Karlsruhe Institute of Technology

Analyse und Korrektur von Disfluenzen in gesprochener Sprache

Bachelorarbeit
von

Robert Hochweiß

An der Fakultät für Informatik
Institut für Programmstrukturen
und Datenorganisation (IPD)

Erstgutachter:	Prof. Dr. Walter F. Tichy
Zweitgutachter:	Prof. Dr. Ralf H. Reussner
Betreuender Mitarbeiter:	Dipl.-Inform. Sebastian Weigelt
Zweiter betr. Mitarbeiter:	M.Sc. Tobias Hey

Bearbeitungszeit: 15.01.2018 – 11.06.2018

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Die Regeln zur Sicherung guter wissenschaftlicher Praxis im Karlsruher Institut für Technologie (KIT) habe ich befolgt.

Karlsruhe, 11.06.2018

. . . . . .

(**Robert Hochweiß**)

Kurzfassung

Disfluenzen sind ein wesentlicher Bestandteil von spontan gesprochenen Äußerungen. Bei Disfluenzen handelt es sich um Unterbrechungen des normalen Sprechflusses, die durch Fehler, Wortwiederholungen, Füllwörter oder ähnliche andere Wörter entstanden sind. Sie erschweren die Bearbeitung einer Äußerung und müssen daher korrigiert werden. Eine automatisierte Korrektur dieser Disfluenzen erweist sich jedoch aufgrund des unregelmäßigen Aufbaus solcher Disfluenzen als schwierig. Deshalb wird in dieser Arbeit die Erkennung und Korrektur von Disfluenzen in natürlichsprachlichen Äußerungen erarbeitet. Hierzu wurde mit Hilfe eines maschinellen Lernverfahrens ein Klassifikator entwickelt, der diese Disfluenzen erkennt und korrigiert. Das maschinelle Lernverfahren basiert auf einem rekurrenten neuronalen Netzwerk mit langen Kurzzeitgedächtnis (engl. long short-term memory - LSTM). Die Funktionalität des entworfenen Werkzeugs wird anhand von händischen Transkriptionen sowie einem Testdatensatz des Switchboard-Korpus getestet. Auf diesen beiden Datensätzen wurde entsprechend ein F1-Wert von 0,71 und 0,792 erreicht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

In der heutigen Zeit nimmt die Anzahl von sprachgesteuerten Systemen im Alltag immer weiter zu. Seien es intelligente persönliche Assistenten wie *Google Now* und Apples *Siri* oder auch Industrieroboter, die sich immer mehr als wachsender Markt etablieren [IFR15]. Doch auch für die intelligenten Haushaltsroboter, wie der am KIT entwickelte ARMAR-III [ARA+06], steigt das Interesse an.
Doch mit gesprochener Sprache als Eingabe für diese Systeme ergeben sich auch Nachteile: Es treten Versprecher, Fehler sowie Wiederholungen auf und Füllwörter wie „umm“ oder „uh“ werden benutzt. Dies gilt vor allem, wenn ein Text nicht vorgelesen, sondern spontan gesprochen wird, was im Alltag dem Normalfall entspricht. Diese Unterbrechungen des normalen Sprechflusses werden als *Disfluenzen* bezeichnet. Ein Beispiel hierfür wäre bei der Befehlsanweisung für den Roboter ARMAR-III:

„*hey Armar take the red cup uh the green cup*“

Während Menschen diese Disfluenzen intuitiv und automatisch auflösen, haben es Maschinen wesentlich schwerer. Sie müssen zuerst die Disfluenzen sprachlich korrekt erfassen und anschließend diese geeignet interpretieren und verarbeiten. Durch eine falsche oder fehlende Interpretation und Korrektur von Disfluenzen kann es vorkommen, dass einige Anweisungen unkorrekt oder nicht umgesetzt werden. Doch wie erfolgt diese Interpretation und Korrektur? Diese Frage soll im Rahmen dieser Bachelorarbeit beantwortet werden.

1.1. Zielsetzung

Das Ziel dieser Bachelorarbeit ist die Entwicklung eines Werkzeugs, das natürlichsprachliche Anweisungen als Eingabe erhält und diese auf sprachliche Disfluenzen hin analysiert und ein oder mehrere Korrekturvorschläge für die entsprechenden Stellen ausgibt.
Dafür müssen die Disfluenzen zunächst in den Anweisungen identifiziert und dann klassifiziert werden. Anschließend wird je nach Art der Disfluenz die entsprechende Korrekturmaßnahme ausgewählt.
Das zu entwickelnde Werkzeug soll dabei ein Agent für das Projekt PARSE (siehe Kapitel 3) sein.

1.2. Aufbau der Arbeit

In dieser Arbeit werden zunächst im Kapitel 2 die für diese Arbeit erforderlichen Grundlagen eingefügt. Danach wird in Kapitel 3 ein Überblick über das Projekt PARSE gegeben. In Kapitel 4 werden die zur Thematik dieser Arbeit verwandten Arbeiten vorgestellt. Danach wird im Kapitel 5 die Problemstellung näher analysiert sowie die Eignung der in den verwandten Arbeiten vorgestellten Ansätzen hinsichtlich des Lösungsentwurfs diskutiert. Das Kapitel 6 beschriebt den Lösungsentwurf sowie dessen Realisierung. Die Funktionsweise des zu entwickelten Werkzeugs wird dann in Kapitel 7 gemäß bestimmten Metriken bewertet. Abschließend fasst Kapitel 8 die wichtigsten Punkte nochmals zusammen und gibt einen Ausblick über mögliche weiterführende Arbeit.

2. Grundlagen

In diesem Kapitel werden die für diese Arbeit erforderlichen häufig genutzte Begriffe sowie Konzepte definiert und erläutert.

2.1. Sprachwissenschaftliche Grundlagen

In diesem Abschnitt werden die für diese Arbeit relevanten Begriffe und Konzepte aus der Sprachwissenschaft definiert und vorgestellt.

2.1.1. Natürliche Sprache

Eine natürliche Sprache ist eine Sprache zwischen Menschen, die regional und sozial geschichtet aus einer historischen Entwicklung entstanden ist [Buß02], wie bspw. die Sprachen Deutsch und Englisch. Im Unterschied dazu fehlt solch eine Entwicklung bei den künstlichen Sprachen, die gezielt für einen bestimmten Verwendungszweck erzeugt wurden, wie bspw. Programmiersprachen. Natürliche Sprachen weisen gegenüber künstlichen Sprachen insbesondere auch eine lexikalische und strukturelle Mehrdeutigkeit bzw. eine Vagheit oder Bedeutungsvielfalt auf.

2.1.2. Syntax

Die Syntax ist ein Teilbereich der Grammatik natürlicher Sprachen und beschreibt die Lehre vom Bau einer Texteinheit [Buß02]. Die Syntax ist dabei ein Regelwerk, welches festlegt, wie sich Zeichen und Wörter zu übergeordneten Einheiten wie Phrasen, Teilsätze oder vollständige Sätze zusammensetzen lassen. Sie stellt demnach die Struktur eines Satzes dar. Eine wichtige syntaktische Regel in der englischen Sprache ist hierbei die vorgeschrieben Satzgliedreihenfolge Subjekt-Prädikat-Objekt, die in jedem syntaktisch korrekten Satz vorliegen muss.

2.1.3. Phrase

Eine Phrase ist eine Menge von syntaktisch zusammengehörigen Elementen, die eine Konstituente, also eine Wortgruppe oder einen relativ selbstständigen Satzteil, bilden [Buß02]. Jede Phrase besitzt dabei einen Kopf, nach dem sie entsprechend bezeichnet wird. So ist im Satz „A blond boy plays in the house.“ bspw. die Phrase „A blond boy“ eine Nominalphrase, die nach dem Nomen „boy“ definiert ist. Die wichtigsten Phrasen sind Nominal-, Verbal- und Präpositionalphrasen.

2.2. Natürliche Sprachverarbeitung

Unter dem Begriff der natürlichen Sprachverarbeitung (engl. *Natural Language Processing - NLP*) wird in der Forschung die computergestützte maschinellen Verarbeitung von natürlicher Sprache verstanden [JM09]. Die Sprache kann dabei in gesprochener oder geschriebener Form vorliegen. Da es sich bei der natürlichen Sprachverarbeitung um ein sehr umfangreiches und komplexes Forschungsgebiet handelt, werden im Folgenden nur die für diese Arbeit relevanten Aspekte und Begriffe vorgestellt.

2.2.1. Wortartmarkierung

Die Wortartmarkierung, im Englischen als *Part-Of-Speech-Tagging (POS-Tagging)* bezeichnet, ist der Zuordnungsprozess eines Worts zu seiner zugehörigen Wortart für jedes Wort innerhalb eines Satzes [JM09, Kap. 5]. Es existieren einige verschiedene Ansätze, wie bspw. regelbasierte oder maschinelle Lernansätze, die jedem Wort eine entsprechende Markierung (engl. Tag) zuweisen. Zur Vereinheitlichung wird dabei ein Markierungssatz (engl. Tagset) verwendet, wie bspw. der am meisten verbreitete Penn-Treebank-Tagset [MMS93]. Durch die Wortartmarkierung lassen sich auch bereits einige Zusammenhänge zwischen den Wörtern und ihren Nachbarn ermitteln, die in nachfolgenden Sprachverarbeitungsschritten von Bedeutung sein können. Im unteren Beispiel ist nun der Satz „Tim go to the computer and book the cheapest flight“ mit den Markierungen des Penn-Treebank-Tagsets dargestellt, wobei die Markierungen tiefgestellt und den entsprechenden Wörtern vorangehend gesetzt sind:

(1) [NNP Tim] [VB go] [TO to] [DT the] [NN computer] [CC and] [VB book] [DT the] [JJS cheapest] [NN flight]

Im Beispiel ist dabei auch zugleich eine Problematik der Wortartmarkierung dargestellt, denn ”book”kann im Allgemeinen neben einem Verb auch ein Nomen sein. Diese Mehrdeutigkeiten müssen durch das gewählte Markierungsverfahren unter Berücksichtigung des aktuellen Kontexts aufgelöst werden.

2.2.2. Phrasenerkennung

Bei der Phrasenerkennung (engl. *Chunking*) werden neben geordnete, sich nicht überlappende Wortgruppen oder Phrasen (siehe Abschnitt 2.1.3) erkannt und entsprechend markiert [JM09]. Es wird dabei eine vorherige Wortartmarkierung vorausgesetzt. Die Phrasen werden dabei je nach dem Kopf und damit je nach dem Typ der Phrase entsprechend annotiert. Da die Informationen der Phrasenerkennung üblicherweise pro Wort gespeichert werden, wird für jedes Wort noch zusätzlich die Postion innerhalb der Phrase markiert. Dabei wird standardmäßig das *IOB*-Format verwendet:
Pro Wort wird markiert, ob es sich am Anfang (Markierung: B), innerhalb (Markierung: I) oder außerhalb (Markierung: O) einer Phrase befindet. Beispiel 1 ist nun nach einer Phrasenerkennung im IOB-Format im Folgenden dargestellt:

(2) [B-NP Tim] [B-VP go] [B-PP to] [B-NP the] [I-NP computer] [O and] [B-VP book] [B-NP the] [I-NP cheapest] [I-NP flight]

Die Markierungen sind dabei in den Klammern dem entsprechenden Phrasen voraus- und tiefgestellt.

2.3. Disfluenzen in gesprochener Sprache

Disfluenzen in gesprochener Sprache (engl. speech disfluencies) sind Unterbrechungen des normalen Sprechflusses. Beispielhaft dafür sind gefüllte Pausen wie „uh, uhm“ oder Wortwiederholungen. Da sich diese Arbeit mit der Analyse und Korrektur von Disfluenzen befasst, werden deren Eigenschaften im Folgenden beschrieben. Falls nicht explizit anders genannt, sind alle Bezeichnungen, Definitionen und die Terminologie der Arbeit von Shriberg [Shr94] entnommen.

2.3.1. Aufbau

Disfluenzen in spontan gesprochener Sprache besitzen einen typischen Aufbau, der im Folgenden beschrieben wird:

(3) „*Hey Armar go [to the kitchen table + uhm I mean to the dishwasher]* “

In Beispiel 3 markiert „+“ den Unterbrechungspunkt (engl. interruption point - IP) des Sprechflusses. Das Interregnum (IM), oft auch als Bearbeitungsphase (engl. editing phase) bezeichnet, ist optional und besteht aus einer Kombination von Verzögerungslauten und Füllwörtern (engl. filler oder filled pauses), Diskursmarker sowie expliziter Bearbeitungsbegriffe (engl. explicit editing terms). Statt eines Interregnums kann auch lediglich eine normale Pause des Sprechflusses erfolgen (engl. unfilled pause).
Das Reparandum (RM) ist der fehlerhafte, zu korrigierende Anweisungsteil, der durch die Reparans (RS), oft auch als Reparatur bezeichnet, ersetzt wird. Das Reparans ist üblicherweise eine Modifikation des Reparandums. Der Korrekturvorschlag für Beispiel 3 lautet nun wie folgt:

„*Hey Armar go to the dishwasher*“

Beispiel 3 stellt eine grundlegende (Selbst-)Reparatur da.

2.3.2. Klassifizierung

Disfluenzen lassen sich laut [Bes06] nach ihrem typischen Aufbau in drei Kategorien einteilen: unkorrigierte und entfernbare Disfluenzen und Reparaturen.

In (Selbst-)Reparaturen oder (Selbst-)Korrekturen, sind die Bestandteile Reparandum und Reparans durch den Sprecher gegeben. Dabei finden Modifikationen der Reparatur gegenüber dem Reparandum statt. So kann je nach Art der Modifikation eine Überarbeitung nochmals als Entfernung, Einsetzung, Wiederholung, Ersetzung oder Neustart klassifiziert werden.

Bei der Entfernung werden in der Reparatur nur einige Teile des Reparandums wiederholt, während die anderen Wörter entfernt wurden. Die gelöschten Wörter müssen sich dabei in zentraler Position innerhalb des Reparandums befinden.
Im Gegensatz dazu werden bei einer Einsetzung zusätzliche Wörter in der Reparatur hinzugefügt, wobei die Wörter nicht erst am Ende der Reparatur hinzugefügt werden dürfen. Ansonsten würde es sich um eine Wiederholung, bei der die Reparatur dem Reparandum gleicht, handeln.
Bei einer Ersetzung werden nur Teile des Reparandums wiederholt, während die anderen Wörter durch neue ersetzt werden.
In einem Neustart (engl. restart oder auch false start oder fresh start) hingegen werden keine Wörter des Reparandums wiederholt und es besteht kein Zusammenhang zwischen

Reparatur und Reparandum.

Im Gegensatz dazu korrigiert der Sprecher bei unkorrigierten Disfluenzen seine Fehler nicht, wodurch diese Disfluenzen auch keinen Reparans-Teil besitzen. So enthalten die Reparanda bei diesen Disfluenzen Sprachfehler, Wortauslassungen oder auch eine Falschordnung der Wörter, wodurch der Satz grammatikalisch unkorrekt wird.

Bei der dritten Gruppe handelt es sich um die entfernbaren Disfluenzen. Diese zeichnen sich dadurch aus, dass sie nur ein Reparandum und keine Reparatur besitzen und sie ohne Auswirkungen auf den Rest der Äußerung entfernt werden können, d.h. sie besitzen keine Bedeutung in der eigentlichen Aussage. Beispielhaft dafür sind gefüllte Pausen wie „um“ oder Wortfragmente, die durch Stottern, Versprecher und andere verbale Unterbrechungen entstanden sind. Zu den entfernbaren Disfluenzen gehören jedoch auch Diskursmarker wie „you know“ oder explizite Bearbeitungsbegriffe wie bspw. „I mean“.
Die entfernbaren Disfluenzen können für sich innerhalb einer Äußerung auftreten, in der ihre Korrektur lediglich das Entfernen der entsprechenden Wörter wäre, oder auch innerhalb des Interregnums in einer Überarbeitung.

2.4. Rauschkanalmodell

Ein Rauschkanal (noisy channel) ist ein Übertragungskanal von Signalen, indem es zu Störungen in Form von Rauschen kommt, die die Signale beeinflussen [Jon11].

In wissenschaftlichen Arbeiten, die sich mit Disfluenzen befassen, werden Rauschkanäle innerhalb eines Modells zur Erkennung von Disfluenzen verwendet. Dies wird als *Rauschkanalmodell* bezeichnet (engl. noisy channel modell - NCM). Dabei werden Disfluenzen als Rauschen innerhalb des Übertragungskanals modelliert [LJ17]. Eine sprachliche Äußerung ohne Disfluenzen wird dabei als Quelläußerung X bezeichnet. Im Übertragungskanal kommt es nun zu Störungen beziehungsweise Rauschen in Form von Disfluenzen, sodass aus der Quelläußerung die disfluente Äußerung Y entsteht. Nun ist anhand von Y die wahrscheinlichste Quelläußerung Z zu finden, sodass gilt: $Z = argmax_X P(Y|X) * P(X)$, wobei $P(Y|X)$ das Kanalmodell und $P(X)$ das Sprachmodell ist. Es wird dabei angenommen, dass X eine Teiläußerung von Y ist, X also durch das Entfernen von Wörtern aus Y generiert wird. Dadurch gibt es nur eine endliche Menge an Lösungskandidaten für Z.
Rauschkanalmodelle setzen die Ähnlichkeiten zwischen Reparandum und dem zugehörigen Reparans als Indikator der Disfluenz ein. Die Anwendung eines effektiven Sprachmodells innerhalb eines Rauschkanals ist jedoch rechenintensiv.

2.5. 1-aus-n-Code

Der 1-aus-n-Code (engl. one-hot code) ist eine binäre Darstellung einer dezimalen Zahl [SW07]. Die dezimale Zahl wird hierbei durch n Bits repräsentiert, wobei nur ein Bit auf „1“ gesetzt ist, während die anderen n-1 Bits auf „0“ gesetzt sind. Die Position der „1“ entspricht dabei dem Wert der jeweils dargestellten dezimalen Zahl, wobei vom niedrigstwertigen Bit (engl. least significant bit) zum höchstwertigen Bit (engl. most significant bit) gezählt wird. Für n=10 wird beispielsweise die dezimale Zahl „9“ durch das Codewort „1000000000“ repräsentiert. In Tabelle 2.1 ist nun die vollständige Codierung für das Beispiel n=10 dargestellt.

Tabelle 2.1.: 1-aus-n-Code für n=10

Dezimalzahl	1-aus-10-codiert
0	0000000001
1	0000000010
2	0000000100
3	0000001000
4	0000010000
5	0000100000
6	0001000000
7	0010000000
8	0100000000
9	1000000000

2.6. Wortvektor

Einige Verfahren benötigen die Wörter einer Eingabe nicht in deren ursprünglichen textuellen Form, sondern in einer vektoriellen Darstellung mit reellen Zahlen als Einträgen. Solch eine vektorielle Repräsentation eines Wortes wird als *Wortvektor* (engl. word vector) bezeichnet [JM17]. Da es bei dieser Abbildung von Wort zu Vektor zu einer Einbettung in einen Vektorraum kommt, wird ein Wortvektor hierbei auch als *Worteinbettung* (engl. word embedding) bezeichnet. Für die Gestaltung dieser Abbildung und den daraus resultieren Wortvektoren existieren verschieden Verfahren. Falls beispielsweise von der Menge der häufigsten Wörter eines Textkorpus ausgegangen wird, was als Vokabular bezeichnet wird, so lässt sich ein Wort jeweils als Index innerhalb des Vokabulars darstellen. Dadurch würde ein 1-aus-n-Vektor entstehen. Solch eine Repräsentation stellt jedoch nur die Position des Wortes innerhalb des Vokabulars dar. Damit lassen sich beispielsweise keine Zusammenhänge zu anderen Wörtern bilden.
Stattdessen wird versucht, pro Wort semantische Informationen abzubilden. Es gilt dabei, dass sich die Bedeutung eines Wortes aus seiner Umgebung, also dessen Kontext ergibt. Dies bedeutet, dass die Bedeutung eines Wortes von der Verteilung der Wörter in dessen Umgebung abhängt. Demzufolge besitzen Wörter mit ähnlichen Bedeutungen auch einen ähnlichen Kontext. Falls nun beispielsweise das Wort *„tesgüino“* nicht bekannt wäre, so würde sich die grundsätzliche Bedeutung aus den folgenden Sätzen herleiten:

*A bottle of **tesgüino** is on the table.*
*Everybody likes **tesgüino**.*
***Tesgüino** makes you drunk.*
*We make **tesgüino** out of corn.*

Anhand dieser Beispielsätze lässt sich schlussfolgern, dass *„tesgüino“* ein alkoholisches Getränk ist, welches aus Getreide hergestellt wird. Dies lässt sich beispielsweise anhand der Umgebung *„... make tesgüino out of corn“* erkennen. Andere Argumente für diese Schlussfolgerung lassen sich im Auftreten der Wörter *„bottle“* und *„drunk“* in anderen Umgebungen des Wortes *„tesgüino“* finden. Die Tatsache, dass sich diese sowie ähnliche Wörter wahrscheinlich auch in der Umgebung des Wortes *„beer“* finden lassen, ist ein Anzeichen dafür, dass die Wörter *„tesgüino“* und *„beer“* eine ähnliche Bedeutung besitzen. Zusätzlich können auch syntaktische Informationen des Kontextes betrachten werden, wie beispielsweise dass das Wort *„tesgüino“* nach dem Wort *„bottle“* auftritt oder das direkte Objekt des Wortes *„likes“* ist.

Falls nun wieder von einem Vokabular eines Textkorpus ausgegangen wird, so lässt sich die Abbildung von Wort zu Vektor in einer Matrix M darstellen, die das gemeinsame Auftreten der Wörter mit anderen Wörtern des Vokabulars in einem geeigneten Kontext beschreibt (engl. co-occurrence matrix). Bei M handelt es sich um eine $|V|x|V| - Matrix$, wobei $|V|$ der Anzahl der Wörter des Vokabulars entspricht. Jede Zeile i entspricht hierbei einem Wort w des Vokabulars und pro Spalte j wird angegeben, wie oft ein Wort v des Vokabulars im Kontext des Wortes w über den gesamten Korpus hinweg auftritt. Alternativ zum Abzählen des gemeinsamen Auftretens der Wörter lässt sich auch ein beliebiger Funktionswert pro Spalte eintragen, beispielsweise um die Gewichtung von bestimmten Wörtern zu beeinflussen. Auch syntaktische Merkmale pro Spalte sind möglich, beispielsweise ob ein Wort das Objekt eines anderen Wortes ist. Insgesamt entspricht der Wortvektor eines Wortes der entsprechenden Zeile in der Matrix M, wobei jeder Spalteneintrag einem Merkmal entspricht. Die Dimensionalität der resultierenden Vektoren entsprechen nun, falls pro Spalte nur das gemeinsame Auftreten von Wörtern in einem Kontext eingetragen wird, der Anzahl der Wörter des zugrunde liegenden Vokabulars. Da ein solches Vokabular üblicherweise mehr als 10 000 Wörter beinhaltet, entstehen durch dieses Verfahren sehr hoch dimensionale Vektoren. Des Weiteren sind viele der Einträge dieser Vektoren gleich Null, da pro Wort nur eine begrenzte Anzahl an anderen Wörtern in einem gemeinsamen Kontext auftreten. Diese Vektoren werden als spärlich besetzt bezeichnet (engl. sparse vectors).

Für einige Verfahren sind solche spärlich besetzten und hoch dimensionalen Vektoren jedoch ungeeignet. Diese Verfahren benötigen Wortvektoren mit wesentlich geringerer Dimensionalität und einer dichteren Besetzung (engl. dense vectors), das bedeutet dass die meisten der Einträge der Vektoren ungleich Null sind [JM16].
Ein kürzerer Wortvektor, da heißt ein Vektor mit niedrigerer Dimension, lässt sich beispielsweise einfacher als Eingabemerkmal in einem maschinellen Lernverfahren einbinden, da das Verfahren pro Wort wesentlich weniger Gewichte lernen muss. Ein dichter Vektor besitzt gegenüber einem spärlich besetzten Vektor den Vorteil, dass er weniger Parameter besitzt, obwohl er die gleichen Informationen repräsentiert. Bei einem spärlich besetzten Vektor muss beispielsweise zur effizienten Weiterverarbeitung neben den eigentlichen Einträgen auch die Indizes der Einträge angegeben werden, die ungleich Null sind. Dadurch sind dichte Vektoren hilfreicher bei der Verallgemeinerung von Zusammenhängen und um Übersättigung zu vermeiden.

Für die Erzeugung von kurzen und dichten Wortvektoren existieren einige Verfahren. Ein besonders häufig genutztes Verfahren wurde von Mikolov et al. [MCCD13] als „word2vec“ vorgestellt. Dieses Verfahren nutzt ein Neuronales Netzwerk mit drei Schichten als maschinelles Lernverfahren zur Erstellung der Wortvektoren. Das Neuronale Netzwerk besteht aus einer Eingabe-, Ausgabe- und einer verborgenen Schicht. Die Verlustfunktion des Verfahrens wird berechnet, indem gemessen wird, wie gut anhand eines Wortes dessen Kontext geschätzt werden kann. Dabei existieren die zwei Modelle Continuous-Bag-Of-Words (CBOW) und Skip-Gram (SG). CBOW versucht anhand des Kontextes das zugehörige Wort zu schätzen, während SG versucht anhand des Wortes dessen Kontext zu schätzen. Nach dem Trainingsvorgang können die resultierenden dichten Wortvektoren extrahiert werden. Die übliche Dimensionalität der Wortvektoren ist dabei 50 bis 300.

Ein anderes Verfahren zur Erzeugung von kürzeren und dichten Vektoren sind die von Pennington et al. [PSM14] entwickelten globalen Vektoren (engl. global vectors - GloVe). Das Verfahren verwendet ebenfalls ein maschinelles Lernmodell zur Erzeugung der Vektoren, jedoch wird bei den Merkmalen nicht nur der lokale Kontext pro Wort, sondern auch das gemeinsame Auftreten von anderen Wörtern in diesem Kontext betrachtet. Dies wird global über den gesamten Korpus durchgeführt. Deshalb muss im Gegensatz zum

Verfahren word2vec, das die Eingabe als Fließtext erhält, zunächst die Matrix M erzeugt werden, die das gemeinsame Auftreten von Wörtern darstellt.

3. PARSE - Programming ARchitecture for Spoken Explanations

Diese Arbeit ist Bestandteil des Projektes *PARSE*. Das Ziel von PARSE ist die Umwandlung von in natürlicher Sprache gesprochenen Anweisungen in einen für ein Zielsystem, wie bspw. den humanoiden Haushaltsroboter ARMAR-III [ARA+06], verständlichen Quelltext. Das Projekt beschäftigt sich mit den für diesen Umwandlungsprozess erforderlichen Anforderungen und Problemen. Die Besonderheit des Projektes ist hierbei der explizite Fokus auf Anweisungen in gesprochener Sprache.

3.1. Projektarchitektur

Im Projekt wird eine auf Agenten basierende Architektur [WT15] eingesetzt, in der die für den Umwandlungsprozess erforderlichen Aufgaben wie bspw. die Auflösung von sprachlichen Mehrdeutigkeiten als Agenten realisiert werden. Die Eingabeerkennung ist in Modulen gekapselt, welche leicht auswechselbar sind. So kann alternativ zu einem automatischen Spracherkenner auch ein Modul zur Gestenerkennung verwendet werden.

Die sprachliche Eingabe wird in einem seichten Sprachverarbeitungsmodul (engl. Shallow Natural Language Processing - SNLP) verarbeitet. Dieses Modul erzeugt anhand einer Wortartmarkierung, einer Phrasenerkennung sowie anhand dem Markieren von semantischen Rollen einen initialen Graphen, der die gesprochenen Anweisungen repräsentiert. Falls die Spracherkennung mehrere alternative Transkriptionen ausgibt, werden entsprechend mehrere Graphen erstellt, die gleichzeitig verarbeitet werden. Der Graph dient außerdem als Schnittstelle und geteilter Datenspeicher zwischen den Agenten. Die Agenten greifen dabei bei Bedarf auf den Graphen zu und geben ihr Ergebnis durch Modifikation des Graphen aus.

Die interne Verarbeitung der Agenten erfolgt unabhängig und transparent voneinander, wodurch auch die Evaluation entsprechend jeweils einzeln erfolgen kann. Die Agenten erhalten über Schnittstellen auch Zugriff auf externe Wissensressourcen wie die lexikalischen Datenbank WordNet [Mil95], wodurch auch die Bearbeitung von komplexeren Aufgaben der Sprachverständnis (egl. Natural Language Understanding - NLU) wie bspw. das Auflösen von Mehrdeutigkeiten möglich ist.

Des Weiteren hat die Architektur von PARSE das Entwurfsziel, dass die Bearbeitung der

Agenten möglichst domänenunabhängig erfolgt, wodurch die Anpassung an neue Domänen beschleunigt werden soll. Die Domänen sind dabei als Ontologien modelliert. Der Zugriff auf Domänenwissen erfolgt dabei nur durch die entsprechende wohl definierte Repräsentation der Ontologie. Bei Änderung der Domäne ändert sich nur der Inhalt der Ontologie, während die Struktur erhalten bleibt. Dadurch verbleiben domänenabhängige Agenten in dieser Situation unbeeinflusst.

3.2. Bisheriger Projektverlauf

Zunächst wurde im Projekt PARSE untersucht, wie Nutzer gesprochene Anweisungen in natürlicher Sprache verwenden, um humanoiden Haushaltsrobotern wie insbesondere dem ARMAR-III neue Anweisungen beizubringen. Dazu wurde in einer vorherigen Bachelorarbeit [Gün15] ein Sprachkorpus entwickelt. Des Weiteren wurde in der Bachelorarbeit von Paskaran [Pas15] ein Werkzeug zur Analyse und zum Vergleich von vier verschieden automatischen Spracherkennern entwickelt. In der Masterarbeit von Scheu [Sch16] wurden die Spracherkennerausgaben aufbereitet, um dadurch die Wortfehlerrate zu verbessern sowie teilweise Disfluenzen und Befehlsgrenzen erkannt. Die Bachelorarbeit von Kocybik [Koc15] führte für die Ausgaben der Spracherkenner erste Sprachverarbeitungsschritte wie die Wortartmarkierung sowie die Phrasenerkennung durch und bildete dies auf einen Handlungsrepräsentationsgraphen ab, welcher in der Bachelorarbeit von [Sch15] entwickelt wurde.

Die Optimierung von den Ergebnissen der Wortartmarkierung unter Einsatz von Klassifikationsverfahren wurde außerdem in der Bachelorarbeit von Kiesel [Kie16] behandelt. Auf den Graphen greifen die anderen Agenten zu und bearbeiten ihre entsprechende Aufgabe des Sprachverständnisses.
Die bisher entwickelten Agenten erkennen Aktionen [Ou16] sowie Strukturen von Bedingungen [Ste16] in gesprochener Sprache und analysieren und lösen Kontext- und Korreferenzsituationen [Hey16] sowie sprachliche Mehrdeutigkeiten [Hof16] auf.

4. Verwandte Arbeiten

Die Erkennung und Behandlung von Disfluenzen in gesprochener Sprache ist seit einiger Zeit ein wichtiges Forschungsthema, zu dem bereits zahlreiche Ansätze existieren.
Die einzelnen Erkennungsverfahren lassen sich oft nicht direkt vergleichen, da sie unter anderem verschiedene Korpora einsetzen, nach unterschiedlichen Sequenzen suchen oder unterschiedliche Annahmen bezüglich der Eingabe treffen. Dennoch sind in Tabelle 4.1 die Verfahren, welche in diesem Kapitel vorgestellt werden dargestellt. Sie sind dabei zunächst nach den Kategorien ihrer Modellierungstechniken und innerhalb dieser Kategorien aufsteigend nach dem F1-Maß geordnet. Das hier angegebene F1-Maß bezieht sich dabei auf die Erkennung von Wörtern, die in einem Reparandum liegen. Eine Ausnahme ist hierbei das Verfahren von [Geo09], bei dem sich der Wert des F1-Maßes auf die Erkennung des ersten Wortes eines Reparandums bezieht. Die Erkennung von gefüllten Pausen, Diskursmarkern sowie explizite Bearbeitungsbegriffen wird üblicherweise als trivial und in den Verfahren selten betrachtet. Auch die Erkennung von Wörtern, die in einem Reparans liegen, wird in den meisten Verfahren nicht behandelt. Die mit * markierten Autoren erlauben in ihren Verfahren prosodische Merkmale oder Wortfragmente in der Eingabe.

Die ersten Lösungsansätze waren regelbasiert. Einer der ersten regelbasierten Ansätze ist von Hindle in [Hin83] vorgestellt worden. Dieser Ansatz beinhaltet Regeln zur Erkennung und Korrektur von Selbstreparaturen und Neustarts. Diese erreichen ein erfolgreiches Auflösen von 95 % der Disfluenzen in den Testdaten. Jedoch benötigen die Regeln immer ein Bearbeitungssignal, welches den Unterbrechungspunkt einer Disfluenz markiert sowie korrekt annotierte Konstituenten. Des Weiteren enthält die Testprobe nur 1512 Sätzen der Transkriptionen aus den insgesamt zwanzig stündigen soziolinguistischen Interviews, die den Testkorpus bilden. Diese soziolinguistische Interviews wurden derart konzipiert, dass sie das unreflektierte Sprechen fördern. Von diesen Sätzen besitzen wiederum nur 404 Sätze ein Bearbeitungssignal und es wird dabei von händisch erstellten Transkriptionen ausgegangen.
Ein anderer, auf Regeln basierender Ansatz wird von Bear et al. in [BDS92] vorgestellt. Dabei werden Disfluenzen durch einfache Mustererkennungstechniken sowie syntaktische und semantische Zerteilungsverfahren erkannt. Die Mustererkennung prüft Eingabesätze auf identische Wortsequenzen sowie einfache syntaktische Anomalien wie *„a the“* oder *„to from“*. Dadurch wird eine Vorauswahl an Sätzen getroffen, die möglicherweise disfluent sind. Diese Sätze werden durch den Gemini-Zerteiler [DGA+93] nochmals überprüft. Falls nun die Zerteilung eines Satzes nicht erfolgreich ist, wird dieser Satz als disfluent markiert, ansonsten als falsch positiv. Auf dem ATIS-Korpus wird dabei eine Präzision von 81,9 %

Tabelle 4.1.: In diesem Kapitel vorgestellte Verfahren mit zugehörigem Test-Korpus und F-Wert

Autoren	Modellierungstechnik	Test-Korpus	F-Wert
[Hin83]	regelbasiert + Bearbeitungssignal	Interviews	k.A.
[BDS92]	Mustererkennung + Gemini-Zerteilung	ATIS	k.A.
[YSM16]	Zerteiler + Arc-Eager	Kaldi-SWBD	62,5 %
[RT13]	gieriger Zerteiler + Arc-Eager	SWBD	81,4 %
[HJ14]	Strahlensuche-Zerteiler + Arc-Eager	SWBD	84,1 %
[HS15]	Inkrementelle Erkennung mit RNN	SWBD	77,9 %
[JC04]	TAG-NCM	SWBD	79,7 %
[Geo09]	CRF + ILP	SWBD	80,1 %
[ZOH14]	CRF + mehr Zustände	SWBD	82,8 %
[QL13]	M^3N	SWBD	84,1 %
* [FDK15]	Semi-CRF	SWBD	85,4 %
* [ZOH16]	BI-LSTM + ILP	SWBD	85,9 %
[LJ17]	TAG-NCM + BI-LSTM + Entscheider	SWBD	86,8 %
[WCLL16]	BI-LSTM-CRF + handgefertigte Merkmale	SWBD	87,1 %

hinsichtlich der Erkennung von disfluenten Sätzen erreicht. Das Verfahren geht jedoch von perfekten Bedingungen bezüglich der Transkription und möglichen Zerteilern aus. Außerdem werden auch prosodische Informationen bezüglich der Wörter sowie Wortfragmente berücksichtigt.

In aktuellen Ansätzen werden jedoch fast ausschließlich nur maschinelle Lernverfahren als Kern der Lösungsansätze angewendet. Diese lassen sich in zwei Hauptkategorien unterteilen: auf Zerteilern (engl. Parser) basierende syntaktische Verfahren sowie auf Sequenzmarkierung basierende Verfahren (engl. sequence tagging oder sequence labeling).

Bei auf Zerteilern basierenden Verfahren werden Zerteiler zur Identifizierung von Disfluenzen sowie der syntaktischen Struktur einer Äußerung verwendet. Für das Training dieser Modelle sind große annotierte Trainingskorpora notwendig, die sowohl Annotationen von Disfluenzen als auch von syntaktischen Baumstrukturen enthalten. Diese Verfahren vereinigen die Erkennung von Disfluenzen und die Syntaxanalyse (engl. parsing) und führen beide Vorgänge gleichzeitig aus. In den aktuelleren Ansätzen werden dabei meistens Zerteiler verwendet, die die Abhängigkeiten zwischen Wörtern analysieren (dependency parsing). So verwenden Rasooli und Tetreault [RT13] einen deterministischen auf Übergängen basierenden Zerteiler für ihr Verfahren. Dieser führt die Syntaxanalyse inkrementell aus. Die Autoren erweitern dabei den Arc-Eager-Algorithmus und wenden einen zweistufigen Klassifikator an. Die erste Stufe wird dabei durch ein durchschnittlich strukturiertes Perzeptron modelliert, während auf der zweiten Stufe ein auf einem gierigen Algorithmus (engl. greedy algorithm) basierender Zerteiler eingesetz wird. Die gemeinsam ausgeführte Erkennung von Disfluenzen und Syntaxanalyse erfolgt dabei in linearer Zeit. Es wird auf dem Switchboard-Korpus damit ein F1-Wert von 81,4 % erreicht.
Das Verfahren von Honnibal und Johnson [HJ14] ist sehr ähnlich, es erweitert ebenfalls den Arc-Eager-Algorithmus und wendet eine zweistufige Klassifikation an. Der Zerteiler in der zweiten Stufe verwendet einen Algorithmus, der auf strahlenförmiger Suche basiert (engl. beam search algorithm). Durch diese Änderungen sowie einer anderen Erweiterung der Aktionen des Arc-Eager-Algorithmus und anders gewählten Merkmalen erreichen die Autoren eine bessere Erkennungsrate der Disfluenzen. So wird ein F1-Wert von 84,1 % auf

dem Switchboard-Korpus erzielt.

Yoshikawa et al. [YSM16] stellt ein auf Übergängen basierendes Zerteiler-Verfahren vor, welches ebenfalls den Arc-Eager-Algorithmus erweitert. Im Gegensatz zu den beiden vorherigen Verfahren wird der Zerteiler jedoch bewusst auf Eingaben optimiert, die nicht aus einer händischen Transkription stammen, sondern der Ausgabe eines automatischen Spracherkenners entsprechen. Die Verfahren von Rasooli und Tetreault sowie Honnibal und Mark trainieren und testen ihre Klassifizierer auf den mit Disfluenzen sowie Syntaxbäumen annotierten händischen Transkriptionen des Switchboard (SWBD)-Korpus [GHM92]. Jedoch ist nur eine Teilmenge der mit Disfluenzen annotierten Transkriptionen auch mit Syntaxbäumen annotiert. Dahingegen verwenden Yoshikawa et al. nur die Sprachaufnahmen des Switchboard-Korpus als Eingabe für das Kaldi-Spracherkennungssytem [PGB+11]. Die Ausgaben werden dann durch eigenes Verfahren der Autoren entsprechend annotiert. Diese annotierte Ausgaben dienen nun als Trainings- und Testdaten. Da nicht mit händisch annotierten Datensätzen trainiert wurde, erreichte der Ansatz eine wesentlich schlechtere Erkennungsrate als die beiden vorherigen Verfahren. So wird nur ein F1-Wert von 62,5 % erzielt.

Bei der *Sequenzmarkierung* werden Wörter unter Einsatz von verschiedenen Modellierungstechniken als fluent oder disfluent markiert. Die meist verwendeten Modellierungstechniken sind (*Conditional Random Fields (CRF)*), *Hidden-Markov-Modelle (HMM)* und *Rekurrente Neuronale Netzwerke (RNN)*. Modelle, die eine Sequenzmarkierung ausführen, benötigen außerdem einen bestimmten Zustandsraum für die Disfluenzerkennung, wie beispielsweise eine Markierung, die festlegt, ob ein Wort innerhalb oder außerhalb der Reparandum-Wort-Sequenz liegt.

Liu et al. [LSS+06] vergleichen Ansätze basierend auf HMMs, Maximale Entropy-Modelle (ME) und CRFs für die Modellierung von Disfluenz- und Satzgrenzerkennung, wobei sowohl lexikalische als auch prosodische Merkmale verwendet werden. Bei der Disfluenzerkennung werden HMMs und MEs dazu trainiert, den Unterbrechungspunkt zu finden, woraufhin heuristische Regeln angewendet werden, um die genauen Wortgrenzen des Reparandums zu bestimmen. Dahingegen werden CRFs direkt dazu trainiert, sowohl den Unterbrechungspunkt als auch die Grenzen des Reparandums zu bestimmen. In der Untersuchung zeigt sich, dass MEs und CRFs meistens deutlich bessere Ergebnisse erzielen als HMMs, wobei von den drei Modellen die CRFs die besten Ergebnisse erzielen. Für die Trainings- und Testdaten werden Korpora von Conversational-Telephone-Speech (CTS) and Broadcast-News (BN) verwendet.

Georgila [Geo09] untersucht, in wiefern sich die Erkennungsrate von Hidden-Event-Modellen (engl. hidden-event language model - HELM), MEs und CRFs verbessert, falls deren Ausgaben nachträglich durch ein ganzzahliges lineares Optimierungsverfahren (engl. integer linear programming - ILP) optimiert werden. Dabei zeigt sich, dass sich die Erkennungsrate von allen drei Modellen verbessert, insbesondere bei HELMs und MEs. Das F1-Maß auf dem Switchboard-Korpus beträgt 80,1 %. In einer anderen Arbeit von Georgila et al. [GWG10] zeigt sich des Weiteren, dass sich durch eine nachträgliche ILP-Optimierung bei CRFs die Erkennungsrate in fremden Domänen (in denen sie nicht ausreichend trainiert worden sind) besonders verbessert.

In [QL13] stellen Quian und Liu ein Max-Margin-Markov-Netzwerk (M^3N) vor. Es handelt sich dabei um eine mehrschrittige Implementierung eines gestapelten Lernverfahrens. Das Verfahren besteht aus drei Schritten. Im ersten Schritt werden gefüllte Pausen, Diskursmar-

ker und explizite Bearbeitungsbegriffe aus der Eingabe entfernt. Aus dieser „gesäuberten“ Eingabe können nun neue N-Gram-Merkmale extrahiert werden. Im zweiten Schritt werden anhand dieser neuen Eingabe und Merkmale nun die Wörter der Reparanda lokalisiert und entfernt. Im letzten Schritt wird nun diese gesäuberte Eingabe und die Merkmale der ursprünglichen Texteingabe eingesetzt, um eine präzise endgültige Entscheidung bezüglich der Wörter der Reparanda zu treffen. Dadurch wird auf dem Switchboard-Korpus ein F1-Wert von 84,1 % erreicht.

Ferguson et al. [FDK15] stellen ein Semi-Markov-CRF (kurz: Semi-CRF) vor. Mit diesem Modell lässt sich eine Unterscheidung zwischen disfluenten und normalen Stellen auf Wortgruppenebene treffen. Dadurch können im maschinellen Lernverfahren nicht nur Merkmale auf Wortebene, sondern auf Wortgruppenebene, beispielsweise bezüglich dem ganzen Reparandum, extrahiert werden. Ein Abhängigkeiten zwischen Reparandum und Reparans lassen sich so besser erkennen. Das Verfahren erzielt auf dem Switchboard-Korpus einen F1-Wert von 85,4 %. Die Autoren verwenden jedoch auch prosodische Merkmale in der Eingabe.

Hough und Schlangen [HS15] setzen ein RNN zur inkrementellen Disfluenzerkennung ein. Anstatt eine Äußerung als Gesamtheit zu betrachten, um dann jedes Element der Äußerung entsprechend zu markieren, werden Disfluenzen bei einer inkrementellen Erkennung so schnell wie möglich markiert. Dadurch können während der Testphase eventuelle Korrespondenzen zwischen Reparanda und Reparans nicht betrachtet und berücksichtigt werden, was zu einer deutlich verschlechterten Erkennungsrate führt. So wird auf dem Switchboard-Korpus nur ein Wert von 77,9 % erreicht. Dafür ist eine inkrementelle Erkennung wesentlich schneller. Dennoch bewerten die Autoren das RNN bezüglich der Erkennungsrate sowie relativ simplen und vielseitigen Vorgabe der Merkmale als angemessen und positiv.

Zayats et al. [ZOH14] verwenden ein CRF mit zusätzlichen Markierungszuständen und Mustererkennungsmerkmalen. Die neuen Markierungszustände dienen dabei zur Markierung der Wörter, die sich innerhalb eines Reparans befinden. Dadurch und durch neue Mustererkennungsmerkmale verbessert sich insbesondere auch die Erkennung der Reparansstellen. Des Weiteren zeigen Zayats et al., dass sich insbesondere durch die neuen Merkmale die Erkennung von Disfluenzen, die eine geringere Korrespondenz zwischen Reparandum und Reparans aufweisen, deutlich verbessert. Dadurch wird ein F1-Wert von 82,8 % erreicht. Außerdem verbessert sich durch die neuen Mustererkennungsmerkmale auch die Erkennungsrate auf den Testdaten von anderen Domänen.

In [ZOH16] wird dieser Ansatz nochmals erweitert. So wird alternativ zu einem CRF auch ein bidirektionales rekurrentes neuronales Netzwerk mit langen Kurzzeitgedächtnis (engl. long short-term memory - LSTM) verwendet. Dabei handelt es sich um eine Weiterentwicklung eines herkömmlichen RNNs. Zusätzlich werden die Ausgaben beider Modelle durch ein ILP-Verfahren nachträglich optimiert. Es zeigen sich dabei ähnliche Ergebnisse wie bereits in [ZOH14]. Jedoch wird deutlich, dass der Einsatz eines bidirektionalen LSTMs die Erkennungsrate vom Reparandum und insbesondere von dem Reparans verbessert. So weist ein LSTM für Disfluenzen mit einer komplexeren Struktur eine bessere Erkennungsrate auf, die in einen F1-Wert von 85,9 % auf dem Switchboard-Korpus resultiert. Die Autoren verwenden jedoch unter anderem auch Wortfragmente in der Eingabe.

In [JC04] führen Johnson und Charniak einen auf einer Baumadjunktions-Grammatik (engl. tree-adjoining grammar - TAG) basierenden Rauschkanal (engl. noisy channel model - NCM) Abschnitt 2.4 ein. Das bereits in [CJ01] verwendete, syntaktische auf Zerteiler basierte Sprachmodell wird hierbei zur Neubewertung der Ausgabe des NCMs eingesetzt.

Damit wird auf dem Switchboard-Korpus ein F1-Wert von 79,7 % erzielt.

Das Verfahren von Lou und Johnson [LJ17] basiert auf [JC04]. Es wird dabei ein NCM zur Analyse einer Textsequenz verwendet. Daraus werden die n-besten Kandidaten für eine nicht disfluente, fehlerfreie Sequenz ermittel, die durch das Entfernen von Wörtern aus der ursprünglichen Sequenz entstehen. Die Textsequenz wird ebenfalls mit einem bidirektionalen LSTM analysiert. Anhand der Bewertungen des NCMs und des LSTMs sowie anderen Eigenschaften bestimmt ein Entscheider, der auf [ZJ11] basiert, die wahrscheinlichste Lösung. Das Verfahren erreicht auf dem Switchboard-Korpus einen F1-Wert von 86,8 % .

Wang et al. [WCLL16] kombinieren die Ausgabe eines bidirektionalen LSTMs mit einem CRF sowie mit handgefertigten Merkmalen. Das CRF wird dabei als Schicht innerhalb des Netzwerks realisiert und wird vor der Ausgabeschicht platziert. Bei den Merkmalen handelt es sich um eine Kombination von diskreten handgefertigten und kontinuierlichen neuralen Merkmalen. Dadurch wird ein F1-Maß von 87,1 % erreicht, was dem aktuellen Stand der Technik bezüglich der Disfluenzerkennung entspricht.

5. Analyse

In dieser Bachelorarbeit soll ein Werkzeug entwickelt werden, welches Disfluenzen in gesprochenen natürlichen Spracheingaben erkennt und entsprechende Korrekturvorschläge anbietet. Die Eingaben dienen zur Kontrolle und Steuerung eines Zielsystems. Das Werkzeug soll dabei als Agent innerhalb des Projektes PARSE Kapitel 3 entwickelt werden. Um dies zu ermöglichen, soll zunächst die Problemstellung in diesem Kapitel genauer analysiert werden. Zur Veranschaulichung soll nachfolgendes Szenario dienen.

Innerhalb einer Küche erhält ein Haushaltsroboter eine Folge von Anweisungen. Als Robotersystem wird hierbei der am KIT entwickelte humanoide Haushaltsroboter ARMAR-III [ARA+06] betrachtet. Der Roboter soll in diesem Szenario Wasser aus dem Kühlschrank holen. Eine mögliche Anweisungsfolge wäre hierbei bspw.:

(4) „*Go to the fridge*
and open the fridge
and bring the water“

Beispiel 4 stellt hierbei jedoch eine ideale Befehlsfolge dar, wie sie in geschriebener oder vorgelesener verbaler Form vorliegen würde. Bei der Kommunikation zwischen einem Menschen und einem Haushaltsroboter, wie es in diesem Szenario der Fall ist, erfolgen die Anweisungen jedoch in spontan gesprochener Form. Dies würde nun beispielsweise wie folgt lauten:

(5) „*Go to the fridge*
and open uh the fridge
and bring [*the orange juice* + *uh I mean the*] water“

Beispiel 4 enthält dabei ein für die gesprochene Sprache charakteristisches Merkmal: das Vorhandensein von Disfluenzen.
Diese stellen eine Unterbrechung des normalen Sprechflusses dar. Menschen erkennen diese Unterbrechungen automatischen und lösen diese intuitiv auf:
Sie entfernen die gefüllte Pause „*uh*“ im zweiten und dritten Befehl sowie den expliziten Bearbeitungsbegriff „*I mean*“ im dritten Befehl. Des Weiteren erkennen sie intuitiv den fehlerhaften, zu ersetzenden Satzteil im dritten Befehl, das *Reparandum (RM)*, und wissen,

dass sie diesen durch den *Reparans (RS)* ersetzen müssen, um die korrekte Anweisung zu erhalten. Außerdem sind für Menschen auch die genauen Wortgrenzen des Reparandums und der Reparatur in der Anweisung offensichtlich und sie würden diese Bestandteile auch ohne kennzeichnende Wörter wie die gefüllte Pause „*uh*“ zuordnen können.

Während dies Prozesse von Menschen intuitiv ausgeführt werden, ist eine Automatisierung dieser Prozesse wesentlich komplexer. So ist das Erkennen und geeignete Annotieren von Disfluenzen in gesprochener Sprache Gegenstand zahlreicher Forschungen.
In diesem Kapitel werden zunächst die Einschränkungen, die sich aus dem Projekt PARSE ergeben, sowie Anforderungen an einen möglichen Lösungsentwurf erläutert. Anschließen werden die bereits im Kapitel 4 vorgestellten verwandten Arbeiten hinsichtlich ihrer Eignung für den Lösungsentwurf dieser Arbeit näher betrachtet. Daraufhin wird auf die drei wesentlichen Aspekte der Disfluenzerkennung, nämlich der Bestimmung des Unterbrechungspunktes, die Klassifizierung der Disfluenzarten sowie der Bestimmung der Wortgrenzen der Bestandteile eingegangen. Abschließend werden potentiell besonders problematische Fälle unter den Disfluenzen näher betrachtet.

5.1. Einschränkungen

Dass das zu entwickelnde Werkzeug im Rahmen des Projektes PARSE entworfen wird, bringt auch Einschränkungen mit sich, die im Folgenden näher erläutert werden.

Im Projekt PARSE wird aktuell der Spracherkenner vom IBM-Watson-System [LPM+12] verwendet, welcher Sprachaufnahmen in textuelle Transkriptionen überführt. Dieser Spracherkenner gibt neben den reinen Transkriptionen nur die entsprechenden Konfidenzwerte und Wortalternativen aus. So werden keine akustischen und prosodischen Merkmale wie unterschiedliche Tonhöhen ausgegeben, wodurch jeder potentielle Lösungsansatz nur auf linguistischen Merkmalen basieren darf.
Des Weiteren werden wie bei den meisten Spracherkennern keine Wortabbrüche, sogenannte Wortfragmente, erkannt. Diese sind zwar oft charakteristisch für Disfluenzen, jedoch werden Wortfragmente in den meisten Fällen dem vorherigen oder nachfolgenden Wort zugeordnet, was zu falsch erkannten Wörtern führen kann [JM09].
Deshalb werden auch nicht die in Abschnitt 2.3.2 genannten Versprecher, Stotterer und andere verbale Unterbrechungen aus der Klasse der entfernbaren Disfluenzen korrekt erkannt, da sich diese stattdessen in falsch erkannten Wörtern äußern. Außerdem gibt der verwendete Spracherkenner auch keine zuverlässigen Informationen zur Interpunktion und Satztrennung aus.

Unabhängig vom Spracherkenner lassen sich auch keine Disfluenzen der Klasse der unkorrigierten Disfluenzen identifizieren und auflösen, da im Projekt bisher keine Instanz existiert, die die Anweisungen auf grammatikalische Korrektheit überprüft und aus dem Kontext heraus fehlende Wörter detektiert. Dadurch können keine Sprachfehler, Wortauslassungen und falsche Anordnungen von Wörtern erkannt werden.

Neben der fehlenden Kontrolle bezüglich der grammatikalischen Korrektheit gibt es des Weiteren auch keine anderen Anforderungen an die Eingabe, außer dass diese auszuführende Aktionen beinhalten muss. So kann eine Eingabe auch nur aus einem Verb bestehen wie beispielsweise im Fall „*go*“. Demnach können alle andere Satzteile beliebig entfallen und die Eingabe muss keine Strukturen einhalten.

Im bisherigen Projektverlauf handelte es sich bei den Eingaben um englische Befehlssequenzen an den Küchenroboter ARMAR-III. Auch in dieser Arbeit werden Eingaben aus

dieser Domäne als Testdaten verwendet. Diese Eingaben sind üblicherweise in Befehlsform gestellt und bestehen aus mehreren Instruktionen. Die Instruktionen können dabei einerseits durch signalisierende Wörter wie die Konjunktion „*and*“ von einander getrennt sein und dadurch aus mehreren syntaktisch und inhaltlich abgeschlossenen Teilsätzen bestehen wie im folgen Fall:

(6) „*Go to the fridge and open the fridge and close it*“

Jedoch können die Instruktionen andererseits auch einfach aneinander gereiht sein, ohne signalisierende Wörter zwischen ihnen:

(7) „*Go to the fridge open the fridge close it*“

Demnach darf nicht von einer in Sätze segmentierten Eingaben ausgegangen werden. Des Weiteren sind selbst in der bisher verwendeten Anwendungsdomäne nicht alle Eingaben in Befehlsform-Form. So können innerhalb der Eingabe jederzeit Teilsätze wie „*I am thirsty*“ vorkommen, die keine Instruktionen sind und demnach sich nicht in Befehlsform befinden. Generell sollte der Lösungsentwurf nicht von bestimmten Wortformen und Formulierungen innerhalb einer bestimmten Domäne abhängen, da ein Entwurfsideal des PARSE-Projektes die domänenunabhängige Arbeitsweise der Agenten ist.

5.2. Anforderungen an den Entwurf

Aus der allgemeinen Analyse der Problemstellung sowie deren Einschränkungen lassen sich einige Anforderungen herleiten, die der Entwurf eines möglichen Lösungsansatzes erfüllen muss:

- Erkennen von disfluenten Stellen:
  Der Lösungsansatz muss disfluente Stellen in der Eingabe erkennen können. Für jedes Wort muss annotiert werden, ob es sich innerhalb einer Disfluenz befindet und zu welchem Bestandteil der Disfluenz es zugehörig ist.

- Korrekte Klassifizierung:
  Im Lösungsansatz muss eine Differenzierung zwischen den unterschiedlichen Arten der Disfluenzen getroffen werden. So ist in der Äußerung „*open **uh** the fridge*“ die Disfluenz lediglich eine Verzögerung, während es sich bei der Äußerung „*bring [the orange juice + uh I mean the] water*“ um eine Reparatur handelt.

- Erkennung der Grenzen einer Reparatur:
  Falls eine Reparatur erkannt wird, müssen die genauen Wortgrenzen von Reparandum, Interregnum und Reparatur festgelegt werden. Dabei darf nicht davon ausgegangen werden, dass ein Unterbrechungspunkt durch eine gefüllte Pause, einen expliziten Bearbeitungsbegriff oder einen Diskursmarker signalisiert wird.
  Des Weiteren muss innerhalb einer Reparatur eine Verbindung zwischen den Bestandteilen Reparandum und Reparatur annotiert werden.

- Verarbeitung von langen Textsequenzen:
  Wie unter Abschnitt 5.1 angegeben, gibt der von PARSE verwendete Spracherkenner keine zuverlässigen Ausgaben zu der Interpunktion und den Satzgrenzen der Eingabe an. Deshalb erfolgt die Modellierung der Eingabe nicht durch eine Unterteilung in Sätzen, sondern als eine komplette, längere Eingabesequenz von Wörtern, die keine Satzzeichen enthält. Bspw. lautet eine mögliche Eingabesequenz (eine händische Transkription, die aus dem Korpus des Projektes entnommen wurde) wie folgt:

„*uhm go to the table uhm take the green cup uhm then go to the dishwasher open the dishwasher uhm put the green green cup ehm in the dishwasher and close the dishwasher*“

Ein möglicher Lösungsansatz muss demnach möglichst beliebig lange Eingabesequenzen verarbeiten können, die keinerlei Interpunktionszeichen besitzen.

5.3. Diskussion der Verwandten Arbeiten

In diesem Abschnitt werden die im Kapitel 4 vorgestellten Ansätze aus verwandten Arbeiten miteinander verglichen und hinsichtlich ihrer Eignung für die aktuelle Arbeit analysiert. Alle Angaben bezüglich dem F-Maß beziehen sich dabei auf die Klassifizierung des Reparandums.
Generell gilt, dass regelbasierte bzw. heuristisch regelbasierte Ansätze wie von Hindle [Hin83] und von Bear et al. [BDS92] unter sehr idealisierten Bedingen arbeiten. So erwartet Hindle, dass innerhalb einer Disfluenz der Unterbrechungspunkt mit einem Bearbeitungssignal markiert ist, was in realen Disfluenzen normalerweise nicht der Fall ist. Ein Unterbrechungspunkt kann zwar durch gewisse Wortarten wie gefüllte Pausen und Diskursmarker gekennzeichnet sein, jedoch sind diese optional und liegen nicht immer vor. Außerdem können mehrere Wörter innerhalb eines Interregnums zwischen Reparandum und Reparans liegen, wodurch die Regeln von Hindle entsprechend angepasst werden müssten.
Des Weiteren dürfen gemäß den Einschränkungen aus Abschnitt 5.1 für den Lösungsentwurf keine prosodischen und akustischen Merkmale einer Äußerung in Betracht gezogen werden, wodurch ein Unterbrechungspunkt auch nicht durch prosodische Anzeichen oder Wortfragmente markiert werden kann. Infolgedessen lassen sich die Regeln von Hindles Ansatz oft nicht zuverlässig alleine anwenden, da zunächst der Unterbrechungspunkts gefunden werden muss. Die Regeln von Bear et al. [BDS92] weisen ähnliche Probleme auf und sowohl Hindle als auch Bear et al. gehen mit Ausnahme von den Disfluenzen von wohlgeformten englischen Sätzen bezüglich deren Grammatik und Syntax aus. Innerhalb des Projektes PARSE kommt es jedoch wie unter den Anforderungen in Abschnitt 5.2 bereits erwähnt meistens nicht zu wohlgeformten Eingabesätzen, sondern zu Reihungen von Befehlen innerhalb einer Eingabesequenz, die nicht in Sätze unterteilt ist und einige Fehler enthalten kann wie beispielsweise folgendes Beispiel:

(8) „*Go to fridge take some water bring water...*“

Insgesamt lässt sich demnach schlussfolgern, dass reine regelbasierte Ansätze alleine nicht ausreichen für einen Lösungsansatz, jedoch eignen sie sich beispielsweise in Kombination mit auf maschinellen Lernverfahren basierenden Ansätzen. So können Bestandteile dieser Regeln wie beispielsweise der Vergleich von Wortarten und Phrasen innerhalb eines bestimmten Beobachtungsintervalls als Merkmale innerhalb eines maschinellen Lernverfahrens oder als Kriterien in einem nachträglichen Optimierungsverfahren nach dem Lernverfahren eingesetzt werden.

Mit maschinellen Lernverfahren lassen sich auch nicht explizite, durch bestimmte Wortarten gekennzeichnete Unterbrechungspunkte und damit Disfluenzen erkennen. Diese Lernverfahren benötigen zwar zusätzlich einen relativ großen Satz an annotierten Trainingsdaten, jedoch sind diese durch den Switchboard-Korpus gegeben, welcher auch in den meisten verwandten Arbeiten als Referenzkorpus verwendet wird. Unter diesen Ansätzen muss jedoch auf Verfahren, die auf syntaktische Zerteilung basieren, wie in [RT13, HJ14, YSM16],

verzichtet werden. Denn diese Verfahren verwenden oder basieren auf Dependenzgrammatiken und -graphen, was sich bereits in der Bachelorarbeit von Kocybik [Koc15] als nicht vielversprechend für die nicht in herkömmliche Sätze eingeteilte Eingaben des Projektes PARSE herausgestellt hat. Des Weiteren existieren wesentlich weniger annotierte Trainings- und Testdatensätze für solche Verfahren, als für die Sequenzmarkierung.

Bei den Sequenzmarkierungsverfahren haben sich Verfahren, die auf Conditional Random Fields (CRF) und rekurrenten neuronalen Netzwerken mit langen Kurzzeitgedächtnis (engl. long short-term memory - LSTM) als Lernmodelle basieren, als am erfolgreichsten bezüglich dem F-Maß herausgestellt. So weist ein CRF eine insbesondere gegenüber HMMs und Maximalen-Entropie-Modellen bessere Erkennungsrate der disfluenten Stellen und des Unterbrechungspunktes auf [LSS+06]. Insbesondere das von Ferguson et al. [FDK15] eingesetzte Semi-Markov-CRF ist erfolgreich beim Erkennen von Disfluenzen, da bei diesem Modell auch Merkmale auf Wortgruppen-Ebene und nicht nur auf Wort-Ebene betrachtet werden können. Ferguson et al. verwenden jedoch auch prosodische Merkmale, was in dieser Arbeit nicht möglich ist. Auch in den Arbeiten von [Geo09] und [ZOH14] werden CRFs mit Erfolg eingesetzt.

Alternativ ist auch das Max-Margin-Markov-Netzwerk (M_3N) aus [QL13] als maschinelles Lernmodell geeignet. Es weist einen ähnlich guten F-Wert wie das Semi-Markov-CRF auf, jedoch ist dafür wesentlich komplexer und es ist fraglich, wie gut sich das mehrstufige Verfahren des M_3N bei einer potentiell langen, nicht wohldefinierten Eingabe einsetzen lässt.

Ein LSTM hingegen, insbesondere ein bidirektionales LSTM, das die Eingabe sowohl vorwärts gerichtet Reihenfolge als auch in umgekehrte Reihenfolge erhält, weist je nach Konfiguration und Parameter eine noch bessere Erkennungsrate als herkömmlich CRFs auf [ZOH16]. Des Weiteren können LSTMs theoretisch auch komplexer aufgebaute und längere Disfluenzen erkennen. Um die Ergebnisse jedoch nicht nur von einem Modell abhängig zu machen, werden maschinelle Lernverfahren öfters miteinander kombiniert. So verwenden Zayats et al. [ZOH16] und [WCLL16] eine Kombination aus CRF- und LSTM-Modellen. Zayats et al. verwendet jedoch Wortfragmente als Eingabemerkmal, was innerhalb dieser Bachelorarbeit nicht möglich ist. Ein andere Kombinationsmöglichkeit findet sich in der Verwendung eines Rauschkanalmodells und eines effektiven Sprachmodells [CJ01, JC04, LJ17]. Ein Rauschkanal, insbesondere wie in den betreffenden Arbeiten durch eine Baumadjunktions-Grammatik formalisiert, hat jedoch Probleme mit evtl. sehr langen und nicht in normalen Sätzen aufgebaute Eingaben, wie es in dieser Arbeit der Fall ist, da unter anderem der Aufwand bezüglich dem Finden und Erzeugen der nicht disfluenten Eingaben exponentiell mit der Länge der Eingabe ansteigt. Außerdem basiert der Erfolg eines Rauschkanals auf der starken Ähnlichkeiten und Wiederholungsrate zwischen Reparandum und Reparans, was bei Neustarts nicht gegeben ist, wodurch diese wesentlich schlechter und seltener bei einem Rauschkanal erkannt werden.

Ein inkrementeller Ansatz, wie beispielsweise der von Hough und Schlangen [HS15], in dem die Autoren ein RNN zur inkrementellen Disfluenzerkennung verwenden, ist für diese Bachelorarbeit ebenfalls nicht sinnvoll. Denn inkrementelle Ansätze versuchen so schnell wie möglich ein Wort bezüglich seiner Zugehörigkeit zu einer Disfluenz zu annotieren, ohne die komplette Eingabeäußerung zu betrachten, wodurch sie zwar eine schnellere Klassifizierung erreichen, die jedoch dafür eine wesentlich schlechtere Erkennungsrate aufweist. Da beim Projekt PARSE die Einhaltung bestimmter Latenzzeiten nicht von großer Bedeutung ist, sind inkrementelle Ansätze nicht sinnvoll.

Die besten Werte bezüglich dem F-Maß weisen unter den Sequenzmarkierungsverfahren die Ansätze von Lou und Johnson [LJ17] sowie Wang et al. [WCLL16] auf. Beide verwenden einen kombinierten Ansatz aus einem bidirektionalen LSTM und einem weiteren Lernverfahren. Wang et al. hingegen kombinieren ihr LSTM mit einer zusätzlichen CRF-Schicht. Es bietet sich demnach in einem möglichen Lösungsansatz für diese Bachelorarbeit an, ebenfalls einen kombinierten Ansatz zu verwenden, der insbesondere ein bidirektionales LSTM als eines der Modelle verwendet. Als zweites Lernmodell lässt sich beispielsweise ebenfalls ein CRF verwenden.

Beide Verfahren kombinieren insbesondere auch handgefertigte Regeln mit ihren maschinellen Lernansätzen, wodurch sie erfolgreich ihre Erkennungsrate steigern. Lou und Johnson [LJ17] realisieren diese zusätzliche Regeln durch einen nachträglichen Reranker, der die Ergebnisse der beiden Lernmodelle und der Regeln zur endgültigen Entscheidungsfindung verwendet. Wang et al. [WCLL16] verwenden ihre Regeln direkt als Eingabemerkmal in ihrem LSTM-Modell oder als zusätzliche Eingabe vor der CRF-Schicht neben der Ausgabe des bidirektionalen LSTMs.
Solch eine Kombination von regelbasierten und maschinellen Lernansätzen hat sich insbesondere auch als vielversprechend für die Anwendung in domänenfremden Testdaten erwiesen, wie beispielsweise in [GWG10], wo nach dem maschinellen Lernansatz eine nachträgliche auf Regeln basierende Optimierung durch ein ganzzahliges lineares Optimierungsverfahren (engl. integer linear programming - ILP) stattfindet. Demnach wäre auch bei einem Lösungsansatz für diese Bachelorarbeit eine nachträgliche auf Regeln basierende Optimierung beispielsweise durch ein ILP-Verfahren sinnvoll, da sich die späteren Testdaten des PARSE-Projektes in einer anderen Domäne befinden als die Trainingsdaten des Switchboard-Korpus.

Abschließend muss noch erwähnt werden, dass sich die meisten der genannten Ansätze lediglich mit der Markierung der Reparanda befassen. So wird die Klassifizierung von Interregna üblicherweise als nahezu trivial betrachtet, da sie nur aus gefüllte Pausen, explizite Bearbeitungsbegriffe sowie Diskursmarker bestehen, bei denen es sich jeweils um eine endlich Anzahl von Wörtern oder um leicht erkennbare Wörter handelt. Die Reparans der Disfluenzen werden auch nur explizit in [Geo09, GWG10, ZOH14, ZOH16, HS15] markiert, wobei sich herausgestellt hat, dass die Erkennungsrate für die Reparans üblicherweise wesentlich schlechter ausfällt. Es hat sich ebenfalls in diesen Arbeiten erwiesen, dass die Kombination von maschinellen Lernverfahren mit regelbasierten Ansätzen sich positiv auf die Erkennungsrate der Reparanda auswirkt. Innerhalb dieser Arbeit werden jedoch nicht nur die Reparanda, sondern auch die Reparans und die gefüllten Pausen, expliziten Bearbeitungsbegriffe sowie Diskursmarker annotiert.

5.4. Suche des Unterbrechungspunktes

Ein Unterbrechungspunkt des Sprechflusses signalisiert eine Disfluenz. Der Unterbrechungspunkt folgt dabei dem letzten Wort eines Reparandums. Das Lokalisieren eines Unterbrechungspunktes erleichtert dadurch auch die nachfolgende Klassifizierung und Festlegung der Wortgrenzen einer Disfluenz [JM09].
Die Lokalisierungsprozess ist dabei trivial, falls nach diesem eine gefüllte Pause, ein Diskursmarker oder ein expliziter Bearbeitungsbegriff auftritt. Die wichtigsten gefüllten Pausen wie „*uh*“ oder „*uhm*“ werden direkt vom IBM-Watson-Spracherkenner erkannt, einige andere gefüllte Pausen werden dabei mit dem Platzhalter „%HESITATION“ als Wert des Tokens versehen. In beiden Fällen annotiert die seichte Sprachverarbeitung von PARSE gefüllte Pausen meistens als Interjektionen oder Fremdwörter bei den Wortarten, was die Suche nach diesen Wörtern weiterhin erleichtert.

Für Diskursmarker wie z.B. „*well*“ und explizite Bearbeitungsbegriffe wie „*I mean*“ lässt sich eine Schlüsselwortsuche durchführen, da in beiden Fällen die Anzahl der zugehörigen Wörter sehr begrenzt ist. Falls keine der drei Wortgruppen vorhanden ist, muss der Unterbrechungspunkt durch einen Abgleich von Wortmustern und syntaktischen Korrespondenzen zwischen möglichen Reparandum und Reparans lokalisiert werden.
Bei diesen Abhängigkeiten und Korrespondenzen handelt es sich üblicherweise um Wiederholungen bezüglich den eigentlichen Wörtern und Wortarten. Der Einfachste Fall ist dabei die direkte Wiederholung von einem Wort wie beispielsweise „ [*the + the*]“, bei der ein Unterbrechungspunkt offensichtlich zwischen den beiden Wörtern liegt. Etwas komplexer ist eine Wiederholung von mehreren Wörtern bzw. Wortfolgen wie „ [*go to the + go to the*]“, da dabei die Anzahl der zu beobachteten Wörter und damit die Fehleranfälligkeit steigt. Noch komplexer ist es, falls die Wörter im möglichen Reparans nicht direkt wiederholt werden, sondern es im Reparans zu Ersetzungen, Entfernungen oder Einfügungen von neuen Wörtern kommt wie im Folgenden beispielhaft dargestellt:

(9) „[*he + she*]“

(10) „[*move the + turn around and move the*] *green block*“

Beispiel 9 ist hierbei eine simple Ersetzung, bei der dennoch die gleiche Wortart beibehalten wird, nämlich ein Personalpronomen. Das Beispiel verdeutlicht, dass neben der reinen Wortwiederholung auch die Wortart bei der Bestimmung des Unterbrechungspunktes relevant ist. Im Beispiel 10 hingegen kommt es zu mehrfachen Einsetzungen von neuen Wörtern bevor das Verb „*move*“ im Reparans wiederholt wird. Die generelle Festlegung, ob ein Unterbrechungspunkt vorliegt, ist zwar hierbei durch die Wiederholung des Verbs „*move*“ sowie des Artikels „*the*“ leicht zu treffen, jedoch ist beispielsweise die genaue Position des Unterbrechungspunktes für eine automatische Suche nicht so einfach erkennbar.

So würde eine naive Suche nach dem Unterbrechungspunkt diesen evtl. direkt vor dem ersten wiederholten Wort setzen, nämliche das Wort „*move*“, anstatt diesen korrekt vor dem Verb „*turn*“ zu setzen. Das Setzen des Punktes vor dem wiederholten „*move*“ würde zwar nach der Korrektur die syntaktisch richtige Aussage „*move the green block*“ ergeben, jedoch würde dadurch die neue Information „*turn around and*“ verloren gehen, dessen Einsetzung bei nachträglicher Auswertung der Äußerung sich als eigentlicher Anlass für die Reparatur festlegen lässt. Demnach würde durch das Entfernen der Wörter „*turn around and*“ aus dem Korrekturvorschlag die eigentlich geplante Aussage des Sprechers verfälscht werden. Es würde hierbei eine komplette Instruktion entfernt werden, was insbesondere für die Kommunikation zwischen Mensch und Maschine, wie beim Projekt PARSE, schwerwiegende Auswirkungen haben kann.
Beispiel 10 verdeutlicht hierbei auch, dass neben den reinen Wörtern und Wortarten auch Informationen auf Ebene der Phrasen hilfreich sein können. So ist das erste „*the*“ der Beginn einer Nominalphrase, die nicht vollständig ist, d. h. keinen Kopf der Phrase besitzt. Dies kann ebenfalls als ein Anzeichen dafür interpretiert werden, dass zumindest eine korrigierende Phrase gleichen Typs für das „*the*“ folgen wird und das erste „*the*“ fehlerhaft ist, womit die Suche nach dem Unterbrechungspunkt eingeschränkt werden kann. Informationen auf Ebene der Phrasen wurden bereits im regelbasierten Ansatz von Hindle [Hin83] eingesetzt. Dabei ging Hindle jedoch auch von wohldefinierten vollständigen Sätzen sowie einem bereits vorhanden Unterbrechungspunkt aus, was hier in dieser Arbeit nicht der Fall ist. Es zeigt sich am Beispiel auch insbesondere, dass ohne kennzeichnende Wortgruppen wie gefüllte Pausen die Suche nach dem Unterbrechungspunkt sowie die Festlegung der Wortgrenzen der zugehörigen Reparatur simultan ablaufen.

Solch eine Untersuchung auf Wiederholungen bezüglich der Wortmuster wird dabei komplexer, je mehr Wörter bzw. Wortarten sich nicht im Reparans wiederholen und je weiter die Wiederholungen voneinander entfernt sind. Des Weiteren ist auch je nach Situation zu bestimmen, ob es sich bei der Wiederholung wirklich um eine Disfluenz handelt oder nicht. So liegt im folgenden Beispiel trotz der wiederholten Wörter „*go to*“ keine Disfluenz vor:

(11) „*go to table go to the fridge go to the door*“

Beispiel 11 ist hierbei eine Folge von Instruktionen, wie sie beispielsweise innerhalb des Projektes PARSE als Eingabe erfolgen kann. Mit solch einer Eingabe muss wie bereits in Abschnitt 5.2 erwähnt ein potentielles Lösungsverfahren umgehen können. Dadurch entfallen jegliche Merkmale und Regeln bei der Suche nach disfluenten Strukturen, die wohldefinierte Sätzen als Eingabe benötigen. Die drei Instruktionen unterscheiden sich lediglich bezüglich einem Nomen und sind ansonsten auf Wort- und Wortartenebene identisch. Da auch keine für eine Disfluenz typische Wortgruppe im Beispiel vorkommt, ist es für ein mögliches Lösungsverfahren schwer, das Beispiel korrekt zu analysieren, denn die Wahrscheinlichkeit für eine Fehlentscheidung ist hier aufgrund der mehrfachen Wortwiederholungen sehr hoch. Ein mögliches Anzeichen für eine fehlende Disfluenz könnte hierbei die Vollständigkeit jeder Phrase sein. Es ist dabei noch anzumerken, dass sehr häufige Wörter wie bestimmte und unbestimmte Artikel oder Konjunktionen, falls sie sich nicht direkt wiederholen, zur Festlegung einer Disfluenz eher ungeeignet sind, da sie zu oft Bestandteile von normalen Satzteilen sind.

In maschinellen Lernverfahren werden solche Zusammenhänge innerhalb eines kurzen Beobachtungsintervalls untersucht und basierend darauf die Entscheidung bezüglich des Unterbrechungspunktes getroffen. Die Lernverfahren entwickeln durch den Musterabgleich mit einem großen Satz an korrekt annotierten Trainingsdaten eigene Kriterien und Zusammenhänge, die jedoch den in diesem Abschnitt untersuchten Ansatzpunkten ähneln. Die Lokalisierung des Unterbrechungspunktes erfolgt dabei meistens simultan zur Festlegung der Grenzen der Disfluenz. Als Eingabemerkmale werden in den Lernverfahren typischerweise die Wörter sowie die zugehörigen Wortarten verwendet. Jedoch wäre auch der Einsatz der Phrasen-Markierungen sinnvoll.

5.5. Klassifizierung der Disfluenz

Falls ein Unterbrechungspunkt und damit eine Disfluenz erkannt wird, muss diese entsprechend je nach ihrer Art behandelt werden. Wie in Abschnitt Abschnitt 5.1 bereits erwähnt können unkorrigierte Disfluenzen weder zuverlässig identifiziert noch korrigiert werden, denn diesen Disfluenzen fehlt ein Reparans und damit eine Korrektur durch den Sprecher. Wegen diesen fehlenden Reparans lässt sich weder durch einen regelbasierten noch durch einen maschinellen Lernansatz zuverlässig ermitteln, ob ein Reparandum und damit fehlerhafte Wörter vorliegen. Und dadurch entfällt zumindest in dieser Arbeit eine mögliche Korrektur solcher Disfluenzen.

In Abschnitt 5.1 wurde ebenfalls bereits erwähnt, dass aufgrund der Rahmenbedingungen im Projekt PARSE keine entfernbaren Disfluenzen erkannt werden, die nur aufgrund von prosodischen Merkmalen identifizierbar sind, wie beispielsweise Disfluenzen ausgelöst durch Stottern oder Wortfragmente. Damit verbleiben die übrigen entfernbaren Disfluenzen: Diskursmarker, explizite Bearbeitungsbegriffe und gefüllte Pausen.

Diese drei Gruppen sind wie im letzten Abschnitt bereits erwähnt aufgrund ihrer Wortarten und endlicher Anzahl an zugehörigen Wörtern leicht zu erkennen. Des Weiteren markieren sie implizit einen Unterbrechungspunkt und signalisieren damit eine Disfluenz. Sie enthalten zwar auch kein Reparans, jedoch ist bei ihnen die Korrektur implizit: sie müssen lediglich entfernt werden. Deshalb genügt es zur Korrektur, diese Wörter mit ihren Bezeichnungen zu annotieren. In den folgenden Beispielen sind jeweils entsprechend gefüllte Pausen, Diskursmarker und explizite Bearbeitungsbegriffe dargestellt:

(12) „*armar give me* ***uh*** *the orange juice and the* ***uh*** *water bottle*“

(13) „***well*** *give me the bottle and* ***anyway*** *go to the fridge*“

(14) „*ok armar go [to the dishwasher +* ***I mean*** *to the fridge]*“

Die Beispiele 12 und 13 verdeutlichen eine der beiden üblichen Situationen, in denen entfernbare Disfluenzen auftreten: sie dienen zur Verzögerung und als Pause. Gefüllte Pausen treten auf, wenn der Sprecher unsicher ist und Zeit zum Nachdenken und zur Formulierung seiner weiteren Aussage benötigt. Es wird vermutet, dass besonders viele gefüllte Pausen auftreten, falls der Sprecher bei der Formulierung seiner Aussage zu viele Optionen besitzt und zwischen ihnen auswählen muss [SCRB91]. Jedoch ist es auch möglich, dass gefüllte Pausen durch eine zu starke Ungewissheit des Sprechers entstehen, falls dieser zu wenig oder keine Sprechoptionen besitzt, aber seine Äußerung fortsetzen will. Außerdem tauchen sie auch auf, falls der Sprecher entweder zu viele Ideen zur aktuellen Thematik oder zu wenige besitzt. Manche Menschen besitzen jedoch einfach die Angewohnheit beim Sprechen gefüllte Pausen von sich zu geben.
Diskursmarker besitzen eine ähnliche Funktion bezüglich Verzögerungen, jedoch dienen sie auch zwischen Strukturierung eines Diskurses und stellen eine Verbindung zur vorherige Diskurseinheit und der aktuellen Äußerung her [Buß02]. Sie besitzen jedoch ähnlich wie gefüllte Pausen wenig semantische Bedeutung.

Beispiel 14 zeigt die zweite übliche Situation, in der entfernbaren Disfluenzen auftreten: Sie trennen als Interregna innerhalb einer Reparatur die Reparanda und Reparans voneinander. Auch in diesem Fall dienen sie zur Verzögerung. Hierbei treten sie auf, falls der Sprecher bemerkt, dass er einen Fehler gemacht hat und Zeit zur Formulierung seiner Korrektur benötigt. Dabei unterbricht der Sprecher mittels ein oder mehrerer entfernbaren Disfluenzen seinen Sprechfluss. Die Reparatur wird genauer im nächsten Abschnitt betrachtet.
Es ist nun für jede entfernbare Disfluenz zu entscheiden, ob diese als alleinige Verzögerung oder als Interregnum innerhalb einer Reparatur dient. Beispiel 14 dient hierbei zur Veranschaulichung von expliziten Bearbeitungsbegriffen. Diese implizieren eine Reparatur, denn sie beschreiben, dass ein Fehler gemacht wurde wie beispielsweise „*I am sorry*“ oder dass etwas genauer definiert wird wie „*I mean*“. Deshalb befinden sich explizite Bearbeitungsbegriffe nahezu immer innerhalb eines Interregnums.

Für gefüllte Pausen oder Diskursmarker muss jedoch einzeln entschieden werden, ob sie sich innerhalb eines Interregnums befinden. Ein stichhaltiges Anzeichen für eine Reparatur ist das auftreten von entfernbaren Disfluenzen zwischen gleichen Wörtern, also zwischen Wiederholungen, befindet. Ersatzweise zu Wortwiederholungen kann es auch ein ähnliches syntaktisches Muster bezüglich der Wortarten sein. Speziell bei einer gefüllten Pause kann ein Anzeichen gegen eine Reparatur sein, falls sich die gefüllte Pause innerhalb einer Nominalphrase befindet und diese nach der Pause direkt fortgesetzt und abgeschlossen wird.

Ansonsten lässt sich mit einem maschinellen Lernverfahren und passendem Datensatz, die Unterscheidung bezüglich reiner Verzögerung und Reparatur erlernen und diese gleichzeitig neben der Bestimmung des Unterbrechungspunktes und der Wortgrenzen der Reparatur festlegen.

5.6. Bestimmung der Wortgrenzen einer Reparatur

Die Erkennung von Reparaturen und deren Wortgrenzen ist der Hauptgegenstand der aktuellen Forschung bezüglich der Erkennung von Disfluenzen. Deshalb ist der Kern dieser Arbeit auch deren möglichst genaue Erkennung. Falls der Unterbrechungspunkt durch ein signalisierendes Wort impliziert wird und damit bereits lokalisiert ist erleichtert dies die genaue Bestimmung der Wortgrenzen der Reparanda und Reparans. Dies soll am folgenden Beispiel verdeutlicht werden.

(15)

„*hello please* [$\underbrace{\textit{go to the fridge}}_{\text{Reparandum}}$ + $\overbrace{\textit{uh no uh yes}}^{\text{Interregnum}}$ $\underbrace{\textit{go to the fridge}}_{\text{Reparans}}$] *and...*“

Das Interregnum einer Reparatur ist leicht zu begrenzen, da es immer vom Unterbrechungspunkt bis zum ersten Wort, das nicht einer entfernbaren Disfluenz entspricht, verläuft. Das Interregnum besteht aus einer Kombination von entfernbaren Disfluenzen. Die Hauptaufgabe ist die Bestimmung des Reparandums und Reparans. Die beiden Bestandteile sind dabei so festzulegen, dass das Reparandum sich komplett durch das Reparans ersetzen lässt und eine syntaktisch korrekte bzw. sinnvolle Äußerung verbleibt. Wie bereits in 5.4 angesprochen, dienen dabei Wörter des Reparandums, die sich im Reparans wiederholen, als Orientierungshilfe für die Festlegung beider Bestandteile.
Beispiel 15 zeigt hierbei die einfachste Form einer Reparatur: eine komplette Wiederholung des Reparandums im Reparans. Noch simpler wäre lediglich die Version, bei der Reparandum und Reparans nur aus einem Wort bestehen:

„*hello please* [*go* + *uh no uh yes go*] *to the fridge and...*“

In beiden Versionen sind die Grenzen leicht zu bestimmen, da jedes Wort ohne Veränderung wiederholt wird. Komplexer wird es, falls im Reparans nur eine Teilmenge der Wörter des Reparandums wiederholt wird. Dafür werden stattdessen neue Wörter eingefügt, bisherige Wörter entfernt oder durch neue ersetzt. Dies ist entsprechend in folgenden Beispielen dargestellt.

(16) „*hello please bring me* [*the cup* + *uh no the red cup*] *and...*“

(17) „*hello please bring me* [*the small green cup* + *uh no the small cup*] *to the fridge and...*“

(18) „*hello please bring me* [*the red cup* + *uh no the green cup*] *and...*“

Anhand der Beispiele wird deutlich, dass zur Bestimmung der Grenzen nach Korrespondenzen zwischen beiden Teilen gesucht werden muss. Diese Korrespondenz kann sich direkt in einer Wortwiederholung oder in einer gleichen Wortart äußern. Die Grenzen muss nun so gesetzt werden, dass auf beiden Seiten die korrespondierenden Wörter jeweils umfasst werden. Die Suche nach solchen Korrespondenzen sollte mit dem letzten Wort vor dem

Unterbrechungspunkt beginnen und falls im Reparans ein korrespondierendes Wort gefunden wird, sind damit zunächst die Grenzen des Reparans bestimmt. Nun muss das Reparans umgekehrt entlang gelaufen werden und für jedes Wort eine Korrespondenz auf Seiten des Reparandums gefunden werden, um damit dessen linke Wortgrenze festzulegen. Wie Beispiel 16 verdeutlicht, können im Reparans jedoch auch neue Wörter eingefügt sein, zu denen es keine Korrespondenz gibt. Deshalb muss festgelegt werden, in welchem Beobachtungsintervall nach Korrespondenzen gesucht wird und ob die erste passende Korrespondenz ausgewählt wird. Als Beobachtungsintervall bietet sich hierbei eine Suche auf phrasaler Ebene an. Es sollten zunächst die ersten Phrasen, die einen Unterbrechungspunkt umgeben, untersucht werden. Eine Korrespondenz auf phrasaler Ebene schränkt dabei die weitere Suche nach Korrespondenzen auf Wortebene ein. Bei der Art der Korrespondenz ist hierbei die Wiederholung des Worts gegenüber der wiederholten Wortart zu bevorzugen. So lässt sich die erste Wortwiederholung als Korrespondenz ansehen, jedoch kann, falls sich innerhalb der korrespondierenden Phrase keine solche Wiederholung finden lässt, auch die gleiche Wortart als Korrespondenz angesehen werden.

In den drei oberen Beispielen sind die Korrespondenzen die Wörter „*the*“ und „*cup*“. Diese werden direkt im Reparans wiederholt und schaffen dadurch eine Korrespondenz zwischen beiden Bestandteilen.
In Beispiel 16 wird nun ein neues Wort zwischen den beiden Korrespondenzen eingefügt. Es wird dadurch deutlich, dass alle eingefügten Wörter zwischen Wörtern im Reparans, die bereits Korrespondenzen zum Reparandum besitzen, ebenfalls zum Reparans gehören. Beispiel 17 verdeutlicht die Problematik bei Wortentfernungen, nämlich dass einige Wörter des Reparandums keine Korrespondenzen im Reparans besitzen. Auch hierbei orientiert sich die Bestimmung der Wortgrenzen anhand der Wörter im Reparans, die Korrespondenzen aufweisen. Für Wortersetzungen wie in Beispiel 18 ist dies analog. Dabei werden Wörter des Reparandums durch neue Wörter im Reparans ersetzt, jedoch weisen die neuen Wörter meist die gleiche Wortart auf, wodurch sich dennoch Korrespondenzen zwischen den alten und neuen Wörtern finden lassen.

Wie bei der Bestimmung des Unterbrechungspunktes erschwert sich die Wortgrenzenbestimmung, falls keine signalisierende Wörter wie gefüllte Pausen in der Disfluenz auftreten. In diesem Fall muss die Bestimmung des Unterbrechungspunktes sowie der Wortgrenzen simultan erfolgen. Die Eingabe ist demnach auf syntaktische Irregularitäten und Wiederholungen bezüglich Wörtern bzw. Wortmustern zu überprüfen. Es treten dabei die bereits im Abschnitt 5.4 angesprochenen Probleme auf, insbesondere falls die Eingabe aus einer Folge von simplen Instruktionen besteht, wodurch häufig Wort- und Wortartenwiederholungen auftreten, die keinen Zusammenhang zu Disfuenzen aufweisen.
In maschinellen Lernverfahren werden all diese Punkte berücksichtigt und es werden je nach Verfahren und Lernmodell eigene Schlüsse und Zusammenhänge aus den annotierten Trainingsdaten gebildet, die evtl. auch sehr abhängig von der verwendeten Domäne sein können.

5.7. Problematische Disfluenzen

Bei der korrekten Erkennung und Klassifizierung von Disfluenzen existieren zwei Sonderfälle, die häufig zu Problemen führen: Neustarts und komplexe Disfluenzen. Diese werden in den folgenden Abschnitten näher erläutert.

5.7.1. Neustarts

Wie bereits in den Grundlagen Abschnitt 2.3.2 definiert, ist ein Neustart im Rahmen dieser Arbeit eine Disfluenz, bei der im Reparans kein Wort des Reparandums wiederholt wird.

Ein Beispiel hierfür ist im Folgenden dargestellt.

(19) „*please bring me [the red cup + uh no a green bottle] and...*“

Das Beispiel verdeutlicht hierbei, dass es dennoch Korrespondenzen bezüglich der Wortart und eventuell auch der bezüglich der Phrase geben kann. Dieses Beispiel stellt demnach einen simplen Neustart dar, welcher sowohl von maschinellen Lern- als auch von regelbasierten Verfahren erkannt werden kann. Problematisch wird es jedoch, falls keine Korrespondenzen existieren.

(20) „*please [bring me a red + uh no I am thirsty and need orange juice] and...*“

In Beispiel 20 wird im Reparandum eine Instruktion begonnen, die im Reparans verworfen wird und es beginnt eine neue Handlungsfolge, die unabhängig von der vorherigen ist. Es existieren keine wirklichen Korrespondenzen bis auf einzelne Wiederholungen von Wortarten, die in dieser Konstellation nicht als Teil einer Disfluenz erkannt werden. Noch weiter erschwert wird die Erkennung, falls die Disfluenz kein Interregnum besitzt, wodurch meistens die Disfluenz gar nicht erkannt wird, da keine signalisierenden Wörter vorhanden sind. Solche Fälle lassen sich nicht von regelbasierten und auch nur sehr schwer von maschinellen Lernansätzen erfassen. Generell gilt dass, je weniger Wörter sich im Reparans wiederholen, desto schlechter ist die Erkennungsrate. Dies gilt insbesondere für einen wie im Beispiel 20 dargestellten Neustart. Neustarts führen demnach zu einer erheblichen Verminderung der Erkennungsrate falls sie häufig auftreten und keinerlei Korrespondenzen aufweisen.

5.7.2. Komplexe Disfluenzen

Komplexe Disfluenzen sind laut Shriberg [Shr94] Disfluenzen, die mehr als einen Unterbrechungspunkt aufweisen. Sie lassen sich in zwei Arten aufteilen: verkettete (engl. chained) und eingebettete (engl. nested) Disfluenzen.

Verkettete Disfluenzen sind mehrere aneinander gehängte Disfluenzen. Dabei ist mit Außnahme des letzten Reparans jedes Reparans einer Disfluenz gleichzeitig auch ein Reparandum der nachfolgenden Disfluenz. Dies ist im folgenden Beispiel ersichtlich.

(21) „*please bring me [a red + a red + a green cup + uh no a green bottle] and...*“

Am Beispiel wird deutlich, dass eine verkettete Disfluenz nacheinander von links nach rechts aufgelöst wird und Korrespondenzen zwischen benachbarten Disfluenzen vorliegen müssen. Die Korrektur des Beispiels 21 lautet nun:

(22) „*please bring me a green bottle and...*“

Im Gegensatz dazu handelt es sich bei eingebetteten Disfluenzen um Disfluenzen innerhalb einer anderen Disfluenz bzw. um eine Reparatur innerhalb einer anderen Reparatur. Die eingebettete Reparatur kann dabei innerhalb des Reparandums oder innerhalb des Reparans liegen. Ein Beispiel für eine eingebettete Reparatur innerhalb des Reparans ist dabei im folgenden dargestellt.

(23) „*please bring me [a red cup + uh no a green [cup + uh bottle]] and...*“

Die Disfluenzen werden hierbei zunächst von innen nach außen und dann nacheinander von links nach rechts aufgelöst. Beide Arten von komplexen Disfluenzen erschweren die erfolgreiche Klassifizierung, insbesondere falls sie eine komplexe bzw. längere Struktur aufweisen.

6. Entwurf und Implementierung

In diesem Kapitel wird der Lösungsentwurf sowie dessen Realisierung beschrieben. Zunächst werden dabei die Erkenntnisse und wichtigsten Punkte der Analyse nochmals näher betrachtet. Anschließend wird der entworfene Klassifikator, der schätzt, ob ein Wort disfluent ist oder nicht, gemäß seinem Aufbau und seinen inneren Komponenten beschrieben. Danach wird der interne Repräsentation des Lösungsentwurfs näher betrachtet und der allgemeine Programmablauf beschrieben. Abschließend wird noch die Abspeicherung der Ergebnisse dargestellt.

6.1. Erkenntnisse aus der Analyse

In diesem Abschnitt werden nochmals die wichtigsten Punkte aus der Analyse (siehe Kapitel 5) hinsichtlich ihrer Bedeutung für den Lösungsentwurf betrachtet.
Die Erkennung und Korrektur von sprachlichen Disfluenzen wurde in der Analyse in drei Aufgaben aufgeteilt, die auch den ersten drei in Abschnitt 5.2 definierten Anforderungen an das zu entwickelnde Werkzeug entsprechen:

1. Die Identifizierung des Unterbrechungspunktes
2. Die Klassifizierung der Disfluenz
3. Die Festlegung der Wortgrenzen der drei grundlegenden Bestandteile einer Disfluenz: Reparandum, Interregnum und Reparans (siehe 2.3.1)

Falls die zu untersuchende Spracheingabe gefüllte Pausen, Diskursmarker oder explizite Bearbeitungsbegriffe enthält, lassen sich die entsprechenden Unterbrechungspunkte direkt identifizieren, da die drei genannten Gruppen meistens Indikatoren für eine Disfluenz sind. In diesem Fall würde der Unterbrechungspunkt direkt vor der entsprechenden Phrase liegen. Die drei Gruppen selbst sind einfach zu identifizieren, da es sich jeweils um eine endliche und feste Anzahl an Phrasen handelt. Falls keine dieser Gruppen in der Eingabe vorkommt, lassen sich die drei Prozesse nicht unabhängig voneinander lösen und müssen demnach simultan ausgeführt werden.
In den Abschnitten 5.4, 5.5 sowie 5.6 wurde bereits erläutert, dass zur Lösung der drei Prozesse die vorhanden Wörter auf Korrespondenzen innerhalb eines geeigneten Beobachtungsintervalls untersucht werden müssen. Diese Korrespondenzen können sich als wörtliche Wiederholungen sowie auch als Wiederholungen auf Ebene der Wortarten und Wortgruppen äußern, wobei der wörtlichen Wiederholung eine größere Gewichtung zuzuordnen

ist. Auf Wortebene kann alternativ zur Wiederholung auch auf Ähnlichkeit hin untersucht werden, wobei ein solches Ähnlichkeitsmaß genau definiert werden muss. Die Suche nach solchen Korrespondenzen ist jedoch von einem Wort ausgehend beidseitig auszuführen, das bedeutet, dass jeweils sowohl vergangene als auch nachfolgende Wörter, also die Umgebung des Wortes betrachtet werden muss.
Des Weiteren ist jedoch darauf zu achten, dass einige Wörter wie beispielsweise die Konjunktion *„and“* häufig im normalen Sprachgebrauch verwendet und wiederholt werden, ohne dass es sich um eine Disfluenz handelt. Die Untersuchung auf Disfluenzen hin wird unter anderem auch durch Spezialfälle wie Neustarts und komplexe Disfluenzen (siehe Abschnitt 5.7) erschwert. Ein anderes Problem sind weitreichende Disfluenzen. Damit sind Reparaturen gemeint, die Abhängigkeiten und Korrespondenzen aufweisen, die eine größere Anzahl an Wörter überspannen. Solche Disfluenzen und deren weitreichende Abhängigkeiten werden oft nicht korrekt erkannt.
Unter den möglichen Lösungsverfahren haben sich in Kapitel 4 und Abschnitt 5.3 die maschinelle Lernansätzen als besser erwiesen, da sie eine wesentlich größere Erkennungsrate aufweisen beziehungsweise ihr F1-Wert wesentlich höher ist. Zwar verwenden die regelbasierten Verfahren aus Kapitel 4 andere Korpora als die maschinellen Lernverfahren, jedoch ist dennoch eine deutliche positive Tendenz für maschinelle Lernverfahren zu erkennen. Diese Verfahren können auch mit realistischeren Bedingungen, das heißt beispielsweise ohne ein festes Bearbeitungssignal wie bei [Hin83], arbeiten und benötigen auch nicht unbedingt wohldefinierte ganze Sätze. Zusammengefasst eignen sich maschinelle Verfahren wesentlich besser, um solch unregelmäßigen Phänomene wie Disfluenzen zu erkennen. Insbesondere komplexe Disfluenzen und Neustarts lassen sich zwar auch mit einem maschinellen Verfahren nicht unbedingt optimal erfassen, jedoch können sie theoretisch mit diesen Spezialfällen besser umgehen als regelbasierte Ansätze. Auch sind sie weniger von den drei Gruppen gefüllte Pausen, Diskursmarker und explizite Bearbeitungsbegriffe abhängig, in dem bezüglich dem F1-Wert besten Lernverfahren [WCLL16] werden die meisten gefüllten Pausen sogar komplett aus den Trainings- und Testdatensätzen entfernt.
Unter den maschinellen Lernverfahren hatten Sequenzmarkierungsverfahren, die auf einem rekurrenten neuronalen Netz mit langen Kurzzeitgedächtnis (engl. long short-term memory - LSTM) basieren, die besten Ergebnisse bezüglich dem F1-Wert erzielt. LSTMs können theoretisch auch Disfluenzen mit weitreichenden Abhängigkeiten erfassen [WCLL16]. Das LSTM muss dabei bidirektional arbeiten, damit Abhängigkeiten auf beiden Seiten eines Wortes erfasst werden können. Als Eingabemerkmale lassen sich leicht pro Wort das Wort selbst sowie dessen Wortart und Wortgruppe verwenden. Die Umgebung eines Wortes kann beispielsweise durch einen diskreten Vektor wie er in [WCLL16] eingesetzt wird als zusätzliches Merkmal verwendet werden. Die drei in der Analyse definierten Aufgaben werden von einem maschinellen Lernverfahren normalerweise simultan ausgeführt. Die vierte Anforderung, dass das Verfahren mit langen Eingabesequenzen zurecht kommen muss, lässt sich beispielsweise durch die Verwendung von möglichst langen Eingabesequenzen in den Trainingsdaten erfüllen. Der Kern des Lösungsentwurfs ist demnach ein auf einem bidirektionalen rekurrenten neuronales Netzwerk mit langem Kurzzeitgedächtnis (engl. long short-term memory - LSTM) basierender Klassifikator.

6.2. Der Klassifikator

Der Kern des Lösungsentwurfs ist ein auf einem bidirektionalen rekurrenten neuronales Netzwerk mit langem Kurzzeitgedächtnis (engl. long short-term memory - LSTM) basierender Klassifikator. Dieser Klassifikator führt dabei eine Sequenzmarkierung durch, das heißt das jedes Wort einer Eingabesequenz einer Ausgabeklasse zugewiesen wird. Die Ausgabeklassen entsprechenden hierbei den verschiedenen Fällen von Disfluenzen beziehungsweise der Position innerhalb der Disfluenz. Das neuronale Netzwerk sowie die dafür

erforderliche Funktionalität wird hierbei durch die Programmbibliothek Deeplearning4j (DL4J) [dl4a] implementiert. Es handelt sich dabei um eine freie, plattformübergreifende Bibliothek, die sich auf neuronale Netzwerke spezialisiert und eine Entwicklung komplett in Java ermöglicht. Die einzige Alternative in Java wäre hierfür die Java-Schnittstelle von TensorFlow [ten], diese wird jedoch zur Zeit noch als experimentell und nicht stabil angesehen.
Die folgenden Abschnitte beschreiben den entworfenen Klassifikator durch die verwendeten Datensätze, die Eingabemerkmale und Ausgabeklassen, den Aufbau des zugrunde liegenden neuronalen Netzwerks sowie durch die zusätzlichen Einstellungen und Parameter.

6.2.1. Verwendeter Datensatz: Der Switchboard-Korpus

Der Switchboard-Korpus [GHM92] dient für die meisten aktuellen maschinellen Lernansätze als grundlegender Datensatz für deren Trainings- und Evaluationsprozess. Dies trifft auch auf alle in Kapitel 4 vorgestellten maschinellen Lernverfahren, die auf Sequenzmarkierung basieren, zu. Aufgrund der häufigen Verwendung als Trainings- und Testdatensatz eignet sich dieser Korpus gut Vergleichsgrundlage zu anderen Verfahren. Der grundlegende Switchboard-Korpus besteht dabei aus ungefähr 2 400 Telefongesprächen zwischen jeweils zwei Leuten [JH93]. Die 543 Sprecher, von denen 302 männlich und 241 weiblich waren, stammen aus allen Regionen der Vereinigten Staaten. Jedes Gespräch handelt dabei über ein vorher bestimmtes Thema aus insgesamt 70 vorgegebenen Themen. Die Wahl der jeweiligen Themen und der Gesprächsteilnehmer wurde dabei so festgelegt, dass zum einen zwei Sprecher niemals mehr als einmal miteinander kommunizieren und jeder Gesprächspartner ein Thema nur einmal erhält.
In dieser Arbeit werden die durch das linguistische Datenkonsortium (engl. Linguistic Data Consortium - LDC) bereitgestellten Transkriptionen des Switchboard-Korpus verwendet, bei denen die Disfluenzen annotiert sind. Es sind ebenfalls diese annotierten Transkriptionen, die auch in den auf Sequenzmarkierung basierenden Verfahren des Kapitel 4 verwendet werden.

Der Datensatz wird gemäß [CJ01] in einen Trainings-, einen Entwicklungs- und einen Testdatensatz unterteilt. In den Datensätzen sind Disfluenzen wie im Abschnitt 2.3 vorgestellt annotiert. Die entwickelte Hilfsklasse `InputProcessor` besitzt einige Methoden mit denen diese Transkription über mehrere Verarbeitungsschritte hinweg in eine gesäuberten Darstellung umgewandelt werden. Dabei werden zunächst alle Annotationen, die in keiner Beziehung zu Disfluenzen stehen, sowie sonstige Trennzeichen und Anmerkungen entfernt. Beispielsweise werden Annotationen bezüglich akustischen Störungen herausgefiltert. Da die für das Projekt PARSE Kapitel 3 üblichen Texteingaben nicht in einzelne Sätze segmentiert sind und theoretisch sehr lang werden können, wird versucht, auch aus den Switchboard-Transkriptionen möglichst lange Eingabesequenzen zu generieren. Deshalb werden jeweils alle von einem Sprecher zu einem Zeitpunkt genanten Sätze bis es zu einem Sprecherwechsel kommt zu einer Eingabesequenz zusammengefasst. Anschließend werden gemäß [JC04] alle Interpunktionen und partiellen Wörter, das bedeutet alle abgebrochenen Wörter, entfernt. Dadurch werden die Eingabe realistischer, da ein Spracherkenner sowohl Interpunktion als auch partielle Wörter nicht oder nicht zuverlässig ausgeben kann. Dadurch halten sich die Eingaben auch an die in Abschnitt 5.1 definierten Einschränkungen des Projektes PARSE. Restannotationen von disfluenten Strukturen, die nun nicht mehr möglich oder disfluent sind, beispielsweise weil ein Reparandum nur aus partiellen Wörtern bestand und nun ein Reparans ohne zugehöriges Reparandum existiert, werden ebenfalls entfernt. Des Weiteren werden die Wörter der Eingabesequenzen klein gesetzt.
Das spätere Lernverfahren benötigt neben den Wörtern auch Wortarten und -gruppen,

weshalb die nun gefilterten Switchboard-Transkriptionen durch die seichte Sprachverarbeitung von PARSE gesendet werden. Dies soll den Trainingsprozess näher an die späteren Eingaben des PARSE-Projektes bringen, für die auch die selbe seichte Sprachverarbeitung die Wortart und -gruppe bestimmt. Die Wortgruppe wird dabei im IOB-Format (siehe Abschnitt 2.2.2) angegeben. In der seichten Sprachverarbeitung werden die selben Werkzeuge wie bereits in [Koc15] verwendet, mit der Ausnahme, dass für die Wortartenmarkierung das Werkzeug aus [MSB+14] eingesetzt wird.
Im Gegensatz zu dem Verfahren in [WCLL16] werden die gefüllten Pausen nicht aus der Eingabe entfernt und die häufigen Phrasen *„i mean“* und *„you know“* werden nicht jeweils zu einem Token vereint. Stattdessen werden die meisten gefüllten Pausen durch das Wort *„uh“* ersetzt, da es die häufigste gefüllte Pause im Datensatz ist und nahezu alle gefüllte Pausen die gleiche Wortart aufweisen und die gleiche Funktion besitzen. Diese Ersetzung geschieht jedoch nur direkt bevor die Wörter an den Klassifikator gesendet werden und da die Position der Eingabewörter erhalten bleiben, werden die in der Zwischenverarbeitung ersetzten gefüllten Pausen dennoch bei der Evaluation wieder mit ihrem korrekten Wert als gefüllte Pause klassifiziert. Schlussendlich werden die Sequenzen wortweise in einer gesäuberten Darstellung nach dem Schema *Wort, Wortart, Wortgruppe in IOB-Format* abgespeichert und für die spätere Weiterverarbeitung in dieser Form eingesetzt. Sowohl die ursprünglichen Switchboard-Transkriptionen als auch die gesäuberten, mit weiteren Informationen angereicherten Transkriptionen befinden sich im Ressourcen-Ordner des Projektverzeichnisses.

6.2.2. Eingabemerkmale

Das Neuronale Netzwerk erhält gemäß Abschnitt 6.1 pro Wort der Sequenz vier Merkmale als Eingabe: das Wort selbst, die Wortart, die Wortgruppe sowie einen diskreten Eingabevektor. Jedes Merkmal wird dabei vorverarbeitet und dem Netzwerk in Form eines Vektors mit reellen Zählen als Einträgen übergeben.

Das Wort selbst wird zur effizienten Weiterverarbeitung in der Vorverarbeitung in einen niedrig dimensionalen dichten Wortvektor umgewandelt (siehe Abschnitt 2.6). Für diese Umwandlung wird ein vortrainiertes Modell eingesetzt, welches globale Vektoren verwendet (engl. global vectors - GloVe) [PSM]. Es wird ein auf globale Vektoren basierendes Modell verwendet, da globale Vektoren gemäß Pennington et al. [PSM14] im Vergleich zu auf Wort-zu-Vektor (engl. word2vec) basierende Wortvektoren bei den von den Autoren ausgeführten Evaluationsaufgaben wie der Eigennamenerkennung zu bessere F1-Werten führen. Des Weiteren benötigen vortrainierte GloVe-Modelle wesentlich weniger Speicherplatz und es existieren auch vortrainierte Modelle für Vektoren mit einer geringeren Dimensionalität. Das eingesetzte GloVe-Modell wurde auf englischen Twitter-Nachrichten, die 27 Milliarden Token umfassen, trainiert und es wurde ein Vokabular gebildet, das Wortvektoren zu den häufigsten 1,2 Millionen Wörter enthält. Dieses GloVe-Modell dient als Umsetzungstabelle zur Abbildung von Wörtern auf feste Wortvektoren. Für die Dimension der Vektoren wurde der Wert 50 gewählt, da eine höhere Dimension zu keiner Verbesserung des F1-Wertes auf dem Entwicklungsdatensatz führte. Den Wörtern, die nicht im Vokabular des GloVe-Modells enthalten sind und damit kein Wortvektor aus dem Modell zugewiesen werden kann, wird ein fester aber zufälliger Vektor mit reellen Einträgen zwischen -1,0 und +1,0 zugewiesen. Dieser zufällig Vektor bleibt für das entsprechende Wort für den gesamten Trainings- und Testprozess gleich. Im Neuronalen Netzwerk des Klassifikators durchläuft der Wortvektor nochmals eine Eingabeschicht, die ihn auf einen Vektor der Dimension 100 abbildet.

Die Wortart und Wortgruppe werden in der Vorverarbeitung jeweils auf einen 1-aus-n-kodierten Vektor (siehe Abschnitt 2.5) abgebildet, da diese Art von Merkmalen sich

lediglich kategorisch unterscheidet, das bedeutet es lässt sich kein direkter funktionaler Zusammenhang zwischen den unterschiedlichen Werten des Merkmals bilden, und dies in der Vorverarbeitung sich durch einen 1-aus-n-Code am einfachsten realisieren lässt. Des Weiteren erfordert die zugrunde liegende verwendete Programmbibliothek diese Form für ein kategorisches Eingabemerkmal. Die Dimension der beiden Vektoren entspricht jeweils der Anzahl aller im Projekt PARSE (siehe Kapitel 3) definierten Wortarten und Wortgruppen. Sowohl Wortart als auch Wortgruppe durchlaufen jeweils eine Eingabeschicht im Neuronalen Netzwerk, die die 1-aus-n-kodierten Vektoren in dichte Vektoren mit verringerter Dimension umwandelt. Die Dimension der Wortart wird von 42 auf 25 reduziert und die Dimension der Wortgruppe wird von 24 auf 12 reduziert. Diese Reduktion der Dimension bewirkt neben einer effizienteren Weiterverarbeitung auch eine unterschiedliche Gewichtung der Eingabemerkmale, so dass das Wort selbst eine größeres Gewicht gegenüber der Wortart und die Wortart ein größeres Gewicht gegenüber der Wortgruppe besitzt.

Unter den bisherigen Merkmalen werden das Wort selbst und die Wortart auch in allen in Kapitel 4 vorgestellten Verfahren, die den Switchboard-Korpus zum Training und Testen verwenden, als Merkmale eingesetzt. Die Wortgruppe wird als Merkmal eingeführt, da es sich in Abschnitt 5.4 und Abschnitt 5.6 als möglicherweise hilfreiche Information bei der Untersuchung auf Disfluenzen erwiesen hat (eine Auswertung dazu siehe Abschnitt 7.3.1). Um die Umgebung eines Wortes zusätzlich zu fokussieren wird ein weiteres Merkmal eingeführt. Es handelt sich dabei grundsätzlich um den von Wang et al. in [WCLL16] eingeführten diskreten Merkmalsvektor mit den nahezu selben dort definierten Funktionen durch die der Vektor gebildet wird. Im Verfahren von Wang et al. für dieser Eingabevektor zu einer erheblich Steigerung des F1-Wertes auf dem Entwicklungsdatensatz um 6,7 %. In diesem Vektor werden Wiederholungen bezüglich dem Wort und der Wortart in der Umgebung durch handgefertigte Merkmale explizit markiert. So wird pro Wort innerhalb einer Umgebung von 15 Wörter auf beiden Seiten für alle Wiederholungen des Wortes und dessen Wortart jeweils eine Eins im Vektor gesetzt und andernfalls eine Null. Im Folgenden sind die Gleichungen für dieses zwei Funktionen dargestellt, wobei w_i das aktuelle Wort und p_i dessen zugehörige Wortart ist:

$$\text{-15} \leq k \leq +15 \text{ and } k \neq 0 :$$

$$duplicate(w_i, w_{i+k}) = \begin{cases} 1 & \text{für } w_i = w_{i+k} \\ 0 & \text{sonst} \end{cases} \tag{6.1}$$

$$duplicate(p_i, p_{i+k}) = \begin{cases} 1 & \text{für } p_i = p_{i+k} \\ 0 & \text{sonst} \end{cases} \tag{6.2}$$

Viele der Disfluenzen in den verwendeten Datensätzen sind relativ kurz, das heißt deren Reparanda bestehen nur aus einem oder zwei Wörtern. Oft handelt es sich bei diesen Disfluenzen in den zugehörigen Reparans um Wiederholungen. Solche Wiederholungen treten bei Disfluenzen, deren Reparanda und Reparans mehr als ein Wort enthalten, oft nur für zwei Wörter direkt hintereinander auf. Des Weiteren treten solche paarweisen Wiederholungen oft relativ kurz nach dem zugehörigen Reparandum auf und die Suche nach Wiederholungen dieser Art kann auf eine kleinere Wortumgebung beschränkt werden. Deshalb werden im Vektor pro Wort und dessen nächstfolgenden Wort in einer Umgebung von vier Wörtern auf beiden Seiten für jede paarweise Wiederholungen bezüglich dem Wort und der Wortart eine Eins gesetzt, ansonsten ein Null. Im Folgenden sind wieder beide Gleichungen der Funktionen dargestellt:

$$\text{-}4 \leq k \leq +4 \text{ and } k \neq 0:$$

$$duplicate(w_i w_{i+1}, w_{i+k} w_{i+k+1}) = \begin{cases} 1 & \text{für } w_i w_{i+1} = w_{i+k} w_{i+k+1} \\ 0 & \text{sonst} \end{cases} \tag{6.3}$$

$$duplicate(p_i p_{i+1}, p_{i+k} p_{i+k+1}) = \begin{cases} 1 & \text{für } p_i p_{i+1} = p_{i+k} p_{i+k+1} \\ 0 & \text{sonst} \end{cases} \tag{6.4}$$

Neben Wiederholungen ist es auch sinnvoll in der nahen Umgebung nach ähnlichen Wörtern zu suchen und für diese entsprechend eine Eins im Vektor zu setzen. Denn es kann vor allem bei längeren Wörtern zu Versprechern kommen, welche üblicherweise unmittelbar wiederholt und korrigiert werden. Hierbei folgt jedoch erstmalig eine Abweichung von den Funktionen aus [WCLL16], denn es wird nicht nur in einer Wortumgebung von einem Wort auf beiden Seiten, sondern in einer Umgebung von zwei Wörtern nach solchen ähnlichen Wörtern gesucht. Diese Erweiterung um ein Wort auf beiden Seiten ist sinnvoll, da es bei komplexeren Wörtern oft mehr als ein Versprecher hintereinander gibt. Des Weiteren wurden in den Datensätzen ein nicht die häufigsten gefüllten Pausen entfernt, wie es die Autoren von [WCLL16] durchgeführt haben. Deshalb kann es möglich sein, dass für die Untersuchung auf Ähnlichkeit zunächst ein gefüllte Pause übersprungen werden muss. Im Folgenden sind wieder die Gleichungen der Funktionen dargestellt, wobei hierbei die Funktion *num_same_letters* die Anzahl der gleichen Buchstaben beider zu vergleichenden Wörter ausgibt:

$$\text{-}2 \leq k \leq +2 \text{ and } k \neq 0:$$

$$similarity = \frac{2 * num\_same\_letters}{length(w_i) + length(w_{i+k})} \tag{6.5}$$

$$fuzzyMatch(w_i, w_{i+k}) = \begin{cases} 1 & \text{für } similarity > 0,8 \\ 0 & \text{sonst} \end{cases} \tag{6.6}$$

Die Dimension des gesamten resultierenden Vektors beträgt 80. Es wurde auch versucht den Vektor durch zusätzliche Informationen bezüglich den Positionen von Wörtern mit gleicher Wortgruppe in der Wortumgebung anzureichen, was jedoch zu einer Verschlechterung der Erkennungsrate der Disfluenzen auf dem Entwicklungsdatensatz führte. Auch die Erweiterung um die Positionen von gefüllten Pausen, Diskursmarkern und expliziten Bearbeitungsbegriffen in der Wortumgebung führte zu einer Verschlechterung der Erkennungsrate. Das gleiche passierte bei einer Vergrößerung der ursprünglichen Untersuchungsgrenzen bezüglich Wort- und Wortartenwiederholungen. Eine verbessernde Modifikation ist beispielsweise die Ersetzung des bisherigen Ähnlichkeitsmaßes durch die Jaro-Winkler-Distanz [CRF03], wodurch auch kürzere Wörter als ähnlich betrachtet werden können. Eine weitere positive Modifikation ist, dass benachbarte Wörter, bei denen das erste Wort ein Präfix des anderen ist, als gleich betrachtet. Die Ergebnisse dieser Modifikationsversuche des Vektors werden im Abschnitt 7.3.1 näher betrachtet. Nach den Er in Abschnitt 7.3.1. Der diskrete Vektor wird im Gegensatz zu den anderen Merkmalen direkt an die nächste Schicht im Netzwerk weitergereicht, ohne eine Eingabeschicht zu durchlaufen, damit die diskreten Umgebungsinformationen direkt Einfluss auf die Verarbeitung der verborgenen Schicht nehmen.

Tabelle 6.1.: Ausgabeklassen des Klassifizierers mit deren Markierungen

Ausgabeklasse	Markierung
Außerhalb einer Disfluenz	O-DF
Beginn eines Reparandums	B-RM
Innerhalb eines Reparandums	I-RM
Beginn eines Reparans	B-RS
Innerhalb eines Reparans	I-RS
Gefüllte Pause	FP
Expliziter Bearbeitungsbegriff	EE
Diskursmarker	DM

6.2.3. Ausgabeklassen

Der Klassifizierer schätzt für jedes Wort der Eingabesequenz dessen Zugehörigkeit zu einer Disfluenz. So wird beispielsweise geschätzt ob ein Wort zu einem Reparandum (RM) oder einem Reparans (RS) gehört. Innerhalb dieser beiden wesentlichen Bestandteile wird noch die Postion des Wortes geschätzt, also ob da Wort zu Beginn (engl. begin - B), innerhalb (engl. inside - I) oder außerhalb (engl. outside - O) liegt. Dabei werden gefüllte Pausen (engl. filled pauses - FP), Diskursmarker (engl. discourse marker - DM) und explizite Bearbeitungsbegriffe (engl. explicite editing terms - EE) entsprechend gesondert markiert. Falls das Wort außerhalb eines Reparandums oder eines Reparans liegt und nicht einer der gesonderten Gruppen angehört wird es als außerhalb einer Disfluenz markiert. Der Klassifizierer weist demnach jedes Wort einer der insgesamt acht Ausgabeklassen zu. Diese Klassen sind nun in Tabelle 6.1 mit ihren Markierungen (engl. labels oder tags) dargestellt.

Für den Trainingsprozess werden die Ausgabemarkierungen der Trainingsdaten jeweils pro Wort 1-aus-8-kodiert an den Neuronale Netzwerk übergeben. Die Eingabesequenz „*armar go [to the dishwasher + I mean to the fridge]*“ würde beispielsweise im Idealfall durch den Klassifizierer wie im folgenden markiert werden:

[O-DF armar] [O-DF go] [B-RM to] [I-RM the] [I-RM dishwasher] [EE I] [EE mean] [B-RS to] [I-RS the] [I-RS fridge]

Es wird darauf verzichtet, noch weitere Klassen für RM und RS zu definieren, wie beispielsweise jeweils eine Klasse für nur aus einem Wort bestehende Blöcke (S-RM oder S-RS) oder eine Klasse zur Markierung des Ende des Blockes (E-RM oder E-RS). Dadurch soll vermieden werden, dass durch zu viele Klassen sich die Erkennungsrate der anderen Klassen verschlechtert. Denn je mehr Klassen, desto mehr Schätzungen für falsche Klassen sind möglich. Aus ähnlichen Gründen sind zwei zusätzliche Klassen, die ursprünglich zur Modellierung von komplexen Disfluenzen (siehe Abschnitt 5.7 eingeführt wurden, wieder entfernt worden. Diese Klassen beschrieben den Fall, dass ein Wort, welches zu einem Reparans gehört und damit ein korrigierendes Wort ist, gleichzeitig auch wieder selbst korrigierendes wird und damit auch ein Teil eines Reparandums ist. Die zugehörige Markierung war „B-RS_RM“ beziehungsweise „I-RS_RM“.

So würde ein Verkettung von Disfluenzen wie in der beispielhaften Sequenz „*bring me [a red + a red + a green] cup*“ wie im folgenden markiert werden:

[O-DF bring] [O-DF me] [B-RM a] [I-RM red] [B-RS_RM a] [I-RS_RM red] [B-RS a] [I-RS green] [O-DF cup]

Eine Einbettung einer Disfluenz hingegen wie in der Beispielsequenz „*bring me [a red cup + uh a green [cup + uh bottle]]*“ würde wie im folgenden markiert werden:

[O-DF bring] [O-DF me] [B-RM a] [I-RM red] [I-RM cup] [FP uh] [B-RS a] [I-RS green] [B-RS_RM cup] [FP uh] [I-RS bottle]

Diese zwei Klassen wurden wieder entfernt, da sie nicht genügend Repräsentationen im Trainingsdatensatz besitzen und dadurch eine zu schlechte Erkennungsrate aufwiesen, insbesondere die Klasse I-RS_RM. Ihr teilweise sehr komplexer Aufbau verschlimmerte diese Situation noch weiter. Insgesamt verschlechterten diese Klassen die zusammengefügte Erkennungsrate aller RM-Klassen. Durch dass entfernen dieser Klassen verbesserte sich insbesondere die Erkennungsrate der Klasse I-RM. Näheres dazu wird in Abschnitt 7.3.1 behandelt.

6.2.4. Aufbau des Neuronalen Netzwerks

Wie bereits erwähnt durchlaufen die drei Eingabemerkmale Wörter beziehungsweise Wortvektoren, Wortarten sowie Wortgruppen jeweils eine Eingabeschicht (siehe Abschnitt 6.2.2). Während die 1-aus-n-kodierten Vektoren der Wortarten und Wortgruppen durch die Eingabeschicht zu dichten Vektoren (siehe Abschnitt 2.6) umgewandelt werden, werden Wortvektoren auf eine höher dimensionale Vektoren der Dimension 100 abgebildet. Dieses nun umgewandelten Eingabevektoren werden zusammen mit dem diskreten Eingabevektor in einem Verbindungsknoten (engl. merge vertex) zu einem zusammengefassten Eingabevektor vereinigt. Der Verbindungsknoten wird öfters auch als Verbindungsschicht bezeichnet. Die Vereinigung der einzelnen Eingaben geschieht durch Konkatenation der Vektoren.
Dies wird nun an die verborgene Schicht gesendet. Diese Schicht ist wie auch in den Verfahren in [ZOH16, WCLL16, LJ17] als ein bidirektionales LSTM realisiert. Das verwendete LSTM muss bidirektional arbeiten, da der Kontext eines Wortes in beide Richtungen betrachtet werden muss, um sowohl die Reparanda, zu denen zugehörige Abhängigkeiten nur bei umgekehrter Wortreihenfolge wirklich sichtbar werden, als auch die Reparans, zu denen zugehörige Abhängigkeiten nur bei vorwärts gerichteter Wortreihenfolge wirklich sichtbar werden, korrekt zu erfassen.
Die verborgene Schicht besteht aus einem herkömmlichen LSTM, dass die Eingabe in normaler Reihenfolge abarbeitet und einem rückwärts arbeitenden LSTM, welches unabhängig vom vorwärts arbeitendem LSTM die Eingabe rückwärts bearbeitet. Die Aktivierungsfunktion und damit die Ausgabe beider LSTMs wird durch einen internen Verbindungsknoten verknüpft. Die Art der Verknüpfung ist eine Addition, welche etwas bessere F1-Werte auf dem Entwicklungsdatensatz als die Konkatenation erzielt, die in [ZOH16, WCLL16, LJ17] als Verknüpfungsfunktion verwendet wird. Die Dimension des Ausgabevektors und damit die Größe, das bedeutet die Anzahl der Knoten der Schicht, der verborgenen Schicht beträgt 100. Dieser Wert ist basierend auf dem Verfahren aus [WCLL16] gewählt. Die Ausgabe der verborgenen Schicht wird abschließend an die Ausgabeschicht gesendet, die nun damit pro Wort die Wahrscheinlichkeit der Zugehörigkeit zu den Klassen ausgibt. Die Klasse mit der höchsten Wahrscheinlichkeit wird dann als geschätzte Klasse gewählt.

6.2.5. Weitere Einstellungen und Parameter für das Neuronale Netzwerk

In diesem Abschnitt werden noch einige weitere Einstellungen und Parameter für das Neuronale Netzwerk behandelt, die bisher nicht betrachte wurden. Die meisten dieser Parameter basieren entweder auf den Verfahren in [WCLL16, LJ17],da diese die besten F1-Werte auf den verwendeten Datensätzen erzielen und ebenfalls ein bidirektionales LSTM für ihr Lernmodell verwenden, oder es handelt um empfohlene Werte aus der Dokumentation der Programmbibliothek DL4J [dl4b] oder es sind empirisch ermittelte Werte.
Die bereits in Abschnitt 6.2.1 angesprochene Aufteilung des Datensatzes in einen Trainings-, Entwicklungs- und Testdatensatz hat den Zweck, den Trainingsvorgang eines maschinellen Lernmodells zu verkürzen und effizienter zu gestalten. So wird nach jedem Trainingsdurchgang durch den kompletten Datensatz (engl. epoch) das bis dahin trainierte Modell auf

dem Entwicklungsdatensatz evaluiert. Es wird dabei dann ein Leistungsmaß, wie beispielsweise das F1-Maß, betrachtet und dann mit den Evaluationsergebnissen, die sich nach den vorherigen kompletten Trainingsdurchgängen ergaben, verglichen. Falls sich in den Ergebnissen keine positive Entwicklung bezüglich des Leistungsmaßes ableiten lässt, so kann der Trainingsprozess vorzeitig abgebrochen werden. Dieser Trainingsvorgang hört frühzeitig auf, im Englischen wird dies als *early stopping* bezeichnet [Pre98]. Durch diesen frühzeitig Abbruch des Trainingsprozess kann Übersättigung vermieden werden. In dem in dieser Arbeit entwickelten Modell wird der Trainingsprozess abgebrochen, falls sich das arithmetische Mittel der F1-Werte der beiden Reparanda-Klassen nicht für zwei komplette, aufeinander folgende Trainingsdurchgänge verbessert. Der Testdatensatz dient dann dazu, nach diesem Trainingsvorgang noch einen möglichst unabhängigen Datensatz für die eigentliche Evaluation zu besitzen,

Die Parameteroptimierung findet mit dem stochastischen Gradientenabstieg (engl. stochastic gradient descent - SGD) statt. Die Lernrate $n_0 = 0,1$ wird nach jedem Trainingsdurchgang durch den kompletten Datensatz (engl. epoch) gemäß folgender Gleichung [WCLL16] aktualisiert, wobei t der Anzahl der Durchgänge über den kompletten Datensatz entspricht:

$$n_t = n_0/(1 + 0,05 * t) \tag{6.7}$$

Eine geringere initiale Lernrate wurde ausprobiert, jedoch führte sie zu einer Verschlechterung der Erkennungsrate. Die Aktivierungsfunktion ist bei allen Netzwerkschichten bis auf die Ausgabeschicht die Hyperbeltangens-Funktion (tanh). In der Ausgabeschicht ist die Aktivierungsfunktion wie in nahezu jeder Klassifizierung die Softmax-Funktion. Die Größe des Mini-Stapels (engl. mini batch) beträgt 20 Datensequenzen, was aus [LJ17] übernommen wurde. Während des Trainingsvorgang müssen die Eingabesequenzen auch auf eine maximale Länge beschränkt werden, unter anderem auch damit der Vorgang nicht zu rechenintensiv und ineffizient wird. In [LJ17] und [WCLL16] werden die Eingabesequenzen auf 50 Wörter beschränkt, aber da die späteren Eingaben des Projektes PARSE eine möglicherweise wesentlich größere Länge besitzen, wird in dem in dieser entwickelten Verfahren die Begrenzung auf 60 Wörter erhöht. Während der Evaluation erfolgt keine Begrenzung dieser Art für die Länge der Wortsequenzen.
Des Weiteren wurde die Methode des zufälligen Ausfallens der Knoten einer Netzwerkschicht (engl. dropout method) [SHK+14] als Methode zur Regularisierung zur Vermeidung von Übersättigung auf die verborgene Schicht des Netzwerks während des Trainingvorgangs angewendet. Dies führte jedoch zu keiner Verbesserung der Erkennungsrate auf dem Entwicklungsdatensatz und auch zu keiner sonstigen erheblichen Leistungsverbesserung wie sie in [WCLL16] beschrieben wurde.

6.3. Interne Repräsentation und grundlegender Programmablauf

Das für diese Bachelorarbeit erstellte Werkzeug ist ein in Java realisiertes Maven-Projekt mit dem Namen „disfluencyanalyzer“. Die Hauptklasse ist der Agent `DisfluencyAnalyzer`. Diese Klasse ist der Startpunkt des ganzen Projektes und ist für die Steuerung der Abläufe verantwortlich. Sie startet die Informationsextraktion über die Hilfsklasse `GraphUtils` und sendet die Eingabeinformation an die Klasse `LSTMClassifier`. Nachdem der geschätzten Ergebnisse feststehen, werden diese wieder an den `DisfluencyAnalyzer` gesendet, der diese Ergebnisse dann über`GraphUtils` auf den Graphen abbildet. Die Hilfsklasse `GraphUtils` wird für den Zugriff auf den Handlungsrepräsentationsgraphen verwendet.

Für die Informationsextraktion in der Eingabe müssen hierbei lediglich die Wörter der Eingabesequenz sowie deren Wortarten und -gruppen extrahiert werden. Dies lässt sich direkt aus den Knoten des Graphen ablesen, da sie pro Knoten direkt in den Attributen abgelesen werden können. Die Ausgabenabbildung und dafür erforderliche Transformation des Graphen wird in Abschnitt 6.4 behandelt. `GraphUtils` prüft auch, ob sich der Graph verändert hat und ein neuer Klassifizierungsvorgang erforderlich ist. Wie dies umgesetzt wird, wird in Abschnitt 6.4 beschrieben. Die in Abschnitt 6.2.3 definierten möglichen Klassen für ein Wort bei der Zuweisung durch den Klassifikator ist die durch Aufzählung (engl. enum) `DisfluencyTag` modelliert. Der eigentliche Klassifizierer ist in der Klasse `LSTMClassifier` modelliert. In dieser Klasse wird ein maschinelles Lernmodell, basierend auf mehreren Schichten an neuronalen Netzen (siehe Abschnitt 6.2) anhand eines vorher abgespeicherten Datensatzes trainiert und wahlweise auch evaluiert. Alternativ lässt sich auch ein vortrainiertes Modell aus dem Ressourcen-Ordner laden und dies anhand eines Datensatzes evaluieren. Außerdem ist es mit einem trainierten Modell auch möglich, die Ausgabemarkierungen für eine einzelne Eingabesequenz zu schätzen, was der normalen Programmausführung entspricht. `LSTMClassifier` lädt des Weiteren auch das vortrainierte Modell an globalen Vektoren (GloVe) (siehe Abschnitt 6.2.2), mit dem Wörter auf Wortvektoren abgebildet werden.
Für den Trainings- und Evaluationsvorgang verwendet `LSTMClassifier` pro Datensatz einen `DisfluencyDataSetIterator`. Die Klasse `DisfluencyDataSetIterator` lädt zunächst den vorher abgespeicherten Datensatz mit allen abgespeicherten Merkmalen wie Wortarten und -gruppen in normale Listen. Während dem Trainings- oder Evaluationsvorgang lädt die Klasse den Datensatz iterativ in das neuronale Netzwerk von `LSTMClassifier`. Davor wandelt `LSTMClassifier` die Eingabemerkmale in eine vektorielle Darstellung um. Des Weiteren wird pro Wort der diskrete Umgebungsvektor als viertes Eingabemerkmal berechnet. Einen Umwandlungsprozess für die Wörter einer einzelnen Eingabesequenz außerhalb eines Trainings-oder Evaluationsvorgangs ist auch möglich. `LSTMClassifier` benötigt jedoch für das Laden des zugehörigen Datensatzes diesen in einer „gesäuberten“ Darstellung (Abschnitt 6.2.1). Die Filterung von den ursprünglichen Transkriptionen des Switchboard-Datensatzes zu dieser neuen Darstellung geschieht durch die Hilfsklasse `InputProcessor`. Diese wird jedoch nur einmalig und vor dem normalen Programmablauf statt. Sowohl in der Klasse `LSTMClassifier` als auch in der Klasse `DisfluencyDataSetIterator` werden viele Konstrukte der Programmbibliothek „Deeplearning4j“ (DL4J) verwendet, um ein auf neuronalen Netzen basierendes Lernverfahren zu implementieren.

Nun wird noch der normale Programmablauf grundlegend beschrieben. Es wird davon ausgegangen, dass bereits ein Lernmodell in `LSTMClassifier` trainiert wurde und es sich im Ressourcen-Ordner des Projektes befindet. In diesem Ordner muss sich außerdem das vortrainierte GloVe-Modell befinden. Es wird auch davon ausgegangen, dass die Eingabesequenz auf den Handlungsrepräsentationsgraphen abgebildet ist und pro Token-Knoten neben dem Wert auch die Attribute der Wortart- und Wortgruppe (im IOB-Format) gesetzt ist.
Zum Programmstart erzeugt `DisfluencyAnalyzer` zunächst eine Instanz des `LSTMClassifier`, welche anschließend zunächst das Glove-Modell und danach ein vortrainiertes Modell des Klassifizierers in den Speicher lädt. Danach werden die Eingabeinformationen über `GraphUtils` aus dem Graphen extrahiert und über `DisfluencyAnalyzer` an den `LSTMClassifier weitergeleitet.` Dieser gibt die Eingabeinformationen mit einer Referenz auf das GloVe-Modell an eine Instanz von `DisfluencyDataSetIterator` weiter, die daraus die vektorielle Darstellung berechnet. Das vortrainierte Modell des Klassifizierers erhält dies nun als Eingabe und gibt dann seine Schätzungen bezüglich der Zugehörigkeit zu den Disfluenz-Ausgabeklassen aus. Diese Ergebnisse werden wieder an den Agenten `DisfluencyAnalyzer` gesendet, der diese Information über `GraphUtils` im Graphen dar-

stellt.

6.4. Ergebnisspeicherung

Da es sich beim Klassifizierer um eine Sequenzmarkierung handelt, in der jedem Wort der Eingabe eine Schätzung bezüglich seiner Zugehörigkeit zu den Ausgabeklassen zugewiesen wird, bietet es sich an, die Token-Knoten des Handlungsrepräsentationsgraphen um das neue Attribut „disfluencyTag“ zu erweitern. Hierbei gilt eine Zuordnung von einem Wort zu der Klasse, für die vom Klassifikator die höchste Zuordnungswahrscheinlichkeit geschätzt wurde. Als Werte des Attributs „disfluencyTag“ werden die in Abschnitt 6.2.3 definierten Markierungen der Ausgabeklassen verwendet.
Vor Durchführung der Schätzung kann nun dieser Attribut-Wert darauf überprüft werden, ob er gesetzt ist, um unnötige Programmausführungen bei nicht veränderten Graphen zu vermeiden. Zusätzlich muss jeder Beginn eines Reparans-Blockes auf den Beginn des zugehörigen Reparandum-Blockes verweisen. Deshalb wird dem Graphen der neue Kantentyp „repair“ hinzugefügt und es wird für jedes Token, das beim „disfluencyTag“-Attribut den Wert B-RS aufweist eine Kante zum nächsten linken Token gesetzt, dass einen „disfluencyTag“-Wert von B-RM aufweist. Falls bei der Suche nach einem passenden Beginn eines Reparandum-Blocks bereits der nächste Reparans-Block durchlaufen wird, so verweist die „repair“-Kante auf das Token mit dem zuletzt angetroffenen I-RM-Token. Falls auch solch ein Token nicht gefunden wurde, so verbleibt der entsprechende Reparans-Block ohne ein zugehöriges Reparandum. Eine Ausnahme muss hierbei für in die Reparans eingebetteten komplexen Disfluenzen berücksichtigt werden: in diesem Fall kommt es auch bei korrekter Schätzung des Klassifikators vor, dass direkt vor einem Token mit „disfluencyTag“-Wert I-RS ein Token mit dem Wert I-RM liegt, da beim beseitigen der vorherige Klassen der komplexen Disfluenzen diese durch die Markierung I-RM ersetzt wurden. In diesem Fall muss die „repair“-Kante von dem ersten I-RS-Token nach solch einer komplexen Disfluenz auf den ersten I-RS-Token der komplexen Disfluenz verweisen, um eine korrekte Korrektur darstellen.

7. Evaluation

In diesem Kapitel wird die Funktionsweise des entwickelten Werkzeugs hinsichtlich der Erkennung von disfluenten Strukturen analysiert und bewertet. Die für diese Bewertung erforderlichen Datensätze werden in Abschnitt 7.1 vorgestellt. Anschließend wird die Vorgehensweise in Abschnitt 7.2 beschrieben. In Abschnitt 7.3 werden die Ergebnisse der Evaluation für alle verwendeten Datensätzen vorgestellt und diskutiert. Abschließend erfolgt in Abschnitt 7.4 eine Zusammenfassung und Bewertung der wesentlichen Ergebnisse. Im Gegensatz zum Kapitel 4 werden die Werte für die in diesem Kapitel verwendeten Metriken nicht prozentual dargestellt.

7.1. Verwendete Datensätze

Um das in dieser Bachelorarbeit entwickelte Werkzeug zu evaluieren wird ein Testkorpus benötigt. Dafür bietet sich der bereits in Abschnitt 6.2.1 vorgestellte Switchboard-Korpus an. Dieser Korpus wurde in der Entwicklungsphase gemäß [CJ01] jeweils in einen Trainings-, Entwicklungs- und Testdatensatz aufgeteilt. Da der Trainingsdatensatz bereits für den Trainingsprozess des maschinellen Lernmodells des Werkzeuges verwendet wurde, lassen sich nur die beiden übrigen Datensätze für die Evaluation verwenden. Dabei sind die Ergebnisse des Testdatensatzes gegenüber denen des Entwicklungsdatensatzes zu bevorzugen, da der Entwicklungsdatensatz bereits im Trainingsprozess für die Bewertung der Erkennungsrate des Lernmodells eingesetzt wird (siehe Abschnitt 6.2.5). Hierbei trainiert das Lernmodell solange, bis auf dem Entwicklungsdatensatz keine besseren Ergebnisse mehr hinsichtlich der Erkennungsrate von Disfluenzen erreicht werden. Der Testdatensatz hingegen wird während des Trainingsprozesses nicht betrachtet und stellt damit einen unabhängigeren Datensatz für die Evaluation dar. Die Verwendung des Switchboard-Korpus ermöglicht ebenfalls die Nutzung der in Kapitel 4 vorgestellten Verfahren für Vergleichszwecke, da viele dieser Verfahren ebenfalls den Switchboard-Korpus mit der gleichen Aufteilung in die drei Datensätze für ihr maschinelles Lernmodell verwenden.

Die eigentliche Aufgabe des entwickelten Werkzeuges ist jedoch die Erkennung und Korrektur von disfluenten Strukturen in Spracheingaben, die innerhalb des Projektes PARSE (siehe Kapitel 3) vorkommen. Beispielhaft dafür sind die Befehlseingaben für den Küchenroboter ARMAR-III [ARA+06]. Diese Eingaben sind als Sequenz von Anweisungen formuliert und dadurch prinzipiell anders aufgebaut, als die Telefongespräche des Switchboard-Korpus. Damit eignet sich der Switchboard-Korpus nur bedingt für die Evaluation des

Werkzeugs und es ist ein weiterer Korpus erforderlich, der für das Projekt PARSE geeignet ist.
In der Arbeit von Günes [Gün15] wurde bereits ein solcher Korpus erstellt. Dazu wurden drei Szenarien konzipiert, in denen die Probanden dem Küchenroboter ARMAR-III selbst formulierte Anweisungen erteilen mussten. Pro Szenario wurden dabei 22 Aufnahmen mit händischen Transkriptionen des Gesagten erstellt. Aufbauend auf diesen Szenarien wurde der Korpus danach um jeweils 14 weitere Aufnahmen und Transkriptionen pro Szenario erweitert. Da diese Szenarien jedoch keine Bedingungsstrukturen in den Anweisungen förderten, wurden in der Arbeit von Steurer [Ste16] zwei weitere Szenarien konzipiert, zu denen jeweils 19 Sprachaufnahmen erstellt wurden. Der daraus resultierende Korpus umfasste nun 146 Sprachaufnahmen und Transkriptionen. Da die bisherigen Szenarien jedoch nur simple Korreferenzen aufwiesen, wurde in der Arbeit von [Hey16] der Korpus um zwei weitere Szenarien mit jeweils zehn Sprachaufnahmen sowie einer wiederholten Aufnahme erweitert. Dieser Korpus besteht nun aus insgesamt 167 Sprachaufnahmen sowie zugehörige Transkriptionen und ist bezüglich dem sprachlichen Aufbau der Eingaben für die Evaluation geeignet.
Der Korpus enthält 60 gefüllte Pausen, jedoch keine expliziten Bearbeitungsbegriffe und Diskursmarker (siehe Abschnitt 2.3.2). Des Weiteren weist dieser Korpus vier Reparaturen (siehe Abschnitt 2.3) auf. Bei der Erkennung von Disfluenzen sind jedoch besonders die Reparaturen von Bedeutung, da gefüllte Pausen, Diskursmarker sowie explizite Bearbeitungsbegriffe jeweils nur eine endliche Anzahl von zugehörigen Wörtern aufweisen beziehungsweise aus festen Phrasen bestehen und damit üblicherweise leicht zu erkennen sind. Für eine sinnvolle Evaluation ist demnach ein Datensatz mit mehr Reparaturen erforderlich. Deshalb wurde der bestehende Korpus um zwei weitere Szenarien erweitert. Die Szenarien wurden gemäß der Voraussetzungen, die in Günes [Gün15] Arbeit definiert wurden, entworfen, jedoch wurde versucht die Szenarien derart zu gestalten, dass in den Spracheingaben mehr Fehler und anschließende Reparaturen durch die Sprecher erfolgen. Die daraus entstanden Szenarien sind in Abschnitt A des Anhangs dargestellt. Im ersten Szenario, *Red cup in the cupboard and orange juice on the table*, muss der Roboter zunächst dazu angewiesen werden, der Spülmaschine einen roten Becher zu entnehmen und diesen im Schrank zu platzieren. Anschließend muss der Roboter den Orangensaft aus dem Kühlschrank holen und diesen auf den Tisch neben den grünen Becher platzieren. Abschließend muss das Fenster geschlossen werden. Im zweiten Szenario, *Filling a red cup and the green cup* muss zunächst das Fenster geöffnet werden und dann der grüne und rote Becher mit Orangensaft gefüllt werden. Abschließend wird das Fenster wieder geschlossen. Bei beiden Szenarien wurde angegeben, dass Fehler akzeptabel seien, jedoch diese zu korrigieren sind, falls sie erkannt werden. Die Szenarien sollten dabei Situationen erzeugen, in denen die Probanden mehr sprachliche Fehler begehen und diese daraufhin korrigieren. Dies sollte unter anderem durch eine in den Szenarien geforderte Feingranularität der Anweisungen erreicht werden, die besagt, dass beispielsweise für die Entnahme eines Gegenstands aus dem Kühlschrank, der Roboter sich erst zum Kühlschrank begeben und diesen öffnen muss. Des Weiteren müssen auch die Postionen bei der Entnahme und Platzierung der Gegenstände spezifiziert werden. Dies und weitere Bedingungen, wie beispielsweise ein Verbot für zuvor erstellte Notizen, soll Wiederholungen und Fehler fördern. Die Anweisungen zu beiden Szenarien wurden für zehn Probanden aufgenommen und gemäß den in [Gün15] definierten Vorschriften händisch transkribiert. Die Probanden konnten ihre Aufnahmen beliebig wiederholen, jedoch wurden die vorherigen Aufnahmen ebenfalls aufbewahrt und transkribiert, da diese meist sprachliche Fehler aufwiesen. Insgesamt wurden 29 Sprachaufnahmen erzeugt.

In Tabelle 7.1 stellt eine kurze Übersicht dieser Sprachaufnahmen und deren Transkriptionen dar. Die Transkriptionen der Aufnahmen enthalten neben 71 gefüllten Pausen ins-

Tabelle 7.1.: Übersicht über die neuen Transkriptionen

	Sz. 1	Sz. 2	Gesamt
Anzahl Aufnahmen	15	14	29
Anzahl wiederholte Aufnahmen	5	4	9
Anzahl Wörter Transkriptionen	1269	1392	2661
Anzahl gefüllte Pausen	54	17	71
Anzahl Reparaturen	12	15	27
Anzahl Eingaben ohne Reparaturen	8	9	17

gesamt 27 Reparaturen, unter denen eine Reparatur eine komplexe Disfluenz (siehe Abschnitt 5.7.2) in Form einer eingebetteten Reparatur darstellt. Es sind jedoch wieder keine expliziten Bearbeitungsbegriffe und Diskursmarker enthalten. Mit 17 der Transkription weisen über die Hälfte der Eingaben keine Reparaturen auf. Jedoch enthalten die übrigen 12 Transkriptionen 27 Reparaturen und weisen damit ein Vielfaches der Anzahl an Reparaturen in Transkriptionen der vorherigen Szenarien auf, wodurch sich deuten lässt, dass die Szenarien erfolgreich bei der Förderung einer größeren Anzahl an Reparaturen waren.

7.2. Vorgehensweise

In diesem Abschnitt wird erläutert, wie die Evaluation durchgeführt wird. Da der Klassifikator des entwickelten Werkzeuges die Eingabesequenzen wortweise klassifiziert, bietet es sich an, die Evaluation ebenfalls wortweise durchzuführen. Dies bedeutet, dass jedem Wort einer Eingabesequenz eine Klasse und damit eine Markierung zugeordnet wird. Als Klassen werden hierbei die in Abschnitt 6.2.3 definierten Ausgabeklassen und ihre entsprechenden Markierungen verwendet. Diese Klassen beschreiben die Zugehörigkeit eines Wortes zu einer bestimmten Art von Disfluenz beziehungsweise bei einer Reparatur die Zugehörigkeit zu einem der Bestandteile Reparandum (RM) oder Reparans (RS) (siehe Abschnitt 2.3.1). Für ein Wort aus einem dieser beiden Bestandteile beschreibt die Klasse auch die Position innerhalb des Bestandteils nach IOB-Format.

Zur Beurteilung der Eignung dieser Zuweisung werden die Metriken Präzision, Ausbeute und das F1-Maß verwendet [JM09]. Dafür werden zunächst die folgenden vier Bezeichnungen für das Ergebnis einer wortweisen Klassifizierung beispielhaft anhand der Klasse Beginn eines Reparandums (B-RM) eingeführt:

- **Richtig positives Ergebnis** (engl. *true positive* kurz *tp*): Es wurde die Klasse B-RM erwartet und das Wort wurde auch als solche markiert.
- **Richtig negatives Ergebnis** (engl. *true negative* kurz *tn*): Es wurde eine Klasse erwartet die nicht B-RM entspricht und das Wort wurde auch nicht als B-RM markiert.
- **Falsch positives Ergebnis** (engl. *false positive* kurz *fp*): Das Wort wurde als B-RM markiert, obwohl es nicht zu dieser Klasse gehört.
- **Falsch negatives Ergebnis** (engl. *false negative* kurz *fn*): Es wurde die Klasse B-RM erwartet, jedoch wurde das Wort nicht als solche markiert.

Bezogen auf alle Wörter einer Klasse ist *tp* die Anzahl der richtig zugeordneten Wörter, *fp* die Anzahl der fälschlicherweise zugeordneten Wörter, *fn* die Anzahl der nicht erkannten Wörter und *tn* die Anzahl der nicht richtig zugeordneten Wörter.
Basierend darauf lassen sich die bereits erwähnten Metriken berechnen. Die Präzision (engl.

precision) gibt hierbei an wie korrekt die Zuweisungen der Wörter zu einer Klasse sind. Sie wird mit folgender Formel berechnet:

$$Przision = \frac{Anzahl\ der\ korrekt\ erkannten\ Wörter\ pro\ Klasse}{Gesamtzahl\ der\ erkannten\ Wörter\ pro\ Klasse} = \frac{tp}{tp + fp} \tag{7.1}$$

Die Ausbeute (engl. *recall*) hingegen gibt den relativen Anteil der korrekt erkannten Wörter hinsichtlich der Gesamtzahl der tatsächlichen Wörter pro Klasse an. Sie wird mit folgender Formel berechnet:

$$Ausbeute = \frac{Anzahl\ der\ korrekt\ erkannten\ Wörter\ pro\ Klasse}{Gesamtzahl\ der\ zu\ findenden\ Wörter\ pro\ Klasse} = \frac{tp}{tp + fn} \tag{7.2}$$

Des weiteren lässt sich mit dem F1-Maß (engl. *F1-Score*) ein Zusammenhang zwischen Präzision und Ausbeute bilden. Das F1-Maß ist das harmonische Mittel aus Präzision und Ausbeute:

$$F1 = 2 * \frac{Präzision * Ausbeute}{Präzision + Ausbeute} \tag{7.3}$$

Des Weiteren werden diese Metriken in den folgenden Abschnitten auch für mehr als jeweils eine Klasse verwendet. Dies erfolgt, falls die Metriken über alle Reparandum- oder über alle Reparans-Klassen gebildet werden. Hierbei wird eine normale Zusammensetzung der Ergebnisse der Klassen von einer binären Zusammensetzung unterschieden. In einer normalen Zusammensetzung werden jeweils die tp, fp, tn und fn der betroffenen Klassen addiert und die Metriken danach gemäß ihren Formeln berechnet. Im binären Fall hingegen werden die Reparandum- und Reparans-Klassen jeweils zu einer Klasse zusammengefasst, es wird also die Anzahl der Klassen reduziert. Diese Betrachtungsweise ist binär, da beispielsweise bei den Reparandum Klassen jedes korrekt geschätzte Wort mit einer RM-Markierung, sei es eine B-RM oder I-RM Markierung, als richtig positives Ergebnis betrachtet. Somit werden die vorherigen falsch positiven Ergebnisse bezüglich der jeweils anderen RM-Klasse nun als zusätzliche richtig positive Ergebnisse betrachtet. Vom Betrachtungspunkt der RM-Klassen existieren in diesem Fall nur zwei Klassen: die RM-Klasse und die Klasse aller anderen Wörter.

Im Evaluationsprozess wird nun das Werkzeug hinsichtlich der in diesem Abschnitt vorgestellten Metriken bewertet, wobei das Werkzeug auf den in Abschnitt 7.1 beschriebenen Datensätzen ausgeführt wird. Die dafür erforderlichen Musterlösungen wurden durch ein händische Annotation der disfluenten Strukturen erzeugt. Die Annotationen erfolgen dabei gemäß dem bereits in Abschnitt 2.3.1 definierten Schema von Shriberg [Shr94]. Dieses Schema lässt sich dann mit der Hilfsklasse `InputProcessor` (siehe Abschnitt 6.3) in eine geeignete Datenstruktur umwandeln.

7.3. Ergebnisse der Evaluation

In diesem Abschnitt werden die Ergebnisse der Evaluation dargestellt und bewertet. Die Evaluation wird dabei sowohl auf dem Switchboard-Korpus als auch auf dem durch neue Szenarien erweiterten PARSE-Korpus ausgeführt. Der Evaluationsprozess auf dem Switchboard-Korpus dient dabei zunächst nur zur grundlegenden Bewertung des im entwickelten Werkzeugs eingesetzten maschinellen Lernverfahrens. Des Weiteren werden verschiedene Kombinationen von Eingabemerkmalen auf dem Testdatensatz des Switchboard-Korpus bewertet. Abschließend wird noch ein Vergleich mit einem in Kapitel 4 vorgestellten Verfahren hinsichtlich der Erkennung von disfluenten Strukturen auf dem Switchboard-Korpus

Tabelle 7.2.: Ergebnisse der Basisversion auf dem Entwicklungsdatensatz

Klasse	Präzision	Ausbeute	F1
O-DF	0,955	0,990	0,972
B-RM	0,895	0,738	0,809
I-RM	0,853	0,586	0,695
B-RS	0,898	0,778	0,834
I-RS	0,830	0,414	0,552
FP	0,986	0,995	0,990
EE	0,972	0,958	0,965
DM	0,963	0,939	0,951

Tabelle 7.3.: Ergebnisse der Basisversion auf dem Testdatensatz

Klasse	Präzision	Ausbeute	F1
O-DF	0,951	0,990	0,970
B-RM	0,895	0,750	0,816
I-RM	0,848	0,591	0,697
B-RS	0,896	0,754	0,819
I-RS	0,824	0,434	0,559
FP	0,989	0,993	0,991
EE	0,970	0,938	0,953
DM	0,960	0,903	0,931

durchgeführt. Anhand dieser Betrachtungen werden die besten Versionen des Lernverfahrens ausgewählt, um diese anschließend auf dem PARSE-Korpus zu evaluieren. Dabei werden die ausgewählten Lernverfahren noch weiter optimiert, um mit den zu den Switchboard-Korpus unterschiedlichen Formulierungen der Eingaben des PARSE-Korpus besser umzugehen.

7.3.1. Ergebnisse auf dem Switchboard-Korpus

In diesem Abschnitt werden die Evaluationsergebnisse, die auf dem Switchboard-Korpus erzielt wurden, präsentiert und bewertet. Als Vergleichsmetriken dienen dabei stets die Präzision, Ausbeute und das F1-Maß (siehe Abschnitt 7.2). Dabei werden zunächst die Ergebnisse einer Basisversion des Werkzeuges betrachtet. Diese Basisversion wurde anhand der in Abschnitt 6.2.2 definierten Eingabemerkmale trainiert, wobei für den diskreten Umgebungsvektor die erste Version verwendet wurde, die auf der Arbeit von Wang et al. [WCLL16] basiert. Anschließend wird diese Basisversion mit anderen Versionen, die Modifikationen bezüglich der Eingangsmerkmale aufweisen, verglichen. Abschließend findet mit der daraus ermittelten besten Version des Werkzeuges ein Vergleich mit dem in Kapitel 4 vorgestellten Verfahren von Wang et al. [WCLL16] statt, dass bezüglich dem F1-Maß auf dem Switchboard-Korpus dem aktuellen Stand der Technik entspricht.

In den Tabellen Tabelle 7.2 und Tabelle 7.3 sind nun die Ergebnisse der Basisversion entsprechend für den Entwicklungs- und Testdatensatz des Switchboard-Korpus dargestellt (siehe Abschnitt 7.1).

Anhand von Tabelle 7.2 und Tabelle 7.3 lässt sich leicht erkennen dass die Klassifizierung von gefüllten Pausen, expliziten Bearbeitungsbegriffen und Diskursmarkern besonders präzise mit F1-Werten ist. So werden in allen Metriken Werte von mehr als 0,9 erreicht. Dies

hat leicht verständliche Gründe, da wie bereits in Abschnitt 5.4 erwähnt diese drei Klassen aus einer endlichen Anzahl an Wörtern oder festen Phrasen bestehen. Innerhalb des Trainingsprozess werden diese nahezu auswendig gelernt und da der Entwicklungs- und Testdatensatz nahezu keine neue Wörter für diese Klassen aufweist, lassen sich diese leicht und zuverlässig während der Evaluation erkennen.
Auch die Wörter die außerhalb jeglicher Disfluenz liegen, werden präzise erkannt und mit sehr hoher Wahrscheinlich korrekt klassifiziert auf beiden Datensätzen. Dies liegt unter anderem auch an der wesentlich höheren Menge an Wörter die zu dieser Klasse gehören. So beträgt die Anzahl der zu dieser Klasse zugehörigen Wörter beispielsweise ca. 25 mal der Anzahl an Wörtern, die zur Klasse B-RM gehören. Da diese vier bisher genannten Klassen stets besonders hohe Werte auf dem Switchboard-Korpus bezüglich den Metriken erzielen und ihre Werte nahezu nicht merkbar variieren, werden diese in den nachfolgenden Betrachtungen nicht weiter berücksichtigt.
Von besonderer Bedeutung sind die Werte der Reparatur-Klassen B-RM, I-RM, B-RS und I-RS, da deren Erkennung der grundlegende Kern dieser Arbeit sowie vieler anderer verwandten Arbeiten wie die aus Kapitel 4 ist. Es wurde auf beiden Datensätzen für die Klassen B-RM und B-RS mit F1-Werten von über 0,8 gute Ergebnisse erzielt. Insbesondere die Präzision erreicht auf beiden Klassen und Datensätzen mit Werten von über 0,89 präzise Ergebnisse. Dies lässt auf eine niedrige Rate an falsch positiven Ergebnisse für beide Klassen schließen. Dies könnte daran liegen, dass ein Wort diesen Klassen nur zugewiesen wird, falls dem Klassifikator sprachliche Strukturen vorliegen, die stets sicher auf eine Disfluenz verweisen. Die Werte für die Ausbeute hingegen verweisen auf eine größere Anzahl an falsch negativen Ergebnissen und damit auf viele nicht erkannte zugehörige Wörter. Dies kann beispielsweise an sehr selten Fällen von Reparaturen liegen, die nur eine geringfügige Repräsentation im Trainingsdatensatz aufweisen. Des Weiteren sind Neustarts (siehe Abschnitt 5.7.1) sowie Disfluenzen, deren Abhängigkeiten eine hohe Menge an Wörtern überspannen (siehe Abschnitt 6.1), teilweise schwer erkennbar. Der Switchboard-Korpus enthält des Weiteren auch Reparaturen, die kein Reparans aufweisen und damit keine Korrektur enthalten,jedoch werden die fehlerhaften Wörter dennoch als Reparanda markiert, was ebenso zu der geringeren Ausbeute der Klasse B-RM beiträgt. Die Werte von den Klassen B-RM und B-RS sind dabei abhängig voneinander, da zwischen Wörtern aus diesen beiden Klassen in den meisten Fällen eine Abhängigkeit und Korrespondenz bestehen muss (siehe Abschnitt 5.6), damit diese erst als zugehörig zu ihren Klassen erkannt werden können. Lediglich bei Einfügungen neuer Wörter oder Entfernungen von Wörtern aus dem Reparandum muss diese Abhängigkeit nicht bestehen.
Die I-RM-Klasse erreicht für die Präzision ebenfalls für beide Datensätze einen hohen Wert von über 0,84. Jedoch sind die Werte der Ausbeute im Vergleich zu denen der Klasse B-RM wesentlich geringer. Die Werte der Ausbeute liegen unter anderem daran, dass nicht immer im Reparans eine passendes korrespondierendes Wort vorliegt. Des Weiteren kann durch komplexe Disfluenzen (siehe Abschnitt 5.7.2) der Klassifikator verwirrt werden, da für diese Disfluenzen keine eigenen Klassen in der Basisversion definiert sind und die komplexen Disfluenzen auf die Klasse I-RM abgebildet werden (siehe Abschnitt 6.2.3). Dadurch ist es möglich, dass der Klassifikator falsche Zusammenhänge im Trainingsprozess lernt. Ansonsten gelten für die geringer Ausbeute auch ähnliche Gründe wie bei Wörtern der Klasse B-RM.
Die Klasse I-RS weist noch geringe Werte für die Ausbeute auf. Dies liegt neben ähnlichen Gründen wie bei der Klasse I-RM auch daran, dass die rechte Grenze einer Reparatur oft nicht leicht bestimmbar ist, da beispielsweise im Reparandum passende Wörter, zu denen eine Abhängigkeit besteht, fehlen können. Dennoch sind die Werte der Ausbeute und damit auch die F1 Werte nicht gut. Es ist jedoch auch hier für alle Reparatur-Klassen anzumerken, dass eine nachträgliche Korrektur in dieser Version fehlt, die unvollständige Reparatur-Blöcke, wie beispielsweise ein fehlendes B-RM, ergänzt. Dies ist in machen

Situationen notwendig, da das neuronale Netzwerk des Klassifikators die Ausgabeklassen pro Wort festlegt, ohne Abhängigkeiten zwischen den Ausgabeklassen zu berücksichtigen. Solch eine nachträgliche Korrektur kann eventuell die bestehen Werte der Metriken, insbesondere die Werte der Ausbeute, für die Reparatur-Klassen verbessern.
Die Tabelle 7.2 und Tabelle 7.3 verdeutlichen ebenfalls, dass die Werte für den Entwicklungs- und Testdatensatz ähnlich sind und nah beieinander liegen. So unterscheiden sie sich in allen Reparatur-Klassen außer B-RS um weniger als 0,01. Deshalb werden im folgenden Evaluationsprozess nur die Werte des Testdatensatzes betrachtet, da dieser nicht im Entwicklungsprozess bereits verwendet wurde und damit eine unabhängigere Wertung aufweist als der Entwicklungsdatensatz.

Im folgenden werden nun basierend auf der Basisversion weitere Versionen des Lernmodells des Klassifikators erzeugt und mit der Basisversion hinsichtlich der bisherigen Metriken verglichen. Diese Versionen unterscheiden sich hinsichtlich ihrer Eingabemerkmale Abschnitt 6.2.2 oder der bezüglich der Anzahl der Ausgabeklassen Abschnitt 6.2.3. So werden zur Basisversion beispielsweise zusätzliche Merkmale hinzugefügt beziehungsweise bisherige erweitert, dies ist in der Modellbezeichnung durch ein „+“ kenntlich gemacht. Oder es werden Merkmale aus der Basisversion entnommen, wobei dies in der Modellbezeichnung durch ein „-“ kenntlich gemacht ist. In Tabelle 7.4 und Tabelle 7.5 sind nun die Ergebnisse auf dem Switchboard-Testdatensatz für die RM- und RS-Klassen hinsichtlich der bisherigen Metriken dargestellt. Diese Metriken werden nun jedoch jeweils für die gesamten RM- oder RS-Klassen berechnet. Dadurch sollen die durch die unterschiedlichen Versionen erzeugten Auswirkungen schneller ersichtlich werden. Dabei gibt es zwei Varianten, die in Abschnitt 7.2 vorgestellt wurden: Die normale Kombination der entsprechenden Klassen sowie die binäre Kombination. Die binäre Kombination führt dabei in der Regel zu höheren Werten, da in dieser Variante auch beispielsweise die falsch positiven Ergebnisse der B-RM-Klasse hinsichtlich der Klasse I-RM als richtig positives Ergebnis gezählt wird. Die in Kapitel 4 vorgestellten verwandten Arbeiten benutzen ebenfalls die Kombination der RM-Klassen als Bewertungsmetrik.
Die Tabelle 7.5 verdeutlicht, dass die kombinierte RS-Klasse wesentlich geringere Werte als die kombinierte RM-Klasse aufweist. So weisen die binären F1-Werte durchschnittlich einen um 0,07 bis 0,08 geringeren Wert auf. Dies liegt an den deutlich schlechten Werten der I-RS-Klasse für die Ausbeute und damit auch für das F1-Maß, was im vorherigen Abschnitt bereits erläutert wurde. Die Basisversion weist bei den RM-Klassen mit einem normalen kombinierten F1-Wert von 0,765 und einen binären F1-Wert von 0,792 einen akzeptablen bis guten Wert auf. Das Entfernen der Wortgruppe als Eingabemerkmal bewirkt nur eine sehr geringfügige Verschlechterung aller RM-Metriken. So fallen die Werte durchschnittlich nur um 0,001 , während sie sich für die Ausbeute- und F1-Werte sogar verbessern. Dies zeigt dass die reine Wortgruppe nur einen geringen Einfluss auf das Erlernen von Zusammenhängen für disfluente Strukturen besitzt. Dabei ist zu beachten, dass die Wortgruppen- beziehungsweise Phrasenerkennung sehr fehleranfällig ist, da bestimmte Wörter wie gefüllte Pausen oft den falschen Wortgruppen zugeordnet werden. Des Weiteren treten gleiche Wortgruppen auch in einer nicht disfluenten Struktur wiederholt auf, weshalb sich das Merkmal trotz der in Abschnitt 5.4 diskutierten zusätzlichen Anhaltspunkte, die die Wortgruppe bei der Untersuchung auf Disfluenzen bietet, für das den in dieser Arbeit entwickelten Klassifikator als nicht besonders hilfreich erwiesen. Ebenso hat sich die Erweiterung des diskreten Vektors um die Positionen der gefüllten Pausen, expliziten Bearbeitungsbegriffen sowie der Diskursmarker in der Umgebung des Wortes als nicht hilfreich erwiesen. Dadurch wurde die Eingabe zusätzlich parametrisiert und aus wurden keine oder geringfügige zusätzliche Informationen aus der Wortumgebung extrahiert. Lediglich in den RS-Klassen kam es zu einer geringfügigen Erhöhung der F1-Werte. Durch diese, sowie die nächste Version, in der die gefüllten Pausen aus der Eingabe entfernt wur-

Tabelle 7.4.: Verschiedene Versionen im Vergleich für die RM-Metriken

Model	Präz.	Ausb.	F1	Präz.-B	Ausb.-B	F1-B
Basisversion	0,876	0,679	0,765	0,906	0,702	0,792
-Wortgruppe	0,877	0,677	0,764	0,908	0,701	0,791
+FP,EE,DM in Vektor	0,863	0,679	0,760	0,900	0,704	0,788
+Ohne FP	0,865	0,678	0,760	0,896	0,702	0,787
+Komplex	0,848	0,661	0,743	0,906	0,706	0,794

Tabelle 7.5.: Verschiedene Versionen im Vergleich für die RS-Metriken

Modell	Präz.	Ausb.	F1	Präz.-B	Ausb.-B	F1-B
Basisversion	0,869	0,600	0,701	0,881	0,607	0,719
-Wortgruppe	0,860	0,608	0,712	0,869	0,614	0,720
+FP,EE,DM in Vektor	0,841	0,608	0,712	0,869	0,624	0,725
+Ohne FP	0,853	0,615	0,715	0,865	0,624	0,725
+Komplex	0,822	0,621	0,707	0,835	0,631	0,719

den, wird deutlich, dass die Lernmodelle nur wenige Zusammenhänge anhand der gefüllten Pausen erlernen können, obwohl diese kennzeichnend für eine Disfluenz sind. Es ist jedoch auch möglich, dass die gefüllten Pausen im Switchboard-Korpus nicht oft genug innerhalb von Interregna auftreten.
Abschließend wird noch eine Version betrachtet, in dem die komplexen Disfluenzen Abschnitt 5.7.2 berücksichtigt und entsprechend modelliert werden. Dafür wurden zwei weitere Ausgabeklassen definiert (siehe Abschnitt 6.2.3). Diese komplexen Disfluenzen besitzen jedoch nur eine geringfügige Anzahl an Repräsentationen im Trainingskorpus, weshalb auch viele falsch positive Ergebnisse erzielt werden, die zur schlechtesten Präzision innerhalb aller Versionen für die normal kombinierten Metriken für die RM- und RS-Klassen führen. Für die binären Metriken werden diese falsch positiven Ergebnisse jedoch oft wieder als richtig positiv gewertet, wodurch der durch zusätzliche Klassen entstandene Nachteil wieder kompensiert wird. Dadurch wird der höchste binäre F1-Wert unter den RM-Klassen erreicht.
Abschließend lässt sich jedoch schlussfolgern dass diese Änderungen nur zu einer unwesentlichen Erhöhung oder Senkung der Werte führte. Im Schnitt über alle Werte hinweg verbleibt die Basisversion als sinnvollstes Modell für die Klassifikation innerhalb des Switchboard-Korpus.

Abschließend wird die Basisversion mit dem Verfahren von Wang et al. [WCLL16] verglichen, das auf dem Switchboard-Testsatz mit einem F1-wert von 0,871 den aktuellen Bestwert erreicht. Da das Verfahren nicht die Reparans berücksichtigt, werden nur die binären Metriken bezüglich der Reparanda betrachtet. In Tabelle 7.6 werden die binären Metriken Präzision, Ausbeute und das F1-Maß mit denen des Verfahren von Wang et al. verglichen. Während die Präzision nahezu gleich ist, ist die Ausbeute und damit der F1-Wert des Vergleichsverfahrens wesentlich höher. Das Verfahren besitzt ein ähnlich aufgebautes Netzwerk als maschinelles Lernmodell und verwendet auch den nahezu gleich aufgebauten diskreten Vektor als Eingabemerkmal. Dennoch werden wesentlich mehr Wörter der RM-Klassen erkannt. Dies kann unterschiedliche Gründe haben. Zum einen verwendet das Vergleichsverfahren nach dem bidirektionalen rekurrenten neuronalen Netz mit langem Kurzzeitgedächtnis (engl. long short-term memory - LSTM) noch zusätzlich ein Conditional Random Field (CRF) als Vorverarbeitungsschicht vor der Ausgabe. Dadurch können

Tabelle 7.6.: Vergleich der binären Metriken auf dem Switchboard-Testdatensatz

Modell	Präzision	Ausbeute	F1-Wert
Basisversion	0,906	0,702	0,792
BI-LSTM-CRF [WCLL16]	0,91	0,836	0,871

noch weitere Abhängigkeiten zwischen den Ausgabeklassen berücksichtigt werden. Dies führte im Verfahren aus [WCLL16] jedoch nur zu einer Verbesserung des F1-Wertes um 0,01. Im Vergleichsverfahren wurde jedoch auch mit Pocket CRF [poc] ein anderes verfahren zur Generierung der Wortartmarkierung eingesetzt. Des Weiteren wurden für die Wortvektoren ein eigenes Modell auf einem Sprachkorpus trainiert und vorkonfiguriert. Ein anderer wesentlicher Grund ist jedoch die Tatsache, dass für den Trainingsprozess der Basisversion im Gegensatz zum Vergleichsverfahren die ursprüngliche Segmentierung des Switchboard-Korpus in Sätze aufgehoben wurde und mehrere Sätze zu einer größeren Eingabesequenz hinzugefügt wurden. Dies sollte eine gegenüber dem Korpus des Projektes PARSE ähnliche Eingabe generieren, die die nachfolgende Klassifizierung erleichtern sollte. Jedoch wurden dadurch neue Wortumgebungen gebildet, durch die womöglich andere Zusammenhänge im Lernverfahren ermittelt werden, wodurch die Basisversion gewisse Disfluenzen schlechter erkennt. Durch diese fehlende ursprüngliche Segmentierung der Eingaben ist die Basisversion nur bedingt mit dem Verfahren aus [WCLL16] vergleichbar, da beide Modelle unterschiedliche Eingaben vor dem Trainingsprozess erhielten. Weitere Unterschiede sind wahrscheinlich in den anderen unterschiedlichen Einstellungen und Konfigurationen der beiden Netzwerke zu finden. Aus diesen möglichen Gründen erzielt die Basisversion dieser Arbeit schlechtere Ergebnisse als das Vergleichsverfahren.

7.3.2. Ergebnisse des PARSE-Korpus

In diesem Abschnitt werden die Ergebnisse des Werkzeug, das ein Klassifikator für Disfluenzen ist, auf den Transkriptionen des PARSE-Korpus, der bereits in Abschnitt 7.1 beschrieben wurde, präsentiert und bewertet. In Tabelle 7.7 sind die Ergebnisse für die Reparatur-Klassen sowie deren binäre Kombination (siehe Abschnitt 7.2) hinsichtlich der Metriken Präzision, Ausbeute und F1-Maß dargestellt. Als Lernmodell wurde die in Abschnitt 7.3.1 beschriebene Basisversion verwendet. Da die gefüllten Pausen leicht zu erkennen sind und ähnliche Werte wie auf dem Switchboard-Korpus aufweisen, wird auf deren Darstellung in diesem Abschnitt verzichtet. Während die Präzision für alle Klassen außer I-RM akzeptabel sind die Werte der Ausbeute sehr niedrig, was auch zu einen niedrigen F1-Wert führt. Die niedrige Ausbeute bedeutet, dass es viele falsch negative Ergebnisse gibt, was bedeutet, dass viele der Reparaturen nicht erkannt werden. Zwar werden das Ergebnisse unter Verwendung der binären Metriken verbessert, jedoch sind diese immer noch zu niedrig für einen möglichst zuverlässigen Klassifikator. Diese niedrige Ausbeute lässt sich teilweise daher erklären, dass das entwickelte Werkzeug auf dem Switchboard-Korpus trainiert wurde, dessen Eingaben anderen Formulierungen und Strukturen aufweisen. Zwar wurde versucht, durch das Entfernen von Satzzeichen und das Zusammenfügen von mehreren Sätzen auf möglichst ähnlichen Eingaben zu trainieren, jedoch scheint dies nicht besonders erfolgreich zu sein.

Deshalb wurde als erster Schritt versucht, die Eingabemerkmale Abschnitt 6.2.2 zu modifizieren, sodass bessere Ergebnisse auf den PARSE-Eingaben erzielt werden. Da sich die Wörter sowie Wortarten und Wortgruppen nicht einfach ändern lassen, wurden der diskrete Umgebungsvektor modifiziert. So wurde mit der Jaro-Winkler-Distanz ein anderes

Tabelle 7.7.: Die Ergebnisse des PARSE-Korpus auf der Basisversion

Ausgabeklasse	Präzision	Ausbeute	F1
B-RM	0,765	0,433	0,553
I-RM	0,615	0,444	0,516
B-RS	0,706	0,400	0,511
I-RS	0,700	0,333	0,452
RM-Bin	0,767	0,479	0,590
RS-Bin	0,741	0,392	0,513

Tabelle 7.8.: Die Ergebnisse des PARSE-Korpus auf einer Basisversion mit neuem diskreten Vektor

Ausgabeklasse	Präzision	Ausbeute	F1
B-RM	0,750	0,500	0,600
I-RM	0,643	0,500	0,563
B-RS	0,737	0,467	0,571
I-RS	0,800	0,381	0,516
RM-Bin	0,765	0,542	0,634
RS-Bin	0,759	0,431	0,550

Ähnlichkeitsmaß für die unmittelbare Wortumgebung ausgewählt und benachbarte Wörter, von denen das vordere ein Präfix des anderen ist, werden in diesem veränderten Vektor als das gleiche Wort betrachtet. Dadurch kann beispielsweise die Disfluenz *„... up upper ...“* erkannt werden. Die Ergebnisse dieser Änderungen sind in Tabelle 7.8 dargestellt. Es wurden wesentlich bessere Ergebnisse für die Ausbeute und damit auch für den F1-Wert erzielt. Die größten Verbesserungen traten dabei für die Klassen B-RM und B-RS auf, da durch die Änderungen einigen Disfluenzen wie beispielsweise *„... up upper ...“* erkannt wurden.

Diese Werte sind jedoch noch nicht ausreichend und deshalb wurde überprüft, was eine Veränderung des Eingabetextes bewirkt. Da in einigen Arbeiten zur Erkennung von Disfluenzen, wie beispielsweise in dem Verfahren aus [WCLL16], in dem der aktuelle Bestwert bezüglich dem F1-Wert erreicht wurde, die gefüllten Pausen vor der Verarbeitung aus den Korpus entfernt werden, wurde untersucht, wie ein Lernmodell, das ohne gefüllte Pausen trainiert wurde, die Eingaben des PARSE-Korpus analysiert. Dabei wurden auch die selben Änderungen bezüglich dem diskreten Umgebungsvektor beibehalten Die Ergebnisse sind in Tabelle 7.9 dargestellt. Während zwar die Präzision und Ausbeute der Klasse B-RM niedrigere Werte aufweisen, stiegen die diese beiden Metriken und damit der F1-Wert für die anderen Klassen teilweise erheblich. Die größten Änderungen weisen dabei die Klassen I-RM und B-RS auf. Durch die Entfernung von gefüllten Pausen sind dabei einige Wiederholungsstrukturen deutlich geworden und es wurde bereits in Abschnitt 7.3.1 gezeigt, dass das verwendete Lernmodell nahezu keine Zusammenhänge für die Erkennung von Disfluenzen aus gefüllten Pausen extrahieren kann.

Abschließend erfolgte noch eine Nachverarbeitung der Schätzungen, in der versuchte wurde, unvollständige Reparanda und Reparans zu vervollständigen. So wurde beispielsweise geprüft, ob vor einer I-RM oder I-RS Markierung eine zugehörige B-RM oder B-RS Markierung liegt und ansonsten wird die entsprechende Markierung gesetzt. Des Weiteren wird auch nach korrespondierende Wörtern für einzelne Reparatur-Markierungen innerhalb ei-

Tabelle 7.9.: Die Ergebnisse des PARSE-Korpus ohne gefüllte Pausen

Ausgabeklasse	Präzision	Ausbeute	F1
B-RM	0,667	0,467	0,549
I-RM	0,6875	0,611	0,647
B-RS	0,833	0,500	0,625
I-RS	0,818	0,429	0,563
RM-Bin	0,730	0,562	0,635
RS-Bin	0,828	0,471	0,600

Tabelle 7.10.: Die Ergebnisse des PARSE-Korpus für eine Nachverarbeitung der Schätzungen

Ausgabeklasse	Präzision	Ausbeute	F1
B-RM	0,690	0,667	0,678
I-RM	0,6875	0,611	0,647
B-RS	0,870	0,667	0,755
I-RS	0,846	0,524	0,647
RM-Bin	0,733	0,688	0,710
RS-Bin	0,861	0,608	0,713

nes kurzen Intervalls mit einer Länge von zwei oder drei Wörtern gesucht. Es wird auch sichergestellt, dass direkte Wortwiederholungen als Reparaturen markiert werden. Die Ergebnisse nach dieser Nachverarbeitung der Schätzungen sind in Tabelle 7.10 dargestellt. Die Werte der Ausbeute und damit auch die F1-Werte für die Klassen B-RM und B-RS stiegen besonders durch diese Änderungen an. das binäre F1-Maß der RM- und RS-Klassen erreicht mit einem Wert von mehr als 0,710 eine akzeptable Stand. Dennoch ließe sich in späteren Entwicklungsschritten noch einiges bessern. So wird die Eingabe „*... next to the green cup the green cup is located ...*“ fälschlicherweise als Disfluenz erkannt, da es sich um eine Wiederholung von mehreren Wörtern handelt. In diesem Fall ist die Bedingung des entsprechenden Szenarios, dass die Position der Gegenstände spezifiziert werden muss nicht förderlich, da solche Formulierungen wiederholt in den Transkriptionen auftreten und stets zu falsch positiven Ergebnisse führen. Zur Behebung solcher Probleme müsste der Klassifikator zwischen Subjekt und Objekt innerhalb einer Eingabe unterscheiden können, was im Rahmen dieser Arbeit nicht berücksichtigt wurde. Des Weiteren konnten auch Korrespondenzen zwischen einigen Wörtern nicht gefunden werden, da diesen durch die seichte Sprachverarbeitung Kapitel 3 ungültige Wortarten zugewiesen wurden. Dennoch wird mit dem binären F1-Maß der RM- und RS-Klassen jeweils ein akzeptabler Wert von mehr als 0,710 erreicht.

7.4. Bewertung der Ergebnisse

Insgesamt hat sich ergeben, dass sich disfluente Strukturen prinzipiell mit dem entwickelten Werkzeug erkennen und korrigieren lassen. So wurde mit der Basisversion auf dem Testdatensatz des Switchboard-Korpus mit einem binären F1-Wert von 0,792 für die Reparandum-Klassen eine geeignete F1-Rate erreicht. Hierbei ist die erreichte Präzision von 0,906 bereits gut, jedoch lässt die Ausbeute von 0,703 noch wesentlich verbessern. So liegt der resultierende F1-Wert noch unterhalb viele der in Kapitel 4 vorgestellten Verfahren, hierbei sind diese Verfahren jedoch nur bedingt mit dem in dieser Arbeit entwickelten Lernmodell vergleichbar, da der Switchboard-Datensatz unterschiedlich vorverarbeitet

wurde und dadurch die Eingabesequenzen verschieden definiert sind.
Beim Vergleich mit unterschiedlichen Kombinationen von Eingabemerkmalen stellt sich heraus, dass beispielsweise die Wortgruppen keinen bedeutsamen Einfluss auf den Lernprozess und die nachfolgende Disfluenzen aufweisen. Auch die in der Analyse als für eine Disfluenz signalisierend bezeichneten gefüllte Pausen, bieten dem Lernverfahren nahezu keine Informationen, durch die sich Lernprozess wesentlich verbessert.
Auf dem PARSE-Korpus wurde in einer erweiterten und auf den Korpus angepasste Version des Lernmodell ein binärer F1-Wert von 0,71 für die Reparandum-Klassen erreicht. Dieser Wert ist zwar akzeptabel, jedoch lässt er sich auch noch wesentlich verbessern. Die zugehörige Präzision weist dabei ein Wert von 0,733 auf. Um die entsprechenden falsch positiven Ergebnisse zu verringern, würde der Klassifikator beispielsweise Kenntnisse zu semantischen Rollen benötigen [CM05], um Wortgruppen korrekt als Subjekt oder Objekt zu unterscheiden zu können. Dadurch könnten gewollte von ungewollten Wiederholungen unterschieden werden. Sowohl die binäre Präzision der RM-Klassen als auch die Ausbeute mit einem Wert von 0,688 würden von einem zusätzlichen Condtional Random Field (CRF) als Verarbeitungsschicht vor der Ausgabe profitieren, da dadurch auch Abhängigkeiten zwischen den Ausgabeklassen, wie beispielsweise, dass eine B-RM Markierung vor einer I-RM Markierung erfolgen muss, berücksichtigt werden.
Diese wäre wahrscheinlich auch weniger fehleranfällig als die manuell durchgeführte Korrektur der Schätzungen. Insgesamt ist jedoch eine erfolgreiche Erkennung von Disfluenzen mit dem modifizierten Lernmodell möglich.

8. Zusammenfassung und Ausblick

Diese Arbeit beschäftigt sich mit der Erkennung und Korrektur von Disfluenzen in gesprochener Sprache. Dazu wurde ein Werkzeug entwickelt, dass disfluente Wörter in der Eingabe erkennt und diese sowie ihre zugehörigen sprachlichen Korrekturen entsprechend annotiert. Dieses Werkzeug sollte als Agent für das Sprachverständnis innerhalb des Projektes PARSE entwickelt werden. Es wurde dabei zunächst die Problemstellung analysiert und diverse verwandte Arbeiten betrachtet, in denen diese Thematik bereits ausführlich behandelt wurde. Basierend auf diesen Arbeiten wurde in der Entwurfsphase entschieden, dass die Erkennung von Disfluenzen durch ein maschinelles Lernverfahren realisiert wird, da diese Verfahren besser mit den unregelmäßige Strukturen von Disfluenzen zurecht kommen. Für das maschinelle Lernverfahren wurde gemäß den verwandten Arbeiten ein bidirektionales neuronales Netzwerk mit langen Kurzzeitgedächtnis (engl. long short-term memory - LSTM) gewählt, da dieses Modell die besten Erkennungsraten für Disfluenzen aufweist. Als Trainingsdatensatz für das Verfahren wurde der Switchboard-Korpus verwendet, der auch in den meisten verwandten Arbeiten im Trainingsprozess für die Disfluenzerkennung verwendet wird. Die Auswahl der Eingabemerkmale basierte dabei auf den in der Analyse betrachteten Zusammenhängen zwischen den Bestandteilen einer Disfluenz sowie anhand der in anderen Arbeiten eingesetzten Merkmalen. Das maschinelle Lernverfahren wurde mit Hilfe der Programmbibliothek Deeplearning4j implementiert.

In der Evaluationsphase wurde die Funktionsweise des entwickelten Werkzeuges anhand von händisch erstellten Transkriptionen sowie mit dem Testdatensatz des Switchboard-Korpus bewertet. Dabei wurde mit einer grundlegenden Version auf dem Switchboard-Testdatensatz eine binäre Präzision von 0,906 sowie eine Ausbeute von 0,703 für die fehlerhaften Wörter einer Disfluenz, dem Reparandum, erreicht. Der zugehörigen F1-Wert ist hierbei 0,792 und damit ein geeigneter Wert, wenn auch dieser sich noch wesentlich verbessern lässt. Diese Werte sind zwar geringer als viele der in den verwandten Arbeiten erreichten Werte, jedoch lässt sich dies nur bedingt vergleichen, da es wesentliche Unterschiede bei der Vorverarbeitung des Switchboard-Datensatzes gibt und die daraus resultierenden Eingabesequenzen verschieden sind. Des Weiteren befassen sich die meisten der verwandten Arbeiten nicht mit der Erkennung der zugehörigen sprachlichen Korrekturen, den Reparans. Auf den Transkriptionen des PARSE-Korpus wurde für die fehlerhaften Wörter der Disfluenzen eine Präzision von 0,733 und eine Ausbeute von 0,688 und damit ein F1-Wert von 0,710 erreicht. Dieser Wert ist ebenfalls akzeptabel, jedoch auch noch deutlich verbesserbar. Es hat sich in der Evaluation auch ergeben, dass die Wahl von

Wortgruppen als Eingabemerkmal zu keiner wesentlichen Verbesserung führt. Insgesamt wurde jedoch gezeigt, dass das erstellte Werkzeug Disfluenzen prinzipiell erkennen und mit ihren geeigneten Korrekturen annotieren kann.

In zukünftigen Arbeiten kann der Klassifikator des Agenten noch mit weiteren Merkmalen, die in dieser Arbeit nicht in Betracht gezogen wurden angereichert werden, wie beispielsweise Informationen bezüglich den semantischen Rollen. Durch diese ließen sich in einigen Situationen ungewollte und gewollten Wiederholungen unterscheiden und damit einige falsch positive Ergebnisse vermeiden, was zu einer Erhöhung der Präzision führen würde. Des Weiteren kann im neuronalen Netzwerk eine weitere Schicht vor der Ausgabe, beispielsweise mit einem Conditional Random Field (CRF), definiert werden. Dadurch ließen sich die Abhängigkeiten zwischen den Ausgabeklassen besser berücksichtigen, was ebenfalls zu einer Verbesserung der Ergebnisse führen kann.

Literaturverzeichnis

[ARA+06] Asfour, Tamim ; Regenstein, Kristian ; Azad, Pedram ; Schroder, J ; Bierbaum, Alexander ; Vahrenkamp, Nikolaus ; Dillmann, Rüdiger: ARMAR-III: An integrated humanoid platform for sensory-motor control. In: *Humanoid Robots, 2006 6th IEEE-RAS International Conference on* IEEE, 2006, S. 169–175 (zitiert auf den Seiten 1, 11, 19 und 45).

[BDS92] Bear, John ; Dowding, John ; Shriberg, Elizabeth: Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. In: *Proceedings of the 30th annual meeting on Association for Computational Linguistics* Association for Computational Linguistics, 1992, S. 56–63 (zitiert auf den Seiten 13, 14 und 22).

[Bes06] Besser, Jana: *A Corpus-Based Approach to the Classification and Correction of Disfluencies in Spontaneous Speech*, Saarland University, Germany, Bachelor's Thesis, November 2006 (zitiert auf Seite 5).

[Bußo2] Bussmann, Hadumod [. (Hrsg.): *Lexikon der Sprachwissenschaft.* 3., aktualisierte u. erw. Aufl. Stuttgart : Kröner, 2002 (zitiert auf den Seiten 3 und 27).

[CJ01] Charniak, Eugene ; Johnson, Mark: Edit detection and parsing for transcribed speech. In: *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies* Association for Computational Linguistics, 2001, S. 1–9 (zitiert auf den Seiten 16, 23, 35 und 45).

[CM05] Carreras, Xavier ; Màrquez, Lluís: Introduction to the CoNLL-2005 shared task: Semantic role labeling. In: *Proceedings of the ninth conference on computational natural language learning* Association for Computational Linguistics, 2005, S. 152–164 (zitiert auf Seite 56).

[CRF03] Cohen, William ; Ravikumar, Pradeep ; Fienberg, Stephen: A comparison of string metrics for matching names and records. In: *Kdd workshop on data cleaning and object consolidation* Bd. 3, 2003, S. 73–78 (zitiert auf Seite 38).

[DGA+93] Dowding, John ; Gawron, Jean M. ; Appelt, Doug ; Bear, John ; Cherny, Lynn ; Moore, Robert ; Moran, Douglas: Gemini: A natural language system for spoken-language understanding. In: *Proceedings of the 31st annual meeting on Association for Computational Linguistics* Association for Computational Linguistics, 1993, S. 54–61 (zitiert auf Seite 13).

[dl4a] *Deep Learning for Java.* `https://deeplearning4j.org/`. – Zuletzt zugegriffen am: 04.06.2018 (zitiert auf Seite 35).

[dl4b] *Deep Learning for Java.* `https://deeplearning4j.org/documentation`. – Zuletzt zugegriffen am: 04.06.2018 (zitiert auf Seite 40).

[FDK15] Ferguson, James ; Durrett, Greg ; Klein, Dan: Disfluency Detection with a Semi-Markov Model and Prosodic Features. In: *HLT-NAACL*, 2015, S. 257–262 (zitiert auf den Seiten 14, 16 und 23).

[Geo09] Georgila, Kallirroi: Using integer linear programming for detecting speech disfluencies. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers* Association for Computational Linguistics, 2009, S. 109–112 (zitiert auf den Seiten 13, 14, 15, 23 und 24).

[GHM92] Godfrey, John J. ; Holliman, Edward C. ; McDaniel, Jane: SWITCHBOARD: Telephone speech corpus for research and development. In: *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on* Bd. 1 IEEE, 1992, S. 517–520 (zitiert auf den Seiten 15 und 35).

[Gün15] Günes, Zeynep: *Aufbau eines Sprachkorpus zur Programmierung autonomer Roboter mittels natürlicher Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, Mai 2015. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/guenes_ba` (zitiert auf den Seiten 12 und 46).

[GWG10] Georgila, Kallirroi ; Wang, Ning ; Gratch, Jonathan: Cross-domain speech disfluency detection. In: *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue* Association for Computational Linguistics, 2010, S. 237–240 (zitiert auf den Seiten 15 und 24).

[Hey16] Hey, Tobias: *Kontext- und Korreferenzanalyse für gesprochene Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Master's Thesis, September 2016. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/hey_ma` (zitiert auf den Seiten 12 und 46).

[Hin83] Hindle, Donald: Deterministic parsing of syntactic non-fluencies. In: *Proceedings of the 21st annual meeting on Association for Computational Linguistics* Association for Computational Linguistics, 1983, S. 123–128 (zitiert auf den Seiten 13, 14, 22, 25 und 34).

[HJ14] Honnibal, Matthew ; Johnson, Mark: Joint incremental disfluency detection and dependency parsing. In: *Transactions of the Association for Computational Linguistics* 2 (2014), S. 131–142 (zitiert auf den Seiten 14 und 22).

[Hof16] Hoffmann, Tizian: *Disambiguierung gesprochener Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, April 2016. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/hoffmann_ba` (zitiert auf Seite 12).

[HS15] Hough, Julian ; Schlangen, David: Recurrent neural networks for incremental disfluency detection. In: *Interspeech 2015* (2015) (zitiert auf den Seiten 14, 16, 23 und 24).

[IFR15] IFR: *World Robotics Survey: Industrial robots are conquering the world*. `https://ifr.org/ifr-press-releases/news/world-robotics-survey-industrial-robots-are-conquering-the-world-`. Version: 09 2015. – Zuletzt zugegriffen am: 14.11.2017 (zitiert auf Seite 1).

[JC04] Johnson, Mark ; Charniak, Eugene: A TAG-based noisy channel model of speech repairs. In: *Proceedings of the 42nd Annual Meeting on Association for*

Computational Linguistics Association for Computational Linguistics, 2004, S. 33 (zitiert auf den Seiten 14, 16, 17, 23 und 35).

[JH93] John, Godfrey ; Holliman, Edward: Switchboard-1 Release 2 LDC97S62. In: *Web Download. Philadelphia: Linguistic Data Consortium* (1993) (zitiert auf Seite 35).

[JM09] Jurafsky, Dan ; Martin, James H.: *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. 2. ed., Pearson International Edition. Prentice Hall, Pearson Education International, 2009 (zitiert auf den Seiten 4, 20, 24 und 47).

[JM16] Jurafsky, Daniel ; Martin, James H.: Semantics with Dense Vectors. In: *Web. Speech and Language Processing* (2016) (zitiert auf Seite 8).

[JM17] Jurafsky, Daniel ; Martin, James H.: *Vector Semantics*. `https://web.stanford.edu/~jurafsky/slp3/15.pdf`. Version: August 2017. – Zuletzt zugegriffen am: 03.06.2018 (zitiert auf Seite 7).

[Jon11] Jondral, Friedrich: *Nachrichtensysteme : Grundlagen, Verfahren, Anwendungen; mit ... 10 Tabellen*. 4., Ã¼berarb. und erweiterte Aufl. Wilburgstetten : Schlembach, 2011 `http://www.schlembach-verlag.de/buecher.php?bnr=68`. – ISBN 978–3–935340–68–7. – Erscheint: MÃ¤rz 2011 (zitiert auf Seite 6).

[Kie16] Kiesel, Viktor: *Optimierung von POS-Tagger-Ergebnissen*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, April 2016. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/kiesel_ba` (zitiert auf Seite 12).

[Koc15] Kocybik, Markus: *Projektion von gesprochener Sprache auf eine Handlungsrepräsentation*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, Juli 2015. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/kocybik_ba` (zitiert auf den Seiten 12, 23 und 36).

[LJ17] Lou, Paria J. ; Johnson, Mark: Disfluency Detection using a Noisy Channel Model and a Deep Neural Language Model. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* Bd. 2, 2017, S. 547–553 (zitiert auf den Seiten 6, 14, 17, 23, 24, 40 und 41).

[LPM+12] Lally, Adam ; Prager, John M. ; McCord, Michael C. ; Boguraev, Branimir K. ; Patwardhan, Siddharth ; Fan, James ; Fodor, Paul ; Chu-Carroll, Jennifer: Question analysis: How Watson reads a clue. In: *IBM Journal of Research and Development* 56 (2012), Nr. 3.4, S. 2–1 (zitiert auf Seite 20).

[LSS+06] Liu, Yang ; Shriberg, Elizabeth ; Stolcke, Andreas ; Hillard, Dustin ; Ostendorf, Mari ; Harper, Mary: Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. In: *IEEE Transactions on audio, speech, and language processing* 14 (2006), Nr. 5, S. 1526–1540 (zitiert auf den Seiten 15 und 23).

[MCCD13] Mikolov, Tomas ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: Efficient estimation of word representations in vector space. In: *arXiv preprint arXiv:1301.3781* (2013) (zitiert auf Seite 8).

[Mil95] Miller, George A.: WordNet: a lexical database for English. In: *Communications of the ACM* 38 (1995), Nr. 11, S. 39–41 (zitiert auf Seite 11).

[MMS93] Marcus, Mitchell P. ; Marcinkiewicz, Mary A. ; Santorini, Beatrice: Building a large annotated corpus of English: The Penn Treebank. In: *Computational linguistics* 19 (1993), Nr. 2, S. 313–330 (zitiert auf Seite 4).

[MSB+14] Manning, Christopher ; Surdeanu, Mihai ; Bauer, John ; Finkel, Jenny ; Bethard, Steven ; McClosky, David: The Stanford CoreNLP natural language processing toolkit. In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, S. 55–60 (zitiert auf Seite 36).

[Ou16] Ou, Yue: *Erkennung von Aktionen in gesprochener Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, September 2016. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/ou_ba` (zitiert auf Seite 12).

[Pas15] Paskaran, Dinesh: *Evaluation unterschiedlicher Spracherkennungssysteme in der Domäne Humanoide Robotik*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, November 2015. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/paskaran_ba` (zitiert auf Seite 12).

[PGB+11] Povey, Daniel ; Ghoshal, Arnab ; Boulianne, Gilles ; Burget, Lukas ; Glembek, Ondrej ; Goel, Nagendra ; Hannemann, Mirko ; Motlicek, Petr ; Qian, Yanmin ; Schwarz, Petr u. a.: The Kaldi speech recognition toolkit. In: *IEEE 2011 workshop on automatic speech recognition and understanding* IEEE Signal Processing Society, 2011 (zitiert auf Seite 15).

[poc] *Pocket CRF*. `https://sourceforge.net/projects/pocket-crf-1/files/`. – Zuletzt zugegriffen am: 10.06.2018 (zitiert auf Seite 53).

[Pre98] Prechelt, Lutz: Early stopping-but when? In: *Neural Networks: Tricks of the trade*. Springer, 1998, S. 55–69 (zitiert auf Seite 41).

[PSM] Pennington, Jeffrey ; Socher, Richard ; Manning, Christopher D.: *GloVe: Global Vectors for Word Representation*. `https://nlp.stanford.edu/projects/glove/`. – Zuletzt zugegriffen am: 04.06.2018 (zitiert auf Seite 36).

[PSM14] Pennington, Jeffrey ; Socher, Richard ; Manning, Christopher: Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, S. 1532–1543 (zitiert auf den Seiten 8 und 36).

[QL13] Qian, Xian ; Liu, Yang: Disfluency detection using multi-step stacked learning. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, S. 820–825 (zitiert auf den Seiten 14, 15 und 23).

[RT13] Rasooli, Mohammad S. ; Tetreault, Joel R.: Joint Parsing and Disfluency Detection in Linear Time. In: *EMNLP*, 2013, S. 124–129 (zitiert auf den Seiten 14 und 22).

[Sch15] Schneider, Michael: *Entwurf einer Handlungsrepräsentation für gesprochene Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, Mai 2015. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/schneider_ba` (zitiert auf Seite 12).

[Sch16] Scheu, Sven: *Aufbereitung von Spracherkennerausgaben*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Master's Thesis, Juli 2016. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/scheu_ma` (zitiert auf Seite 12).

[SCRB91] Schachter, Stanley ; Christenfeld, Nicholas ; Ravina, Bernard ; Bilous, Frances: Speech disfluency and the structure of knowledge. In: *Journal of Personality and Social Psychology* 60 (1991), Nr. 3, S. 362 (zitiert auf Seite 27).

[SHK+14] Srivastava, Nitish ; Hinton, Geoffrey ; Krizhevsky, Alex ; Sutskever, Ilya ; Salakhutdinov, Ruslan: Dropout: A simple way to prevent neural networks from overfitting. In: *The Journal of Machine Learning Research* 15 (2014), Nr. 1, S. 1929–1958 (zitiert auf Seite 41).

[Shr94] Shriberg, Elizabeth E.: *Preliminaries to a theory of speech disfluencies*, University of California, Berkeley, Diss., 1994 (zitiert auf den Seiten 5, 30 und 48).

[Ste16] Steurer, Vanessa: *Strukturerkennung von Bedingungen in gesprochener Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, April 2016. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/steurer_ba` (zitiert auf den Seiten 12 und 46).

[SW07] Schneider, Uwe [. (Hrsg.) ; Werner, Dieter [. (Hrsg.): *Taschenbuch der Informatik : mit ... 108 Tabellen.* 6., neu bearb. Aufl. Leipzig : Fachbuchverl. Leipzig im Carl-Hanser-Verl., 2007. – ISBN 3–446–40754–5; 978–3–446–40754–1 (zitiert auf Seite 6).

[ten] *org.tensorflow.* `https://www.tensorflow.org/api_docs/java/reference/org/tensorflow/package-summary`. – Zuletzt zugegriffen am: 05.06.2018 (zitiert auf Seite 35).

[WCLL16] Wang, Shaolei ; Che, Wanxiang ; Liu, Yijia ; Liu, Ting: Enhancing Neural Disfluency Detection with Hand-Crafted Features. In: *China National Conference on Chinese Computational Linguistics* Springer, 2016, S. 336–347 (zitiert auf den Seiten 14, 17, 23, 24, 34, 36, 37, 38, 40, 41, 49, 52, 53 und 54).

[WT15] Weigelt, Sebastian ; Tichy, Walter F.: Poster: ProNat: An Agent-Based System Design for Programming in Spoken Natural Language. In: *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on* Bd. 2 IEEE, 2015, S. 819–820 (zitiert auf Seite 11).

[YSM16] Yoshikawa, Masashi ; Shindo, Hiroyuki ; Matsumoto, Yuji: Joint Transition-based Dependency Parsing and Disfluency Detection for Automatic Speech Recognition Texts. In: *EMNLP*, 2016, S. 1036–1041 (zitiert auf den Seiten 14, 15 und 22).

[ZJ11] Zwarts, Simon ; Johnson, Mark: The impact of language models and loss functions on repair disfluency detection. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* Association for Computational Linguistics, 2011, S. 703–711 (zitiert auf Seite 17).

[ZOH14] Zayats, Victoria ; Ostendorf, Mari ; Hajishirzi, Hannaneh: Multi-domain disfluency and repair detection. In: *Fifteenth Annual Conference of the International Speech Communication Association*, 2014 (zitiert auf den Seiten 14, 16, 23 und 24).

[ZOH16] Zayats, Vicky ; Ostendorf, Mari ; Hajishirzi, Hannaneh: Disfluency detection using a bidirectional LSTM. In: *arXiv preprint arXiv:1604.03209* (2016) (zitiert auf den Seiten 14, 16, 23, 24 und 40).

Anhang

A. Erweiterung des Korpus

Der PARSE-Korpus wurde um zwei Szenarien erweitert, die entsprechend so entworfen wurden, dass mehr Disfluenzen auftreten. Die Probanden wurden dabei zunächst über die Projektsituation informiert, weshalb ihnen bekannt ist, dass das Zielsystem ARMAR-III in einer Küchenumgebung darstellt und dem Küchenroboter grundlegende Aktionen bekannt sind.

A.1. Szenario 1: Red cup in the cupboard and orange juice on the table

In this scene you want the robot to take a red cup (located in the top rack of the dishwasher) and put it somewhere in the cupboard (assume the cupboard is empty). Afterwards the robot has to take the orange juice out of the fridge and place it next to the green cup on the table. Finally the robot has to close the window. Try to keep the instructions in the specified order.

Note:*Some task may require that you have to describe more fine-grained actions. E.g. to get an object from the fridge, you have to tell the robot to go to the fridge and open it and afterwards close it. Assume that at the beginning of the scene the fridge, cupboard and dishwasher are closed while the window is opened.*

Note: *The robot is aware of the locations of the fridge, dishwasher, cupboard, table and window. You only have to specify the location of the objects inside them, e.g., the instructions for taking the multivitamin juice out of the fridge may include:*
„..., take the multivitamin juice, it is located on the forth shelf between the orange juice and the milk, ...“

Note: *Errors within your instructions are acceptable but try to correct them as you spot them. Also do not be afraid to use hesitation words such as uhm, ehm and the like.*

A.2. Szenario 2: Filling a red cup and the green cup

In this scene you want the robot to first open the window and then fill a red cup (located in the top rack of the dishwasher) and the green cup (located on the table) with orange juice (located in the fridge). Afterwards the robot has to close the window. Except for opening

and closing the window the order of your instructions is unimportant but you have to follow the following conditions for the instructions:

- *the robot can only carry one object at the same time*
- *to fill an object it has first to be placed on the table*

Note:*Some task may require that you have to describe more fine-grained actions. E.g. to get an object from the fridge, you have to tell the robot to go to the fridge and open it and afterwards close it. Assume that at the beginning of the scene the fridge, cupboard and dishwasher are closed while the window is opened.*

Note: *The robot is aware of the locations of the fridge, dishwasher, cupboard, table and window. You only have to specify the location of the objects inside them, e.g., the instructions for taking the multivitamin juice out of the fridge may include:*
„..., take the multivitamin juice, it is located on the forth shelf between the orange juice and the milk, ...“

Note: *Errors within your instructions are acceptable but try to correct them as you spot them. Also do not be afraid to use hesitation words such as uhm, ehm and the like.*