is much larger than our corpus, but most of the videos are of relatively low quality and therefore too challenging for state-of-the-art video processing to extract relevant information. The videos are typically short and summarized with a single sentence. Our corpus contains coherent textual descriptions of longer video sequences, where each sentence is associated with a particular timeframe.

Another large multimodal resource combining language and (static) visual information resulted from the ESP game (von Ahn & Dabbish, 2004). The dataset contains many images tagged with several one-word labels each.

Gupta *et al.* (2009) present another useful resource: their model learns the alignment of predicate-argument structures with videos and uses the result for action recognition in videos. However, the corpus contains no natural language texts.

The connection between natural language sentences and videos has so far been mostly explored by the computer vision community, where different methods for improving action recognition by exploiting linguistic data have been proposed (Gupta & Mooney, 2010; Motwani & Mooney, 2012; Cour *et al.*, 2008; Tzoukermann *et al.*, 2011; Rohrbach *et al.*, 2012b, among others). Our resource is intended to be used for action recognition as well, but in this paper, we focus on the inverse effect of visual data on language processing.

Feng & Lapata (2010) were the first to enrich topic models for newspaper articles with visual information, by incorporating features from article illustrations. They achieve better results when incorporating the visual information, providing an enriched model that pairs a single text with a picture.

Bruni *et al.* (2011) used the ESP game data to create a visually grounded semantic model. Their results outperform purely text-based models using visual information from pictures for the task of modeling noun similarities. They model single words, and mostly visual features lead only to moderate improvements, which might be due to the mixed quality and random choice of the images. Dodge *et al.* (2012) recently investigated which words can actually be grounded in images at all, producing an automatic classifier for visual words.

An interesting in-depth study by Mathe *et al.* (2008) automatically learnt the semantics of motion verbs as abstract features from videos. The study captures 4 actions with 8-10 videos for each of the actions, and would need perfect object recognition from a visual classifier to scale up.

Steyvers (2010) and later Silberer & Lapata (2012) present an alternative approach to incorporating visual information directly: they use so-called *feature norms*, which consist of human associations for many given words, as a proxy for general perceptual information. Because this model is trained and evaluated on those feature norms, it is not directly comparable to our approach.

The *Restaurant Game* by Orkin & Roy (2009) grounds written chat dialogues in actions carried out in a computer game. While this work is outstanding from the social learning perspective, the actions that ground the dialogues are clicks on a screen rather than real-world actions. The dataset has successfully been used to model determiner meaning (Reckman *et al.*, 2011) in the context of the *Restaurant Game*, but it is unclear how this approach could scale up to content words and other domains.

Chapter 8

A Text-Based Application: Paraphrase Extraction

To compute script representations from event sequence descriptions, we mine event paraphrases in a first step, which enables the generalization over different sequences. In the resulting script representation, each event is represented by a set of its realizations from the input data. These realizations are (scenario-specific) paraphrases of each other, and they may vary widely on the surface level (cf. Chapter 4).

The paraphrase sets come about merely as a by-product of the script mining process, but they are also an interesting resource by themselves. Generic paraphrase collections are commonly recognized as useful resources for applications like document summarization (Barzilay *et al.*, 1999), recognizing textual entailment (Dagan *et al.*, 2005), natural language generation (Zhao *et al.*, 2010; Ganitkevitch *et al.*, 2011), and machine translation (Marton *et al.*, 2009). As a consequence, many methods have been proposed for generating large paraphrase resources (Lin & Pantel, 2001; Szpektor *et al.*, 2004; Dolan *et al.*, 2004).

Our script-centered collection of event paraphrases stems from a very focused set of common sense scenarios, which do not provide enough coverage for a direct text-based application. However, we can apply our sequence-based extraction algorithm to standard texts, and thus yield a much larger and more general paraphrase resource. The core insight here is that sets of event sequence descriptions are actually monolingual comparable corpora, whereas the texts (= the ESDs) all have very similar discourse structures (= the event order). We can then apply the alignment-based model to other corpora of multiple texts that share comparably sequential structures. Similar to the ESD-based approach, we assume that sentences with similar discourse contexts can be paraphrases, even if a semantic similarity model does not consider them to be very similar.

In our experiments, we create a source corpus containing multiple summaries of individual TV show episodes (from the show *House*), which all describe the events of that episode in the same sequential order (the same order in which they showed on TV). Our evaluation results suggest that our system can extract accurate paraphrases, and further that it clearly beats a baseline which does not consider discourse context.

Long sentences are often an impractical unit for paraphrasing, because their information is too specific, and they contain context-dependent references. We thus post-process the paraphrases provided by the sequence alignment and extract so-called *paraphrase fragments*. With paraphrase fragments, we refer to paraphrases on a sub-sentential level, typically phrases or clauses. Our results show that the fragment extraction crucially depends on the precision gained by adding discourse context to the preceding sentence matching step.

This chapter is structured as follows: first we summarize previous methods for paraphrasing and explain how our new method differs from those (Section 8.1). Then we describe the parallel corpus we collected out of TV-show recaps and compare it to the script data used for our other experiments (Section 8.2). In the technical part, we first show how we apply our MSA architecture to these texts and evaluate the results (Section 8.3). The final fragment extraction step is outlined (Section 8.4) and evaluated (Section 8.5) in the following two sections. The chapter concludes with a survey of related work on paraphrasing in general (Section 8.7).

The work described in this chapter was published by Regneri & Wang (2012).

8.1 Paraphrasing from Monolingual Comparable Corpora

In general, comparable and parallel corpora have emerged as advantageous resources for finding paraphrases. Several methods have been proposed to extract paraphrases from different types of source corpora. One line of research uses multilingual corpora and considers multiple (English) translations of one source language. Paraphrase extraction starts then either from given sentence alignments (Barzilay & McKeown, 2001; Bannard & Callison-Burch, 2005) or with reproducing such alignments by matching similar translations (Zhao *et al.*, 2008). In all three approaches, the matched sentences are then post-processed to find smaller equivalent units (which we call paraphrase fragments).

Other systems extract paraphrases from monolingual comparable corpora, which means that they have to find matching sentences without any given alignments to a common pivot. Such sentential paraphrases are usually extracted by grouping sentences according to their surface similarity, e.g. measuring n-gram overlap (Barzilay & Lee, 2003;

Dolan *et al.*, 2004) or BLEU scores (Wang & Callison-Burch, 2011). Discourse structure has only marginally been considered for this task. For example, Dolan *et al.* (2004) extract the first sentences from comparable articles and take them as paraphrases. Deléger & Zweigenbaum (2009) match similar paragraphs in comparable texts in a pre-processing step, which means that they create smaller comparable documents for paraphrase extraction.

Paraphrasing and discourse structure

We believe that discourse structure delivers important information for the extraction of paraphrases, just like the sequential structure does for event sequence descriptions. In our experiments, we analyze recaps of TV shows: summaries of the same episode in fact have the same underlying event sequence (the episode), and thus all have the same sequential linear discourse structure (which mirrors the event order). Consider the following (made-up) examples:

(4) (1.1) House keeps focusing on his aching leg.
(1.2) The psychiatrist suggests him to get a hobby
(1.3) House joins a cooking class.

(5) (2.1) He tells him that the Ibuprofen is not helping with the pain.
(2.2) Nolan tells House to take up a hobby.
(2.3) Together with Wilson he goes to a cookery course.

Read as a whole, it is clear that the two texts describe the same three events, in the same order, and thus, e.g. 1.2 and 2.2 are paraphrases. However, they are not very similar on the surface level, with varying references (*Nolan* vs. *the psychiatrist*; various pronominalizations), different perspectives (cf. 1.1 & 2.1), different levels of granularity (the reference to *Wilson* is omitted in 2.3) and lexical variances (*suggest to get* vs. *tells to take up*, *goes to a cookery course* vs. *joins a cooking class*).

Integrating discourse structure would help to identify the matching sentences: a system that considers the sequence of events in the two texts and compares not only the sentences in isolation, but rather considers their discourse context, would have a much better chance of matching the paraphrases. Further, coreference resolution as a discourse-based preprocessing step would be essential for finding matching paraphrase fragments, e.g. to match phrases like *suggest him* and *tells House*, or *the psychiatrist* and *Nolan*.

System overview

We propose to match sentences using multiple sequence alignment, which considers both their semantic similarity and their position in the discourse. Because sentence pairs are usually too long and too specific to be used in NLP applications, we will then extract smaller paraphrase fragments from the matched sentence pairs. The complete system pipeline is sketched in Figure 8.1:

1. **Create a corpus:** First, we create a comparable corpus of texts with *highly comparable discourse structures*. Complete discourse structures like in the RST Discourse Treebank (Carlson *et al.*, 2002) may be very useful for paraphrase computation, however, they are hard to obtain. Discourse annotation is difficult and work-intensive, and full-blown automatic discourse parsers are neither robust nor very precise. To circumvent this problem, we assemble documents that have parallel sequential discourse structures by default: we compile multiple plot summaries of TV show episodes. The textual order of those summaries typically mirrors the underlying event order of the episodes, in the same sequence in which they happened on screen. We take sentence sequences of recaps as parallel discourse structures.

2. **Extract sentence-level paraphrases:** Our system finds sentence pairs that are either paraphrases themselves, or at least contain paraphrase fragments. This procedure uses our MSA-based architecture, treating the linear discourses like ESDs. For processing event sequence descriptions, we developed a tailor-made semantic similarity measure. In the context of ESDs, this WordNet-based measure outperforms all standard similarity measures, because it is robust enough for the syntactically noisy data, and it circumvents problems of scenario-specific equivalences, which we could not cover with systems trained on standard corpora. In the case of TV show summaries, we deal with standard texts with complete sentences, and less specific vocabulary. This allows us to use a state-of-the-art vector space model to compute semantic similarities for sentence pairs.

3. **Extract paraphrase fragments:** Sentence-level paraphrases are too specific and too long for most applications. Further they contain context-dependent Named Entities (e.g. *House*) or time references. To show that the advantages we gain from using sequential structures indeed carry over to the applicational level, we take a second step and extract finer-grained paraphrase *fragments* from the sentence pairs matched in step 2. The resulting matched phrases should be grammatical and interchangeable regardless of context. We propose and compare different fragment extraction algorithms.

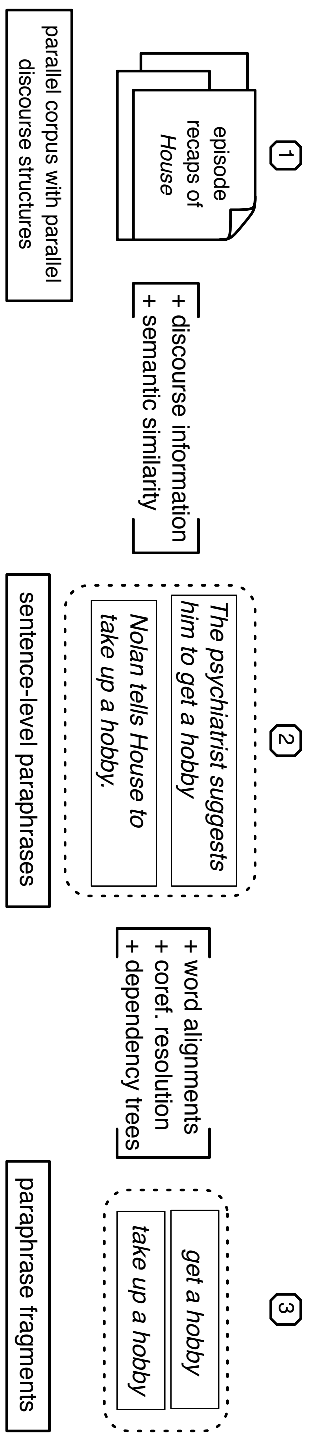

1
episode recaps of *House*
parallel corpus with parallel discourse structures
+ discourse information
+ semantic similarity
2
The psychiatrist suggests him to get a hobby
Nolan tells House to take up a hobby.
sentence-level paraphrases
+ word alignments
+ coref. resolution
+ dependency trees
3
get a hobby
take up a hobby
paraphrase fragments

Figure 8.1: System pipeline for paraphrase extraction.

8.2 A Corpus with Highly Parallel Discourses

Our quest for a suitable parallel corpus was driven by the idea to find texts that not only share the same topic, but also the same narrative structure. We specifically searched for texts of different sources that have the same (external) stimulus, like timelines or live blogs for events of public interest (e.g. the Academy Awards), reports about the same sports match, or recipes for the same dish. For several reasons, summaries of TV shows turned out to be the most promising data source. In the following, we describe the particular advantages of this text type, how we collected the actual data set, and compare the TV show corpus to our corpus of event sequences in terms of complexity (cf. Chapter 3).

TV show summaries as a corpus

Out of all text types we considered, different summaries of the same TV show episodes seemed the most favorable ones:

- **Accessibility:** For various TV shows, there are a good number of different summaries on Wikipedia, IMDB and pages by fans all over the web, which are easily and persistently accessible. This is not the case for e.g. timelines of catastrophes or articles on terrorist attacks, because such articles often disappear after some time (except the ones on Wikipedia). Older newspaper stories are often only available as short executive summaries. More sensitive topics like medical case histories for the same disease are often restricted to registered and authorized users.

- **Variety of domains:** There is a huge variety of TV shows, thus the recaps offer texts of many different more or less specific domains (medical domain, crime-related domains, kitchen domain, daily life...). Episode recaps thus can serve for domain-specific paraphrase extraction as well as being a domain-independent source.

- **Text length:** TV shows (in contrast to longer movies) have a medium length, thus the recaps can be expected to be productive with respect to paraphrase extraction but still convenient to process in terms of tractability.

- **Parallel discourse structures:** The textual order of the recaps is strictly sequential, and it is usually the exact order of the events as they happened on screen. This guarantees that different recaps reflect the same sequential order in their sentence order, and also accounts for a similar length of the recaps. This is not the case e.g. for news reports of any kind, because partial summaries, subjective commentaries or related stories intermingle with the actual event descriptions.

	Complexity measure	*House*	*Scripts*
In a Recap	Average Length	86.67	8.97
	Words per Sentence	15.60	3.60
In an Episode	Type Overlap	0.45	0.29
	Token Overlap	0.44	0.23
	Sense Overlap	0.68	0.34

Table 8.1: Average complexity scores of the *House* summaries and the script corpus.

As the basis for our data collection, we picked the show *House*, a famous show (aired from 2004 – 2012 in the U.S.) that has many fans and thus many different summaries on the web. We collected summaries for all episodes of Season 6. As a preprocessing step, we split each summary into single sentences using the "Splitta"[1] sentence splitter (Gillick, 2009), and treating the result simply as a sequence of sentences. If we interpret the sequential event order as a simple linear discourse structure, the segmented summaries can be treated as event sequence descriptions. Note, however, that the texts are not instances of the same prototypical scenario, but rather different descriptions of the same (scripted) scenario instance. This means that all texts share the same underlying event structure, and we do not expect any reordering in the text sequences.

Corpus statistics

The corpus contains 8 summaries for each of the 20 Episodes in Season 6. This results in 160 documents, containing 14735 sentences. Table 8.1 shows some more detailed statistics that describe the corpus complexity in comparison with the script data. The measures we used are the same we introduced before: the first two features describe the overall complexity of the recaps, and the last three numbers indicate the similarity of summaries of the same episode (cf. Chapter 3 for details):

- **Average length** is the average number of sentences per summary.
- **Words per sentence** describes the average sentence length.
- **Overlap** of types, tokens and senses describes which proportion of their lexical inventory (taken as types, tokens or WordNet senses) two sequences of the same recap share on average.

House episodes have an intermediate length for a regular TV show (40 minutes), resulting in summaries with a mean length of 87 sentences. An average summary is

[1]https://code.google.com/p/splitta/

consequently 10 times longer than an average event sequence descriptions (which have an upper bound of 16 events per sequence). Additionally, the sentences from the recaps are about five times longer than the event descriptions, which meets the intuition about unrestricted sentences compared with the bullet point style. While these two factors could possibly complicate further processing, everything else indicates that the *House* summaries are actually less difficult to handle than the common sense ESDs:

The word overlap features, which were highly indicative for the processing results in the ESDs, is around twice as high as for the ESD corpus (2 times for sense overlap, 1.6 times for type overlap).

Another important factor is reordering. For the ESD corpus, we approximated event reordering by counting how many predicate pairs do not have a fixed sequential order within the scenario corpus. While this was indicative for the short event descriptions, this measure is not suitable for the *House* corpus. For *House*, we have much longer texts, with many more sentences, and most of them contain more than one predicate. Further we have many more support verb constructions and dialogues than in the ESD corpus, and multiple occurrences of *have*, *get* or *say* would all wrongly count as events without sequential order. However, in contrast to the script data, the text type restricts the sequential order. While ESDs describe different variants of executing the same scenario, parallel *House* recaps all describe the same course of events, which is given by the episode as the one common stimulus. Thus reordering is not an issue, because we can safely assume that the events on the screen were seen in the same order by all summary writers.

In our previous evaluation (cf. Chapter 4), we showed that a high type overlap correlates with a very good system performance, while reordering is the biggest obstacle for multiple sequence alignment. Both of these benchmarks are in favor of the *House* corpus, because we have more lexical overlap and virtually no reordering. We also showed earlier that our system works just as well with longer sequences as with shorter ones, so the text length is not a problem at all.

A relevant difference between event descriptions and sentences from recaps are the sentences themselves. *House* sentences are much longer and more complex. When comparing two event descriptions, our hand-crafted similarity measure mostly ignores syntactic structure. This works well for short clauses in bullet-point style, but is unsuitable for longer sentences with multiple clauses. On the other hand, the recaps consist of grammatical sentences with little noise, and the vocabulary is more similar to that found in standard corpora. Thus we can compute sentence similarities with a state-of-the-art vector space model (Thater *et al.*, 2011), which takes syntactic structure into account.

With this one modification, we can apply our system directly to the new corpus.

8.3 Sentential Paraphrase Extraction

In a first step, we apply our MSA-based system to the TV show summaries to match sentential paraphrases. In an intermediate evaluation, we show that the integration of structural discourse information via sequence alignment leads to considerably better results for matching equivalent sentences.

MSA with comparable standard texts

We extract sentential paraphrases with the same system we showed in Section 4.2. However, instead of using the hand-crafted scoring measure, we can apply a state-of-the-art vector space model to assess semantic similarity. To compute alignment match scores for sentence pairs, we use the same syntactically contextualized vector space model (Thater *et al.*, 2011) which we applied to the action descriptions in our multimodal experiments (cf. Chapter 7).

The remainder of the general alignment procedure is the same as the one we applied to event sequence descriptions. In particular, we maintain the progressive alignment that aligns two sequences at a time, always grouping the two most similar sequences of the current set together. Similarity for the alignment order is determined by the *Bag of words aligner*, which takes the lexical overlap of two sequences as similarity indicator (cf. Section 4.2).

Figure 8.2 shows an excerpt from an alignment table produced for the summaries. Each column contains the sentences of a summary, and the sentences of the same row are *aligned*. Gaps denote insertions or deletions, which means that the respective recap does not contain a sentence describing the event denoted in the row (or that the sequence aligner could not match it appropriately).

Evaluation measures and baselines

To evaluate the extraction of sentential paraphrases, we adapt the baselines we used for aligning event sequence descriptions (cf. Chapter 4) and create a new gold standard. We compute precision, recall and f-score with respect to the gold standard taking f-score as follows:

$$f\text{-}score = \frac{2 * precision * recall}{precision + recall}$$

We also compute accuracy as the overall proportion of correct category assignments (paraphrase vs. not paraphrase).

Our main system uses MSA (denoted by MSA afterwards) with vector-based similari-

row	recap 1	recap 2	recap 3	recap 4	recap 5
34	She gives Foreman one shot.	Cuddy tells Foreman he has one chance to prove to her he can run the team.	∅	Cuddy agrees to give him one chance to prove himself.	Foreman insists he deserves a chance and Cuddy gives in, warning him he gets one shot.
35	∅	∅	∅	Foreman, Hadley, and Taub get the conference room ready and Foreman explains that he'll be in charge.	Foreman gives the news to Thirteen and Taub and they unpack the conference room and go with a diagnosis of CRPS.
36	They decide that it might be CRPS and Foreman orders a spinal stimulation.	∅	Foreman says to treat him for complex regional pain syndrome with a spinal stimulation.	∅	∅
			…		
45	Thirteen and Taub go to see the patient, who thinks he has mercury poisoning from eating too much fish.	When they talk to Vince, he tells them that he has been doing research online and believes that he has mercury poisoning.	He suggests they give him a blood test for mercury poisoning.	He thinks he must have mercury poisoning and wants to be tested.	The millionaire has checked on the Internet and believes that he has mercury poisoning caused by sushi.

Table 8.2: Excerpt from an alignment table for 5 sample recaps of Episode 2 (Season 6).

ties (VEC) as a scoring function. The gap costs are optimized for f-score on a held-out episode, resulting in $c_{gap} = 0$. (Remember that gap costs directly influence precision and recall: "cheap" gaps lead to a more restrictive system with higher precision, and more expensive gaps give more recall. As in the script experiment, we choose f-score as our objective.)

To show the contribution of MSA's structural component and compare it to the vector model's contribution, we create a second MSA-based system that uses MSA with BLEU scores (Papineni *et al.*, 2002) as scoring function (MSA+BLEU). BLEU establishes the average 1-to-4-gram overlap of two sentences. The gap costs for this baseline were optimized separately, ending up with $c_{gap} = 1$.

In order to quantify the contribution of the alignment, we create a discourse-unaware baseline by dropping the MSA and using a state-of-the-art clustering algorithm (Noack, 2007) fed with the vector space model scores (CLUSTER+VEC). The algorithm partitions the set of sentences into paraphrase clusters such that the most similar sentences end up in one cluster. This does not require any parameter tuning.

We also show a baseline that uses the clustering algorithm with BLEU scores (CLUSTER+BLEU), which resembles common systems for paraphrase classification based on surface-level similarity. The comparison of this baseline with the other clustering baseline that uses vector similarities helps to underline the vector model's advantage compared to pure word overlap.

We also show the results for completely random label assignment, which constitutes a lower bound for the baselines and for our system.

Gold standard

We aim to create an evaluation set that contains a sufficient amount of genuine paraphrases. Finding such sentence pairs with random sampling and manual annotation is infeasible: There are more than 200,000,000 possible sentence pairs, and we expect fewer than 1% of them to be paraphrases. We thus sample pairs that either the system or one of the baselines has recognized as paraphrases and try to create an evaluation set that is not biased towards the actual system or any of the baselines. The evaluation set consists of 2000 sentence pairs: 400 that the system recognized as paraphrases, 400 positively labelled pairs for each of the three baselines (described in the following section) and 400 randomly selected pairs. For the final evaluation, we compute precision, recall, f-score and accuracy for our main system and each baseline on this set.

Two annotators labelled each sentence pair (S_1, S_2) with one of the following labels:

1. **paraphrases**: S_1 and S_2 refer to exactly the same event(s).

2. **containment**: S_1 contains all the event information mentioned in S_2, but refers to at least one additional event, or vice versa.
3. **related**: S_1 and S_2 overlap in at least one event reference, but both refer to at least one additional event.
4. **unrelated**: S_1 and S_2 do not overlap at all.

This scheme has a double purpose. The main objective is judging whether two sentences contain paraphrases (1-3) or if they are unrelated (4). We use this coarser distinction for system evaluation by collapsing the categories 1-3 into one *paraphrase*$_{coll}$ category. This category is the reference for all evaluation measures: we take a sentence pair as paraphrase if it is a member of *paraphrase*$_{coll}$. Secondly, the annotation shows how many sentences actually fit each other's content perfectly, and what proportion of the matched sentences need further processing to extract the sentence parts with the same meaning.

The inter-annotator agreement according to Cohen's Kappa (Cohen, 1960) is $\kappa = 0.55$ ("moderate agreement"). The distinction between *unrelated* cases and elements of *paraphrase*$_{coll}$ reaches $\kappa = 0.71$ ("substantial agreement"). For the final gold standard, a third annotator resolved all conflict cases.

Among all gold standard sentence pairs, we find 158 *paraphrases*, 238 *containment* cases, 194 *related* ones and 1402 *unrelated*. We had to discard 8 sentence pairs because one of the items was invalid or empty. The high proportion of *unrelated* cases results from the 400 random pairs and the low precision of the baselines. Looking at the paraphrases, 27% of the 590 instances in the *paraphrase*$_{coll}$ category are proper paraphrases, and 73% of them contain additional information that does not belong to the paraphrased part.

Results

Overall, our system extracts 20379 paraphrase pairs. Table 8.3 shows the evaluation results on our gold standard.

The MSA based system variants outperform the two clustering baselines significantly (all levels refer to $p = 0.01$ and were tested with a resampling test (Edgington, 1986)).

The clustering baselines perform significantly better than a random baseline, especially considering recall. The more elaborate vector-space measure even gives 10% more in precision and accuracy, and overall 14% more in f-score. This is already a remarkable improvement compared to the random baseline, and a significant one compared to CLUSTER+BLEU.

Adding structural knowledge with MSA improves the clustering's accuracy performance by 24% (CLUSTER+VEC vs. MSA+VEC), and precision goes up by 39%.

System	*Prec.*	*Recall*	*F-score*	Acc.
RANDOM	0.30	0.49	0.37	0.51
CLUSTER+BLEU	0.35	0.63	0.45	0.54
CLUSTER+VEC	0.40	0.68	0.51	0.61
MSA+BLEU	0.73	**0.74**	**0.73**	0.84
MSA+VEC	**0.79**	0.66	0.72	**0.85**

Table 8.3: Results for sentence matching.

Intuitively we expected the MSA-based systems to end up with a higher recall than the clustering baselines, because sentences can be matched even if their similarity is moderate or low, but their discourse context is highly similar. However, this is only the case for the system using BLEU scores, but not for the system based on the vector space model. One possible explanation lies in picking f-score as the objective for optimization of the MSA gap costs: for the naturally more restrictive word overlap measure, this leads to a more recall-oriented system with a low threshold for aligning sentences, whereas the gap costs for the vector-based system favor a more restrictive alignment with more precise results.

The comparison of the two MSA-based systems highlights the great benefit of using structural knowledge: MSA+BLEU and MSA+VEC have comparable f-scores and accuracy. The advantage from using the vector-space model compared to pure word overlap is nearly evened out when we add discourse knowledge as a backbone. However, the vector model still results in nominally higher precision and accuracy, thus we consider the system using MSA with the vector space model (MSA+VEC) as our main system for the fragment extraction step.

8.4 Extracting Paraphrase Fragments

Taking the output of the sentence alignment as input, we extract shorter phrase-level paraphrases (*paraphrase fragments*) from the matched sentence pairs. The core principle for fragment extraction is to extract phrase pairs that contain continuous spans of pairwise equivalent words, similar to phrase table computation for machine translation (MT). We experiment with different algorithms to find matching phrase pairs of different sizes. In the basic case, we extract chunks of pairwise aligned words. Using the same alignments, we then extend this approach with syntactic information to extract equivalent verbal and prepositional phrases.

Preprocessing

Before extracting paraphrase fragments, we first preprocess all documents as follows:

Stanford CoreNLP [2] provides a set of natural language analysis tools. We use the part-of-speech tagger, the named-entity recognizer, the parser (Klein & Manning, 2003), and the coreference resolution system (Lee *et al.*, 2011). In particular, the parser's dependency structures are used for syntax-based fragment extraction. The output from the coreference resolution system is used to cluster all mentions referring to the same entity and to select one as the *representative* mention. If the representative mention is not a pronoun, we modify the original texts by replacing all coreferent mentions in the cluster with the syntactic head of the representative mention. We apply the coreference resolution to each recap as a whole.

GIZA++ (Och & Ney, 2003) is a widely used word aligner for MT systems. We amend the input data by copying identical word pairs 10 times and adding them as additional 'sentence' pairs (Byrne *et al.*, 2003), in order to emphasize the higher alignment probability between identical words. We run GIZA++ for bi-directional word alignment and obtain a lexical translation table from our training corpus, as well as word alignment probabilities for each sentence pair.

Basic fragment extraction

Following the approach by Wang & Callison-Burch (2011), we base our system on word alignments, for which there are many techniques and tools from machine translation. Wang and Callison-Burch's system is a modified version of an approach by Munteanu & Marcu (2006) on translation fragment extraction. We briefly review the three-step procedure here and refer the reader to the original paper for more details:

1. For each sentence pair, we first retrieve all word-word alignment probabilities from GIZA++ (trained on all documents). We don't use the one best guess that GIZA produces, but rather compute our own alignment based on the likelihoods GIZA computes as a second-to-last step.
2. We smooth the alignment considering the association strength of the neighboring words. For words with 0 alignment probability (i.e. without a match in the other sentence), we set the alignment probability with words in the other sentence to the average alignment probabilities of the adjacent words (in a window size of 4 to both sides). This allows us to match phrases like *your book* and *book of you*, which could not be aligned if *of* had no matched equivalent in the first phrase.

[2] `http://nlp.stanford.edu/software/corenlp.shtml`

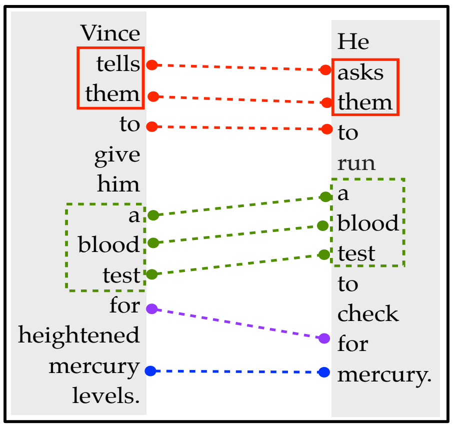


Figure 8.2: Basic paraphrase fragment extraction.

3. In a final step, we extract fragment pairs using different heuristics, e.g., non-overlapping n-grams, chunk boundaries, or dependency trees.

After obtaining a lexical translation table by running GIZA++, we compute an optimal alignment of word pairs by maximizing the sum of alignment probabilities. All word alignments (excluding stop-words) with positive scores are selected as candidate fragment elements. With a modified longest common substring algorithm, we extract pairs of longest common word chunks, with *common* meaning *having alignment links*. Provided with those candidate fragment elements, Wang & Callison-Burch (2011) use a chunker to constrain the boundaries of the output fragments, in order to yield mostly grammatical fragments with phrasal character. We use the same OpenNLP chunker[3] for comparability. We discard trivial fragment pairs, such as string-identical ones or complete sentence pairs that have not been pruned at all.

Figure 8.2 displays the basic fragment extraction. In the picture, the lines denote the alignments computed in step 1 and 2, and the boxes indicate the fragments extracted from those alignments (step 3). The post-processing with a chunker avoids the inclusion of the particle *to* into the first (red) fragment. In the next step, we replace the chunker with a dependency parser to determine the fragment boundaries.

Dependency-based fragment extraction

The chunk-constrained fragment pairs are sometimes ungrammatical, and are often too short to capture meaningful paraphrases. In order to yield a more accurate output, we add another layer of linguistic information to the extraction mechanism: based on

[3]`http://opennlp.sourceforge.net/`

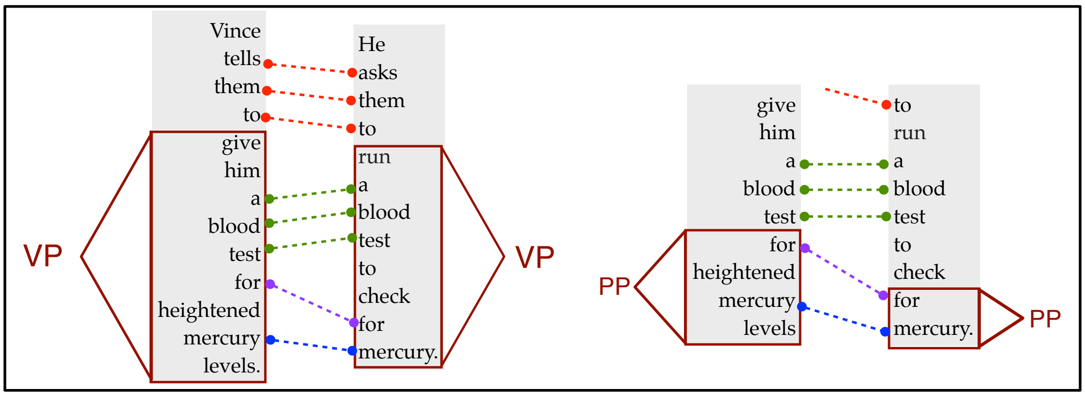

Vince tells them to give him a blood test for heightened mercury levels.
He asks them to run a blood test to check for mercury.
VP
VP
give him a blood test for heightened mercury levels
to run a blood test to check for mercury.
PP
PP

Figure 8.3: Dependency-based fragment extraction.

the dependency parses produced during preprocessing, we extract clauses and prepositional phrases. In both cases, we match the respective head words of the phrases and their complements. More precisely, we match two phrases if their respective subtrees t_1 and t_2 satisfy the following conditions:

- The subtrees mirror a complete subset of the GIZA++ word alignment, i.e., all words aligned to a given word in t_1 are contained in t_2, and vice versa. For empty alignments, we require an overlap of at least one lemma (ignoring stop words).
- The root nodes of t_1 and t_2 have the same roles within their trees, e.g., we match clauses with an `xcomp`-label only with other `xcomp`-labeled clauses. While paraphrases can have different syntactic types (like *what he decided* and *his decision*), we found this constraint useful for reducing the number of fragment pairs that are not equivalent but rather have a one-directional entailment relation (which we called *containment* for sentences).
- We restrict t_1 and t_2 to certain phrase types: they must either contain one verb with at least one complement (typically verb phrases and subsidiary clauses), or they must be prepositional phrases. Verbal (para-)phrases mostly account for single events, and prepositional (para-)phrases are more fine-grained parts of the event.
- Again, trivial pairs that are prefixes or suffixes of each other are excluded.

Figure 8.3 shows a sample sentence pair with two extracted fragment pairs. The algorithm matches the verbal phrases of the sentences as well as the included prepositional phrase. The main advantage of using dependencies lies in the output's grammaticality, because the subtrees always match complete phrases. This method also functions as a filtering mechanism for mistakenly aligned sentences: If only the two sentence root nodes are returned as possible matching partners, the pair is discarded from the results.

8.5 Evaluation of Paraphrase Fragment Extraction

Similar to our intermediate evaluation for sentence alignments, we intrinsically evaluate paraphrase fragments as the final output of our system. Based on a manually annotated sample, we compute precision of the fragment pairs. Because recall for fragment extraction is not defined, we instead describe the *productivity* of the different setups, providing some intuition about the methods' quantitative performance.

We compare several system configurations, and we re-use one of the sentence matching baselines to show the influence of discourse integration on the fragment extraction.

Gold standard

For a gold standard, we randomly sample 150 fragment pairs for each of the five system configurations (explained in the following section). Each fragment pair (f_1, f_2) is annotated with one of the following categories:

1. **paraphrases**: f_1 and f_2 convey the same meaning, i.e., they are well-formed and good matches on the content level.
2. **related**: f_1 and f_2 overlap in their meaning, but one or both phrases have additional unmatched information.
3. **irrelevant**: f_1 and f_2 are unrelated.

This labeling scheme again assesses precision as well as paraphrase granularity. For precision rating, we collapse categories 1 and 2 into one *paraphrase$_{coll}$* category.

Each fragment pair is labelled by two annotators, who were shown both the fragments and their whole source sentences. Overall, the raters had an agreement of $\kappa = 0.67$ ("substantial agreement"), and the agreement for the distinction between the *paraphrase$_{coll}$* categories and *irrelevant* instances reaches a level of $\kappa = 0.88$ (also "substantial agreement"). All conflicts were adjudicated by a third annotator.

The kappa values suggest that the task was easier than sentence level annotation, which was to be expected: first, the fragments are much shorter than the sentences, and thus it is in general easier to judge whether they are equivalent or not. This also shows that the fragment size is a much more suitable and intuitive category for paraphrase extraction in general. Another reason for the better agreement is the simple fact that we dropped one category, because we did not distinguish between entailing and overlapping pairs. The simplification of the annotation scheme resulted from a very pragmatic decision: for the time being, we were not planning to use the entailment information, and the easier annotation task allowed us to meet certain time constraints.

Overall, the gold standard contains 190 paraphrases, 258 *related* pairs and 302 *irrelevant* instances. Unlike previous approaches to fragment extraction, we do not evaluate *grammaticality*, given that the dependency-based method implicitly constrains the output fragments to be complete phrases.

Configurations & results

We take the output of the sentence matching system MSA+VEC (multiple sequence alignment with vector model similarities) as input for paraphrase fragment extraction. As detailed in Section 8.4, our core fragment module uses the word-word alignments provided by GIZA++ and uses a chunker for fragment extraction. We successively enrich this core module with more information, either by longest common substring (LCS) matching or by operating on dependency trees (DEP). In addition, we evaluate the influence of coreference resolution by preprocessing the input texts with coreference resolution (COREF).

We mainly compute precision for this task, as the recall of paraphrase fragments is difficult to define. However, we do include a measure we call *productivity* to indicate the algorithm's completeness. It is defined as the ratio between the number of resulting fragment pairs and the number of sentence pairs used as input.

Extraction Method	*Precision*	*Productivity*
MSA	0.57	**0.76**
MSA+LCS	0.45	0.30
MSA+DEP	0.81	0.42
MSA+DEP+COREF	**0.84**	0.45

Table 8.4: Results of paraphrase fragment extraction.

Table 8.4 shows the evaluation results. The dependency-based heuristic yields the most precise results, and is still more productive than the LCS method. The grammatical filter gives us a higher precision compared to the purely alignment-based approaches. Enhancing the system with coreference resolution raises the score even further.

We cannot directly compare this performance to other systems, as all other approaches use different input data. However, precision is usually manually evaluated, so the figures are at least indicative for a comparison with previous work. The most comparable system is that of Wang & Callison-Burch (2011), which is roughly equivalent to our core module. They report a precision of 0.67 using parallel news data. Another state-of-the-art system introduced by Zhao *et al.* (2008) extracts paraphrase fragments from bilingual parallel corpora, matching English phrases that have the same translations. They also reach a precision of 0.67, on a much larger corpus, but for the considerably easier task of matching equivalent translations. Our approach outperforms both fragment extraction systems by 17%, reaching similar estimated productivity.

As a final comparison, we show how the performance of the sentence matching methods directly affects the fragment extraction. We use the dependency-based fragment extraction system (DEP), and compare the performances by using either the outputs from our main system (MSA+DEP) or alternatively the baseline that replaces MSA with a clustering algorithm (CLUSTER+DEP). Both variants use the vector-based semantic similarity measure.

Sentence matching	*Precision*	*Productivity*
CLUSTER+DEP	0.31	0.04
MSA+DEP	**0.81**	**0.42**

Table 8.5: Impact of MSA on fragment extraction.

As shown in Table 8.5, the precision gain from using MSA becomes tremendous during further processing. We beat the baseline by 50% here, and productivity increases by a factor of 10. This means that the baseline produces on average 0.01 good fragment pairs per matched sentence pair, and the final system extracts 0.3 of them. Those numbers show that for any application that acquires paraphrases of arbitrary granularity, sequential event information provides an invaluable source to achieve a lean paraphrasing method with high precision.

Sample results

Figure 8.6 shows sample results from our system pipeline, using the dependency-based extraction method with full coreference resolution, applied to the sentence pairs computed with sequence alignment (MSA+DEP+COREF). The results reflect the importance of discourse information for this task: sentences are correctly matched in spite of relatively low surface similarity, and differences in syntactic structures. Additionally, the coreference resolution allows us to match the character names *Rachel* (example 1) and *Wilson* (example 5) to the correct corresponding pronouns. All examples show that this fragment matching technique could even help to make coreference resolution better, because we can easily identify *Cameron* with *his wife*, *Lydia* with the respective pronouns, *Nash* with *The Patient* or the nickname *Thirteen* with *Hadley*, the character's actual name.

	Sentence 1 [*with fragment 1*]	**Sentence 2 [*with fragment 2*]**
1	Taub meets House for dinner and claims [*that Rachel had a pottery class*].	Taub shows up for his dinner with House without Rachel, explaining [*that she's at a ceramics class*].
2	House doesn't want her to go and she doesn't want to go either, but [*she can't leave her family*.]	Lydia admits that she doesn't want to leave House but [*she has to stay with her family*].
3	Thirteen is in a cab to the airport when she finds out that [*her trip had been canceled*].	Hadley discovers that [*her reservation has been cancelled*].
4	Nash asks House [*for the extra morphine*].	The patient is ready [*for more morphine*].
5	House comes in to tell Wilson that Tucker has cancer and [*shows him the test results*].	House comes in and [*informs Wilson that the tests have proven positive*]: Tucker has cancer.
6	Foreman tells him [*to confide in Cameron*].	When Chase points out they can't move Donny without alerting Cameron, Foreman tells Chase [*to be honest with his wife*].
7	Thirteen breaks [*into the old residence*] and tells Taub that she realizes that he's been with Maya.	Taub and Thirteen break [*into Ted's former residence*].
8	He finds [*a darkened patch on his right foot near the big toe*].	House finally finds [*a tumorous mole on his toe*].

Table 8.6: Example results; fragments extracted from aligned sentences are bracketed and *emphasized*.

8.6 Discussion

We showed that incorporating discourse information provides important advantages both for extraction of sentential paraphrases and for finding paraphrase fragments within those sentences. For this task, we proposed different summaries of the same TV show episodes as highly parallel input corpora, because they feature both matching content and matching sequential discourse structures.

It is hard to do a direct comparison with state-of-the-art paraphrase *recognition* systems, because most are evaluated on different corpora, e.g., the Microsoft paraphrase corpus (Dolan & Brockett, 2005, MSR). We cannot apply our system to the MSR corpus, because we take complete texts as input, while the MSR corpus solely delivers sentence pairs. While the MSR corpus is larger than our collection, the wording variations in its paraphrase pairs are usually less extensive than for our examples. Thus the final numbers of previous approaches might be vaguely comparable with our results: Das & Smith (2009) present two systems reaching f-scores of 0.82 and 0.83, with a precision of 0.75 and 0.80. Both precision and f-scores of our MSA-based systems lie within the same range. Heilman & Smith (2010) introduce a recall-oriented system, which reaches an f-score of 0.81 by a precision of 0.76. Compared to this system, our approach results in better precision values.

Using multiple sequence alignment as a core component for discourse-based paraphrasing, our approach as it stands is restricted to texts with strict sequential order. One can, however, carry this idea over to more complex discourse structures: while texts do not necessarily have the same sequential structure, they can have comparable discourse structures. Shortly after we proposed this extraction method, Métais *et al.* (2013) showed how to apply a graph-based system to discourse trees.

In follow-up studies, we have begun to enhance the fragment extraction algorithm, too: by using semantic dependencies instead of syntactic ones, we can extract less restricted fragments, even matching phrases with different syntactic categories. At the point of this thesis' publication, this extension is still work in progress.

8.7 Related Work

Previous paraphrase extraction approaches can be roughly characterized under two aspects: 1) data source and 2) granularity of the output.

Both parallel corpora and comparable corpora have been quite well studied. Barzilay & McKeown (2001) use different English translations of the same novels (i.e., monolingual parallel corpora), while others (Quirk *et al.*, 2004) experiment on multiple sources of the same news/events, i.e., monolingual comparable corpora. Commonly used (candidate) comparable corpora are news articles written by different news agencies within a limited time window (Wang & Callison-Burch, 2011). Other studies focus on extracting paraphrases from large bilingual parallel corpora, which the machine translation (MT) community provides in many varieties. Bannard & Callison-Burch (2005) as well as Zhao *et al.* (2008) take one language as the pivot and match two possible translations in the other languages as paraphrases if they share a common pivot phrase. As parallel corpora have many alternative ways of expressing the same foreign language concept, large quantities of paraphrase pairs can be extracted in this way.

The paraphrasing task is also strongly related to cross-document event coreference resolution, which is tackled by string-similarity based techniques, just like most available paraphrasing systems (Bagga & Baldwin, 1999; Tomadaki & Salway, 2005).

As far as output granularity is concerned, most work in paraphrase acquisition has dealt with sentence-level paraphrases, e.g., (Barzilay & McKeown, 2001; Barzilay & Lee, 2003; Dolan *et al.*, 2004; Quirk *et al.*, 2004). Our approach for sentential paraphrase extraction is related to the one introduced by Barzilay & Lee (2003), who also employ multiple sequence alignment (MSA). However, they use MSA at the sentence level rather than at the discourse level.

From an applicational point of view, sentential paraphrases are difficult to use in other NLP tasks, because they are too specific and too long to be matched as a whole opaque unit. At the phrasal level, interchangeable patterns (Shinyama *et al.*, 2002; Shinyama & Sekine, 2003) or inference rules (Lin & Pantel, 2001) are extracted. In both cases, each pattern or rule contains one or several slots, which are restricted to certain type of words, e.g., named entities (NE) or content words. They are quite successful in NE-centered tasks, like information extraction.

The research on *general* paraphrase fragment extraction at the sub-sentential level is mainly based on phrase pair extraction techniques from the MT literature. Munteanu & Marcu (2006) extract sub-sentential translation pairs from comparable corpora using the log-likelihood-ratio of word translation probability. Quirk *et al.* (2007) extract fragments using a generative model of noisy translations. Wang & Callison-Burch (2011) extend the first idea to paraphrase fragment extraction on monolingual parallel and comparable corpora. Our current approach also uses word-word alignment, however, we use syntactic dependency trees to compute *grammatical* fragments. Our use of dependency trees is inspired by the constituent-tree-based experiments of Callison-Burch (2008).

Chapter 9

Tying up some Loose Ends

Ongoing Work and Directions for Future Research

In this thesis, we have presented new approaches to previously unsolved challenges for script mining: we have use crowdsourcing for data collection to overcome the problem of implicitness (Chapter 3), and we apply new context-aware algorithms for event and participant paraphrasing (Chapter 4 & 5). Further, we present more advanced applications, namely grounding event sequences in videos (Chapter 7) and generalizing the paraphrasing algorithms to standard texts (Chapter 8).

The work described in this thesis was carried out within the SMILE (*"**S**cript **M**ining as **I**nternet-based **LE**arning"*) project.[1] While this thesis needed to be finished at some point in time, the research within the SMILE project of course went on: there are still many more script-centered topics and open research questions to investigate. Some techniques and algorithms we have introduced can be extended and enhanced, and some aspects (like the *coverage problem*) do not have a solution at all yet. This chapter presents our own ongoing work and further research directions for different aspects of script processing.

To approach the *coverage problem*, we are working on a collaborative game that will help to scale up our crowdsourcing experiments (Section 9.1). In the same context, we also mine computer games and the web for an appropriate set of scenarios and seed sequences for game-based data collection.

Several other enhancements concern our component for finding *event paraphrases* (Section 9.2): we are currently experimenting with different alignment algorithms which find globally optimal alignments, without requiring strict sequential constraints. Further, we are developing new semantic similarity measures trained on scenario-specific corpora. As a more direct approach to finding equivalent event descriptions, we are working on a game-based mechanism for paraphrasing.

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

With more fine-tuned paraphrasing techniques, we can then also develop *new data structures* for script representation (Section 9.3).

Lastly, we are pursuing the goal of *applying script data for automated action recognition* in videos, for which we already have first results (Section 9.4).

9.1 Coverage

The coverage problem for script learning has two different dimensions (cf. Section 1.3). First, we need to collect large numbers of event sequence descriptions for each scenario to cover as many structural and linguistic variants as possible. The second coverage problem concerns the scenarios themselves: we need a wide-coverage set of seed scenarios for which to collect data, but it is unclear what "wide coverage" means for common sense scripts.

We propose new solutions for both coverage problems: for large-scale script data collection, we develop a collaborative online game that is meant to replace Mechanical Turk as a crowdsourcing platform. To collect seed scenarios, we leverage a wiki-like collection of how-to articles, and additionally extract scenarios from a life simulation game.

A game interface for data collection

We have explained earlier why we think that crowdsoucing is the most suitable way to collect data for common sense scripts (cf. Chapter 3): using crowdsourcing, we can get corpora for arbitrary scenarios by directly asking people about the scenarios we target, thus avoiding the problem of implicitness. In our experiments, we used Amazon Mechanical Turk as a commercial crowdsourcing platform.

In recent years, using games instead of such commercial platforms has become more and more popular. Once developed, a game can run for a long time with manageable maintenance effort, and thus deliver more data while needing much less intervention than similar experiments with Mechanical Turk. As a consequence, game-based data collection scales up much easier than any other crowdsourcing approach – provided the game finds a large audience. Further, motivated participants will play the game for fun, which means that nobody needs to pay them. Having only volunteer players as annotators not only reduces costs, but also reduces the number of fraudulent participants.

A game has also advantages from an ethical point of view. Collecting data with a game is ethically completely unquestionable, which is not the case for Mechanical Turk. While we have paid special attention to fair worker treatment in our experiments, supporting Mechanical Turk with the obligatory fees means supporting an unmoderated platform that often attracts abusive requesters (Fort *et al.*, 2011).

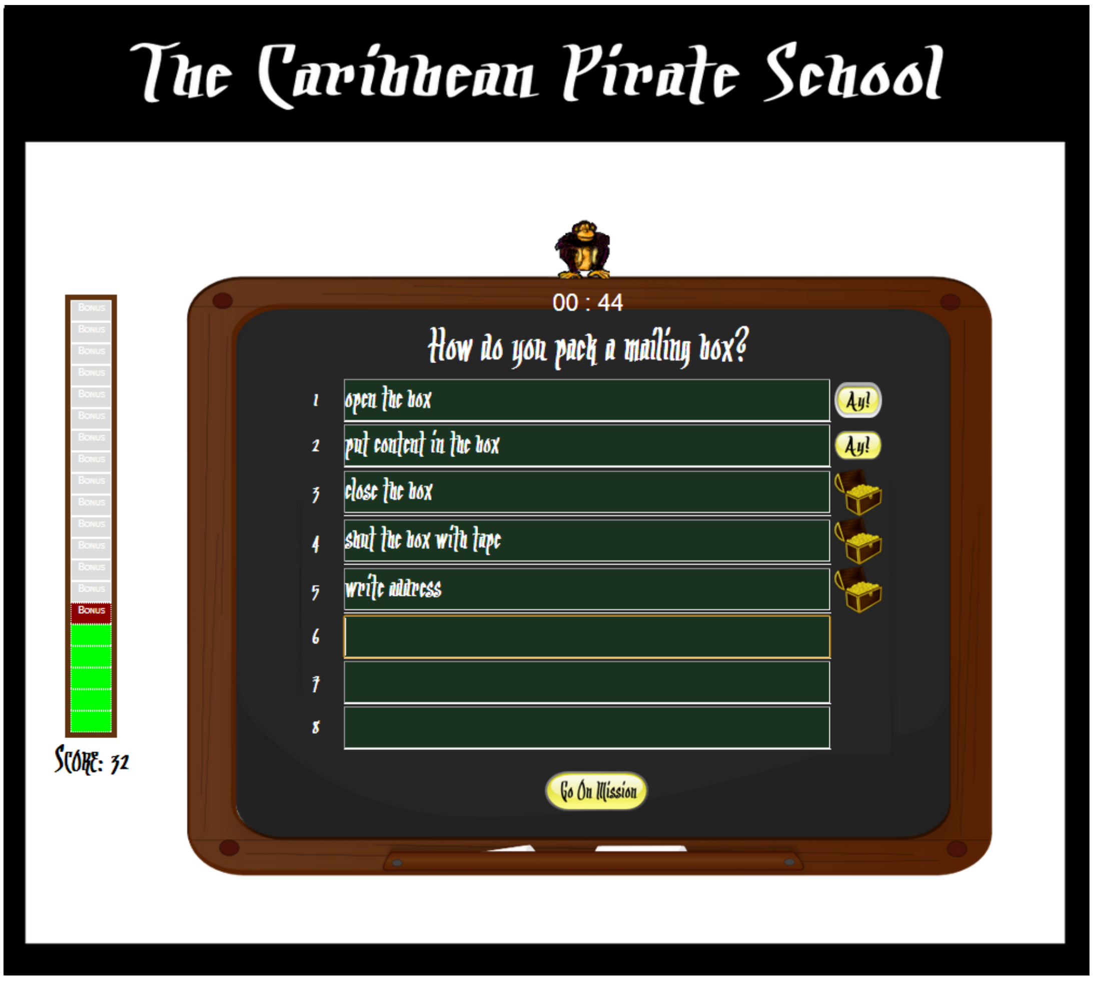
Following earlier examples of crowdsourcing games, we develop a so-called *collaborative game* for collecting event sequence descriptions.[2] In a collaborative game, several people solve a task together rather than in competition. Such games rely on the principles of redundancy and peer-control: if many people answer a question in the same way, the answer is likely to be valid. The ESP Game (von Ahn & Dabbish, 2004) was the first game of this style, in which two people were shown the same image and had to enter a label describing the image's content. The players gained points if they entered the same label. Several other collaborative games were modeled on the principle of the ESP game (von Ahn *et al.*, 2006; Chamberlain *et al.*, 2009).

In the game we are developing, the game story turns around pirates. The pirates have run out of money and need to find jobs among normal (landlubber) people. Being pirates, they are not very civilized and need to learn how to perform the most elementary tasks required for living in a normal society. The necessary skills are taught at the pirate school (Figure 9.1), where a participant is paired with another (fellow pirate) player, and both need to prove their progress by describing how to behave in everyday situations. The questions that they need to answer are of the same sort as the ones we used in the Mechanical Turk setup (e.g. HOW DO YOU MAKE COFFEE?). If both players enter very similar descriptions for one or more sub-events of the scenario, both of them gain extra points (marked with a treasure chest in the picture). Similarity is determined by a WordNet-based measure that considers two event descriptions as equal if one of them contains only words that have a synonym in the other description. (For example *taste coffee* and *savor the fresh coffee* would be scored as matches, because *taste* and *savor* are synonyms in WordNet, and thus the second description contains synonyms for each token of the first description.)

Internally, the game has two modes: the main mode is genuinely collaborative; at least two players need to be online at the same time, and they are randomly paired to play the game together. For the case that an odd number of people (or, in particular, only one player) is logged in, we develop a fallback single-player mode: in this mode an automated simulation replaces the second player, so one participant can also play game without a second (human) partner. To simulate a computer player, we simply match the human player's entries with an ESD that was collected in an earlier game run.

It is a challenging task to develop a game for linguistic data collection that is fun, playable, delivers useful data and consequently has the potential to become popular. A prototype of the game is finished, but to put it into a releasable state, there are still technical issues that take some time to be resolved. Most technical difficulties are related

[2]We worked on concept and implementation together with the students Jonas Sunde, Carolin Shihadeh, Marc Schmit, Uwe Boltz and Nils Bendfeldt.



How do you pack a mailing box?

1 open the box
2 put content in the box
3 close the box
4 shut the box with tape
5 write address
6
7
8

Figure 9.1: Screenshot of the pirate school game interface for collecting event sequence descriptions.

to configuring a reliable synchronization of the two human players in the main mode.

Seed scenarios & sequences

Apart from the need for many event sequence descriptions, there is a second crucial aspect to coverage: our crowdsourcing approach needs a little supervision, because we have to present concrete scenarios to Turkers or players. For creating corpora with Mechanical Turk, we picked the scenarios manually or explored a well-defined domain and took a list of scenarios from the web (cf. Chapter 3). In the long run, we can only scale up our data collection if we find a sufficiently large resource of suitable common sense scenarios to ask people about.

We set up two experiments to get a list of everyday activities that is as comprehensive as possible: we mine actions from a life simulation computer game, and we crawl a huge online collection of how-to articles.

In a first experiment,[3] we collected scenarios from "SIMS 2", a computer game that simulates real life and contains hundreds of everyday activities that the player can take part in (like EATING, ORDERING A PIZZA, SHOWERING, and so on). Our data collection resulted in around 2400 scenarios which vary in their complexity (TAKING A JOB OFFER VS. LIFTING AN OBJECT) as well as in their compatibility with real life (for example, the game features actions involving vampire muffins).

To extend the list we obtained from the SIMS game with more common sense scenarios, we are running a second experiment:[4] we crawl WikiHow,[5] an online collection of how-to instructions for a huge variety of different tasks. WikiHow features more than 150,000 step-by-step instructions for all kinds of activities, ranging from simple everyday tasks (HOW TO WASH YOUR HANDS) to very specialized expert knowledge (HOW TO SET UP A MYSQL SEVER). Compared to all other scenario sources we have found, WikiHow is probably as close to a comprehensive list as we can get. In current experiments, we are testing different filters for the WikiHow data that will separate actual common sense scenarios from advanced expert knowledge.

Another advantage of WikiHow is that it also provides event sequence descriptions: each how-to article is basically one event sequence description for its scenario. Crawling ESDs from WikiHow exclusively is unfortunately insufficient as a basis for script mining, because each ESD in WikiHow provides only one (linguistic and structural) variant of a scenario. However, for our data collection, this data can then serve as seed sequences for the single-player mode of our game, which will later deliver more scenario variants.

[3]Student project by Marc Schmit.
[4]B.Sc. Thesis by Jonas Sunde, ongoing.
[5]`http://www.wikihow.com`

9.2 Optimizing Event Paraphrasing

The first essential step in our script processing pipeline consists in finding paraphrases that describe the same event in different event sequence descriptions, with possibly very different words (cf. Chapter 4). Our current approach uses multiple sequence alignment together with a WordNet-based similarity measure to compute such paraphrases. We see several possibilities for improvements to our algorithm:

First, we enhance the alignment algorithm itself: we want to apply a globally optimal sequence aligner instead of using an alignment tree that only approximates optimal alignment. Further we want to get rid of the strict ordering constraints of MSA in order to deal with reordering in the input sequences.

As a second major enhancement, we are working on new scenario-specific semantic similarity measures that we can combine with the sequence alignment algorithms.

Finally, we show a complementary approach to paraphrasing, in which we are integrating a game interface for paraphrase annotation into our collaborative game.

Optimizing and relaxing MSA

We extend the sequence alignment algorithm in two ways:[6]

First, we want to apply a globally optimal alignment algorithm. In our current setup, the order in which we align the sequences is not optimal, but determined by a rather crude heuristic. First experiments with an optimal MSA algorithm (Althaus *et al.*, 2002) increased precision by up to 10% compared to our simpler heuristic. While this improvement looks promising, the optimal alignment has performance issues when incorporating many sequences at one time (in our experiments, it managed up to 15 ESDs). We are currently developing a cascaded technique that successively feeds the optimal MSA with tractable blocks of sequences, to compute a partially optimal alignment.

A second enhancement tackles an inherent drawback of standard multiple sequence alignment: MSA cannot cope with event reordering, which we encounter in almost every scenario. Take the scenario MAKE COFFEE as an example: some people put the water into the machine before they put the coffee powder in, and some people do it the other way around. Because sequence alignment assumes a strict ordering of all events, we cannot identify all event paraphrases. Consequently, our current algorithm has the worst results for scenarios with much reordering (cf. Section 4.5). We are working on an algorithm that keeps the strong preference to align elements in similar positions in

[6]The algorithms are developed in collaboration with Ernst Althaus (University of Mainz), and partially carried out in a student project by David Przybilla.

the sequences but allows for local reordering at a certain costs. The first results look promising, in particular with respect to recall, but we still need to work out several efficiency and optimization issues.

Independently from MSA, we investigate a different paraphrasing algorithm that has no strict sequential ordering constraints in the first place.[7] This alternative algorithm models the task of finding event paraphrases as a constraint solving problem (CSP), which takes a complementary view on the task compared to the alignment-based approach. The sequence alignment algorithm mainly orders the sequences according to structural constraints and enhances this technique with semantic similarity information. The CSP first groups event descriptions according to their semantic similarity, and then includes structural information on sequence ordering to refine those groupings. The first results look promising, but this approach relies heavily on a precise semantic similarity measure, which is still under development.

In cooperation with the SMILE project, Frermann (2013):[8] developed an approach for probabilistic script modeling. Her method uses hierarchical Bayesian models to compute probabilistic script models featuring events, event orderings and participants. As far as the script mining process is concerned, this system computes event and participant paraphrases synchronously, rather than applying a cascaded approach (like we do). In future work, this paradigm could also be applied for other models integrating both participants and events.

Scenario-specific similarity measures

Our paraphrasing algorithm combines syntactic information (which is inherent in MSA) and a measure of semantic similarity to find equivalent event descriptions. The semantic similarity measure we currently use is hand-crafted for our specific purpose, and solely relies on WordNet. This measure still is superior to all standard measures we tried (word overlap, standard vector space models). Vector space models are usually superior to most other semantic similarity approaches, but in our case, their main problem lies in scenario-specific paraphrases: e.g. the phrases *open the door* and *open the washing machine* are generally not paraphrases, but they are exchangeable within the DO LAUNDRY scenario. This is a common phenomenon in event sequence descriptions, which generic domain-independent vector space models cannot capture. As a consequence, a state-of-the-art vector model trained on a large newspaper corpus (Thater *et al.*, 2011) performed much worse than our hand-crafted measure based on WordNet.

The set of ESDs we have for one scenario is too small to serve as a training corpus for

[7]M. Sc. Thesis by David Przybilla, ongoing.
[8]Lea Frermann (2013): *Generative Modeling of Script Knowledge*. M.Sc. Thesis.

vector models on its own. To tune vector models to recognize scenario-specific similarities, we are working on a way to incorporate scenario-specific training corpora.[9] We have already collected such corpora from the web, using specific search terms from ESDs to retrieve scenario-specific documents. The resulting collection is ranked according to the documents' scenario-specificity. In ongoing work, we use such corpora to develop scenario-specific vector space models. The remaining steps are preprocessing of the noisy web documents to extract the actual textual content, and finally training a vector space model with the cleaned corpora.

A game interface for paraphrasing

With regard to a large-scale data collection, we decided to additionally try an alternative and rather direct approach to event paraphrasing: within the online game for data collection, we developed a second game part that integrates paraphrase annotation into the game.[10] In the pirate game story, the pirates are allowed to go on a pirate mission (with shooting and plundering) from time to time. In the actual interface, two players are playing collaboratively in a shooter. The players have two types of "weapons": one missile is destructive (a bomb or a sword), and the other one fixes things together (a lock). The players synchronously see the same pairs of sentences moving vertically over the screen, and they have to shoot at a chain between each sentence pair (cf. Figure 9.2). If they hit the chain with the bomb, the chain will break, which means that the two sentences aren't paraphrases. If they shoot using the lock, the chain will be fixed together, meaning that the sentences are paraphrases.

The players earn points for each annotation decision they agree on (i.e. if they use the same weapon on the same sentence pair). In order to evaluate the players' reliability and to avoid frauds, we also add pairs from a pre-annotated gold standard and compare the gold annotation with the player decision.

This game setup doesn't allow for comprehensively annotating big datasets. We can, however, try to select the event description pairs carefully, such that we get direct annotations for particularly difficult cases. To realize such a selection process, we could e.g. consider this task as an active learning problem (see Settles (2012) for a survey). Before the game can be released, we have to overcome the same stability issues that we have for the data collection game's multiplayer mode. Further we want to develop a fallback single-player mode for the paraphrasing shooter, too - but at this point in time, it is not clear how we can realize this.

[9]Carolyn Ladda (2012): *Automatic Acquisition of Scenario-Specific Corpora.* B.Sc. Thesis.

[10]Student project by Carolin Shihadeh, Jonas Sunde and Marc Schmit.

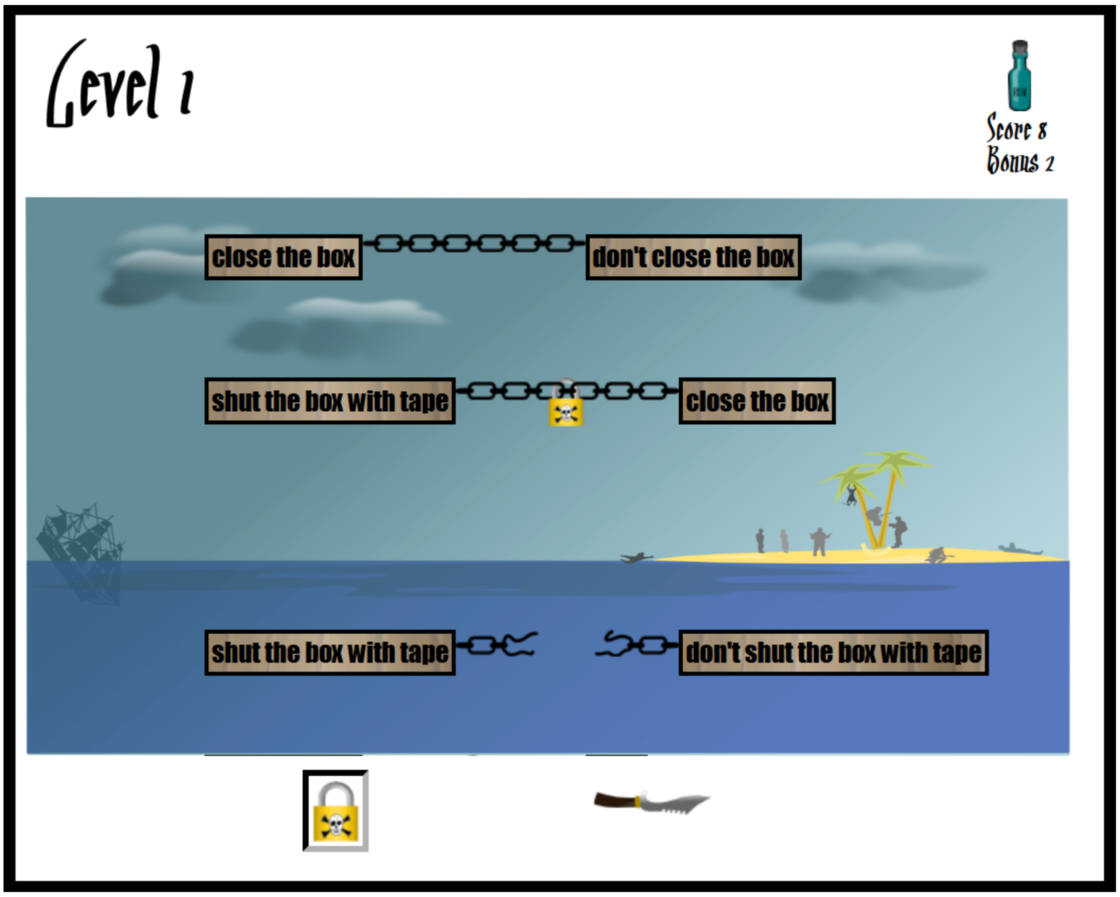


Figure 9.2: Screenshot of the shooter game interface for annotating paraphrases.

1. buy dog food	1. take food from cupboard
2. open can	2. open food container
3. put meat into bowl	3. scoop food in dog dish
4. call the dog	4. call dog
5. trash can	5. put water in bowl
6. fill water bowl	6. dump empty food container
7. watch dog eating	7. dog eats

Figure 9.3: Event paraphrases in two sequences of the FEEDING A DOG scenario.

9.3 Advanced Script Models

Using sets of equivalent event descriptions as a proxy for events enables the generalization over the information from the input data: we can mine events with their possible realizations and valid temporal ordering constraints over them. In the approach described earlier (cf. Chapter 4), we introduce temporal script graphs to model scripts. Temporal script graphs contain sets of event paraphrases as event representations and partial temporal ordering constraints between the events. We decided to use this kind of representation because it is very robust towards mistakes in the paraphrasing step. However, after computing event paraphrases from the input event sequence descriptions, we can also straight-forwardly compute more complex models.

In general, computing more advanced script representation starts with translating the input sequences of *event descriptions* into sequences of *events*, simply by replacing each event description with the set of all its paraphrases (obtained in the event paraphrasing step). From these translated sequences, we can then compute arbitrary compact representations. As a rule of thumb, the robustness of a modeling approach decreases with the complexity of the final representation.

The general problem of mining patterns in general, and graphs in particular, from ordered sequences is a well-researched topic in computer science (cf. Mabroukeh & Ezeife (2010) for an overview). Finite state automata and business process models are two standard ways to represent groups of sequences, but have never been interpreted as script representations before. In this section we show how these representations can be applied as script representations, each with its own advantages and disadvantages.

Finite State Machines

Finite state machines (Rabin & Scott, 1959) are well-researched data structures for representing regular languages. A finite state automaton (FSA) consists of an alphabet, a

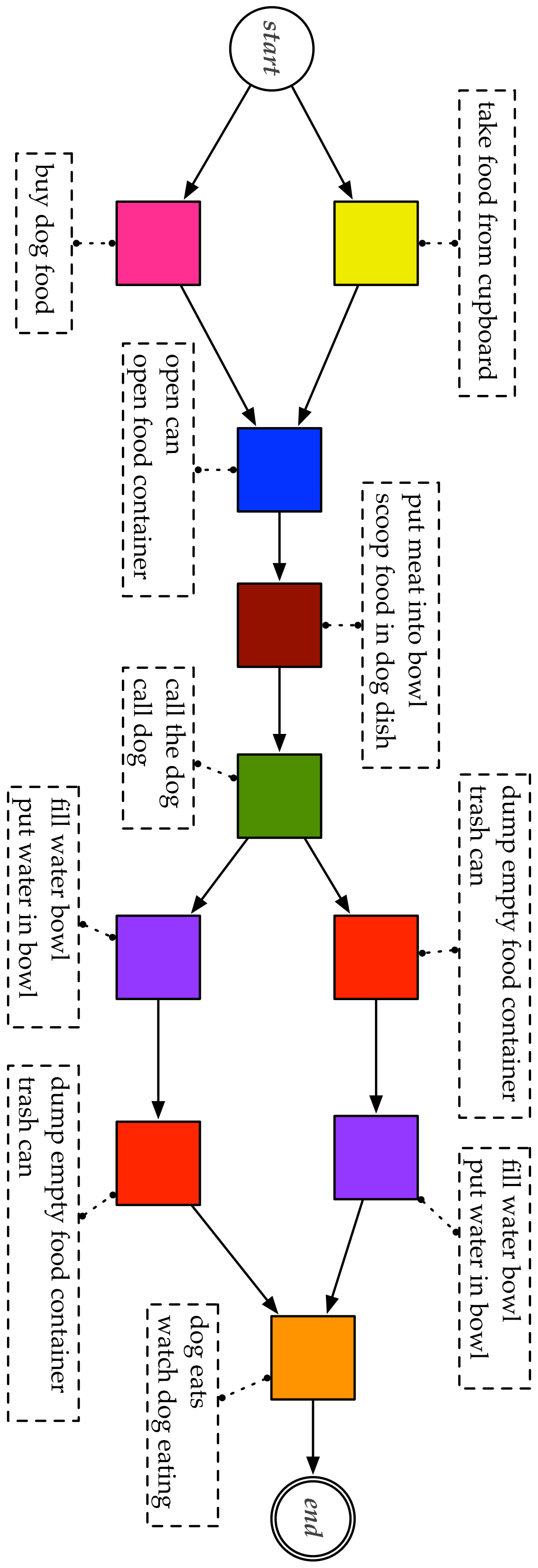


Figure 9.4: A finite state automaton for FEEDING A DOG.

finite set of states with one start state and several final states, and a transition function that defines valid transitions between pairs of states if a certain element of the language is encountered. The language produced by an automaton is the set of words (= concatenations of alphabet elements) that one can extract by traversing all valid paths from the start state to an end state.

Apart from representing (possibly infinite) regular languages, automata can also naturally be used to capture a finite set of sequences. A sample FSA that captures the two sequences for FEEDING A DOG from Figure 9.3 (repeating Figure 1.7) is shown in Figure 9.4. The event description sets are the FSA's alphabet (indicated by the same colors as were used in Figure 9.3), and transitions mark the event ordering.

In order to display the automaton as an event-based graph, we choose a non-standard visualization: the states correspond to events and are labelled with their set of possible realizations. In a standard FSA, the event description sets would be associated with transitions rather than states. We can trivially translate our equivalent representation to the standard variant by moving any state label to the incoming transition of its node and re-naming the states with unique symbols.

The language defined by the example FSA is the set of all *event* courses we can derive from its input data. By translating all valid event sequences into actual event sequence descriptions, we can derive more lexical sequence variants than we put into the FSA: if we have mined the event paraphrases properly, we can also generate sequences with new combinations of event descriptions that inherit sequential constraints from the input data (e.g. by exchanging *put meat into bowl* with *scoop food in dog dish* in the first sequence).

If all paraphrases in the sequences are properly resolved, we can mine an adequate FSA representation of the underlying script by applying standard algorithms. The informativity of the resulting script representation crucially depends on the accuracy of the paraphrases: if no paraphrases are recognized at all (= low recall), we can only infer an automaton that trivially represents the input ESDs. If wrong paraphrases are matched (= low precision), say, *trash empty food container* and *open food container*, this mistake leads to a FSA with invalid temporal constraints, like *open can* can happen after *put food in dog dish*, and the order of *trash can* and *put food in dog dish* would look arbitrary.

Finite state machines are well understood and frequently used in natural language processing, e.g. in computational morphology (Beesley & Karttunen, 2003). They are also extendable to *weighted* automata, which allows to incorporate probabilities for event transitions. A probabilistic automaton e.g. could express how likely it is that somebody *washes carrots* before he or she *peels* them. Such information could be highly useful as prior knowledge for visual action recognition algorithms (cf. Section 9.4).

Business Process Models

Business process mining (BPM) offers more sophisticated techniques and representations that fit our task of script mining extremely well: BPM aims to recover a *business process* from event logs that emerged from execution of the underlying process. Business processes contain everything we need to represent scripts. They consist of events in a certain order, participants, pre- and post-conditions for events and overall goals of the scenario. The mining algorithms developed in this context are tuned to recognize the most general, suitable process that captures all important structural variants of the input data.

A business process (or *workflow*) is a plan defining standardized procedures that a company executes regularly. One example could be CUSTOMER ORDERS A BOOK for an online book seller. Such a process then defines who takes part in the process (e.g. the customer, the sales agent, the packing agent, the delivery service), which resources need to be present (e.g. *online interface*, *book*, *money*) and which events lead to the final goal of the process (e.g. *clicking on a button*, *initiating the payment process*, *processing the customer's order*). Business processes can capture large variance and underspecified constraints, e.g. the customer can *pay* immediately or on delivery, or the customer can optionally *add a gift wrapping*, and it does not matter in which order the *book is retrieved* from the store and the *delivery slip is printed*.

Consequently, we can model scripts as business processes, and thus also profit from the well-researched algorithms (de Medeiros *et al.*, 2003; Dongen *et al.*, 2009) and the available software (van der Aalst *et al.*, 2009) to compute such a business process out of sequences that instantiate the process.

One possibility for representing business processes are Petri Nets, which can represent structural variants like alternatives, concurrent events and loops. Figure 9.5 shows an example workflow representing the two sequences from Figure 1.6. We won't repeat the theory of Petri Nets here, because it is not necessary to understand the mechanisms of the workflow net, but rather give a high-level description of how it works: similar to a finite state automaton, the net has a start node and a final node. The circles represent states, the squares are "task nodes". Arrows represent valid transitions between the nodes. A process can be executed by "sending" elements through it; in this case, the protagonist of the scenario (the *dog owner*) needs to accomplish certain tasks to complete the process. An execution of the process generates one valid sequence of events that instantiates the workflow.

There are several core rules for executing the process represented by a workflow net: 1) everything that comes into a task node will also move out of the node, 2) no task node can be passed by more than one element at a time, and 3) elements can be split (i.e. split

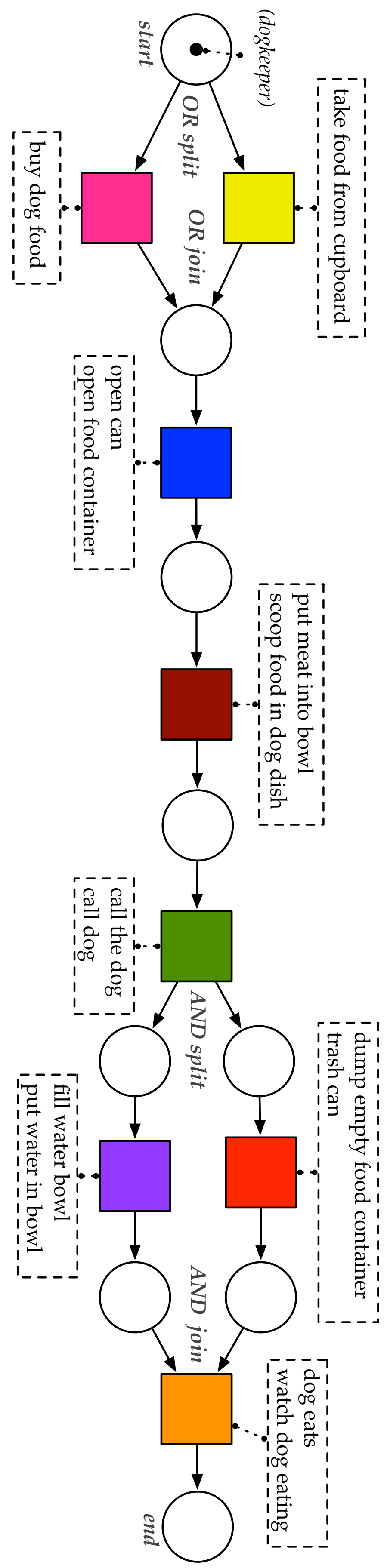


Figure 9.5: A Workflow Net for FEEDING A DOG.

into two states) or joined again (if they have been split before). In consequence, the valid temporal order of tasks can be either linear (like anything between *opening the can* and *calling the dog*), or have alternatives (the *protagonist* can either *buy dog food* or *get some from the cupboard*), and there can also be events that need to be accomplished in arbitrary order, like *filling the water bowl* and *trashing the empty food can*.

Originally, business processes were designed with completely different objectives than e.g. the simpler FSAs: they are not simply models of a language, but also a means to actually execute processes and monitor the state of process completion, and to evaluate the reasons why processes fail. As such they include a more complex machinery than is probably necessary for script processing. In comparison to finite state machines, workflow nets can represent the same "event sequence languages", which means that for generating or validating event sequence descriptions, the two are largely equivalent. However, workflows can directly represent participants, and they offer more compact structures for visualizing scripts.

From the algorithmic point of view, BPM offers great possibilities and powerful tools for work with existing algorithms, but using them would require a lot of source data and perfect algorithms for event and participant paraphrasing.

9.4 Applying Script Data for Video Processing

We have already shown how to ground action descriptions in video data, which leads to accurate multimodal representations of event semantics (cf. Chapter 7). In addition to the benefits this has from semantic point of view, the connection of textual script data and videos can also help visual algorithms. Just like scripts drive humans' expectations about upcoming events, they can also serve as prior knowledge that predicts events in a video: if an algorithm e.g. has recognized a cucumber and a cutting board, script information can predict that an upcoming action like *cut the cucumber* or *peel the cucumber* some short time after. In combination with probabilities, one could also decide whether *cutting* or *peeling the cucumber* should be expected first.

We have some first results that show how scripts can serve as prior knowledge for automated action recognition in videos, reported by Rohrbach *et al.* (2012b). In this study, we applied event sequence descriptions for so-called "zero-shot recognition" of complex events. The algorithm infers which scenario (like PREPARING A CUCUMBER) is shown in a video, based on objects (e.g. *knife*, *cucumber*, *cutting board*) and low-level actions (e.g. *wash*, *grate*) recognized. In particular, this algorithm can recognize scenarios in previously unseen contexts (for which "zero shots" were in the training data): the classifier that recognizes the low-level actions can be trained on other scenarios, because the different objects, ingredients and actions occur in many kitchen-related videos. This classifier can then be applied to arbitrary videos with similar thematic context, and we derive the video's scenario by querying our script data for the recognized actions and objects.

On the linguistic side, we used TF-IDF values to find out which objects and actions are most specific for which scenarios. In the example case, we would e.g. extract the prior knowledge that *cucumber* is very specific for PREPARING A CUCUMBER, and *grating* is also a good indicator that the video shows HOW TO PREPARE A CUCUMBER (and not something that involves a very similar looking *zucchini* e.g.). If these features are then recognized by the visual classifier with a certain confidence, the system can pick PREPARING A CUCUMBER as the best scenario fit.

This scenario recognition algorithm only uses shallow script information and does not consider ordering constraints or associations of certain participants and events. In future work, we plan to use more elaborate script models as prior knowledge for action recognition.

Chapter 10

Conclusions

This final chapter summarizes the main results of this thesis (Section 10.1) and points to possible starting points for future work (Section 10.2).

10.1 Main Contributions

In this thesis, we have presented various new techniques and resources for automated script and event processing. Our results can be divided into two major parts.

The first part of our contributions is centered around a **new system architecture for script mining** (Chapter 2 - 6), and provides new solutions for many script mining challenges. In the second part (Chapter 7 & 8), we have shown two **advanced applications for script data and our script mining algorithms**. First, we have demonstrated how to ground action descriptions in video data to enhance event semantics, and second, how to apply the script-targeted paraphrasing algorithms to standard texts.

Script mining

Our script-mining architecture first uses **crowdsourcing to collect raw script data** (cf. Chapter 3). The crowsdourcing technique is superior to previous approaches that either collect scripts from a few expert annotators, which does not scale up, or try to learn script data from texts, which does not yield sufficient results for common sense scenarios that usually remain implicit. Using Amazon Mechanical Turk, we collected two corpora with raw script data.

To enable generalization over the raw data, we developed **structure-aware paraphrasing algorithms** for event descriptions (Chapter 4) and participants (Chapter 5). These algorithms treat event sequence descriptions for the same scenario as parallel texts in a micro domain. For computing paraphrases, our algorithms use the similar sequential structures of event sequence descriptions as well as semantic similarity information. For both events and participants, the inclusion of structural information allows us to compute more accurate paraphrases and match phrases that have low surface similarity, but are exchangeable within the narrow context of the scenario. Additionally, we have developed preprocessing techniques that exploit the inherent redundancy of the parallel data and thus allow the paraphrasing algorithms to be applied to domains of varying content and language style (Chapter 6).

Applications

For grounding and applying script data, we have taken a multimodal approach and **grounded event descriptions in video data**. As a resulting resource, we have presented the TACoS corpus, which contains videos and synchronized event descriptions. Based on this corpus, we have developed the first genuinely multimodal model for action semantics. In an evaluation against human judgments of action similarity, we have shown that the multimodal model, which combines lexical semantic information with visual features, significantly outperforms each of the individual modalities in isolation.

As a textual application of our structure-aware paraphrasing algorithms, we showed how to use **discourse information for paraphrasing**. We applied the alignment-based system to monolingual comparable texts which have highly parallel sequential discourse structures. The outcome is a new type of parallel corpus along with high-quality sentential paraphrases that have high lexical variance. We also showed that the resulting paraphrases are a good basis for collecting shorter paraphrase fragments, which are more suitable than sentence pairs for end-to-end applications.

10.2 Outlook

This thesis has answered some fundamental research questions around script processing, which had been open since scripts were invented in the seventies. Chapter 9 has discussed some ongoing work on extensions that build directly on the work presented in this thesis. Most importantly, we have shown methods that can produce script collections with high coverage of scenarios and inner-scenario variants.

For a longer-term perspective, we think that the script mining ideas we introduced will have a strong impact on a number of applications:

As soon as script representations are available as a comprehensive resource, they can be applied to diverse text understanding applications, like question answering, resolution of bridging anaphora, textual entailment and processing very short texts, such as microblogs (cf. Section 1.2). Before we can implement full-fledged script-based systems, there is still some work to do, and there are still some open research questions to answer. This thesis lays the groundwork for this future work by introducing basic strategies for mining scripts on a large scale, and by demonstrating their applicability in several concrete systems.

It is our hope that the ideas and methods we have introduced will enable the integration of script processing into the massively data-supported natural language processing of the 21st century.

List of Figures

List of Tables

Bibliography

van der Aalst, Wil M. P., van Dongen, Boudewijn F., Günther, Christian W., Rozinat, Anne, Verbeek, Eric, & Weijters, Ton. 2009. ProM: The Process Mining Toolkit. *In: BPM (Demos)*.

Adams, Lea T., & Worden, Patricia E. 1986. Script development and memory organization in preschool and elementary school children. *Discourse Processes*, **9**(2), 149–166.

von Ahn, Luis, & Dabbish, Laura. 2004. Labeling images with a computer game. *In: Proceedings of SIGCHI 2004*.

von Ahn, Luis, Kedia, Mihir, & Blum, Manuel. 2006. Verbosity: a game for collecting common-sense facts. *In: CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA: ACM.

Althaus, Ernst, Caprara, Alberto, Lenhof, Hans-Peter, & Reinert, Knut. 2002. Multiple sequence alignment with arbitrary gap costs: Computing an optimal solution using polyhedral combinatorics. *In: Proceedings of the European Conference on Computational Biology*.

Althaus, Ernst, Karamanis, Nikiforos, & Koller, Alexander. 2004. Computing Locally Coherent Discourses. *In: Proceedings of ACL 2004*.

Bagga, Amit, & Baldwin, Breck. 1998. Algorithms for Scoring Coreference Chains. *In: Proceedings of LREC 1998*.

Bagga, Amit, & Baldwin, Breck. 1999. Cross-document event coreference: annotations, experiments, and observations. *In: Proceedings of the Workshop on Coreference and its Applications*.

Baker, Collin F., Fillmore, Charles J., & Lowe, John B. 1998. The Berkeley FrameNet Project. *In: Proceedings of ACL 1998*.

Bannard, Colin, & Callison-Burch, Chris. 2005. Paraphrasing with Bilingual Parallel Corpora. *In: Proceedings of ACL 2005*.

BARZILAY, REGINA, & LAPATA, MIRELLA. 2006. Aggregation via Set Partitioning for Natural Language Generation. *In: Proceedings of HLT-NAACL 2006*.

BARZILAY, REGINA, & LEE, LILLIAN. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. *In: Proceedings of HLT-NAACL 2003*.

BARZILAY, REGINA, & MCKEOWN, KATHLEEN R. 2001. Extracting Paraphrases from a Parallel Corpus. *In: Proceedings of ACL 2001*.

BARZILAY, REGINA, MCKEOWN, KATHLEEN, & ELHADAD, MICHAEL. 1999. Information Fusion in the Context of Multi-Document Summarization. *In: Proceedings of ACL 1999*.

BEESLEY, K.R., & KARTTUNEN, L. 2003. *Finite State Morphology*. Studies in Computational Linguistics, 3. Center for the Study of Language and Information Publica Tion.

BEN AHARON, RONI, SZPEKTOR, IDAN, & DAGAN, IDO. 2010. Generating Entailment Rules from FrameNet. *Page 241–246 of: Proceedings of the ACL 2010 Conference Short Papers*. Uppsala, Sweden: Association for Computational Linguistics, for Association for Computational Linguistics.

BERANT, JONATHAN, DAGAN, IDO, & GOLDBERGER, JACOB. 2010. Global learning of focused entailment graphs. *In: Proceedings of ACL 2010*.

BERKELAAR, MICHEL, EIKLAND, KJELL, & NOTEBAERT, PETER. 2004. *lp_solve, a Mixed Integer Linear Programming (MILP) solver Version 5.0*. Website.

BLOEM, JELKE, REGNERI, MICHAELA, & THATER, STEFAN. 2012. Robust processing of noisy web-collected data. *In: Proceedings of KONVENS 2012*.

BRUNI, ELIA, TRAN, GIANG BINH, & BARONI, MARCO. 2011. Distributional semantics from text and images. *In: Proceedings of GEMS 2011*.

BUITELAAR, P., BUITELAAR, P., & CIMIANO, P. 2008. *Ontology Learning and Population: Bridging the Gap between Text and Knowledge - Volume 167 Frontiers in Artificial Intelligence and Applications*. Amsterdam, The Netherlands, The Netherlands: IOS Press.

BURGER, JOHN D., HENDERSON, JOHN, KIM, GEORGE, & ZARRELLA, GUIDO. 2011. Discriminating gender on Twitter. *In: Proceedings of EMNLP 2011*.

BYRNE, W., KHUDANPUR, S., KIM, W., KUMAR, S., PECINA, P., VIRGA, P., XU, P., & YAROWSKY, D. 2003. The Johns Hopkins University 2003 Chinese-English machine translation system. *In: In Proceedings of the MT Summit IX*.

CAI, JIE, & STRUBE, MICHAEL. 2010. Evaluation Metrics For End-to-End Coreference Resolution Systems. *In: Proc. of SIGDIAL 2010*.

Callison-Burch, Chris. 2008. Syntactic Constraints on Paraphrases Extracted from Parallel Corpora. *Pages 196–205 of: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii: Association for Computational Linguistics.

Carlson, Lynn, Marcu, Daniel, & Okurowski, Mary Ellen. 2002. *RST Discourse Treebank*.

Chamberlain, Jon, Poesio, Massimo, & Kruschwitz, Udo. 2009. A demonstration of human computation using the Phrase Detectives annotation game. *In: KDD Workshop on Human Computation*.

Chambers, Nathanael, & Jurafsky, Dan. 2008a. Jointly combining implicit constraints improves temporal ordering. *In: Proceedings of EMNLP 2008*.

Chambers, Nathanael, & Jurafsky, Dan. 2008b. Unsupervised Learning of Narrative Event Chains. *In: Proceedings of ACL 2008: HLT*.

Chambers, Nathanael, & Jurafsky, Dan. 2009. Unsupervised Learning of Narrative Schemas and their Participants. *In: Proceedings of ACL-IJCNLP 2009*.

Chambers, Nathanael, Wang, Shan, & Jurafsky, Dan. 2007. Classifying temporal relations between events. *In: Proceedings of ACL 2007: Interactive Poster and Demonstration Sessions*.

Charniak, E., & Elsner, M. 2009. EM works for pronoun anaphora resolution. *Pages 148–156 of: Proceedings of EACL*. Association for Computational Linguistics.

Chen, David L., & Dolan, William B. 2011. Collecting Highly Parallel Data for Paraphrase Evaluation. *In: Proceedings of ACL 2011*.

Cohen, Jacob. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, **20**(1), 37.

Cour, Timothee, Jordan, Chris, Miltsakaki, Eleni, & Taskar, Ben. 2008. Movie/Script: Alignment and Parsing of Video and Text Transcription. *In: Computer Vision – ECCV 2008*. Springer Berlin Heidelberg.

Coyne, Bob, Rambow, Owen, Hirschberg, Julia, & Sproat, Richard. 2010. Frame semantics in text-to-scene generation. *In: Proceedings of the 14th international conference on Knowledge-based and intelligent information and engineering systems: Part IV*.

Coyne, Robert, & Rambow, Owen. 2009. LexPar: A Freely Available English Paraphrase Lexicon Automatically Extracted from FrameNet. *In: Proceedings of the Third IEEE International Conference on Semantic Computing (ICSC 2009)*.

Cullingford, R. 1981. SAM. *Pages 75–119 of:* Schank, R. C., & Riesbeck, C. K. (eds), *Inside Computer Understanding: Five Programs Plus Miniatures*. Hillsdale, NJ: Erlbaum.

Dagan, Ido, Glickman, Oren, & Magnini, Bernardo. 2005. The PASCAL Recognising Textual Entailment Challenge. *In: Proceedings of MLCW 2005*.

Dahl, Deborah A., Bates, Madeleine, Brown, Michael, Fisher, William, Hunicke-Smith, Kate, Pallett, David, Pao, Christine, Rudnicky, Alexander, & Shriberg, Elizabeth. 1994. Expanding the scope of the ATIS task: the ATIS-3 corpus. *In: Proceedings of the HLT-94*. HLT '94.

Das, D., & Smith, N. A. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. *In: Proceedings of ACL-IJCNLP 2009*.

DeJong, Gerald F. 1982. An Overview of the FRUMP System. *Pages 149–176 of:* Lehnert, Wendy G., & Ringle, Martin H. (eds), *Strategies for Natural Language Processing*. Hillsdale, NJ: Lawrence Erlbaum.

Deléger, Louise, & Zweigenbaum, Pierre. 2009. Extracting Lay Paraphrases of Specialized Expressions from Monolingual Comparable Medical Corpora. *In: Proceedings of the ACL-IJCNLP BUCC-2009 Workshop*.

Denis, Pascal, & Baldridge, Jason. 2007. Joint Determination of Anaphoricity and Coreference Resolution using Integer Programming. *In: Proceedings of HLT-NAACL 2007*.

Dodge, Jesse, Goyal, Amit, Han, Xufeng, Mensch, Alyssa, Mitchell, Margaret, Stratos, Karl, Yamaguchi, Kota, Choi, Yejin, III, Hal Daumé, Berg, Alexander C., & Berg, Tamara L. 2012. Detecting Visual Text. *Pages 762–772 of: HLT-NAACL*.

Dolan, Bill, Quirk, Chris, & Brockett, Chris. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. *In: Proceedings of COLING*.

Dolan, W. B., & Brockett, C. 2005. Automatically constructing a corpus of sentential paraphrases. *In: Proceedings of the 3rd International Workshop on Paraphrasing*.

Dongen, B.F., Alves de Medeiros, A.K., & Wen, L. 2009. Process Mining: Overview and Outlook of Petri Net Discovery Algorithms. *Pages 225–242 of:* Jensen, Kurt, & Aalst, WilM.P. (eds), *Transactions on Petri Nets and Other Models of Concurrency II*. Lecture Notes in Computer Science, vol. 5460. Springer Berlin Heidelberg.

Durbin, Richard, Eddy, Sean, Krogh, Anders, & Mitchison, Graeme. 1998. *Biological Sequence Analysis*. Cambridge University Press.

Edgington, Eugene S. 1986. *Randomization tests*. New York, NY, USA: Marcel Dekker, Inc.

Ellsworth, Michael, & Janin, Adam. 2007. Mutaphrase: Paraphrasing with FrameNet. *In: Proceedings of the Workshop on Textual Entailment and Paraphrasing*. Prague: Association for Computational Linguistics, for Association for Computational Linguistics.

Elsner, Micha, & Charniak, Eugene. 2010. The same-head heuristic for coreference. *Pages 33–37 of: Proceedings of the ACL 2010 Conference Short Papers*. ACLShort '10. Stroudsburg, PA, USA: Association for Computational Linguistics.

Erk, Katrin, & McCarthy, Diana. 2009. Graded Word Sense Assignment. *In: Proceedings of EMNLP 2009*.

Erk, Katrin, McCarthy, Diana, & Gaylord, Nicholas. 2009. Investigations on Word Senses and Word Usages. *In: Proceedings of ACL/AFNLP 2009*.

Erk, Katrin, McCarthy, Diana, & Gaylord, Nick. 2012. Measuring Word Meaning in Context. *CL*.

Fellbaum, Christiane. 1998. *WordNet: An Electronical Lexical Database*. Cambridge, MA: The MIT Press.

Feng, Yansong, & Lapata, Mirella. 2010. Visual Information in Semantic Representation. *In: Proceedings of HLT-NAACL 2010*.

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

Finkel, Jenny Rose, & Manning, Christopher D. 2008. Enforcing transitivity in coreference resolution. *In: Proceedings of ACL 2008: HLT*.

Finlayson, Marc A. 2009. Deriving Narrative Morphologies via Analogical Story Merging. *New Frontiers in Analogy Research*.

Flake, Gary W., Tarjan, Robert E., & Tsioutsiouliklis, Kostas. 2004. Graph Clustering and Minimum Cut Trees. *Internet Mathematics*, **1**(4).

Fort, Karën, Adda, Gilles, & Cohen, K. Bretonnel. 2011. Amazon Mechanical Turk: Gold Mine or Coal Mine? *Computational Linguistics*, **37**(2), 413–420.

Francis, W. N., & Kucera, H. 1979. *Manual of Information to Accompany a Standard Corpus of Present-Day Edited American English, for use with Digital Computers.* Department of Linguistic, Brown University.

Gale, William A., Church, Kenneth W., & Yarowsky, David. 1992. One sense per discourse. *In: Proceedings of the workshop on Speech and Natural Language.*

Ganitkevitch, Juri, Callison-Burch, Chris, Napoles, Courtney, & Van Durme, Benjamin. 2011. Learning Sentential Paraphrases from Bilingual Parallel Corpora for Text-to-Text Generation. *Pages 1168–1179 of: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing.* Edinburgh, Scotland, UK.: Association for Computational Linguistics.

Ganitkevitch, Juri, VanDurme, Benjamin, & Callison-Burch, Chris. 2013. PPDB: The Paraphrase Database. *In: Proceedings of NAACL 2013.*

Gardent, Claire, & Striegnitz, Kristina. 2001. Generating Indirect Anaphora. *In: Proceedings of IWCS-4.*

Gillick, Dan. 2009. Sentence boundary detection and the problem with the U.S. *Pages 241–244 of: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers.* NAACL-Short '09. Stroudsburg, PA, USA: Association for Computational Linguistics.

Glenberg, A. M. 2002. Grounding language in action. *Psychonomic Bulletin & Review.*

Gordon, Andrew S. 2001. Browsing image collections with representations of commonsense activities. *JASIST*, **52**(11).

Gordon, Andrew S., & Swanson, Reid. 2009 (May). Identifying Personal Stories in Millions of Weblog Entries. *In: Third International Conference on Weblogs and Social Media, Data Challenge Workshop.*

Graesser, Arthur C., Gordon, Sallie E., & Sawyer, John D. 1979. Recognition memory for typical and atypical actions in scripted activities: Tests of a script pointer + tag hypothesis. *Journal of Verbal Learning and Verbal Behavior*, **18**(3), 319 – 332.

Gupta, Abhinav, Srinivasan, Praveen, Shi, Jianbo, & Davis, Larry S. 2009. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. *In: Proceedings of CVPR 2009.*

Gupta, Sonal, & Mooney, Raymond J. 2010 (July). Using Closed Captions as Supervision for Video Activity Recognition. *Pages 1083–1088 of: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2010).*

Hajishirzi, Hannaneh, & Mueller, Erik T. 2012. Question Answering in Natural Language Narratives Using Symbolic Probabilistic Reasoning. *In: FLAIRS Conference.*

Heilman, Michael, & Smith, Noah A. 2010. Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. *Pages 1011–1019 of: Proceedings of NAACL-HLT.*

Higgins, Desmond G., & Sharp, Paul M. 1988. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, **73**(1).

Howell, Steve R., Jankowicz, Damian, & Becker, Suzanna. 2005. A model of grounded language acquisition: Sensorimotor features improve lexical and grammatical learning. *JML*.

Hudson, Judith A., Fivush, Robyn, & Kuebli, Janet. 1992. Scripts and episodes: The development of event memory. *Applied Cognitive Psychology*, **6**(6), 483–505.

Jones, Dominic R., & Thompson, Cynthia A. 2003. Identifying events using similarity and context. *In: Proceedings of CoNNL-2003.*

Klein, Dan, & Manning, Christopher D. 2003. Accurate unlexicalized parsing. *In: Proceedings of ACL 2003.*

Kolomiyets, Oleksandr, & Moens, Marie-Francine. 2011. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, **181**(24), 5412 – 5434.

Lee, Heeyoung, Peirsman, Yves, Chang, Angel, Chambers, Nathanael, Surdeanu, Mihai, & Jurafsky, Dan. 2011. Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task. *In: Proceedings of the CoNLL-2011 Shared Task.*

Lee, Heeyoung, Recasens, Marta, Chang, Angel, Surdeanu, Mihai, & Jurafsky, Dan. 2012. Joint Entity and Event Coreference Resolution across Documents. *In: Proceedings of EMNLP-CoNLL 2012.*

Lin, Dekang. 1998. An Information-Theoretic Definition of Similarity. *Pages 296–304 of: ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Lin, Dekang, & Pantel, Patrick. 2001. DIRT - Discovery of Inference Rules from Text. *In: Proceedings of the ACM SIGKDD.*

Lytinen, Steven L. 1992. Conceptual dependency and its descendants. *Computers & Mathematics with Applications*, **23**(2–5), 51 – 73.

Mabroukeh, Nizar R., & Ezeife, C. I. 2010. A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys (CSUR)*, **43**(1).

MANSHADI, MEHDI, SWANSON, REID, & GORDON, ANDREW S. 2008. Learning a Probabilistic Model of Event Sequences from Internet Weblog Stories. *In: Proceedings of the 21st FLAIRS Conference.*

MARCUS, MITCHELL P., MARCINKIEWICZ, MARY ANN, & SANTORINI, BEATRICE. 1993. Building a large annotated corpus of English: the penn treebank. *Comput. Linguist.*, **19**.

MARTON, YUVAL, CALLISON-BURCH, CHRIS, & RESNIK, PHILIP. 2009. Improved Statistical Machine Translation Using Monolingually-Derived Paraphrases. *Pages 381–390 of: EMNLP.*

MATHE, S., FAZLY, A., DICKINSON, S., & STEVENSON, S. 2008. Learning the abstract motion semantics of verbs from captioned videos. *In: Proceedings of SLAM08.*

MCTEAR, MICHAEL. 1987. *The Articulate Computer.* Cambridge, MA, USA: Blackwell Publishers, Inc.

DE MEDEIROS, A.K.A., VAN DER AALST, W.M.P., & WEIJTERS, A. J. M. M. 2003. Workflow Mining: Current Status and Future Directions. *In: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, volume 2888 of Lecture Notes in Computer Science*Springer-Verlag, for Springer-Verlag.

MÉTAIS, ELISABETH, MEZIANE, FARID, SARAEE, MOHAMAD, SUGUMARAN, VIJAYAN, VADERA, SUNIL, BACH, NGOXUAN, MINH, NGUYEN, & SHIMAZU, AKIRA. 2013. *Lecture Notes in Computer Science.* Vol. 7934. Springer Berlin Heidelberg. Pages 65–76.

MINSKY, M.L. 1974. *A Framework for Representing Knowledge.* Artificial intelligence memo. Defense Technical Information Center.

MOONEY, RAYMOND J. 1990. Learning Plan Schemata From Observation: Explanation-Based Learning for Plan Recognition. *Cognitive Science*, **14**(4).

MOTWANI, TANVI S., & MOONEY, RAYMOND J. 2012 (August). Improving Video Activity Recognition using Object Recognition and Text Mining. *Pages 600–605 of: Proceedings of the 20th European Conference on Artificial Intelligence (ECAI-2012).*

MUELLER, ERIK T. 1998. *Natural Language Processing with Thought Treasure.* Signiform.

MUELLER, ERIK T. 2000. A database and lexicon of scripts for thoughttreasure. *Computing Research Repository (CoRR)*, **2000**, Article No. 0003004.

MUNTEANU, DRAGOS STEFAN, & MARCU, DANIEL. 2006. Extracting Parallel Sub-Sentential Fragments from Non-Parallel Corpora. *In: Proceedings of ACL.*

Narayanan, Srini, & Harabagiu, Sanda M. 2004. Question Answering Based on Semantic Structures. *In: COLING.*

Navigli, Roberto. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, **41**(2), 10:1–10:69.

Needleman, Saul B., & Wunsch, Christian D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, **48**(3).

Noack, Andreas. 2007. Energy Models for Graph Clustering. *J. Graph Algorithms Appl.*, **11**(2), 453–480.

Och, Franz Josef, & Ney, Hermann. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, **29**(1).

Orkin, Jeff, & Roy, Deb. 2009. Automatic learning and generation of social behavior from collective human gameplay. *In: AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Pak, Alexander, & Paroubek, Patrick. 2010. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. *In:* Chair), Nicoletta Calzolari (Conference, Choukri, Khalid, Maegaard, Bente, Mariani, Joseph, Odijk, Jan, Piperidis, Stelios, Rosner, Mike, & Tapias, Daniel (eds), *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA).

Palmer, M.S., Gildea, D., & Xue, N. 2010. *Semantic Role Labeling*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Papineni, Kishore, Roukos, Salim, Ward, Todd, & Zhu, Wei-Jing. 2002. BLEU: a method for automatic evaluation of machine translation. *In: Proceedings of ACL.*

Pennacchiotti, Marco, & Popescu, Ana-Maria. 2011. Democrats, republicans and starbucks afficionados: user classification in twitter. *Pages 430–438 of: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '11. New York, NY, USA: ACM.

Quirk, Chris, Brockett, Chris, & Dolan, William B. 2004. Monolingual Machine Translation for Paraphrase Generation. *In: Proceedings of the 2004 Conference on Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

QUIRK, CHRIS, UDUPA, RAGHAVENDRA, & MENEZES, ARUL. 2007. Generative Models of Noisy Translations with Applications to Parallel Fragment Extraction. *In: Proceedings of MT Summit XI.*

RABIN, M. O., & SCOTT, D. 1959. Finite automata and their decision problems. *IBM J. Res. Dev.*, **3**(2), 114–125.

RAGHUNATHAN, KARTHIK, LEE, HEEYOUNG, RANGARAJAN, SUDARSHAN, CHAMBERS, NATHANAEL, SURDEANU, MIHAI, JURAFSKY, DAN, & MANNING, CHRISTOPHER. 2010. A multi-pass sieve for coreference resolution. *Pages 492–501 of: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. EMNLP '10. Stroudsburg, PA, USA: Association for Computational Linguistics.

RASKIN, VICTOR. 1985. *Semantic mechanisms of humor*. Synthese language library ; 24. Dordrecht [u.a.]: Reidel.

RAU, LISA F., JACOBS, PAUL S., & ZERNIK, URI. 1989. Information extraction and text summarization using linguistic knowledge acquisition. *Information Processing and Management*, **25**(4), 419 – 428.

RECKMAN, HILKE, ORKIN, JEFF, & ROY, DEB. 2011. Extracting aspects of determiner meaning from dialogue in a virtual world environment. *In: Proceedings of CCS 2011*. IWCS '11.

REGNERI, MICHAELA, & WANG, RUI. 2012. Using Discourse Information for Paraphrase Extraction. *In: Proceedings of EMNLP-CoNLL 2012.*

REGNERI, MICHAELA, KOLLER, ALEXANDER, & PINKAL, MANFRED. 2010. Learning Script Knowledge with Web Experiments. *In: Proceedings of ACL 2010.*

REGNERI, MICHAELA, KOLLER, ALEXANDER, RUPPENHOFER, JOSEF, & PINKAL, MANFRED. 2011. Learning Script Participants from Unlabeled Data. *In: Proceedings of RANLP 2011.*

REGNERI, MICHAELA, ROHRBACH, MARCUS, WETZEL, DOMINIKUS, THATER, STEFAN, SCHIELE, BERNT, & PINKAL, MANFRED. 2013. Grounding Action Descriptions in Videos. *Transactions of the Association for Computational Linguistics (TACL)*, **1**, 25–36.

ROHRBACH, MARCUS, STARK, MICHAEL, SZARVAS, GYÖRGY, GUREVYCH, IRYNA, & SCHIELE, BERNT. 2010. What Helps Where – And Why? Semantic Relatedness for Knowledge Transfer. *In: CVPR.*

ROHRBACH, MARCUS, AMIN, SIKANDAR, ANDRILUKA, MYKHAYLO, & SCHIELE, BERNT. 2012a. A Database for Fine Grained Activity Detection of Cooking Activities. *In: Proceedings of CVPR 2012.*

Rohrbach, Marcus, Regneri, Michaela, Andriluka, Micha, Amin, Sikandar, Pinkal, Manfred, & Schiele, Bernt. 2012b. Script Data for Attribute-based Recognition of Composite Activities. *In: Proceedings of ECCV 2012*.

Ruiz, Eduardo J., Hristidis, Vagelis, Castillo, Carlos, Gionis, Aristides, & Jaimes, Alejandro. 2012. Correlating financial time series with micro-blogging activity. *Pages 513–522 of: Proceedings of the fifth ACM international conference on Web search and data mining*. WSDM '12. New York, NY, USA: ACM.

Schank, R.C., Treusch, B., Lestuzzi, F., Rova, S., Goldman, N.M., Riesbeck, C.K., Charniak, E., Debiasi, G.B., Mioni, A.M., Galassi, R., *et al.* 1974. *Working Papers: Causality and reasoning*. Working Papers. Istituto per gli Studi Semantici e Cognitivi.

Schank, Roger C. 1999. *Dynamic memory revisited*. Cambridge [u.a.]: Cambridge Univ. Press.

Schank, Roger C., & Abelson, Robert P. 1975. Scripts, plans, and knowledge. *Pages 151–157 of: Proceedings of the 4th international joint conference on Artificial intelligence - Volume 1*. IJCAI'75. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Schank, Roger C., & Abelson, Robert P. 1977. *Scripts, Plans, Goals and Understanding*. Hillsdale, NJ: Lawrence Erlbaum.

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

Schierle, M., Schulz, S., & Ackermann, M. 2008. From spelling correction to text cleaning–using context information. *Data Analysis, Machine Learning and Applications*, 397–404.

Seitz, Aaron R, & Watanabe, Takeo. 2009. The phenomenon of task-irrelevant perceptual learning. *Vision Res*, **49**(21), 2604–10.

Settles, Burr. 2012. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Shen, Dan, & Lapata, Mirella. 2007. Using Semantic Roles to Improve Question Answering. *Pages 12–21 of: Proceedings of EMNLP-CoNLL 2007*.

Shinyama, Yusuke, & Sekine, Satoshi. 2003. Paraphrase acquisition for information extraction. *In: Proc. of the ACL PARAPHRASE '03 Workshop*.

Shinyama, Yusuke, Sekine, Satoshi, & Sudo, Kiyoshi. 2002. Automatic paraphrase acquisition from news articles. *In: Proc. of HLT 2002*.

Silberer, Carina, & Lapata, Mirella. 2012. Grounded Models of Semantic Representation. *In: Proceedings of EMNLP-CoNLL 2012*.

Singh, Push, Lin, Thomas, Mueller, Erik T., Lim, Grace, Perkins, Travell, & Zhu, Wan L. 2002. Open Mind Common Sense: Knowledge Acquisition from the General Public. *In: On the Move to Meaningful Internet Systems - DOA, CoopIS and ODBASE 2002*. London, UK: Springer-Verlag.

Smith, Dustin, & Arnold, Kenneth C. 2009. Learning hierarchical plans by reading simple English narratives. *In: Proceedings of the Commonsense Workshop at IUI-09*.

Snow, Rion, O'Connor, Brendan, Jurafsky, Daniel, & Ng, Andrew Y. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. *In: Proceedings of EMNLP 2008*.

Stefanowitsch, A., & Gries, S.T. 2006. *Corpus-based Approaches to Metaphor And Metonymy*. Mouton select. Mouton de Gruyter.

Steyvers, Mark. 2010. Combining feature norms and text data with topic models. *Acta Psychologica*, **133**(3), 234 – 243.

Suchanek, Fabian M., Kasneci, Gjergji, & Weikum, Gerhard. 2007. Yago: A Core of Semantic Knowledge. *In: 16th international World Wide Web conference (WWW 2007)*. New York, NY, USA: ACM Press.

Swallow, Khena M, & Zacks, Jeffrey M. 2008. Sequences learned without awareness can orient attention during the perception of human activity. *Psychon Bull Rev*, **15**(1), 116–22.

Swanson, Reid, & Gordon, Andrew S. 2008. Say Anything: A Massively Collaborative Open Domain Story Writing Companion. *In: Proceedings of ICIDS 2008*.

Szpektor, Idan, Tanev, Hristo, Dagan, Ido, & Coppola, Bonaventura. 2004. Scaling Web-based Acquisition of Entailment Relations. *Pages 41–48 of:* Lin, Dekang, & Wu, Dekai (eds), *Proceedings of EMNLP 2004*. Barcelona, Spain: Association for Computational Linguistics.

Thater, Stefan, Fürstenau, Hagen, & Pinkal, Manfred. 2011. Word Meaning in Context: A Simple and Effective Vector Model. *In: Proc. of IJCNLP 2011*.

Tomadaki, Eleftheria, & Salway, Andrew. 2005. Matching Verb Attributes for Cross-Document Event Coreference. *In: Proc. of the Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*.

Turney, Peter D., & Pantel, Patrick. 2010. From Frequency to Meaning. Vector Space Models for Semantics. *JAIR*.

Tzoukermann, E., Neumann, J., Kosecka, J., Fermuller, C., Perera, I., Ferraro, F., Sapp, B., Chaudhry, R., & Singh, G. 2011. Language Models for Semantic Extraction and Filtering in Video Action Recognition. *In: AAAI Workshop on Language-Action Tools for Cognitive Artificial Agents*.

Verberne, Suzan, Boves, Lou, Oostdijk, Nelleke, & Coppen, Peter-Arno. 2010. What Is Not in the Bag of Words for Why-QA? *Computational Linguistics*, **36**(2).

Vondrick, Carl, Patterson, Donald, & Ramanan, Deva. 2012. Efficiently Scaling up Crowdsourced Video Annotation. *IJCV*.

Wang, Heng, Kläser, Alexander, Schmid, Cordelia, & Liu, Cheng-Lin. 2011. Action Recognition by Dense Trajectories. *In: Proceedings of CVPR 2011*.

Wang, Rui, & Callison-Burch, Chris. 2011. Paraphrase Fragment Extraction from Monolingual Comparable Corpora. *In: Proc. of the ACL BUCC-2011 Workshop*.

Wolsey, Laurence. 1998. *Integer programming*. Wiley-Interscience.

Zacks, Jeffrey M, Speer, Nicole K, & Reynolds, Jeremy R. 2009. Segmentation in reading and film comprehension. *Journal of Experimental Psychology: General*, **138**(2).

Zhao, Shiqi, Wang, Haifeng, Liu, Ting, & Li, Sheng. 2008. Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora. *In: Proceedings of ACL*.

Zhao, Shiqi, Wang, Haifeng, Lan, Xiang, & Liu, Ting. 2010. Leveraging Multiple MT Engines for Paraphrase Generation. *Pages 1326–1334 of: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Beijing, China: Coling 2010 Organizing Committee.

Appendix A

Details of the Event Sequence Corpus

We provide additional analyses of the event sequence description corpus discussed in Chapter 3:

- **Table A.1** shows the the most frequent content words for each scenario. Content words comprise adjectives, nouns and verbs. The table shows that the content word lexicon is partially very scenario-specific (e.g. *iron*), partially shared across some scenarios (cf. CREDIT CARD and PAY), and partially common to many different scenarios (e.g. *get*).
- **Table A.2** shows the inter-scenario word overlap for all scenario pairs in a matrix. All ESDs for a scenario are collapsed to one bag of words here, and the overlap is calculated on the basis of those bags. (See Chapter 3 for details.) The matrix is symmetric with respect to the middle diagonal; both halves contain the same values.
- **Table A.3** shows the sequence-based word overlap for all scenario pairs. These values indicate how well two scenarios blend with respect to the lexical similarity of their sequences. The matrix is symmetric with respect to the middle diagonal; both halves contain the same values.
- **Table A.4** shows the reordering values for blended scenario pairs. The values show how well two scenarios blend with respect to structural constraints. The matrix is asymmetric; the upper right (grey) half shows the expected reordering, in case the structural constraints of the blended scenarios would not interact at all. The lower left (white) half shows the actual re-ordering for the blended scenarios, which is in some cases much higher than expected (marked in bold).

RESTAURANT		FAST FOOD		RETURN FOOD		SCRAMBLED EGGS		FLY		AIRPORT CHECK-IN		TAKE BUS		TAKE TRAIN	
food	28	food	32	food	27	eggs	68	plane	28	check	23	bus	76	train	43
order	25	order	30	waiter	14	pan	44	check	25	get	21	stop	33	ticket	20
leave	19	wait	12	new	14	add	30	go	24	go	20	wait	24	get	14
waiter	16	eat	12	server	12	bowl	26	flight	19	gate	17	get	24	station	13
wait	16	pay	11	wait	8	put	22	airport	19	ticket	14	seat	17	destination	10
eat	16	table	8	problem	7	heat	19	ticket	17	wait	11	find	12	go	8
menu	15	take	7	plate	7	stir	18	board	14	airport	11	go	9	wait	7
waitress	14	place	7	is	6	get	15	seat	13	luggage	10	pay	8	seat	7
pay	14	menu	7	explain	6	salt	14	luggage	13	counter	10	fare	8	platform	7
meal	14	look	7	dish	6	cook	14	boarding	13	boarding	9	look	7	find	7
DRIVING LESSON		IRON		CREATE HP						TAKE COPIES		POSTER		CHILDHOOD	
car	27	iron	75	add	13					copy	36	poster	54	learn	15
instructor	22	board	26	page	12					copies	30	wall	37	school	8
get	17	put	16	web	10					machine	21	tape	14	go	8
driving	11	ironing	16	homepage	8					press	18	put	13	walk	5
drive	11	turn	15	images	7					select	16	is	10	talk	5
follow	8	shirt	14	content	7					lid	16	use	9	get	5
start	7	set	14	write	6					original	15	make	9	grown	3
seat	7	get	14	use	6					start	13	corners	9	friends	3
practice	6	water	13	upload	6					button	13	sure	8	tooth	2
parking	6	plug	11	find	6					number	12	level	7	start	2
GO SHOPPING		PAY		CREDIT CARD		HAIRCUT		WEDDING		TAKE SHOWER		FIX TIRE		CLEAN UP	
items	39	card	22	card	13	hair	58	bride	38	turn	34	tire	64	clean	19
store	27	wallet	20	cashier	7	stylist	32	groom	29	water	33	tube	29	floor	16
go	21	get	17	receipt	6	is	22	vows	16	get	32	remove	27	dust	12
get	16	cashier	17	credit	6	get	21	aisle	15	wash	28	bike	21	vacuum	11
pay	14	credit	15	wallet	5	pay	18	guests	13	hair	27	put	18	kitchen	10
car	14	take	12	sign	5	cut	18	father	13	body	25	patch	14	put	9
cart	11	receipt	10	wait	3	haircut	17	couple	13	rinse	24	wheel	13	pick	9
look	9	purse	10	take	3	sit	14	rings	12	shower	22	air	10	wash	8
list	9	hand	9	slip	3	wait	13	wedding	11	soap	20	take	8	mop	8
leave	9	cash	9	put	3	chair	13	enter	11	dry	19	rim	7	furniture	8

Table A.1: The 10 most frequent content words (nouns, verbs, adjectives) for each scenario.

	RESTAURANT	FAST FOOD	RETURN FOOD	SCRAMBLED EGGS	FLY	AIRPORT CHECK–IN	TAKE BUS	TAKE TRAIN	SHOPPING	PAY	CREDIT CARD	HAIRCUT	WEDDING	TAKE SHOWER	FIX TIRE	CLEAN UP	DRIVING LESSON	IRON	CREATE HP	TAKE COPIES	POSTER	CHILDHOOD
RESTAURANT	●	**0.50**	0.35	0.18	0.34	0.31	0.36	0.34	0.36	0.37	0.32	0.36	0.28	0.24	0.22	0.19	0.32	0.26	0.26	0.32	0.21	0.23
FAST FOOD	**0.50**	●	0.26	0.19	0.37	0.35	0.35	0.34	**0.44**	**0.41**	0.37	0.33	0.25	0.25	0.28	0.27	0.36	0.30	0.21	0.35	0.26	0.22
RETURN FOOD	0.35	0.26	●	0.17	0.21	0.23	0.23	0.19	0.24	0.23	0.21	0.29	0.20	0.21	0.21	0.16	0.24	0.22	0.18	0.24	0.18	0.20
SCRAMBLED EGGS	0.18	0.19	0.17	●	0.18	0.22	0.19	0.18	0.21	0.19	0.14	0.22	0.18	0.30	0.26	0.23	0.20	0.32	0.16	0.27	0.25	0.14
FLY	0.34	0.37	0.21	0.18	●	**0.48**	**0.42**	**0.46**	0.36	0.31	0.24	0.29	0.26	0.23	0.23	0.21	0.33	0.23	0.22	0.29	0.24	0.20
AIRPORT CHECK IN	0.31	0.35	0.23	0.22	**0.48**	●	**0.40**	0.35	0.35	0.32	0.26	0.35	0.28	0.29	0.26	0.22	0.31	0.26	0.21	0.33	0.27	0.18
TAKE BUS	0.36	0.35	0.23	0.19	**0.42**	**0.40**	●	**0.47**	0.36	0.32	0.22	0.34	0.26	0.29	0.25	0.20	0.31	0.29	0.24	0.31	0.26	0.21
TAKE TRAIN	0.34	0.34	0.19	0.18	**0.46**	0.35	**0.47**	●	0.36	0.34	0.35	0.32	0.27	0.26	0.26	0.22	0.31	0.26	0.25	0.30	0.23	0.22
SHOPPING	0.36	**0.44**	0.24	0.21	0.36	0.35	0.36	0.36	●	**0.44**	0.36	0.36	0.26	0.28	0.26	0.26	0.35	0.31	0.26	0.35	0.28	0.22
PAY	0.37	**0.41**	0.23	0.19	0.31	0.32	0.32	0.34	**0.44**	●	**0.47**	0.30	0.21	0.25	0.26	0.22	0.26	0.26	0.27	0.33	0.26	0.19
CREDIT CARD	0.32	0.37	0.21	0.14	0.24	0.26	0.22	0.35	0.36	**0.47**	●	0.24	0.20	0.26	0.20	0.18	0.24	0.22	0.19	0.23	0.20	0.21
HAIRCUT	0.36	0.33	0.29	0.22	0.29	0.35	0.34	0.32	0.36	0.30	0.24	●	0.27	0.33	0.27	0.22	0.36	0.30	0.26	0.34	0.26	0.19
WEDDING	0.28	0.25	0.20	0.18	0.26	0.28	0.26	0.27	0.26	0.21	0.20	0.27	●	0.20	0.19	0.16	0.26	0.22	0.19	0.19	0.18	0.16
TAKE SHOWER	0.24	0.25	0.21	0.30	0.23	0.29	0.29	0.26	0.28	0.25	0.26	0.33	0.20	●	0.26	0.28	0.30	0.35	0.20	0.31	0.27	0.18
FIX TIRE	0.22	0.28	0.21	0.26	0.23	0.26	0.25	0.26	0.26	0.26	0.20	0.27	0.19	0.26	●	0.23	0.26	0.32	0.21	0.31	0.31	0.19
CLEAN UP	0.19	0.27	0.16	0.23	0.21	0.22	0.20	0.22	0.26	0.22	0.18	0.22	0.16	0.28	0.23	●	0.20	0.29	0.17	0.24	0.25	0.15
DRIVING LESSON	0.32	0.36	0.24	0.20	0.33	0.31	0.31	0.31	0.35	0.26	0.24	0.36	0.26	0.30	0.26	0.20	●	0.30	0.24	0.32	0.23	0.22
IRON	0.26	0.30	0.22	0.32	0.23	0.26	0.29	0.26	0.31	0.26	0.22	0.30	0.22	0.35	0.32	0.29	0.30	●	0.23	0.36	0.34	0.22
CREATE HP	0.26	0.21	0.18	0.16	0.22	0.21	0.24	0.25	0.26	0.27	0.19	0.26	0.19	0.20	0.21	0.17	0.24	0.23	●	0.26	0.21	0.19
TAKE COPIES	0.32	0.35	0.24	0.27	0.29	0.33	0.31	0.30	0.35	0.33	0.23	0.34	0.19	0.31	0.31	0.24	0.32	0.36	0.26	●	*0.38*	0.19
POSTER	0.21	0.26	0.18	0.25	0.24	0.27	0.26	0.23	0.28	0.26	0.20	0.26	0.18	0.27	0.31	0.25	0.23	0.34	0.21	*0.38*	●	0.15
CHILDHOOD	0.23	0.22	0.20	0.14	0.20	0.18	0.21	0.22	0.22	0.19	0.21	0.19	0.16	0.18	0.19	0.15	0.22	0.22	0.19	0.19	0.15	●

Table A.2: Inter-scenario word type overlap. The average overlap is 0.27. Particularly high values (> 0.40) are marked in **boldface**.

	RESTAURANT	FAST FOOD	RETURN FOOD	SCRAMBLED EGGS	FLY	AIRPORT CHECK IN	TAKE BUS	TAKE TRAIN	GO SHOPPING	PAY	CREDIT CARD	HAIRCUT	WEDDING	TAKE SHOWER	FIX TIRE	CLEAN UP	DRIVING LESSON	IRON	CREATE HP	TAKE COPIES	POSTER	CHILDHOOD
RESTAURANT	●	**0.31**	0.16	0.06	0.13	0.11	0.17	0.15	0.15	0.10	0.11	0.15	0.08	0.05	0.06	0.04	0.09	0.07	0.07	0.08	0.07	0.09
FAST FOOD	**0.31**	●	0.17	0.08	0.15	0.13	0.19	0.16	0.18	0.13	0.13	0.16	0.08	0.08	0.08	0.07	0.11	0.10	0.09	0.11	0.09	0.11
RETURN FOOD	0.16	0.17	●	0.06	0.09	0.08	0.13	0.11	0.10	0.09	0.11	0.11	0.06	0.06	0.08	0.04	0.08	0.07	0.07	0.08	0.08	0.07
SCRAMBLED EGGS	0.06	0.08	0.06	●	0.08	0.08	0.09	0.08	0.08	0.07	0.08	0.08	0.06	0.12	0.09	0.05	0.09	0.12	0.08	0.09	0.10	0.08
FLY	0.13	0.15	0.09	0.08	●	**0.27**	0.20	0.23	0.16	0.11	0.11	0.13	0.08	0.09	0.08	0.06	0.12	0.12	0.08	0.10	0.10	0.11
AIRPORT CHECK IN	0.11	0.13	0.08	0.08	**0.27**	●	0.17	0.19	0.14	0.11	0.11	0.12	0.07	0.09	0.07	0.05	0.10	0.10	0.07	0.09	0.09	0.11
TAKE BUS	0.17	0.19	0.13	0.09	0.20	0.17	●	**0.25**	0.17	0.13	0.13	0.16	0.08	0.13	0.09	0.05	0.13	0.12	0.10	0.11	0.10	0.12
TAKE TRAIN	0.15	0.16	0.11	0.08	0.23	0.19	**0.25**	●	0.17	0.12	0.13	0.13	0.08	0.11	0.09	0.05	0.12	0.11	0.09	0.11	0.10	0.14
GO SHOPPING	0.15	0.18	0.10	0.08	0.16	0.14	0.17	0.17	●	0.15	0.16	0.15	0.07	0.09	0.10	0.06	0.14	0.10	0.10	0.12	0.11	0.12
PAY	0.10	0.13	0.09	0.07	0.11	0.11	0.13	0.12	0.15	●	**0.27**	0.09	0.05	0.09	0.09	0.05	0.10	0.08	0.07	0.10	0.08	0.09
CREDIT CARD	0.11	0.13	0.11	0.08	0.11	0.11	0.13	0.13	0.16	**0.27**	●	0.10	0.06	0.08	0.08	0.05	0.09	0.09	0.07	0.09	0.08	0.10
HAIRCUT	0.15	0.16	0.11	0.08	0.13	0.12	0.16	0.13	0.15	0.09	0.10	●	0.07	0.14	0.08	0.05	0.11	0.10	0.07	0.09	0.09	0.11
WEDDING	0.08	0.08	0.06	0.06	0.08	0.07	0.08	0.08	0.07	0.05	0.06	0.07	●	0.05	0.04	0.04	0.06	0.06	0.04	0.06	0.06	0.08
TAKE SHOWER	0.05	0.08	0.06	0.12	0.09	0.09	0.13	0.11	0.09	0.09	0.08	0.14	0.05	●	0.10	0.09	0.11	0.16	0.06	0.09	0.09	0.08
FIX TIRE	0.06	0.08	0.08	0.09	0.08	0.07	0.09	0.09	0.10	0.09	0.08	0.08	0.04	0.10	●	0.05	0.07	0.10	0.06	0.09	0.11	0.06
CLEAN UP	0.04	0.07	0.04	0.05	0.06	0.05	0.05	0.05	0.06	0.05	0.05	0.05	0.04	0.09	0.05	●	0.04	0.08	0.04	0.06	0.06	0.04
DRIVING LESSON	0.09	0.11	0.08	0.09	0.12	0.10	0.13	0.12	0.14	0.10	0.09	0.11	0.06	0.11	0.07	0.04	●	0.10	0.08	0.09	0.09	0.12
IRON	0.07	0.10	0.07	0.12	0.12	0.10	0.12	0.11	0.10	0.08	0.09	0.10	0.06	0.16	0.10	0.08	0.10	●	0.07	0.11	0.11	0.08
CREATE HP	0.07	0.09	0.07	0.08	0.08	0.07	0.10	0.09	0.10	0.07	0.07	0.07	0.04	0.06	0.06	0.04	0.08	0.07	●	0.07	0.09	0.07
TAKE COPIES	0.08	0.11	0.08	0.09	0.10	0.09	0.11	0.11	0.12	0.10	0.09	0.09	0.06	0.09	0.09	0.06	0.09	0.11	0.07	●	0.10	0.07
POSTER	0.07	0.09	0.08	0.10	0.10	0.09	0.10	0.10	0.11	0.08	0.08	0.09	0.06	0.09	0.11	0.06	0.09	0.11	0.09	0.10	●	0.08
CHILDHOOD	0.09	0.11	0.07	0.08	0.11	0.11	0.12	0.14	0.12	0.09	0.10	0.11	0.08	0.08	0.06	0.04	0.12	0.08	0.07	0.07	0.08	●

Table A.3: Sequence-based word overlap between ESDs of different scenarios. The average overlap within *the same scenario* is 0.29. Particularly high values (> 0.25) are marked in **boldface**.

	RESTAURANT	FAST FD.	RETURN FOOD	EGGS	FLY	AIRPORT	BUS	TRAIN	SHOP	PAY	CREDIT CARD	HAIRCUT	WEDDING	SHOWER	FIX TIRE	CLEAN	DRIVING	IRON	HP	COPIES	POSTER	CHILDH.
RESTAURANT	●	*0.10*	0.09	0.11	0.11	0.13	0.10	0.09	0.12	0.11	0.10	0.11	0.12	0.12	0.10	0.11	0.11	0.10	0.10	0.11	0.09	0.11
FAST FOOD	**0.14**	●	0.08	*0.10*	0.10	0.12	0.09	0.08	0.11	0.10	0.09	0.10	0.11	0.11	0.09	0.10	0.10	0.09	0.09	0.10	0.08	0.10
RETURN FOOD	0.10	0.07	●	*0.09*	0.08	0.10	0.08	0.07	0.09	0.08	0.08	0.09	0.09	0.09	0.08	0.09	0.08	0.08	0.07	0.08	0.07	0.08
SCRAMBLED EGGS	0.12	**0.12**	**0.12**	●	0.13	0.15	0.12	0.11	0.14	0.13	0.12	0.13	0.14	0.14	0.12	0.13	0.13	0.12	0.12	0.13	0.11	0.13
FLY	0.11	0.10	0.09	0.12	●	*0.13*	0.10	0.09	0.12	0.11	0.10	0.11	0.12	0.12	0.10	0.11	0.11	0.10	0.10	0.11	0.09	0.11
AIRPORT CHECK IN	0.15	0.15	0.14	0.16	**0.16**	●	0.14	0.13	0.16	*0.15*	*0.14*	0.15	0.16	0.16	0.14	0.15	0.15	0.14	0.14	0.15	0.13	0.15
TAKE BUS	0.12	0.09	0.08	0.11	0.10	0.14	●	0.08	0.11	0.10	0.09	0.10	0.11	0.11	0.09	0.10	0.10	0.09	0.09	0.10	0.08	0.10
TAKE TRAIN	0.08	0.07	0.04	0.10	0.11	0.13	0.08	●	0.08	0.07	0.07	0.08	0.08	0.08	0.07	0.08	0.07	0.07	0.06	0.07	0.06	0.07
GO SHOPPING	0.14	0.13	0.12	0.15	0.13	0.17	0.13	**0.11**	●	*0.13*	*0.12*	0.13	0.14	0.14	0.12	0.13	0.13	0.12	0.12	0.13	0.11	0.13
PAY	0.12	0.10	0.10	0.14	0.11	**0.17**	0.10	0.07	**0.15**	●	0.11	0.12	0.13	0.13	0.11	0.12	0.12	0.11	0.11	0.12	0.10	0.12
CREDIT CARD	0.10	0.08	0.04	0.13	0.10	**0.17**	0.08	0.04	**0.14**	0.12	●	*0.08*	0.09	0.09	0.07	0.08	0.08	0.07	0.07	0.08	0.06	0.08
HAIRCUT	0.13	0.12	0.11	0.13	0.12	0.15	0.11	0.10	0.13	0.12	**0.11**	●	0.13	0.13	0.11	0.12	0.12	0.11	0.11	0.12	0.10	0.12
WEDDING	0.13	0.12	0.12	0.14	0.12	0.16	0.11	0.11	0.15	0.14	0.13	0.13	●	0.14	0.12	0.13	0.13	0.12	0.12	0.13	0.11	0.13
TAKE SHOWER	0.12	0.12	0.12	0.15	0.12	0.17	0.11	0.10	0.15	0.15	0.13	0.13	0.14	●	0.13	0.14	0.13	0.13	0.12	0.13	0.12	0.13
FIX TIRE	0.10	0.09	0.07	0.13	0.10	0.15	0.09	0.06	0.13	0.11	0.08	0.11	0.12	0.12	●	0.11	0.10	0.10	0.09	0.10	0.09	0.10
CLEAN UP	0.12	0.12	0.11	0.14	0.12	0.16	0.10	0.09	0.14	0.13	0.12	0.12	0.14	0.14	0.12	●	0.12	0.12	0.11	0.12	0.11	0.12
DRIVING LESSON	0.11	0.10	0.09	0.12	0.11	0.15	0.10	0.08	0.14	0.12	0.10	0.13	0.13	0.13	0.10	0.12	●	0.10	0.10	0.11	0.09	0.11
IRON	0.10	0.09	0.09	0.13	0.10	0.14	0.09	0.08	0.12	0.11	0.10	0.11	0.12	0.13	0.10	0.11	0.10	●	0.10	0.11	0.09	0.11
CREATE HP	0.09	0.08	0.06	0.12	0.10	0.15	0.08	0.05	0.12	0.10	0.06	0.11	0.12	0.12	0.08	0.11	0.09	0.09	●	0.09	0.07	0.09
TAKE COPIES	0.11	0.10	0.09	0.13	0.11	0.15	0.10	0.08	0.14	0.13	0.10	0.12	0.13	0.14	0.11	0.12	0.11	0.10	0.09	●	0.09	0.11
POSTER	0.09	0.07	0.05	0.11	0.09	0.12	0.08	0.05	0.11	0.09	0.06	0.10	0.11	0.11	0.08	0.10	0.08	0.08	0.06	0.09	●	0.09
CHILDHOOD	0.11	0.09	0.09	0.14	0.11	0.17	0.09	0.06	0.15	0.13	0.11	0.12	0.14	0.15	0.11	0.14	0.11	0.11	0.10	0.11	0.08	●
AVERAGE EXPECTED	0.11	0.10	0.08	0.12	0.11	0.14	0.10	0.08	0.12	0.11	0.09	0.11	0.12	0.12	0.10	0.11	0.11	0.10	0.10	0.11	0.09	0.11
AVERAGE	0.11	0.10	0.09	0.13	0.11	0.15	0.10	0.08	0.14	0.12	0.10	0.12	0.13	0.13	0.10	0.12	0.11	0.11	0.09	0.11	0.09	0.12

Table A.4: Change of predicate reordering after combining two scenarios. The grey part gives the expected reordering value (=average of both scenarios); the white part lists the actual values. Some notable deviances from the expected value ($> +0.02$) are marked in **boldface**, their corresponding expected values appear in *italics*.