Karlsruhe Institute of Technology

Aufbereitung von Spracherkennerausgaben

Masterarbeit
von

Sven Scheu

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

Bearbeitungszeit: 18.12.2015 – 15.07.2016

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, 15.07.2016

. . . . . .

(**Sven Scheu**)

Kurzfassung

Diese Arbeit befasst sich mit der Aufbereitung von Spracherkennerausgaben. Hierbei geht es um die Verbesserung der Wortfehlerrate, sowie die Erkennung von Sprachunstetigkeiten und Befehlsgrenzen.

Die Verbesserung der Wortfehlerrate wird genauer betrachtet. Hierbei wird mit Hilfe von Verwirrungsnetzwerken und externen Datenquellen die Erzeugung einer neuen Ausgabe beschrieben, wobei mehrere Spracherkenner gleichzeitig verwendet werden können.

Insgesamt konnte die Wortfehlerrate um bis zu $\approx$ 30%, relativ zum Wert des Gold-Standards der Spracherkennung, verbessert werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Diese Arbeit beschäftigt sich mit der Aufbereitung von Spracherkennerausgaben. Hierbei geht es um die Verbesserung der Wortfehlerrate, sowie die Erkennung von Befehlsgrenzen und Unstetigkeiten. Die Aufbereitung ist nötig, da derzeit kein Spracherkenner existiert, der fehlerfrei Sprache erkennt oder zumindest in einer Qualität, die die Weiterverarbeitung der Spracherkennerausgaben fehlerfrei ermöglicht [Koc15]. Zudem enthält selbst fehlerfrei transkribierte Sprache Fehler, die dem Sprechverhalten des Sprechers entstammen. Zu diesen Fehlern gehören beispielsweise die genannten Unstetigkeiten, die Abweichungen vom normalen Sprachfluss wie der Verzögerungslaut „ähm“ oder Wiederholungen.

Von den Teilbereichen der Aufbereitung beschäftigt sich diese Arbeit im Wesentlichen mit der Verbesserung der Wortfehlerrate. In diesem Rahmen wurde auch ein System implementiert, das die Ausgaben mehrerer Spracherkenner vereinigt. Mit Hilfe externer Informationsquellen zur Bewertung dieser Ausgaben kann eine neue Ausgabe generiert werden, die in der Regel eine geringere Wortfehlerrate aufweist.

Diese Arbeit ist zudem Teil des *PARSE*-Projekts des Instituts für Programmstrukturen und Datenorganisation (IPD) am KIT. Mit diesem Projekt soll die Möglichkeit geschaffen werden, Roboter wie Armar III mithilfe von gesprochener Sprache zu programmieren. Die Aufbereitung nimmt nach der Spracherkennung den ersten Schritt in der Verarbeitung der Sprache ein und soll es späteren Verarbeitungsschritten möglich machen, die Ziele des Gesamtprojekts leichter und besser umzusetzen.

2. Grundlagen

Dies Arbeit setzt sich mit Themen aus dem Umfeld der Computerlinguistik auseinander. In diesem Kapitel werden einige Fachbegriffe aus diesem Feld erklärt, sofern sie für das Verständnis dieser Arbeit relevant sind.

2.1 N-Gramm

Ein N-Gramm ist eine N-elementige Menge gleichartiger Einheiten. Beispielsweise sind „a b“ oder „x y“ Buchstaben-2-Gramme, wohingegen „Hund Katze Haustier“ ein Wörter-3-Gramm ist. In dieser Arbeit wird N-Gramm im Sinne der Wörter-N-Gramme verwendet.

2.2 Verarbeitung natürlicher Sprache

Die Verarbeitung natürlicher Sprache (engl. Natural Language Processing – NLP) ist ein Teilgebiet der Computerlinguistik und befasst sich mit der Verarbeitung natürlicher Sprache durch Computer. Zu diesem Feld gehören beispielsweise die Spracherkennung und die Wortartenbestimmung.

2.2.1 Wortartenbestimmung

Die Wortartenbestimmung (engl. Part-of-speech Tagging — POS Tagging) beschreibt die Zuweisung der entsprechenden Wortart zu jedem Wort innerhalb eines Textes (siehe [JM09]). Je nach POS-Tagger können dabei noch zusätzliche Wortinformationen anfallen, wie zum Beispiel Tempus oder Numerus.

In dieser Arbeit wird zum POS-Tagging das von Kocybik entwickelte *ShallowNLP* verwendet [Koc15].

2.3 Spracherkenner

Ein Spracherkenner (engl. Automatic Speech Recognizer – ASR) ist ein Werkzeug, das Sprache in Text umwandelt. Dafür gibt es eine Reihe von Ansätzen. In der Regel besteht ein Spracherkenner aus: „Signalverarbeitung & Merkmalsextraktion, Akustikmodell, Sprachmodell und Hypothesensucheinheit“ [JM09, YD14]. Die einzelnen Komponenten werden im Folgenden beschrieben.

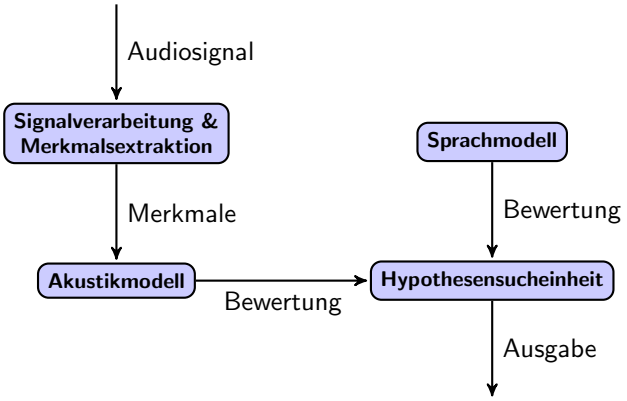

Abbildung 2.1: Spracherkenner Grundaufbau [YD14]

2.3.1 Signalverarbeitung & Merkmalsextraktion

Während der Signalverabeitung & Merkmalsextraktion werden das Audiosignal von Störsignalen bereinigt und für das Akustikmodell relevante Merkmale extrahiert.

2.3.2 Akustikmodell

Das Akustikmodell enthält die nötigen Informationen, um das Audiosignal in Phoneme zu übersetzen. Ein Phonem ist eine Gruppe von Phonen (ein Phon ist die kleinste unterscheidbare Einheit eines Audiosegments), die in einer Sprache zwar unterscheidbar, aber austauschbar sind. Im Englischen werden beispielsweise die „**p**“s in „**p**an“ und „s**p**an“ unterschiedlich ausgesprochen – haben also unterschiedliche Phone, allerdings würde der Austausch dieser Phone nicht die Bedeutung der Worte ändern – es handelt sich also um das gleiche Phonem [Bes95]. (siehe auch [JM09])

2.3.3 Sprachmodell

Das Sprachmodell weißt N-Grammen eine Bewertung zu, die in der Texterstellung verwendet wird. Wichtig ist hierbei, dass das Sprachmodell auf einem Sprachkorpus trainiert werden muss. Demzufolge ist die Qualität der Spracherkennung umso besser, je ähnlicher sich die Domäne des Sprachkorpus und die Domäne der Eingabe des Spracherkenners sind.

2.3.4 Hypothesensucheinheit

Die Hypothesensucheinheit verwendet das Akustik- und das Sprachmodell, um aus dem aufbereiteten Audiosignal Hyphothesen abzuleiten.

2.3.5 Ausgabe

Die in dieser Arbeit verwendeten Spracherkenner können eine oder mehrere Hypothesen ausgeben. Bezeichnet werden die besten N Alternativen hier mit N-Best.

2.4 Sprachkorpus

Ein Sprachkorpus (kurz Korpus) besteht aus einer Sammlung von Sprachaufnahmen und ihren zugehörigen Referenztranskriptionen. Er kann zum Training und zur Bewertung von Spracherkennern verwendet werden. Der in dieser Arbeit verwendete Korpus wurde in den Bachelorarbeiten von Günes, Paskaran und Steurer erstellt beziehungsweise erweitert [Gü15, Pas15, Ste16] und zeichnet sich dadurch aus, dass er Aufnahmen von Aufgabenstellungen an einen Roboter enthält (siehe Abschnitt 4.3).

2.5 Sprachunstetigkeiten

Eine Sprachunstetigkeit (engl. Speech Disfluency) ist jede im natürlichen Sprachfluss auftretende Abweichung vom korrekten, geschriebenen Text. Zu den Sprachunstetigkeiten gehören beispielsweise [Shr94]:

- **Verzögerungslaute:** wie „ähm“ oder „mhh“ (bzw. engl. „erm“, „ya“, „uhm“ usw.).
- **Wiederholungen:** Wiederholungen von Wortteilen bis zu Satzteilen zur Korrektur oder besseren Verständlichkeit.

2.6 Graphenrepresentation

Die N-Best Ausgabe eines Spracherkenners lässt sich auch als Graph repräsentieren, der die Weiterverarbeitung vereinfacht.

2.6.1 Wortgitter

Ein Wortgitter ist eine mögliche Graphendarstellung der N-Best mit folgenden Eigenschaften:

- Gerichtet
- Beschriftet
- Zusammenhängend
- Kreisfrei
- Alle Pfade starten an einem Quellknoten.
- Alle Pfade enden in einem Senkenknoten.

In solchen Graphen ist jede Kante mit einem Wort beschriftet; jeder Pfad durch das Wortgitter repräsentiert eine Satzalternative. Der Vorteil an der Darstellung der N-Best als Wortgitter ist, dass eine große Anzahl an Alternativen durch einen relativ kleinen Graphen dargestellt werden kann.

Betrachtet man beispielsweise folgende N-Best:

- Armar bring mir bitte den Saft.
- Armar bring mir den Saft.
- Armar bring mir bitte Saft.

So können diese Sätze in das deutlich (gemessen an der Wortzahl) kleinere Wortgitter abgebildet werden, das in Abbildung 2.2 dargestellt ist.

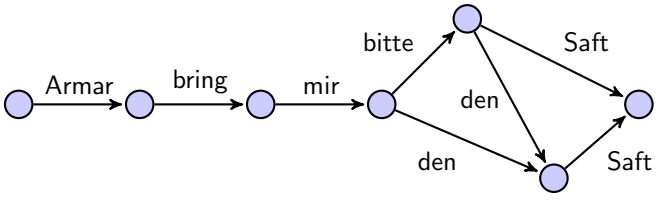

Abbildung 2.2: Wortgitter

2.6.2 Verwirrungsnetzwerk

Wortgitter können weiter zu Verwirrungsnetzwerken (engl. Confusion Network (CN) oder Word Mesh) vereinfacht werden. Ein Verwirrungsnetzwerk hat gegenüber dem Wortgitter noch zwei zusätzliche Eigenschaften: Jeder Pfad muss durch alle Knoten gehen und Auslassungen können durch Kanten ohne Beschriftung repräsentiert werden.

Durch diese Änderungen ist es allerdings nicht mehr möglich, nur die N-Best darzustellen, da Informationen über unmögliche Pfade verloren gehen. Dies ist dann vorteilhaft, wenn nicht nur eine Alternative aus den N-Best ausgewählt, sondern eine bestmögliche Hypothese aus den gegebenen N-Best abgeleitet werden soll.

Das aus dem in Abbildung 2.2 abgeleitete Verwirrungsnetzwerk ist in Abbildung 2.3 dargestellt.

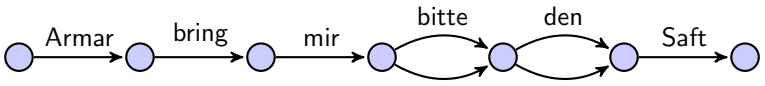

Abbildung 2.3: Verwirrungsnetzwerk

Dieses Beispiel zeigt auch, dass das CN mehr Alternativen enthalten kann als die N-Best, da „Armar bring mir Saft“ zwar nicht in den N-Best enthalten ist, aber ein Pfad im CN darstellt.

2.7 WordNet

WordNet [Pri10] ist eine von der Princeton University veröffentlichte Datenbank, die „englische Nomen, Verben, Adjektive und Adverbien in Synonymgruppen einteilt, die jeweils ein lexikalisiertes Konzept darstellen“ [Mil95]. Diese Synonymgruppen werden in WordNet als „*synset*“ bezeichnet. Zusätzlich zur Kategorisierung liefert WordNet für jedes enthaltene Wort eine Kurzbeschreibung, die „*gloss*“ genannt wird.

Mittlerweile gibt es auch eine Reihe anderer Datenbanken, die derer der Princeton University ähnlich sind und teilweise andere oder zusätzliche Sprachen unterstützen.

2.8 Phonetische Algorithmen

Ein phonetischer Algorithmus weist jedem Wort einen phonetischen Fingerabdruck zu, der für verschiedene Worte gleicher Aussprache gleich sein soll.

Metaphone sowie dessen Weiterentwicklungen (Double Metaphone und Metaphone 3) sind eine Familie phonetischer Algorithmen, die von Lawrence Philips entwickelt wurden, wobei Metaphone 3 von *Anthropomorphic Software LLC* vertrieben wird [Phi00, Ant09].

In dieser Arbeit wird ausschließlich Double Metaphone verwendet (und als Metaphone bezeichnet), das sich durch freie Verfügbarkeit und die Erzeugung von zwei Fingerabdrücken pro Wort auszeichnet. Diese werden erzeugt, um Wörter unterschiedlicher Herkunft zu berücksichtigen. Von gleicher Aussprache wird ausgegangen, wenn einer der beiden Schlüssel eines Wortes mit einem der eines anderen Wortes übereinstimmt. Die Homophone „site“ und „sight“ haben beispielsweise beide die Schlüssel „ST“ und „ST“.

2.9 Wortfehlerrate

Die Wortfehlerrate (engl. word error rate – WER) ist eine Metrik zur Qualität einer Spracherkennerausgabe und gibt an, welcher Anteil der Worte der Ausgabe fehlerhaft erkannt wurde. Sie kann aus den nötigen Änderungen um einen Text in seinen Referenztext zu überführen, sowie der Länge des Referenztextes berechnet werden [JM09]. Demnach gilt:

$$WER = \frac{Einfügungen + Substitutionen + Löschungen}{Wortzahl\ des\ Referenztexts}$$

3. Verwandte Arbeiten

Schon seit einiger Zeit wird an der Verbesserung von Spracherkennern gearbeitet. Dieses Kapitel setzt sich mit einigen dieser Ansätze auseinander.

3.1 Verbesserung der Wortfehlerrate

In diesem Abschnitt werden Arbeiten vorgestellt, die sich mit der Verbesserung der Wortfehlerrate von Spracherkennern auseinandersetzen.

3.1.1 Kombination mehrerer Spracherkenner

Die Kombination mehrerer Spracherkenner zur Verbesserung der Wortfehlerrate wurde von Fiscus et al. in der ihrer Arbeit vorgestellt [Fis97]. Hier wird an jeder Stelle der Ausgabe die häufigste Alternativen, unter allen Hypothesen der verwendeten Spracherkenner, ausgegeben. Zum Einsatz kommt hier hier immer nur eine Hypothese pro Spracherkenner. Zur Bestimmung dieser häufigsten Alternativen kommen Verwirrungsnetzwerke zum Einsatz, die von den Autoren „Worttranspositionsnetzwerk“ [Fis97] gennant werden, wobei mehrere Möglichkeiten zur Auswahl dieser Alternativen vorgestellt werden.

3.1.2 Verwirrungsnetzwerke

In der Arbeit von Mangu et al. wird die Optimierung von Spracherkennerausgaben durch Verwendung von Verwirrungsnetzwerken behandelt [MBS00]. In ihrer Arbeit benutzen die Autoren Verwirrungsnetzwerke, um die normalerweise für eine geringe Satzfehlerrate optimierten Ausgaben der Spracherkenner in einen einzigen Satz mit möglichst geringer Wortfehlerrate zu überführen.

3.1.3 Konsensübersetzung

Die Arbeiten von Bangalore et al. und Matusov et al. [BBR01, MUN06] befassen sich mit dem Finden einer Konsensübersetzung. Dabei sollen die Ausgaben mehrerer maschineller Übersetzer kombiniert werden, um so eine bessere Übersetzung als die Einzelübersetzungen zu erlangen. Auch diese Autoren verwenden hierbei Verwirrungsnetzwerke zur Kombination der Ausgaben (siehe Abschnitt 2.6.2).

3.2 WordNet Ähnlichkeit

Es gibt eine Reihe von Arbeiten, die sich mit der Ähnlichkeit von Worten auseinandersetzen und verschiedene Verwandheitsmaße definieren. Diese Maße lassen sich grob in vier Gruppen einteilen: abstands-, informationsgehalts-, gloss- und tiefenvergleichsbasierte Maße.

Abstandsbasierte Maße

Die in den Arbeiten von Leacock et al. [LC98] und Hirst et al. [HSO98] sind abstandsbasierte Maße und orientieren sich an der Länge des Pfads der die Eingabeworte über verwandte Synsets verbindet.

Informationsgehaltsbasierte Maße

Die Arbeiten von Jiang et al. [JC97], Resnik [Res95] und Lin [Lin98] beschreiben Maße, die auf dem Informationsgehalt des speziellsten gemeinsamen Synets beruhen, wobei die Arbeiten außer derer Resniks auch den Informationsgehalt der Eingabeworte miteinbeziehen.

Glossbasierte Maße

Ein anderer Ansatz wird in den Arbeiten von Banerjee et al. und Patwardhan [BP02, Pat03] vorgestellt. Die in diesen Arbeiten behandelten Maße bewerten die Verwandtschaft der Eingabeworte anhand der Überschneidung in ihren Glossen.

Tiefenvergleichsbasierte Maße

In ihrer Arbeit stellen Wu et al. [WP94] ein Maß vor, das die Verwandtschaft zweier Worte über deren Distanz zum allgemeinsten Synset definiert.

3.3 Erkennung von Unstetigkeiten

Es gibt eine Reihe von Arbeiten zur Erkennung und Entfernung von Unstetigkeiten aus gesprochener Sprache.

Beispielsweise modelliert die Arbeit von Honal et al. die unstetigkeitsbehafteten Rohäußerungen als Kanal mit Rauschen [HS05]. Die unstetigkeitsfreie Version der Äußerung soll dann durch Entfernung des Rauschens erreicht werden. In ihrer Arbeit haben die Autoren gezeigt, dass dieser Ansatz gut funktionieren kann, vor allem wenn das verwendete Vokabular relativ begrenzt ist.

So erreichten sie an dem EVM und MCC Korpus F_1 Maße von 0.44 bzw. 0.32, wobei der MCC Korpus der Größere mit weitreichendem Vokabular ist. Allerdings haben die Autoren auch dargelegt, dass die Qualität der Spracherkennerausgaben Einfluss auf die Erkennung der Unstetigkeiten hat. Die oben genannten F_1 Werte konnten nur mit manuell erstellten Transkriptionen erreicht werden und der Wert lag beim EVM Korpus mit maschinell erstellten Transkriptionen im Durchschnitt bei etwa 0.37.

Ein anderer Ansatz findet sich in der Arbeit von Liu et al. [LSS+06]. In dieser Arbeit werden sowohl Satzgrenzen als auch Unstetigkeiten erkannt. Dazu stellen die Autoren drei verschiedene Ansätze vor: Einen auf *Hidden Markov Models*, einen auf der Maxium-Entropie-Methode und einen auf *Conditional Random Fields* basierenden. Alle diese Ansätze haben gemeinsam, dass die Ergebnisse basierend auf maschinell erstellten Transkriptionen eine etwa 50% bis 100% erhöhte Fehlerrate aufweisen als diese, die die manuell erstellte Referenztranskription verwenden.

3.4 Websuche basierte N-Gramme

Die Arbeiten von [NH05, KL03] beschreiben, wie sich Bewertungen für N-Gramme finden lassen, die nicht in dem Trainingskorpus aufgetreten sind. Die Autoren führen dazu Suchanfragen zu diesen „unbekannten“ N-Gramme durch und bestimmen aus der Anzahl der Suchergebnisse eine Bewertung für dieses N-Gramm. Allerdings beziehen sich diese Arbeiten auf das Training von NLP-Werkzeugen, die auf Text und nicht auf Sprache arbeiten.

4. Analyse

Ziel dieser Arbeit ist die Aufbereitung von Spracherkennerausgaben. Die Aufbereitung umfasst die Reduktion der Wortfehlerrate sowie die Bestimmung von Befehlsgrenzen und Sprachunstetigkeiten. Dieses Kapitel beschreibt, welche dieser Ziele sich derzeit potenziell erreichen lassen.

4.1 Spracherkenner

Spracherkenner liefern die grundlegenden Daten, auf denen diese Arbeit aufgebaut ist. Welche Daten dies umfasst, wie sich Spracherkenner in der Ausgabe von Daten unterscheiden und worin dies begründet ist, wird hier dargelegt. Dabei beschränkt sich die Betrachtung auf Spracherkenner im Allgemeinen, wohingegen Abschnitt 4.4 speziell auf die Eigenschaften der hier verwendeten Spracherkenner eingeht.

Trainings- und Anwendungsdomäne

Den größten Einfluss auf die Spracherkennerausgaben haben das Akustik- und Sprachmodell Abschnitt 2.3, sowie die Domäne der Trainingsdaten und die Domäne der Anwendungsdaten. Domänenspezifische Worte und Satzkonstrukte treten beim Trainieren von Akustik- und Sprachmodell häufiger bzw. überhaupt auf und werden somit als wahrscheinlicher angesehen. Weichen die Domänen der Trainings- und Anwendungsdaten voneinander ab, erzeugt der verwendete Spracherkenner in der Regel Ausgaben mit einer höheren Wortfehlerrate.

Für die hier relevante Domäne gibt es allerdings keinen bestehenden Korpus zum Training eines Spracherkenners, weswegen an diesem Institut einer eigener Sprachkorpus aufgebaut wird, der in Abschnitt 4.3 beschrieben ist. Allerdings ist dieser Korpus mit seiner Gesamtgröße von etwa 24 Minuten derzeit noch deutlich zu klein, um einen Spracherkenner zu trainieren.[1]

Daher kann für die Erstellung dieser Arbeit kein speziell für diese Domäne trainierter Spracherkenner verwendet werden, sondern bestehende Spracherkenner mit vortrainierten Akustik- und Sprachmodellen. Aus deren Ausgaben soll dann eine verbesserte Ausgabe abgeleitet werden.

[1] Zum Vergleich: Die in [YD14] verwendeten Trainingsteile der Sprachkorpora haben eine Gesamtlaufzeit von – je nach Korpus, zwischen 24 bis 2000 Stunden

Ausgabe

Als Ausgabe hat ein Spracherkenner immer mindestens eine Hypothese, was der Inhalt des Gesprochenen war, diese wird hier mit Haupthypothese (HH) bezeichnet. Neben seiner Haupthypothese kann ein Spracherkenner in der Regel noch zusätzliche Informationen ausgeben. Welche Informationen ein Spracherkenner im Speziellen ausgeben kann, hängt dabei vom jeweiligen Spracherkenner ab. Im Folgenden werden einige der möglichen Informationen, die ein Spracherkenner neben der Haupthypothese ausgeben kann, diskutiert. Es handelt sich hier allerdings nicht um eine abschließende, sondern nur eine beispielhafte Liste.

- ***N*-Best:** Wird mehr als eine Hypothese ausgegeben, so werden die Hypothesen neben der Haupthypothese als Nebenhypothese (NH) und die Gesamtheit der Hypothesen als N-Best bezeichnet (siehe Abschnitt 2.3.5). Diese Alternativhypothesen sind eine mögliche Grundlage, um eine andere Ausgabe als die Haupthypothese zu erstellen.
- **Konfidenzen:** Ein numerischer Wert der Worten, Sätzen oder ganzen Hypothesen zugewiesen wird. Er gibt an wie zuversichtlich der Spracherkenner in die Richtigkeit seiner Ausgabe ist. Diese können als Anhaltspunkt zur Erstellung einer neuen Ausgabe verwendet werden.
- **Satzzeichen:** Satzzeichen können, je nach Spracherkenner, gar nicht erkannt werden, nur Satzgrenzen in Form von Punkten umfassen, bis hin zur vollständigen Erkennung aller Satzzeichen. Der Vorteil der Zeichensetzung durch den Spracherkenner anstelle der nachträglichen Zeichensetzung anhand des Transkripts ist, dass der Spracherkenner zusätzliche Anhaltspunkte zur Zeichensetzung verwenden kann. Zu diesen Anhaltspunkten zählen beispielsweise Sprechpausen oder die Betonung. Allerdings hat eine nachträgliche Satzzeichensetzung den Vorteil, eventuell von einer geringeren Wortfehlerrate zu profitieren.
- **Wortalternativen:** Neben den N-Best können auch Wortalternativen für jedes Wort angegeben werden, die dann ein Verwirrungsnetzwerk bilden. Auch diese könnten als mögliche Grundlage zur Erstellung einer neuen Ausgabe verwendet werden.
- **Zeitinformationen:** Informationen über Start- und Endzeiten eines Wortes oder Satzes relativ zum Anfang der Audiodatei. Diese Informationen sind vor allem dann interessant, wenn bei der Auswertung der N-Best nicht eindeutig ersichtlich ist welches Wort einer Hypothese die Alternative zu welchem anderen Wort einer Alternativhypothese darstellt.

Des Weiteren ist auch zu beachten, dass die meisten Informationen, die ein Spracherkenner während der Auswertung einer Sprachaufnahme geniert in der Ausgabe nicht mehr enthalten sind. Zu diesen Informationen gehören beispielsweise Informationen über die erkannten Phoneme aus denen die Hypothesen abgeleitet wurden, sowie über das Akustik- und das Sprachmodell, dadurch ist es nicht möglich weitere Wort- oder Satzalternativen direkt aus den Phonemen abzuleiten, um so eine Anpassung der Ausgaben an die Domäne zu erreichen. Außerdem ist aus den N-Best nicht mehr mit absoluter Sicherheit abzuleiten welche Worte vom Spracherkenner als Alternativhypothesen zueinander betrachtet wurden, es sei den die Wortalternativen sind explizit in der Ausgabe enthalten.

Um aus den N-Best enthaltenen Worten einen optimierten Satz generieren zu können muss also eine neue Informationsanreicherung stattfinden.

4.2 Auswahl der Spracherkenner

Wie im vorherigen Abschnitt beschrieben, stehen für eine Weiterverarbeitung nur eine geringe Menge an Informationen zur Verfügung, insbesondere im Vergleich zu den im Audiosignal enthaltenen Informationen. Umso wichtiger ist es deshalb, dass die Qualität und Quantität dieser Informationen hoch ist. Es muss also eine Auswahl an Spracherkennern getroffen werden, die verwendet werden sollen, wobei im Wesentlichen folgende Kriterien interessant sind:

Wortfehlerrate
Da ein Ziel die Minimierung der Wortfehlerrate ist, ist es sinnvoll, einen Spracherkenner mit ohnehin niedriger Wortfehlerrate zu wählen.

Zuverlässigkeit
Manche Spracherkenner liefern nicht immer eine Ausgabe, diese werden hier als unzuverlässig bezeichnet. Unzuverlässigkeit ist besonders problematisch, da ohne Ausgaben auch keine Weiterverarbeitung möglich ist.

Zusätzliche Informationen
Je mehr zusätzliche Informationen der Spracherkenner bereitstellen kann und je korrekter diese Informationen sind, desto besser lassen sich seine Ausgaben weiterverarbeiten. Interessant sind hier vor allem Informationen über Konfidenzen des Spracherkenners bezüglich ganzer Sätze und einzelner Wörter, Satzzeichen und die direkte Ausgabe von Verwirrungsnetzwerken (siehe Abschnitt 5.3.1.1).

In der Praxis sind Spracherkenner mit besserer Wortfehlerrate allerdings nicht unbedingt die Spracherkenner, die die qualitativ und quantitativ besten Informationen liefern, oder am zuverlässigsten sind. In Tabelle 4.1 ist dies am Beispiel des Google-, des IBM Watson- und des AT&T-Spracherkenners dargelegt[2] [SW16, IBM16, Dev15].

Tabelle 4.1: Spracherkenner Eigenschaften

Spracherkenner	HH WER	Zusätzliche Informationen	Zuverlässigkeit
Google	$\approx 21\%$[3]	Konfidenz der Haupthypothese[4]	Teilweise keine Haupthypothese Teilweise keine Hypothesen
IBM Watson	$\approx 28\%$	Konfidenzen der Haupthypothese Konfidenzen der Wörter der Haupthypothese Wortalternativen Zeitinformationen Schlüsselworterkennung[5]	Liefert immer Hypothesen
AT&T	$\approx 48\%$	Konfidenzen aller Hypothesen Konfidenzen der Wörter der Haupthypothese	Liefert immer Hypothesen

[2] Die Wortfehlerraten wurden mit äquivalenter Methodik zu der in Kapitel 7 angefertigt. Als Datengrundlage wurde der vollständige Korpus aus Abschnitt 4.3 verwendet. Die Diskrepanz der Werte beruht auf sich wandelnden Ausgaben der Spracherkenner im Verlauf der Erstellung dieser Arbeit.
[3] Es handelt sich um die Durschnittswortfehlerrate für Ausgaben mit Hypothesen.
[4] Der Google Spracherkenner hat die Besonderheit, das Nebenhypothesen ohne die Haupthypothese ausgegeben werden können. In dieser Arbeit wird in solch einem Fall die erste Nebenhypothese als Haupthypothese erachtet. Allerdings fehlt dann die Konfidenz für diese Hypothese.
[5] Es können Schlüsselworte angegeben werden, die in der Ausgabe hervorgehoben werden sollen.

Diese Spracherkenner bilden nur eine kleine Auswahl aller Spracherkenner. Unter ihnen ist der Spracherkenner von Google offensichtlich der mit der niedrigsten Wortfehlerrate. Zusätzlich hat Paskaran in seiner Arbeit gezeigt, das der Google Spracherkenner für die Domäne der Roboterprogrammierung auch im Vergleich zu einer größeren Menge an Spracherkennern die geringste Wortfehlerrate aufweist [Pas15]. Bezüglich der zusätzlichen Informationen liefert der IBM Watson Spracherkenner die Meisten, allerdings fehlen die Konfidenzen der Nebenhypothesen, die wiederum der AT&T-Spracherkenner ausgeben kann.

Bei dieser Sachlage ist es naheliegend, nicht nur einen Spracherkenner als Datenquelle zu verwenden, sondern mehrere. Die einfachste Möglichkeit wäre es beispielsweise, den Google Spracherkenner zu verwenden und bei Bedarf auf den IBM Watson Spracherkenner zurückzugreifen. Wie diese Kombination im Detail gestalt werden kann, wird in Abschnitt 5.3.1.1 beschrieben.

4.3 Korpus

Für die Domäne der Robotterprogrammierung wurde von Günes ein Korpus erstellt, der in späteren Arbeiten von Paskaran und Steurer erweitert wurde [Gü15, Pas15, Ste16]. Dieser Korpus umfasst 151 englische Aufnahmen sowie deren Transkriptionen. Die Gesamtlaufzeit der Aufnahmen beläuft sich auf etwa 24 Minuten.

Die Aufnahmen sind von Probanden, die Aufgaben an einen Roboter stellen (in diesem Fall Armar III [ARA+06]). Hierbei sind die Szenarien vorgegeben, die Wortwahl wurde den Probanden überlassen. Insgesamt gibt es fünf Szenarien, die von 61 verschiedenen Probanden gesprochen wurden, wobei jeder Proband maximal drei dieser Szenarien besprochen hat.

Eine Übersicht über diese Szenarien ist in Tabelle 4.2 dargelegt. Hier sind für jedes Szenario eine Kurzbeschreibung, die Anzahl der enthaltenen Aufnahmen (#), die durchschnittliche Sprachkenntnis, Qualität und Quelle der Aufnahmen gelistet. Die durchschnittliche Sprachkenntnis basiert auf der Selbsteinschätzung der Probanden, die ihre Sprachkenntnis auf einer Skala zwischen 1 (elementar) und 5 (Muttersprachler) bewerten konnten. Schlechtere Sprachkenntnisse gehen in diesem Korpus oft mit Aussprache- und Grammatikfehlern einher und führen damit zu höheren Wortfehlerraten. Die Spalte „Qualität“ gibt die Rahmenbedingungen der Aufnahme an. Mit „+“ gekennzeichnete Zeilen wurden in einer stillen Umgebung mit einem professionellen Mikrophon[6] erstellt, wohingegen mit „-“ gekennzeichnete Zeilen mit unbestimmtem Mikrophon und Umgebung aufgezeichnet wurden. Auch schlechte Aufnahmequalität geht oft einher mit Erkennungsfehlern (siehe [Pas15]).

4.3.1 Spracheigenschaften

Die begrenzte Anzahl an Szenarien und die Tatsache, dass die Aufnahmen ausschließlich von Nicht-Muttersprachlern gesprochen wurden, hat zur Folge, dass der Korpus eine Reihe von Eigenschaften hat, die die meisten der Aufnahmen teilen. Beispielsweise sind die meisten Sätze im imperativ und relativ kurz[7]. Demgegenüber stehen große Unterschiede zwischen den Szenarien, bei der Länge der Aufnahmen. Ein Überblick über die verwendete Wortzahl, aufgeschlüsselt nach Szenario, ist in Tabelle 4.3 abgebildet. Wie hier deutlich zu erkennen ist sind sich, längenmäßig, Szenario 1 bis 3 und Szenario 4 bis 5 sehr ähnlich.

An dieser Stelle sei nochmals darauf hingewiesen, dass die Wortwahl den Probanden überlassen wurde, und somit die Tatsache, dass nahezu alle Sätze relativ kurz und im Imperativ sind, Zufall ist.

[6] RØDE NT1-A
[7] weniger als 10 Worte

Tabelle 4.2: Szenarien

Szenario	Kurzbeschreibung der Aufgabenstellung	#	Sprachk. ∅	Qualität	Quelle
1.	Armar soll Popcorn holen.	22	2.77	+	Günes
		14	2.14	-	Paskaran
2.	Armar soll einen Becher in die Spülmaschine stellen.	26	2.77	+	Günes
		14	2.14	-	Paskaran
3.	Armar soll Saft bringen.	23	2.77	+	Günes
		14	2.14	-	Paskaran
4.	Armar soll die Spülmaschine aus- und einräumen.	19	2.22	+	Steurer
5.	Armar soll einen Drink mixen.	19	2.22	+	Steurer

Tabelle 4.3: Aufnahmelänge in Worten pro Szenario

Szenario	Wörter		
	min.	max.	∅
1.	6	32	13.42
2.	8	62	21.36
3.	5	43	18.59
4.	17	65	29.16
5.	20	51	32.26

4.3.2 Unstetigkeiten

Der Korpus enthält nur eine sehr geringe absolute Anzahl an Sprachunstetigkeiten, wobei diese fast ausschließlich in der Form von Verzögerungslauten auftreten. Die Verzögerungslaute, die im Korpus enthalten sind, decken auch nur einen Teil der in Abschnitt 2.5 beschriebenen Verzögerungslaute ab. Des Weiteren werden meist Verzögerungslaute verwendet, die normalerweise nicht im englischen Sprachgebrauch vorkommen, da die Sprecher oft die Verzögerungslaute ihrer Muttersprache verwenden (zum Beispiel das deutsche „äh“).

Neben den Verzögerungslauten sind im Korpus Wiederholungen und durch Zögern bedingte Aussprachefehler enthalten (beispielsweise „upe- per“).

4.4 Ausgabequalität

Für die in den Abschnitten 4.2 beziehungsweise 4.3 beschriebenen Spracherkenner und den Korpus ist für den Entwurf einer Problemlösung vor allem relevant, welche Fehler unter Verwendung der Spracherkenner auf diesem Korpus auftreten und in welcher Häufigkeit. Diese Analyse wird im Folgenden durchgeführt.

4.4.1 Zuverlässigkeit

Bei dem Zuverlässigkeitskriterium ist zwischen dem IBM Watson und Googles Spracherkenner zu differenzieren: Während Google in manchen Fällen überhaupt keine Ergebnisse liefert, fehlen bei IBM Watson meist nur einzelne Worte.

Woran dieses Fehlen von Worten und Sätzen genau liegt, kann hier allerdings nicht geklärt werden, da dazu ein genauer Einblick in die Funktionsweise des jeweiligen Spracherkenners nötig wäre. Auffällig ist, dass das komplette Fehlen von Ergebnissen bei Google meist mit mangelhafter Sprach- oder Aufnahmequalität einhergeht. Schlechte Sprachqualität in

diesem Sinne sind viele oder sehr lange Pausen oder eine Anhäufung von Sprachunstetigkeiten. Mindere Aufnahmequalität beschreibt das Vorhandensein von technisch bedingten Problemen der Aufnahme, wie beispielsweise Hintergrundrauschen oder extrem geringe Lautstärke.

Auch bei IBM Watson gehen fehlende Worte oft mit schlechter Aufnahmequalität einher. Eine eingeschränkte Sprachqualität scheint hier aber nicht so relevant zu sein, sie führt lediglich zu falsch erkannten Worten, nicht zu garnicht erkannten Worten.

4.4.2 Verzögerungslaute

Die beiden Spracherkenner gehen mit Verzögerungslauten unterschiedlich um: Bei IBM Watson werden sie durch „%HESITATION“ ersetzt, wohingegen der Google Spracherkenner sie komplett entfernt.

Demzufolge lassen sich nur schlecht Aussagen über die Qualität der Erkennung von Unstetigkeiten durch Google's Spracherkenner treffen, da nicht erkannte Verzögerungslaute zu falsch erkannten benachbarten Worten führen. (Nachträglich lässt sich nicht erkennen, ob ein Verzögerungslaut als Teil eines falsch erkannten Wortes aufgefasst wurde, oder der Verzögerungslaut richtig erkannt wurde und der Wortfehler eine andere Ursache hat.)

IBM Watson erkennt die Verzögerungslaute in der Regel richtig und markiert diese, selbst wenn eigentlich ein Verzögerungslaut einer anderen Sprache verwendet wurde, der aber dem Englischen in der Aussprache ähnlich ist („äh“ „eh“). Wie bei Google ist eine Rekonstruktion des Verzögerungslautes an sich nicht möglich.

Da die Spracherkenner für alle Sprachaufnahmen des Korpus keinen einzigen Verzögerungslaut in Wortform ausgeben, ist die Entfernung von Verzögerungslauten aus den Spracherkennerausgaben weder möglich noch nötig. Dies schließt allerdings nicht aus, dass diese Verzögerungslaute bei einer Vergrößerung des Korpus nicht irgendwann auftauchen. Dennoch kann dieser Teil der Aufbereitung hier nicht betrachtet werden.

4.4.3 Befehlsgrenzen und Satzzeichen

In dieser Arbeit wird zwischen Befehlen und Sätzen unterschieden. Ein Befehl ist jede Teilmenge eines Satzes, die semantisch vollständig genug ist, um sie zu verstehen. Beispielsweise hat der Satz „Bring mir den Saft und schließe die Tür.“ zwei Befehle („Bring mir den Saft“ und „Schließe die Tür“).

Die verwendeten Spracherkenner können Satzgrenzen ausgeben, Befehlsgrenzen nicht. Allerdings unterscheiden sie sich erheblich in der Frequenz der Satzgrenzenausgabe. Während der Google Spracherkenner nur sehr selten Satzgrenzen ausgibt[8], gibt der IBM Watson Spracherkenner regelmäßig Satzgrenzen aus. Da die Aufnahmen, wie in Abschnitt 4.3 beschrieben, allerdings recht kurz sind, wird auch von diesem meist nur eine Satzgrenze ausgegeben.

Es wäre also zumindest möglich, die Satzgrenzen des IBM Watson Spracherkenners zu verwenden, um auf dieser Basis Befehlsgrenzen zu bestimmen. Da die meisten Ausgaben allerdings nur sehr wenige Befehlsgrenzen enthalten wird hier von dem Entwurf einer neuen Befehlsgrenzenbestimmung abgesehen und es werden stattdessen bestehende Werkzeuge verwendet.

[8] Der Google Spracherkenner gibt auf dem hier verwendeten Korpus genau eine Satzgrenze aus

4.4.4 Wortfehleranalyse

Bei den Wortfehlern gibt es zumindest subjektiv große Unterschiede zwischen Worten, die in dem Trainingskorpus der Spracherkenner zu finden sind (im Weiteren als allgemeine Worte bezeichnet) und Worten, die dort nicht zu finden waren (im Weiteren als spezielle Worte bezeichnet). In diesem Fall sind die speziellen Worte beschränkt auf den Namen des Roboters ARMAR. Grundsätzlich kann man hier Folgendes feststellen:

Allgemeine Worte

In der Regel werden allgemeine Wörter zumindest in einer Alternative der N-Bestrichtig erkannt, falsch erkannte Worte beschränken sich subjektiv sehr ähnlich klingende Alternativen wie zum Beispiel „french“ für „fridge“ oder „cap“ oder „cop“ für „cup“.

Spezielle Worte

Spezielle Worte werden nie richtig erkannt, außerdem weichen die falschen Alternativen meist deutlich stärker von der Referenztranskription ab, als es bei allgemeinen Wörtern der Fall ist. Anstatt „Armar“ geben die Spracherkenner beispielsweise „Amour“, „I'mma“, „are mom“ aus.

Diese Diskrepanz zwischen den Worttypen hat zur Folge, dass diese unterschiedlich behandelt werden müssen. Während es für allgemeine Worte in der Regel ausreicht, die richtige Version dieses Wortes aus den N-Bestzu bestimmen, müssen spezielle Worte zu den Alternativen hinzugefügt werden. Wie dieses Hinzufügen von Worten funktionieren kann wird, in Abschnitt 5.3.1.4 dargelegt.

5. Entwurf

Ziel dieser Arbeit ist die Aufbereitung von Spracherkennerausgaben. Im vorherigen Kapitel wurde dieses Ziel auf die Verbesserung der Wortfehlerrate konkretisiert, da der zur Verfügung stehende Korpus nicht umfangreich genug ist, die anderen Ziele zu verfolgen.

Wie das Ziel der Verbesserung der Wortfehlerrate erreicht werden kann, wird in diesem Kapitel diskutiert.

5.1 Gesamtkonzept

Für die Erreichung des Ziels der Wortfehlerratenminierung soll in dieser Arbeit ein System entworfen und implementiert werden. Da allerdings nicht alle Teile der Aufbereitung erreicht werden können, sollte dieses so konzipiert sein, dass sich das System so erweitern lässt, dass die anderen Teilbereiche der Aufbereitung erreicht werden können. Die drei essentiellen Komponenten sind:

1. Spracherkennung
2. Aufbereitung und Informationsanreicherung
3. Erstellung einer neuen Ausgabenhypothese

Was unter diesen Punkten im Einzelnen zu verstehen ist, wird im Folgenden beschrieben.

Spracherkennung

Im ersten Schritt der Spracherkennung muss eine Anbindung für die Spracherkenner erstellt werden, die die Ausgaben auf ein einheitliches Format bringt. Auch hier ist ein erweiterbares Konzept vorteilhaft, um neue Spracherkenner einbinden zu können.

Aufbereitung und Informationsanreicherung

Bei der Aufbereitung und Informationsanreicherung sollen Ausgabefehler entfernt und Informationen aus anderen Quellen herangezogen werden. Diese sollen es später ermöglichen, eine neue Ausgabenhypothese zu erzeugen.

Erstellung einer neuen Ausgabenhypothese

Im finalen Schritt muss anhand der vorhanden Informationen eine neue Ausgabenhypothese erstellt werden.

5.2 Spracherkennung

Bevor die Ausgaben der Spracherkenner überhaupt verarbeitet werden können, müssen diese in einem ersten Schritt beschafft werden. Dazu wird eine Schnittstelle zwischen den einzelnen Spracherkennern und der Aufbereitung und Informationsanreicherung erstellt. Da Spracherkenner weiterhin aktuelles Forschungsthema sind und es immer wieder neue Entwicklungen in diesem Bereich gibt, ist es vorteilhaft, hier ein modulares System zu entwerfen, das die einfache Einbindung neuer Spracherkenner erlaubt.

Dieses System muss in erster Linie zwei Kriterien erfüllen: Umwandlung der Quellaudiodatei in ein vom jeweiligen Spracherkenner akzeptiertes Format und die Standardisierung der Ausgaben der Spracherkenner. Die Umwandlung soll mit möglichst geringem Informationsverlust erfolgen. Die Standardisierung der Ausgaben umfasst in erster Line die Umwandlung der Spracherkenner-spezifischen Ausgabe in ein Spracherkenner-agnostisches Format.

5.3 Aufbereitung und Informationsanreicherung

Wie in Abschnitt 4.1 beschrieben, geben die Spracherkenner sowohl die N-Best, als auch eine Reihe von Zusatzinformationen aus. Mit der Gefahr, dass die Informationsmenge undurchsichtig und somit schlecht zu verarbeiten wird. Beispielsweise müssen zur Verarbeitung der 5-Best von 2 Spracherkennern in jedem Schritt 10 Ausgabealternativen betrachtet werden, inklusive aller etwaigen Zusatzinformationen wie Konfidenzen.

Aus diesem Grund muss der erste Schritt der Aufbereitung eine Vereinfachung der Quelldaten sein.

5.3.1 Vereinfachung der Spracherkennerausgaben

Die Ausgaben der Spracherkenner sind nicht nur umfangreich, sondern im Bezug auf die enthaltenen Wörter auch hochgradig redundant. Diese Redundanz ist zumindest immer dann gegeben, wenn für vom Spracherkenner für alle Hypothesen eine möglichst geringe Wortfehlerrate erreicht werden soll.[1] Dies liegt daran, dass die Ausgaben Resultat desselben Akustik- und Sprachmodells sind und sich im Regelfall extrem ähneln (Ein starkes Abweichen der N-Best eines Spracherkenners untereinander geht in der Regel mit sehr geringen Wort- und Satzkonfidenzen einher). Da redundante Informationen die Verarbeitung erschweren aber keinen Mehrwert beitragen, ist es sinnvoll, die in den N-Best enthaltenen Informationen auf die wichtigsten Informationen zu reduzieren.

5.3.1.1 Graphendarstellung

Eine Möglichkeit, dies zu bewerkstelligen, ist die Überführung der N-Best in einen Graphen, wobei hiermit gleich sämtliche N-Best aller Spracherkenner kombiniert und damit die Probleme „Informationsredundanz“ und „mehrere Spracherkenner“ in einem Schritt gelöst werden können. Zur Grapherstellung gibt es zwei Ansätze, die im Folgenden erläutert werden.

[1] Seien A und B zwei Hypothesen mit gleicher Länge. Außerdem sei die Wortfehlerrate von A bekannt und gleich w_a. Dann muss B, auf die Worte bezogen, mindestens in einen Anteil von $1 - w_a * 2$ zu A identisch sein, um die gleiche Wortfehlerrate zu erreichen. Für den Extremfall, dass B sich komplett von A unterscheidet, ist die Wortfehlerrate von B mindestens $1 - w_a$.

Wortgitter

Wortgitter ermöglichen die N-Best auf einen Graphen abzubilden, der die Redundanz der N-Best reduziert und somit die Weiterverarbeitung erleichtert (siehe Abschnitt 2.6.1). Dabei ist ein Vorteil von Wortgittern, dass die Konfidenzen des Spracherkenners bei der Erstellung des Wortgitters mit einbezogen werden können. So kann aus den N-Best, die für eine möglichst geringe Satzfehlerrate optimiert sind, ein Wortgitter erzeugt werden, das für eine möglichst geringe Wortfehlerrate optimiert ist. Dies kann erreicht werden, indem die Satzfehlerrate als a-priori Wahrscheinlichkeit angesehen wird (siehe [MBS00]).

Betrachtet man beispielsweise die in Tabelle 5.1 aufgelisteten N-Best, so kann daraus das in Abbildung 5.1 dargestellte Wortgitter abgeleitet werden.

Tabelle 5.1: Beispielausgabe

Satz	Satzkonfidenz
Armar bring mir bitte den Saft	0.9
Armar bring mir den Saft	0.8
Armar bring mir bitte Saft	0.8
Armar bring mir die Säfte	0.7

Der Wert in Klammern ist hierbei die a-posteriori Übergangswahrscheinlichkeit, also die Wahrscheinlichkeit, dass ein Wort im Ausgabesatz enthalten ist. Wie sich erkennen lässt, ist die Komplexität der Ausgabe stark gesunken: Die in den N-Best enthaltenen redundanten Informationen wurden entfernt und die Satzkonfidenzen in Übergangswahrscheinlichkeiten umgewandelt. Somit kann eine pro-Wort-Analyse durchgeführt werden. Anzumerken ist, dass dieses Beispiel mit seinen darin enthaltenen Konfidenzen synthetisch ist, um es anschaulich zu halten. Die generelle Aussage ist nichtsdestoweniger auch auf reale Spracherkennerausgaben anzuwenden, sowie es auch in der Implementierung geschieht in Kapitel 6.

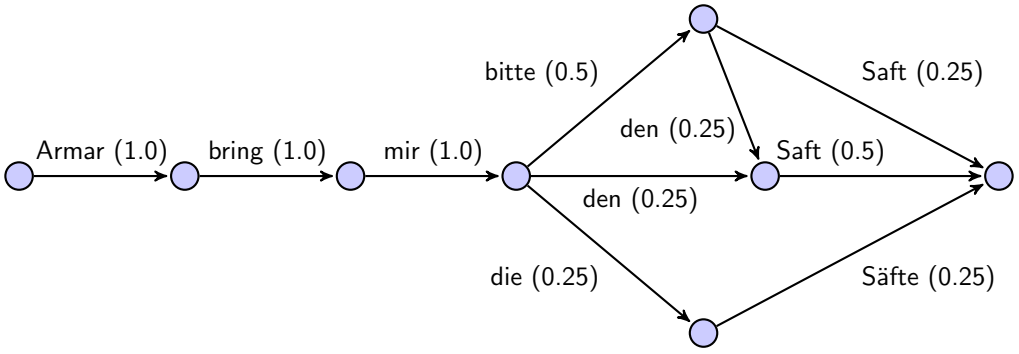

Abbildung 5.1: Wortgitter

Problematisch an Wortgittern ist allerdings die Tatsache, dass Wortalternativen nicht eindeutig aus dem Graphen hervorgehen, und weiterhin Redundanz in Form von duplizierten Worten in dem Wortgitter enthalten ist. Ein weiteres Problem ist, dass das Wortgitter die Ausgaben des Spracherkenners unverändert abbildet, so dass sich aus diesem keine neuen Sätze ableiten lassen. Dies bedeutet, dass es vorkommen kann, dass das Wortgitter zwar theoretisch die nötigen Worte beinhaltet um einen vollständig korrekten Satz zu rekonstruieren, aber keiner der Pfade durch das Wortgitter diese Kombination zulässt.

Verwirrungsnetzwerke

Verwirrungsnetzwerke sind ein Wortgitter, an das zusätzliche Bedingungen gestellt werden (siehe Abschnitt 2.6.2). Dabei hat das Verwirrungsnetzwerk mehrere Vorteile gegenüber

dem Wortgitter, bezogen auf die Verwendung in dieser Arbeit.

Der Hauptvorteil ist, dass die Worte hier in „Alternativengruppen“ aufteilt sind. In dem in Abbildung 5.2 dargestellten Verwirrungsnetzwerk bilden beispielsweise „den“, „die“ und „—“ eine Alternativengruppe, wobei „—“ stellvertretend für eine Auslassung steht.

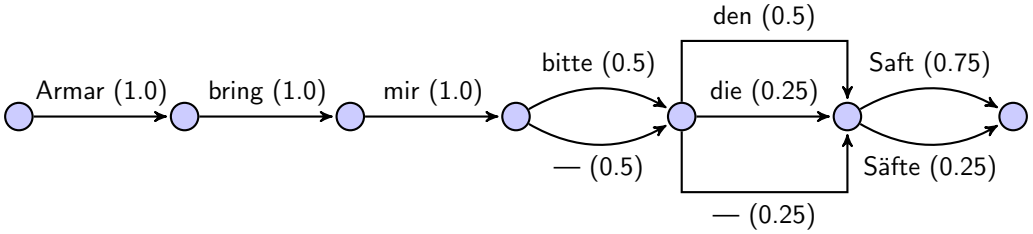

Abbildung 5.2: Verwirrungsnetzwerk

Dies ermöglicht es, jede Alternativengruppe getrennt voneinander zu betrachten, um dann aus jeder Alternativengruppe eine Alternative auszuwählen. Alle ausgewählten Alternativen zusammen bilden dann die Ausgabenhypothese.

Die Alternativengruppen haben auch den Vorteil, dass sie nicht an die N-Best gebunden sind und somit auch Hypothesen erstellt werden können, die nicht in den N-Best enthalten sind. Dies ist vor allem günstig im Hinblick auf die beschränkte Anzahl an N-Best, die ein Spracherkenner ausgeben kann. Neben den Wortkombinationen, die ein Spracherkenner in seinen N-Best ausgibt, sind noch weitere Kombinationen der selben Worte denkbar, die aber auf Grund der beschränkten Anzahl an N-Best nicht ausgegeben werden können. Da auch in einem Verwirrungsnetzwerk jeder Pfad eine Hypothese repräsentiert, hat beispielsweise das in Abbildung 5.2 dargestellte Netzwerk 12 mögliche Ausgabenhypothesen verglichen mit den 4-Best, aus denen dieses Verwirrungsnetzwerk erstellt wurde.

Ein weiterer Vorteil ist, dass jedes Wort an einer bestimmten Stelle nur ein einziges Mal vorkommt. Damit können Wahrscheinlichkeiten pro Wort und nicht mehr nur pro Übergang berechnet werden. Beispielsweise gibt es in dem Wortgitter das Wort „den“ 2 mal und das Wort „die“ 1 mal mit jeweils einer Übergangswahrscheinlichkeit von 25%. In dem Verwirrungsnetzwerk wird den Worten allerdings eine Übergangswahrscheinlichkeit von 50% bzw. 25% zugeordnet, was bezogen auf die N-Best der realistischere Wert ist.

Zusammenfassend ist somit zu sagen, dass Verwirrungsnetzwerke den Nachteil haben, dass sich die N-Best nicht mehr aus ihnen rekonstruieren lassen, aber den Vorteil, dass sie leichter weiterzuverarbeiten sind und trotzdem die Wortredundanz vermindert wird. Aus diesem Grund werden in dieser Arbeit Verwirrungsnetzwerke zur Vereinfachung der N-Best verwendet.

5.3.1.2 Wortverschmelzung

Da verschiedene Spracherkenner unterschiedliche Sprachmodelle verwenden, kann es vorkommen, dass sie zusammengesetzte Wörter unterschiedlich ausgeben.

So kann beispielsweise ein Spracherkenner „dishwasher“ und der andere „dish washer“ ausgeben, was zur Folge hat, dass das Verwirrungsnetzwerk zwei Alternativengruppen für dieses eine Wort hat, so wie es in dem Verwirrungsnetzwerk-Ausschnitt in Abbildung 2.1 dargestellt ist.

Allerdings kann nicht jede dieser Konstellationen verschmolzen werden, denn es muss sichergestellt werden, dass sich die beiden Schreibweisen nicht grundlegend in ihrer Bedeutung unterscheiden. WordNet kann herangezogen werden, um diese Fälle auszuschließen, indem bestimmt wird, ob sich beide Schreibweisen ein Synset teilen.

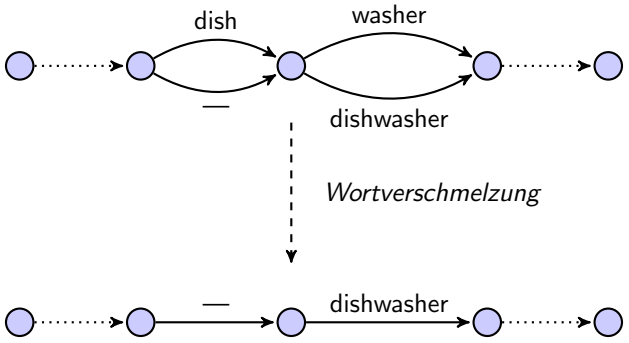

Abbildung 5.3: Wortverschmelzung

5.3.1.3 Lemmatisierung

Die unterschiedlichen Sprachmodelle der verschiedenen Spracherkenner können auch dazu führen, dass nach Erstellung des Verwirrungsnetzwerkes mehrere Flexionen des gleichen Wortes in einer Alternativengruppe enthalten sind. Dies ist vor allem im Hinblick auf die Übergangswahrscheinlichkeiten aus dem Verwirrungsnetzwerk problematisch, was in einem späteren Teil dieser Arbeit relevant wird (siehe Abschnitt 5.3.3).

Diese Problematik wird anhand des Beispiels in Abbildung 5.4 offensichtlich. Hier hat das Wort „Flaggen“ eine höhere Übergangswahrscheinlichkeit als beide Flexionen des Wortes „Flaschen“. Allerdings ist die Gesamtübergangswahrscheinlichkeit der Flaschen-Flexionen höher als die der Flaggen-Flexionen. Demzufolge würde eine Bewertung der Alternativen anhand der Übergangswahrscheinlichkeiten zugunsten von „Flaggen“ verzerrt werden.

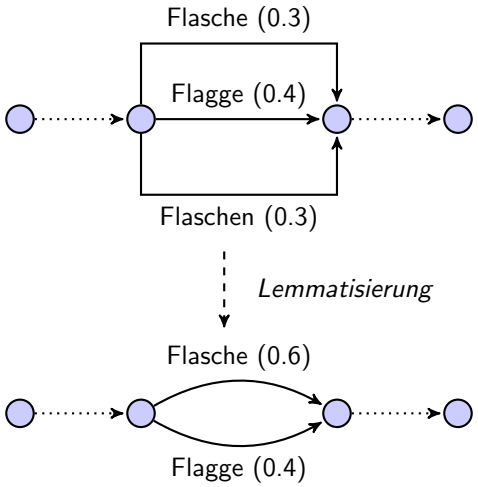

Abbildung 5.4: Lemmatisierung

Zur Behebung dieses Problems können die Lemmata aller Alternativen einer Alternativengruppe bestimmt werden. Daraufhin können alle Alternativen mit identischen Lemmata zu einer Alternative verschmolzen werden.

Welche Alternativen dabei gelöscht werden und welche Alternativen bestehen bleiben, ist dabei gesondert zu betrachten. Hierfür gibt es im Wesentlichen zwei Möglichkeiten: die Vereinigung aller Alternativen mit identischen Lemmata auf die Alternativen mit der höchsten Übergangswahrscheinlichkeit oder auf die Alternative, die das entsprechende Lemmata enthält. In dieser Arbeit wird die zweite dieser Möglichkeiten verwendet, sofern die Alternativengruppe das entsprechende Lemma enthält. Ist dem nicht so, wird die erste Möglichkeit angewandt.

5.3.1.4 Alternativenausweitung

Um das Problem der speziellen Worte aus Abschnitt 4.4.4 zu beheben, müssen diese zum Verwirrungsnetzwerk hinzugefügt werden. In den selteneren Fällen, in denen ein allgemeines Wort in keiner der möglichen Alternativen richtig erkannt wird, ist dieser Schritt natürlich auch sinnvoll. Ein möglicher Ansatz hierfür ist es, eine für den Anwendungsfall zugeschnittene Ontologie aufzustellen und daraus Wörter auszuwählen, die einzelnen Alternativen hinzugefügt werden sollen. Für alle Szenarien aus Abschnitt 4.3 würde beispielsweise eine Ontologie, die die Objekte der Küche sowie den Roboter Armar und seine Handlungsmöglichkeiten beinhaltet, diese Anforderung erfüllen.

Da nicht jedes Wort der Ontologie zu jeder Alternative hinzugefügt werden kann, sondern nur eine Auswahl, müssen infrage kommende Elemente der Ontologie bestimmt werden. Dies sind im wesentlichen ähnlich klingende Elemente zu einem Wort aus dem Verwirrungsnetzwerk. Eine Möglichkeit, diese infrage kommenden Elemente zu bestimmen, wäre einen Abgleich anhand der Schreibweise durchzuführen und Elemente mit ähnlicher Schreibweise aus der Ontologie dem Verwirrungsnetzwerk hinzuzufügen. Dieser Ansatz funktioniert allerdings nur in sehr beschränktem Umfang, da viele Wörter mit ähnlicher Schreibweise stark in der Aussprache voneinander abweichen und sich Wörter mit sehr ähnlicher Aussprache teilweise stark in der Schreibweise unterscheiden. Vor allem bei Wörtern unterschiedlicher Herkunft ist dies oft der Fall.

Deutlich besserer wäre es, diese Entscheidung anhand der Phoneme zu treffen, die zur Auswahl eines Wortes durch den Spracherkenner geführt haben. Da die Informationen über die Phoneme allerdings nicht in den Spracherkennerausgaben enthalten sind, kann sich nicht an diesen orientiert werden. Demzufolge muss eine andere Methode verwendet werden, um ähnlich klingende Worte zu finden. Dazu kann ein phonetischer Algorithmus verwendet werden (siehe Abschnitt 2.8), der jedem Wort einen Schlüssel zuweist, der an die Aussprache eines Wortes abstrahiert und nicht dessen Schreibweise. Ob dann ein genauer Vergleich der Schlüssel verwendet, oder ähnliche aber nicht gleiche Schlüssel zugelassen werden sollen, muss dann den Umständen entsprechend entschieden werden. Für diese Arbeit ist ein Ähnlichkeitsmaß anstatt eines exakten Vergleiches angebrachter, da auch spezielle Worte mit der Alternativenerweiterung erfasst werden sollen. Hierfür kann kein exakter Vergleich verwendet werden, da die Spracherkenner bei Worten, die nicht in ihrem Sprachmodell enthalten sind, Worte ausgeben, die sich in der Aussprache so stark von der Referenztranskription unterscheiden, dass ein genauer Vergleich nicht die erwünschten Resultate erzielen würde (siehe Abschnitt 4.4.4). Je nach Implementierung könnte eine Alternativenerweiterung dann beispielsweise das in Abbildung 5.5 dargestellte Ergebnis liefern.

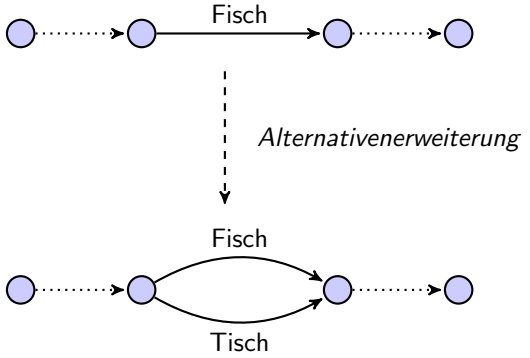

Abbildung 5.5: Alternativenerweiterung

5.3.2 Informationsanreicherung

Im Informationsanreicherungsschritt sollen externe Datenquellen verwendet werden, um neue Informationen zur Neubewertung des Verwirrungsnetzwerkes zu erhalten.

Da sich allerdings keine Informationen über das Sprachmodell aus den Spracherkennerausgaben ableiten lassen und das Sprachmodell an sich sowieso domänenfremd[2] ist, muss zur Auswahl der Alternativen des Verwirrungsnetzwerkes eine andere Bewertungsmethode der N-Gramme gefunden werden.

Dazu sollen sogenannte „Bewertungsmodule“ verwendet werden, die den in dem Verwirrungsnetzwerk enthaltenen Alternativen numerische Werte zuzuweisen.

5.3.3 Bewertungsmodule

Die Bewertungsmodule sollen eine Entscheidungsgrundlage zwischen den Alternativen einer Alternativengruppe liefern. Dafür soll jedes Bewertungsmodul den Wörtern im Verwirrungsnetzwerk einen numerischen Wert zuweisen. Dieser Wert wird hier als Bewertung der Bewertungsmodule bezeichnet. Welche Worte dabei genau mit einer Bewertung durch ein bestimmtes Bewertungsmodul versehen werden, kann von dem Bewertungsmodul abhängen.

Verwirrungsnetzwerk
: Die Übergangswahrscheinlichkeiten aus dem Verwirrungsnetzwerk können direkt als Bewertung dieses Moduls verwendet werden.

Domäne
: Ein Roboter bewegt sich in der Regel in einer festgelegten Domäne. Die in dieser Domäne enthaltenen Objekte und Akteure sowie deren mögliche Aktionen, lassen sich in einer Ontologie abbilden. Anhand einer solchen Ontologie können dann die Alternativen in einem Verwirrungsnetzwerk bewertet werden.

WordNet-Abstand
: Zwischen Synsets können, wie in Abschnitt 3.2 beschrieben, Abstandsmaße bestimmt werden. Zwar ist zu diesem Zeitpunkt der Auswertung noch nicht bekannt, welches Synset genau durch ein Wort dargestellt wird, doch lässt sich der Abstand zwischen allen Synsets zweier Wörter bestimmen. Die minimale Distanz kann dann als 2-Gram herangezogen werden.

FrameNet
: FrameNet ist eine vom Internationalen Institut für Computerwissenschaft (orig. International Computer Science Institute – ICSI) herausgegebene Datenbank, die Sprache systematisch abbilden soll [Fra15]. FrameNet besteht im Wesentlichen aus Frames, die eine Handlungssituation darstellen wie beispielsweise „das Kontrollieren von Etwas “ und Lexikalischen-Einheiten. Jedes Frame enthält dabei Informationen über welche Komponenten dieses Frame enthalten kann oder muss sowie deren Beziehung. Beispielsweise kann das „Kontrollieren-Frame“ nur entweder eine kontrollierende Einheit oder eine kontrollierende Situation beinhalten – nicht beides. Die Lexikalischen-Einheiten hingegen geben Aufschluss darüber, welche Komponenten und welche Frames von einem bestimmten Wort besetzt werden können.

  Durch Abgleich des Verwirrungsnetzwerks mit FrameNet können dann Pfade positiv bewertet werden, die zu einem der Frames in FrameNet passen. Der Ausschluss von Pfaden, die nicht zu einem Frame passen, ist allerdings nicht möglich, da FrameNet nur einen Bruchteil aller möglichen Frames enthält.

[2] Für die hier verwendeten Spracherkenner

Google
Ein weiters Konzept wäre es, aus der Anzahl der Suchergebnisse für eine Wortgruppe, einen N-Gramm Wert abzuleiten. Hier konnte beispielsweise die Google Suchmaschine [Goo16a] verwendet werden. Dazu könnte, ähnlich wie in den in Abschnitt 3.4 beschriebenen Arbeiten die Anzahl der Suchergebnisse für eine Wortgruppe in Relation zur Anzahl der Suchergebnisse der darin enthaltenen Einzelworte gestellt werden. Der Unterschied zu den genannten Arbeiten besteht darin, dass hier nicht das Sprachmodell des Spracherkenners direkt angepasst wird, sondern seine Ausgaben neu bewertet werden, außerdem handelt es sich hier um gesprochene und nicht geschriebene Sprache.

Google Ngram [MSA+11]
Google Ngram ist eine N-Gramm Datenbank, die von Google im Rahmen der Bemühungen zur Digitalisierung von Büchern erstellt wurde. Diese Datenbank enthält Daten über die Häufigkeit von $1-5$-Grammen in diesen Büchern. Ein Bewertungsmodul auf der Basis von Google Ngram könnte ähnlich zu dem Google Bewertungsmodul aussehen, mit dem einzigen Unterschied, dass nicht die Suchergebnisse, sondern die Häufigkeit eines N-Gramms verwendet werden.

5.3.4 Vorverarbeitung

Vor der eigentlichen Anwendung der Bewertungsmodule ist es eventuell sinnvoll, einen Vorverarbeitungsschritt einzubauen, der die Zahl der zu bewertenden Alternativen einschränkt. Dies ist vor allem für Bewertungsmodule relevant, die N-Gramme und nicht einzelne Worte bewerten sollen, wie das Google Bewertungsmodul.

Ein Grund hierfür liegt darin, dass die Bewertung eines N-Gramms nur Sinn macht, solange der Kontext sich nicht ändert. Deutlich wird diese Problematik beispielsweise an dieser Befehlsfolge: „Armar ich bin durstig, bring mir bitte den Saft. Danach stell den Saft zurück in den Kühlschrank. Sobald du dies getan hast, bring mir die Zeitung“. Ohne Einschränkung, welche Worte zu einem N-Gramm gehören können, kann es hier vorkommen, dass ein Bewertungsmodul das 2-Gramm „Zeitung„ und „Saft“ bewertet, was die Bewertung beider Worte fälschlicherweise negativ beeinflussen kann.

Dies ist zudem sinnvoll, um Bewertungen zu vermieden, die nicht zielführend sind. Eine Bewertung ist immer dann zielführend, wenn sie falsche Alternativen mit einer niedrigen Bewertung versieht oder richtige Alternativen mit einer hohen Bewertung. Bei Bewertungsmodulen, die mehrere Wörter gleichzeitig bewerten (wie das Google Bewertungsmodul), kann es leicht zu nicht zielführenden Bewertungen kommen, falls Wortkombinationen bewertet werden, die keinen semantischen Bezug untereinander haben. Dies ist vor allem dann der Fall, wenn ein Wort dieses 2-Gramms eine Konjunktion ist.

Um diese Problematik zu umgehen, kann versucht werden die Eingabe in Befehle zu teilen und Wörter mit niedriger Entscheidungskraft zu ignorieren. Ein Sprachverarbeitungswerkzeug könnte die Eingabe in Befehle teilen und die Wortarten der im Verwirrungsnetzwerk enthaltenen Wörter bestimmen.

5.4 Alternativenauswahl

Sind die im vorherigen Abschnitt beschriebenen Bewertungen angefertigt, muss aus diesen eine Entscheidung abgeleitet werden, welche Alternative jeder Alternativengruppe ausgewählt werden soll, um eine neue Ausgabehypothese zu bilden.

Zur Klassifizierung der Alternativen einer Alternativengruppe könnte ein Maschinenlernverfahren eingesetzt werden, das die Alternativen als korrekt oder inkorrekt markiert. Hier

könnte beispielsweise einer der Klassifizierer von WEKA verwendet werden[HFH+09]. Allerdings hat dieser Ansatz das Problem, dass die Klassifikation nicht eindeutige Ergebnisse für eine Alternativengruppe ergeben kann, wenn mehr als eine Alternative als korrekt markiert wird.

Um diese Problematik zu umgehen, kann statt eines Klassifizieres ein Ranglernverfahren (engl. learning to rank) verwendet werden, das jeder Alternative einen Rang anstatt einer Klasse zuweist.

Eine wiederum andere Möglichkeit diese Entscheidung zu treffen ist es, alle Bewertungen einer Alternative in eine einzige Bewertung zu überführen und dann für jede Alternativengruppe die Alternative mit der höchsten Bewertung auszuwählen.

6. Implementierung

Die in Kapitel 5 beschrieben Ansätze wurden im Rahmen dieser Arbeit teilweise implementiert und deren Umsetzung wird in diesem Kapitel beschrieben. Hierbei wurde jede Komponente des Entwurfs als möglichst eigenständiges Werkzeug implementiert. Die vier Werkzeuge *MultiASR*, *ConfusionNetworkBuilder*, *Revise* und *ReviseAnalyser* decken dabei ein breites Spektrum aus der Aufbereitung von Spracherkennerausgaben ab - von der Spracherkennung, über die Verarbeitung bis hin zur Auswertung.

MultiASR ermöglicht die Erkennung von Sprache mithilfe einer erweiterteren Zahl von Spracherkennern, sowie die Umwandlung der Ausgaben dieser Spracherkenner in ein spracherkennerunabhängiges Format. Mit Hilfe von *ConfusionNetworkBuilder* lassen sich die Ausgaben der Spracherkenner dann in Verwirrungsnetzwerke überführen. Für die eigentliche Aufbereitung steht dann *Revise* zur Verfügung, das in Zusammenarbeit mit *ReviseAnalyser* dazu verwendet werden kann, neue Ausgaben aus den Spracherkennerausgaben abzuleiten, die im Bezug auf ihre Wortfehlerrate optimiert wurden. Angemerkt sei hierbei, dass die Beschränkungen der Funktionsweise wahrscheinlich oft zu restriktiv sind, aber sinnvoll sind um das Prinzip aufzuzeigen ohne zusätzliche Fehler zu erzeugen.

Die genaue Funktionsweise der einzelnen Werkzeuge wird innerhalb dieses Kapitels erläutert.

6.1 Spracherkennung

Zur Spracherkennung wurde das Werkzeug *MultiASR* entwickelt, ein modulares System, das darauf ausgelegt ist, mit einer möglichst großen Zahl an Spracherkennern arbeiten zu können. *MultiASR* besteht im Wesentlichen aus drei Komponenten: einer Hauptkomponente sowie zugehörige Schnittstellen für die Erstellung von Spracherkenner- und Nachverarbeitungsmodulen.

Die einzelnen Komponenten von *MultiASR* werden im Folgenden beschrieben, wobei das zugehörige Klassendiagramm in Abbildung 6.2 dargestellt ist.

6.1.1 Spracherkennermodul

Jedes Spracherkennermodul implementiert die `IASR`-Schnittstelle, die einheitliche Methoden zur Erkennung von Sprache definiert.

Ein Spracherkennermodul überschreibt außerdem die `getSupportedAudioFormats`-Methode zur Beschreibung unterstützer Audioformate der Sprachdaten. Zur Beschreibung der Audioformate gehören beispielsweise die Kanalzahl, Abtastrate und Kodierung.

Um die verschiedenen Informationsgehalte der Spracherkenner auf ein gemeinschaftliches und erweiterbares System abzubilden, kann jeder Spracherkenner optional die Methode `getSupportedCapabilites` überschreiben, die eine Liste unterstützter Zusatzinformationen zurückgibt. Die Zusatzinformationen werden hierbei mit einer Zeichenkette identifiziert.

6.1.2 Nachbearbeitungsmodul

Ein Nachbearbeitungsmodul kann durch Implementierung der `IPostProcessor`-Schnittstelle erstellt werden. Mit Hilfe der Nachbearbeitungsmodule können die Spracherkennerausgaben direkt verändert oder gefiltert werden. Hierzu erhält die `process`-Methode jeweils eine Instanz von `ASROutput`, um diese zu verarbeiten. Der Rückgabewert wird als neue Version dieser Instanz angesehen, wobei die Rückgabe von `null` diese Ausgabe entfernt.

Die Nachbearbeitungsmodule können beispielsweise verwendet werden, um Ausgaben mit niedriger Konfidenz auszufiltern oder die Konfidenzen anzupassen. Die letztere Variante wird für die `-w` Option von *ReviseAnalyser* verwendet (siehe Abschnitt 6.3).

6.1.3 Hauptkomponente

Die Hauptkomponente (Schnittstelle `IMultiASR`) ist für die Verwaltung der Spracherkenner- und Nachverarbeitungsmodule zuständig.

Zur Verwaltung gehört in erster Linie die Möglichkeit, Spracherkenner- und Nachverarbeitungsmodule mit den Methoden `register` bzw. `registerPostProcessor` zu registrieren. Sobald ein Modul registriert wurde, wird es für alle zukünftige Spracherkennungsanfragen verwendet. Zur Registrierung von Spracherkenner- und Nachverarbeitungsmodulen unterstützt die Referenzimplementierung des Hauptmoduls in der Klasse `MultiASR` außerdem die automatische Registrierung aller verfügbaren Module mit den Methoden `autoRegisterASRs` beziehungsweise`autoRegisterPostProcessors`. Ein Modul ist dann verfügbar, wenn es in Form eines *service providers* zur Verfügung steht (siehe [Ora16]).

Außerdem ist das Hauptmodul zuständig für die Bereitstellung der richtigen Audioformate an die Spracherkennermodule sowie die Abwicklung der Nachverarbeitung durch die gleichnamigen Module. Diese Schritte werden als Teil der `recognize`-Methode ausgeführt und laufen nach dem in Abbildung 6.1 gezeigten Schema.

1. **Überprüfung:** In der Überprüfungsphase wird getestet, ob ein Spracherkenner die angeforderten Zusatzinformationen unterstützt, die beim Aufruf der Methode festgelegt werden können. Dabei wird zwischen optionalen und benötigten Zusatzinformationen unterschieden. Anforderungen beider Typen werden an die Spracherkennermodule weitergeleitet. Spracherkenner, die nicht alle der benötigten Zusatzinformationen unterstützen, werden allerdings von der Transkription ausgeschlossen.
2. **Umwandlung:** Ist ein Spracherkenner kompatibel mit den angeforderten Zusatzinformationen, wird in der Umwandlungsphase die beim Aufruf der Methode angegebene Audiodatei in ein vom Spracherkenner unterstütztes Format umgewandelt. Um Qualitätsverlust der Audiodatei zu vermeiden, findet diese Umwandlung allerdings nur bei Bedarf statt. Zur Umwandlung wird hier *FFmpeg* [FFm16] verwendet. *FFmpeg* ist ein Werkzeug, dass die Verarbeitung inklusive der Umwandlung von vielen gängigen, freien und lizenzpflichtigen Formaten unterstützt und sich bei Bedarf um weitere Formate erweitern lässt.

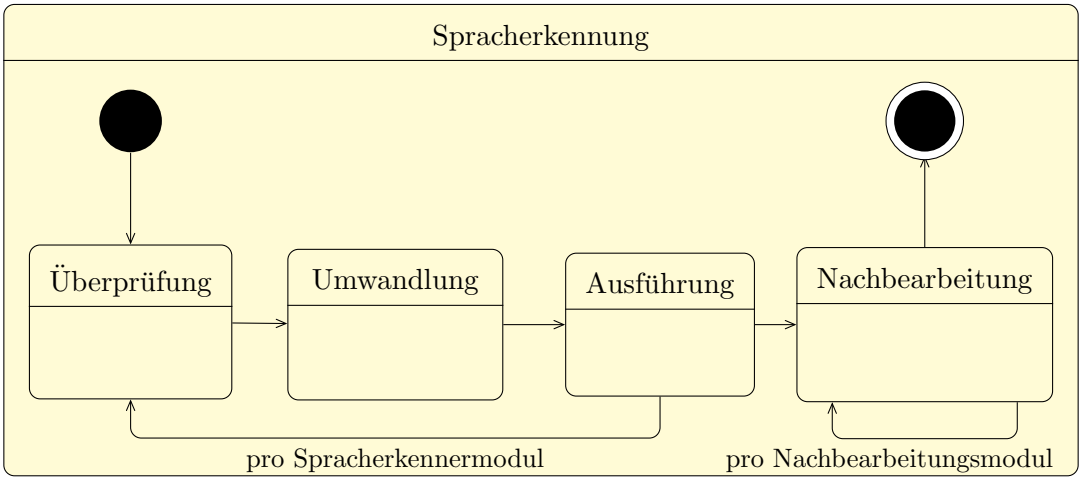

Abbildung 6.1: MultiASR Ablauf

3. **Ausführung:** In der Ausführungsphase wird das Spracherkennermodul aufgerufen. Gibt das Spracherkennermodul eine Ausgabe zurück, so wird diese an die Liste aller Spracherkennerausgaben angehängt.
4. **Nachbearbeitung:** Während der Nachbearbeitungsphase werden in Registerierungsreihenfolge alle Nachbearbeitungsmodule aufgerufen. Hier kann jedes dieser Module die Spracherkennerausgaben überarbeiten, löschen oder neue Ausgaben anfügen.

6.1.3.1 Ausgabe

MultiASR gibt erkannte Sprache in einer vereinheitlichten Form aus. Hierfür wurde die `ASROutput`-Klasse entworfen, deren wichtigste Komponenten in Abbildung 6.3 dargestellt sind. `ASROutput` ist so konzipiert, dass alle zusätzlichen Informationen von Spracherkennern eingebunden werden können, auch wenn sie zum Zeitpunkt der Erstellung dieser Arbeit noch nicht existierten. Um dies zu erreichen ist `ASROutput` im Wesentlichen eine Liste von `Token`s mit der Möglichkeit Metadaten anzuknüpfen.

Ein `Token` ist dabei ein identifizierbarer Teil der textuellen Spracherkennerausgabe, wie beispielsweise Wörter oder Satzzeichen, die in den Klassen `Word` bzw. `Delimiter` implementiert wurden. Die hier gewählte Implementierung macht zwar diese Unterscheidung, aber theoretisch könnten auch Wortgruppen oder ganze Sätze mithilfe eines Tokens repräsentiert werden. Zusätzlich können auch an `Token`s Metadaten angehängt werden.

Das Anhängen von Metadaten, sowohl an Tokens als auch an `ASROutput` geschieht dabei mithilfe der `DataGetter`-Schnittstelle. DataGetter ist eine generische Schnittstelle, die ein Objekt des generischen Typs `T` mithilfe der `get`-Methode zurückgibt. Nach der Instanziierung kann das DataGetter-Objekt an ein Token oder `ASROutput` angehängt werden, wobei als Schlüssel `T` verwendet wird. Der Umweg über die `DataGetter`-Objekte anstelle des direkten Anhängens des gekapselten Objekts wurde dabei gewählt, um die Änderung der angehängten Daten ohne deren expliziten Austausch zu ermöglichen. Ein Beispiel für solche Metadaten ist die `Score`-Klasse, die für Bewertungen und Konfidenzen zum Einsatz kommt. Die `Score`-Klasse wird in dieser Implementierung für Spracherkennerkonfidenzen, sowohl von Worten als auch von kompletten Ausgaben, verwendet.

6.2 Aufbereitung und Informationsanreicherung

Zur Verbesserung der Wortfehlerrate wurde das Werkzeug Revise erstellt, das einen Großteil der in Abschnitt 5.3 beschriebenen Konzepte realisiert. Revise besteht aus einer Reihe

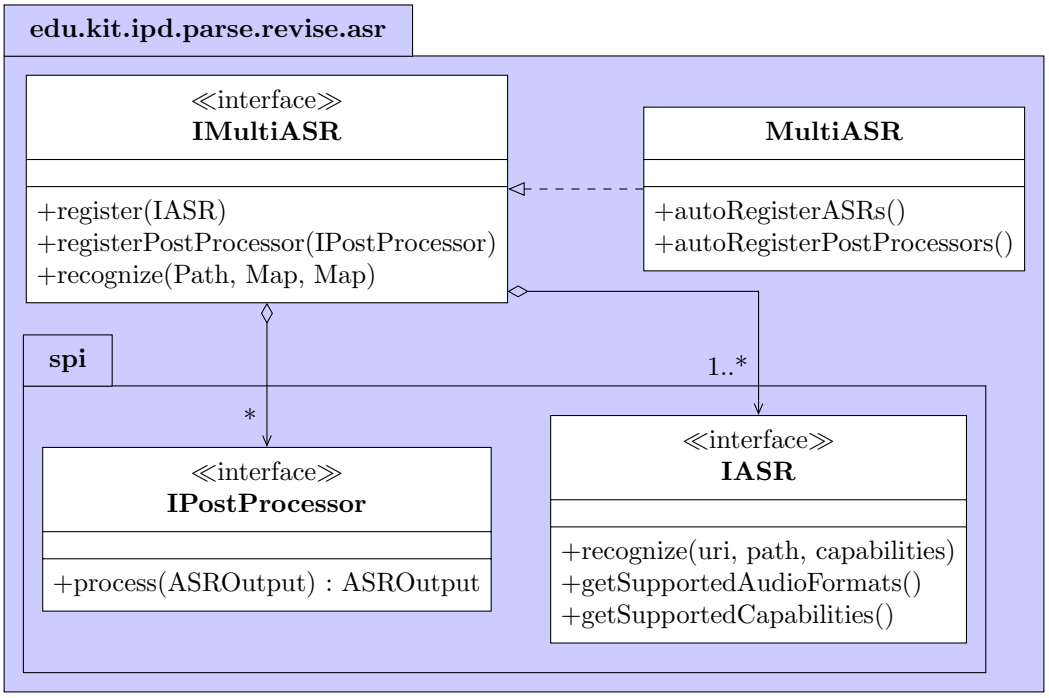

Abbildung 6.2: MultiASR

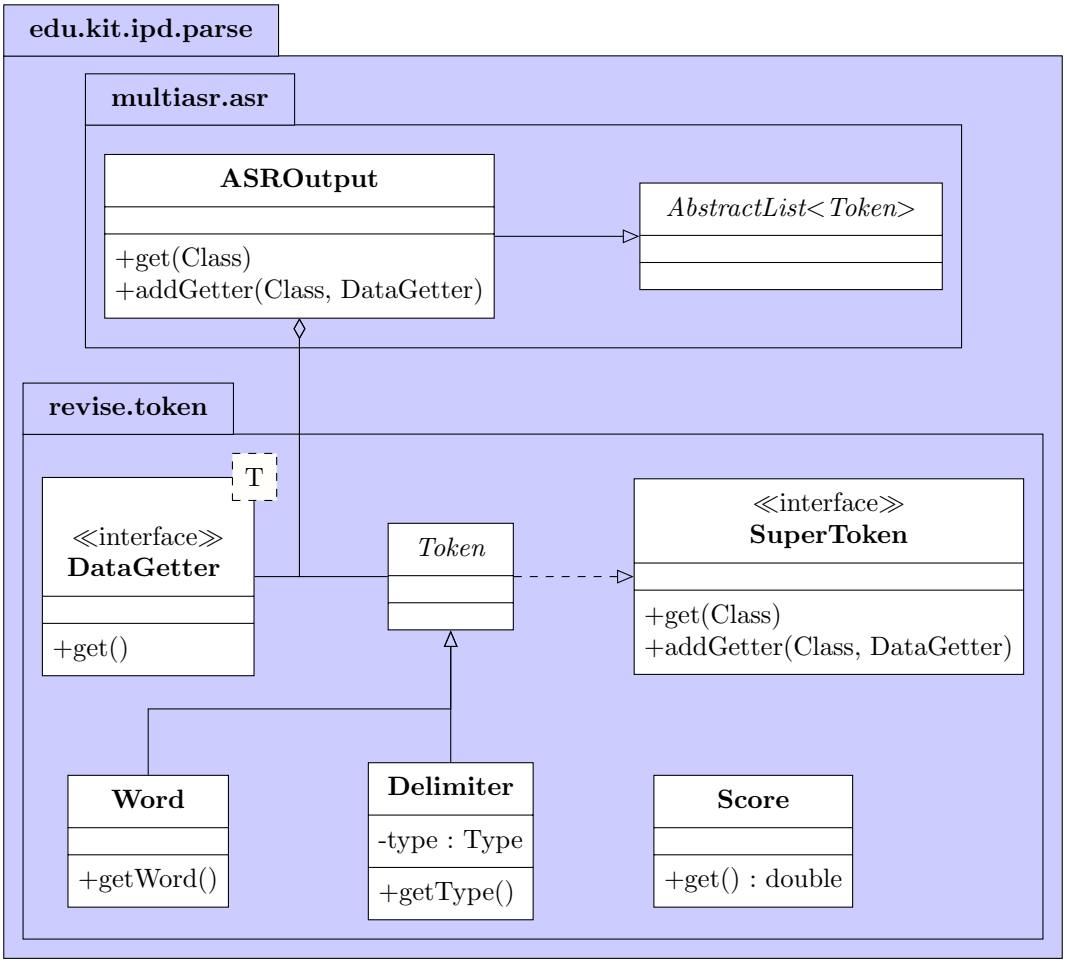

Abbildung 6.3: ASROutput Ausschnitt

von Vorverarbeitungs- und Bewertungseinheiten sowie einer zentralen Hauptkomponente. Es ist so aufgebaut, dass weitere Vorverarbeitungs- und Bewertungseinheiten erstellt und eingebunden werden können. Zusätzlich zu Revise wurde ein eigenständiges Werkzeug mit dem Namen ConfusionNetworkBuilder zur Erstellung von Verwirrungsnetzwerken implementiert.

Die genauen Einzelheiten der Implementierung beider Werkzeuge werden innerhalb dieses Abschnitts beschrieben.

6.2.1 Verwirrungsnetzwerke

Für die Erzeugung von Verwirrungsnetzwerk aus den Spracherkennerausgaben aus Abschnitt 6.1 wurde das Werkzeug *ConfusionNetworkBuilder* erstellt. *ConfusionNetworkBuilder* ist dabei vollkommen unabhängig von *Revise* und kann mit allen Spracherkennerausgaben arbeiten, die in Form eines `ASROuput`-Objekts zur Verfügung stehen. Die einzige Anforderung an die Ausgabe ist, dass die Worte durch `Token`s des Typs `Word` repräsentiert werden (siehe Abschnitt 6.1.3.1 insbesondere Abbildung 6.3). Um die Verwirrungsnetzwerke aus diesen Spracherkennerausgaben zu erzeugen, wird hier *SRILM* verwendet (SRI Language Modeling Toolkit) bewerkstelligt [Sto02].

SRILM ist eine von *SRI Speech Technology and Research Laboratory* entwickelte Sammlung von Werkzeugen im Bereich der Spracherkennung. Aus den in *SRILM* enthaltenen Werkzeugen wird zur Generierung der Verwirrungsnetzwerke das Werkzeug *nbest-lattice* verwendet. Es ermöglicht aus Sätzen und ihren zugehörigen Konfidenzen Verwirrungsnetzwerke zu erstellen. Da die hier verwendeten Spracherkenner allerdings nur für ihre Haupthypothese Konfidenzen ausgeben, werden zur Konstruktion der Verwirrungsnetzwerks gleiche Konfidenzen für alle Hypothesen angenommen.

Diese Wahl wurde getroffen, da Versuche gezeigt haben, dass dies zur Erstellung von Verwirrungsnetzwerken mit minimaler Größe und somit manuell erstellten Verwirrungsnetzwerken am nächsten kommt.

Die Einbindung des *nbest-lattice* funktioniert dabei nach dem in Abbildung 6.4 dargestellten Schema.

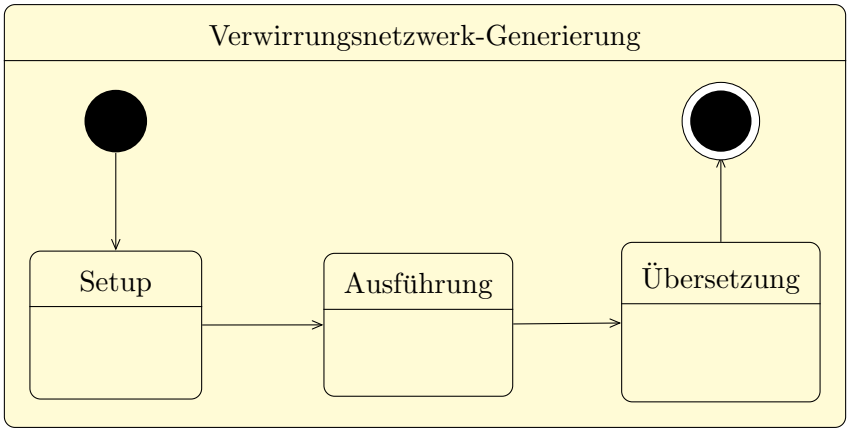

Abbildung 6.4: Verwirrungsnetzwerk-Generierung

1. **Setup:** In der Setup-Phase werden die Spracherkennerausgaben in das von *nbest-lattice* benötigte *nbest-Format* umgewandelt. Hierbei werden die Konfidenzen von Ausgaben ohne Konfidenz auf die niedrigste Konfidenz unter den Ausgaben gesetzt, um bestmögliche Ergebnisse der Verwirrungsnetzwerkerzeugung zu garantieren.

Zusätzlich werden Verzögerungslaute durch eine Zeichenkette ersetzt, die im normalen Sprachgebrauch nicht vorkommt (hier `*hesitation*`). Diese Ersetzung hat praktische Gründe: der Google Spracherkenner gibt keine Verzögerungslaute in seiner Ausgabe aus und der IBM-Watson Spracherkenner erzeugt nur eine Markierung ähnlich zur der eben beschriebenen. Um die Einbindung weiterer Spracherkenner zu ermöglichen wird deshalb an dieser Stelle jeglicher Verzögerungslaut, sofern er als solcher erkannt wurde, ersetzt.

Auch werden hier jegliche Satzzeichen entfernt, da die beiden verwendeten Spracherkenner zu sehr von einander abweichen und das Einbinden der Satzzeichen somit kein Mehrwert hat und nur zu größeren (und somit schlechter verarbeitbaren) Verwirrungsnetzwerken führt.

2. **Ausführung:** Die Ausführungsphase ruft `nbest-lattice` auf, wobei die Eingabedatei und Ausgabedatei festgelegt (`-rescore` *EINGABEPFAD* beziehungsweise `-write` *AUSGABEPFAD*), die Ausgabe von Verwirrungsnetzwerken forciert („-use-mesh“) und die Verzögerungsmarker als Rauschen markiert (`-noise` *hesitation*) werden. Die Verzögerungsmarker werden allerdings durch Setzen der Option `-keep-noise` nicht gelöscht, sondern wie andere Worte im Verwirrungsnetzwerk verbaut.
3. **Übersetzung:** Nach der Ausführung von `nbest-lattice` wird die Ausgabedatei in ein Verwirrungsnetzwerk übersetzt dessen Klassendiagramm in Abbildung 6.5 zu finden ist. Das Verwirrungsnetzwerk repräsentiert durch vier Schnittstellen `ConfusionNetwork`, `ClusterToken`, `Alternative` und `Token`. Die `Token` Schnittstelle ist hierbei die gleiche die auch für die Spracherkennerausgaben verwendet werden (siehe Abschnitt 6.1.3.1). Die anderen drei Klassen repräsentieren das Verwirrungsnetzwerk an sich, eine Alternativengruppe bzw. eine Alternative.

   Eine Alternativengruppe ist hierbei die Menge aller Pfade zwischen zwei Knoten des Verwirrungsnnetzwerkes, eine Alternative ein einziger dieser Pfade. Die Alternativen wiederum können eines oder mehrere Tokens enthalten, wobei hier Wörter, Verzögerungslaute und Löschungen als `Token`-Typen verwendet werden. Diese Typen werden mit den Klassen `Word`, `Hesitation` bzw. `Definiert`. Die Übergangswahrscheinlichkeiten der Alternative werden hier mithilfe der `Score` Klasse an die Tokens angehängt.

   Um eine Anbindung weiterer Spracherkenner in der Zukunft zu vereinfachen, wurde hier `Hesitation` als Unterklasse von `Word` implementiert, die auch Verzögerungslaute in Wortform zulässt. Außerdem kann jede Alternative mehrere Tokens enthalten. `nbest-lattice` gibt keine Alternativen mit mehr als einem Wort aus. Daher werden hier nur Alternativen verwendet, die jeweils ein Wort enthalten.

6.2.2 Revise

Nach der Erstellung der Verwirrungsnetzwerke mittels *ConfusionNetworkBuilder* kann das Werkzeug *Revise* verwendet werden, um aus den Verwirrungsnetzwerken eine neue Ausgabenhypothese zu generieren. Hierzu werden eine Reihe der in Abschnitt 5.3 beschriebenen Vorverarbeitungs- und Bewertungseinheiten verwendet, um das Verwirrungsnetzwerk mit Informationen anzureichern. Die Umsetzung dieser Komponenten wird in diesem Abschnitt beschrieben. Soweit wie möglich werden hierbei die Änderungen an dem in Abbildung 6.6 befindlichen Verwirrungsnetzwerk verdeutlicht. Es handelt sich hier allerdings aus Platzgründen um ein synthetisches Beispiel.

6.2.2.1 Annotierung

In diesem Schritt soll eine Auswahl getroffen werden, welche Alternativen und Paarungen dieser zur Analyse durch die Bewertungseinheiten infrage kommen. Um dies zu bewerkstel-

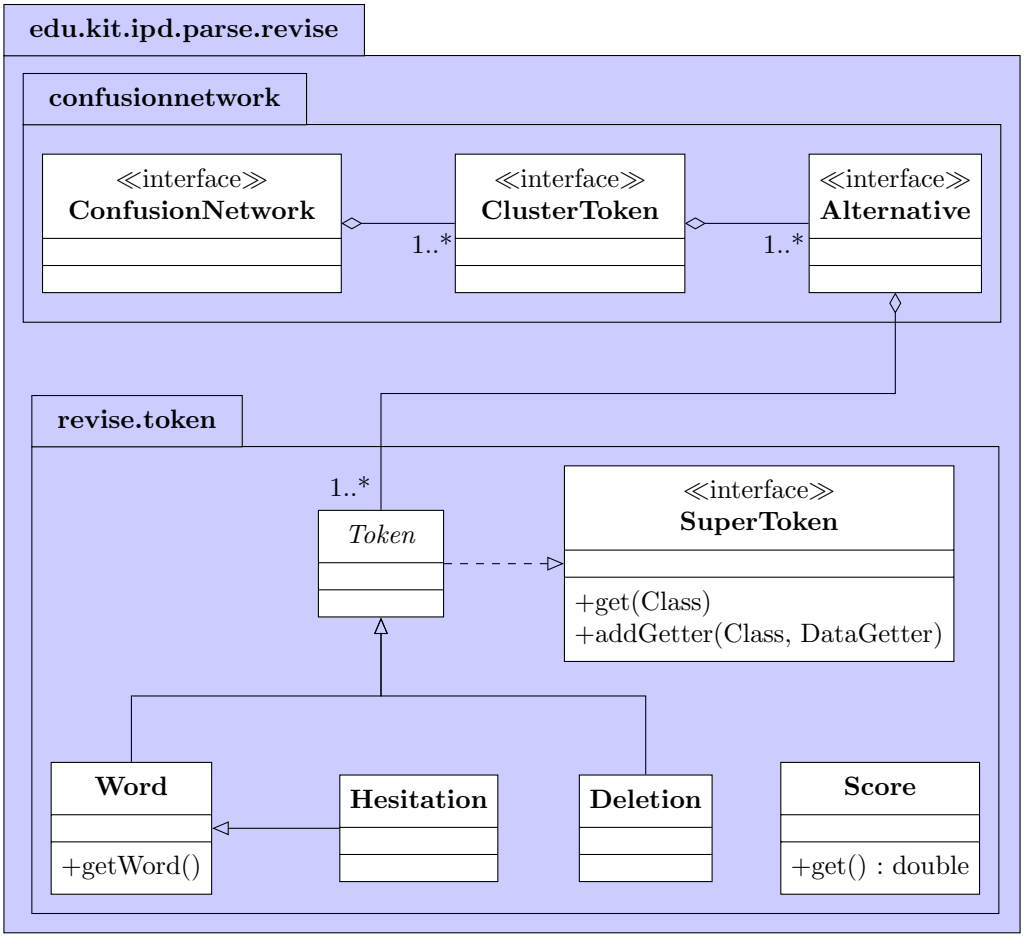

Abbildung 6.5: Verwirrungsnetzwerk Ausschnitt

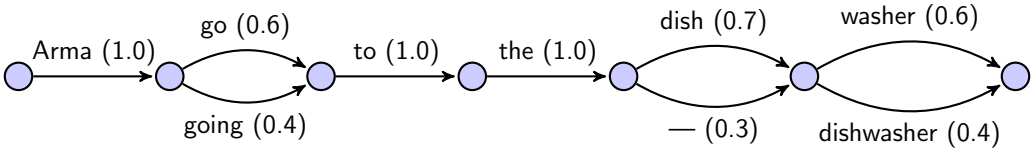

Abbildung 6.6: Ausgangsverwirrungsnetzwerk

ligen, soll wie in Abschnitt 5.3.4 beschrieben, ein Sprachverarbeitungswerkzeug verwendet werden, der Befehlsgrenzen bestimmt und Wortarten feststellt.

Da allerdings kein Sprachverarbeitungswerkzeug existiert, der direkt auf einem Verwirrungsnetzwerk arbeiten kann, werden in dieser Implementierung stattdessen die 25 wahrscheinlichsten Pfade durch das Verwirrungsnetzwerk mit einem Sprachverarbeitungswerkzeug ausgewertet. Als Gesamtwahrscheinlichkeit[1] eines Pfades wird hierbei die Summe aller Übergangswahrscheinlichkeiten des Pfades angenommen. Sind eine oder mehrere Alternativen nicht Teil dieser 25 wahrscheinlichsten Pfade, werden diese zusätzlich bewertet. Dazu nehmen diese Alternativen den Platz der wahrscheinlichsten Alternative im global wahrscheinlichsten Pfad ein.

Zur Veranschaulichung ist in Abbildung 6.7 ein Verwirrungsnetzwerk dargestellt, wobei hier eine Obergrenze von 6 anstelle von 25 angenommen wird. Dieses Verwirrungsnetzwerk enthält 8 Pfade[2], die annotiert werden können. Nach der Annotierung der wahrscheinlichsten 6 Pfade sind die Worte „b“ und „y“ ohne Annotation, die dann als Teil der Pfade 1-b-x und 1-a-y annotiert werden.

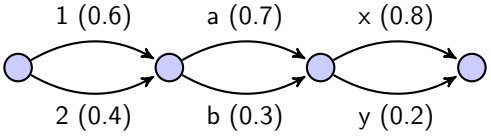

Abbildung 6.7: Beispiel Annotiertung

Diese Beschränkung der zu überprüfenden Pfade hat hauptsächlich zwei Gründe:

1. **Laufzeit:** Die Zahl der Pfade durch das Verwirrungsnetzwerk hängt ab vom Aufbau des Verwirrungsnetzwerks und beträgt im schlimmsten Fall $\mathcal{O}(2^a)$, wobei $a = \mathit{Alternativenanzahl}$. Demzufolge würde auch dieser Verarbeitungsschritt potenziell extrem lange dauern. Beispielsweise benötigt der hier verwendete Sprachverarbeitungswerkzeug zur Analyse eines Pfades etwa 1 Sekunde und die hier verwendeten Verwirrungsnetze enthalten teilweise über 600 mögliche Pfade.[3]

2. **Verwertbarkeit der Daten:** Da jeder Knoten Teil mehrerer Pfade sein kann, gibt es auch für jedes Wort mehrere mögliche Ausgaben des Sprachverarbeitungswerkzeugs bezüglich der Wortart und der Zugehörigkeit zu Befehlen. Mit der Analyse zusätzlicher Pfade können somit weitere mögliche Wortarten bzw. Befehlszugehörigkeiten für die Alternativen hinzukommen. Das Ergebnis wird dann weniger eindeutig und schwerer zu verarbeiten. Insbesondere für Pfade mit relativ geringer Gesamtübergangswahrscheinlichkeit (die demzufolge niedrigen Spracherkennerkonfidenzen entstammen) ist dies problematisch. Die händische Analyse bei der Erstellung dieser

[1] Es handelt sich hier somit nicht um eine echte Wahrscheinlichkeit, sondern ein Vergleichsmaß.
[2] Eine vollständige Liste aller Pfade ist in Tabelle 8.2 abgebildet.
[3] Eine ausführliche Behandlung des Laufzeitproblems findet sich in Abschnitt 7.3.7.

Implementierung ergab, dass für Pfade mit relativ niedriger Gesamtübergangswahrscheinlichkeit nur sehr selten die richtige Wortart und Befehlszahl ermittelt werden konnten.

Die Auswertung an sich wird hierbei mit *ShallowNLP* bewerkstelligt [Koc15]. *ShallowNLP* wurde im Rahmen des *Parse*-Projekts, zu dem auch diese Arbeit gehört, von Kocybik entwickelt. Die Verarbeitung mithilfe von *ShallowNLP* läuft hierzu in diesen Schritten ab:

1. **Umwandlung:** Die ausgewählten Pfade (s.o.) werden in Textfragmente umgewandelt, wobei Auslassungen und Verzögerungslautmarkierungen komplett entfernt werden.
2. **Auswertung:** *ShallowNLP* wird zur Auswertung der Textfragmente eingesetzt.
3. **Annotierung:** Die Ausgaben von *ShallowNLP* werden an das Verwirrungsnetzwerk angehängt. Die Ausgabe ist besteht aus einer Wortfolge, ergänzt durch Informationen über ihre Befehlszahl und Wortart. Um diese Informationen an die Verwirrungsnetzwerkalternativen anzuhängen, wird für jedes dieser Worte die entsprechende Alternative im Verwirrungsnetzwerk gesucht und das `Token`-Objekt an den Wortknoten angehängt. (Es handelt sich hierbei um eine andere `Token`-Klasse, als die in Abschnitt 6.1.3.1 beschriebene.)

6.2.2.2 Vorverarbeitung

Bevor die Alternativen bewertet werden, können Änderungen am Verwirrungsnetzwerk an sich vorgenommen werden. Hierzu wurden beide in Abschnitt 5.3.1 beschriebenen Vorverarbeitungsmodule implementiert, die einige häufige Fehler bei der Erstellung von Verwirrungsnetzwerken aus unterschiedlichen Datenquellen entfernen sollen. Diese Vorverarbeitungsmodule werden im Folgenden genauer beschrieben:

1. **Lemmatisierung:** Anhand ihrer Lemmata werden Alternativen derselben Alternativengruppe verschmolzen, sofern sie sich ein Lemma teilen. Um die Lemmatisierung zu bewerkstelligen, wird *WordNet* als Datenquelle und als Schnittstelle das Perl-Werkzeug *WordNet::Query* [Ren16] verwendet. Da manche Worte keinen eindeutigen Worttyp besitzen und somit kein eindeutiges Lemma, gibt *WordNet::Query* teilweise mehrere mögliche Lemmata zurück, zum Beispiel „walking“ und „walk„ als Lemmata zum Wort „walking“, unabhängig davon, ob das Ausgangswort ein Adjektiv, Nomen oder Verb war. Um diese Unschärfe aufzulösen, wird zusätzlich die Wortart der Lemmata mit der Wortart des Ausgangswortes abgeglichen. Im Falle der Lemmata geschieht dies mit *WordNet::Query* und im Falle des Ausgangswortes anhand der Annotierung des Sprachverarbeitungswerkzeugs. Da die beiden Datenquellen unterschiedlich detaillierte Ergebnisse zurückgeben, müssen diese aufeinander abgestimmt werden. Dazu wird der in Abschnitt 6.2.2.3 beschriebene Teil des WordNet-Bewertungsmoduls verwendet. Um bei dieser Auswahl der richtigen Lemmata Fehler zu vermeiden, wird die Lemmatisierung auf Alternativengruppen beschränkt, in denen sich alle Worte eine Wortart teilen. (Hat das Sprachverarbeitungswerkzeug mehrere mögliche Wortarten für ein Wort festgestellt, so muss es sich mindestens eine Wortart mit den übrigen Worten teilen — Auslassungen werden hierbei ignoriert.)

   Die eigentliche Verschmelzung folgt dann diesen Regeln:

   - Existiert das Lemma als eine der Alternativen, so wird dieses als Ziel der Verschmelzung gewählt.
   - Existiert das Lemma nicht, wird die Alternative mit der höchsten Übergangswahrscheinlichkeit zum Ziel bestimmt.

Bei der Verschmelzung werden alle Alternativen, außer der Zielalternative gelöscht. Zusätzlich werden alle Übergangswahrscheinlichkeiten der entfernten Alternativen aufaddiert und zur Zielalternative hinzugefügt. Im Falle des Beispielsverwirrungsnetzwerks resultiert aus diesem Verarbeitungsschritt das in Abbildung 6.8 gezeigte Netzwerk.

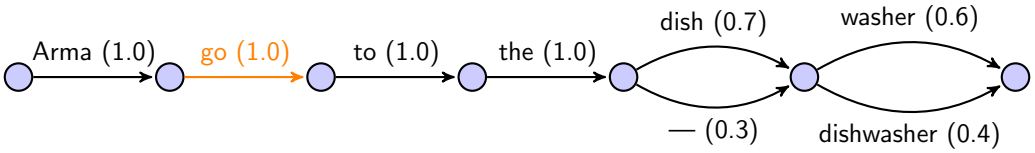

Abbildung 6.8: Verwirrungsnetzwerk nach der Lemmatisierung

2. **Wortverschmelzung:** Bei der Wortverschmelzung sollen zwei Worte zu einem Wort verschmolzen werden, falls für diese Worte eine Alternative existiert, die zusammen geschrieben wird. In der derzeitigen Implementierung werden allerdings nur die Fälle betrachtet, in denen zwei aufeinanderfolgende Alternativengruppen mit jeweils zwei Alternativen auftreten. In solch einem Fall müssen dann, wie in Abbildung 6.6, eine Auslassung sowie drei Worte enthalten sein. Mithilfe von *WordNet* wird ermittelt, ob die beiden Schreibweisen, getrennt und zusammen, ein Synset teilen. Ist dem so, werden die Einzelworte gelöscht und alle anderen Knoten behalten und deren Übergangswahrscheinlichkeit, auf 1 gesetzt. Für das Beispielsnetzwerk resultiert dann das reduzierte Netzwerk in Abbildung 6.9.

   Die Verschmelzung ist allerdings auf Wortartgruppen identischen Types beschränkt. Hierfür wird ebenfalls die Logik der in Abbildung 6.2.2.3 beschriebenen WordNet-Bewertungseinheit verwendet. Diese Einschränkung soll durch diesen Schritt eingeführte Fehler vermeiden.

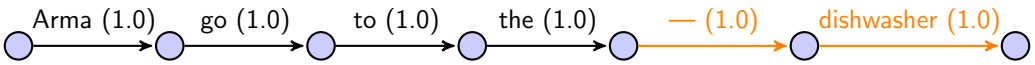

Abbildung 6.9: Verwirrungsnetzwerk nach der Wortverschmelzung

6.2.2.3 Bewertungsmodule

Von den in Abschnitt 5.3.3 beschriebenen Bewertungsmodulen wurden vier implementiert: Domäne, Google und WordNet-Abstand. Das Verwirrungsnetzwerk-Bewertungsmodul existiert auch, allerdings nicht als eigenständiges Modul. Stattdessen werden die Übergangswahrscheinlichkeiten aus dem Verwirrungsnetzwerk direkt als Bewertung angesehen. Die übrigen implementierten Bewertungsmodule werden in diesem Abschnitt genauer beschrieben.

Domäne

Die Domänen-Bewertungseinheit gleicht Worte aus dem Verwirrungsnetzwerk mit Worten aus der Domäne des Roboters ab. Hierzu wurde eine Domänenontologie entworfen, die Objekte und Handlungen enthält, die den Szenarien der im Korpus enthaltenen Aufnahmen entsprechen. Ein Ausschnitt der Ontologie ist in Abbildung 6.10 und die komplette Ontologie in Abbildung 8.1 dargestellt. Wie sich hier erkennen lässt wird zwischen Methoden, Objekten und Robotern unterschieden. Methoden werden mit Verben des Verwirrungsnetzwerks abgeglichen Objekt und Roboter werden mit Substantiven des Verwirrungsnetzwerkes abgeglichen.

- Thing
  - Method
    - bring
    - open
  - Object
    - cup
    - fridge
  - Robot
    - Armar

Abbildung 6.10: Domänenontolgie Ausschnitt

Dazu folgt die Bewertung dieser Einheit den in Algorithmus 6.1 dargestellten Schritten.

Im Wesentlichen werden hier die Levenshtein- und Jaro-Winkler-Distanzen der Double-Metaphon (siehe Abschnitt 2.8) Schlüssel mit einem festgelegten Maximum bzw. Minimum verglichen, um herauszufinden wie ähnlich die einzelnen Worte aus der Domäne zu einer der Alternativen und sowie zu allen Alternativen einer Alternativengruppe ist.

Die Levenshteindistanz [Lev66] gibt an, wie viele Änderungen (durch Löschen, Tauschen, Hinzufügen) nötig wären, um eine Zeichenkette in eine andere zu Überführen.

Die Jaro-Winkler-Distanz [Win06] hingegen gibt in einem Wert im Bereich $[0, 1]$ an, wie ähnlich sich zwei Zeichenketten sind, wobei ein besonderes Augenmerk auf das erste Zeichen gelegt wird. Bei diesem Distanzmaß ist entgegen dem, was der Name vermuten lässt, 1 der Wert der maximalen Gleichheit.

Sind alle Abstände ausgewertet, wird die Bewertung durch dieses Modul entsprechend gesetzt, wobei zwischen dem Fall, ob das Domänenwort in den Alternativen enthalten ist oder nicht, unterschieden wird.

Einige Aspekte des Algorithmus sind allerdings besonders hervorzuheben:

- Es werden die Double-Metaphon Schlüssel und nicht die Wörter direkt verglichen aus folgender Überlegung heraus: Die in dem Verwirrungsnetzwerk enthaltenen Wörter sind Resultat des Akustikmodells eines Spracherkenners, der diese Worte anhand ihrer Phoneme gewählt hat. Demzufolge ist die Schreibweise des Wortes weniger relevant als seine Aussprache. Double-Metaphon ist eine Möglichkeit, auf eine schreibweisenagnostische Version des Wortes zu kommen. Dabei müssen immer beide Double-Metaphon-Schlüssel eines Wortes mit beiden Schlüsseln des anderen Wortes verglichen werden.
- Die gewählten Grenzwerte beruhen auf der manuellen Auswertung einiger Verwirrungsnetzwerke. Eine Optimierung dieser Werte ist durchaus sinnvoll, wurde allerdings wegen des hohen Zeitaufwands nicht durchgeführt.
- Es wird sowohl die Levenshtein- als auch die Jaro-Winkler-Distanz[4] berechnet, um die Schwächen Beider zu minimieren. Die Levenshtein-Distanz ist ein absoluter Wert und somit gut geeignet, wenn ein Wort sehr ähnlich klingen muss, allerdings mit dem Problem, dass sehr kurze Schlüssel anhand dieser Metrik immer sehr ähnlich

[4]Für die Jaro-Winkler-Distanz gilt, entgegen ihrem Namen, dass höhere Werte größere Ähnlichkeit bedeuten

Eingabe : Verwirrungsnetzwerk, Domänenonthologie
für alle *Wortart* **tue**
  für alle *Verwirrungsnetzwerk.Alternativengruppen* **tue**
    $Min\_Distanz := 2$;
    für alle *Alternativengruppe.Alternative* ***und*** *Domänenonthologie.Wortart.Domänenworte* **tue**
      Bestimme Levenshtein- und Jaro-Winkler-Distanz zwischen Double-Methaphon von Alternative und Domänenwort;
      wenn *Levenshtein Distanz* $< Min\_Distanz$ **und** *Jaro Winkler Distanz* $<= 0.85$ **dann**
        $Min\_Distanz$ = Levenshthein-Distanz;
      Ende
    Ende
    wenn $Min\_Distanz <= 1$ **und** $\varnothing$ *Levenshtein Distanz* $<= 1.2$ **und** $\varnothing$ *Jaro-Winkler-Distanz* $>= 0.75$ **dann**
      wenn *Alternativengruppe enthält ein Domänenwort* **dann**
        Setze Bewertung für alle Domänenworte auf 1.0
      sonst
        Füge Domänenwort hinzu mit Bewertung 1.0;
        Setze Bewertung der übrigen Worte = $\frac{1}{1 + \text{Levenshtein Distanz}}$;
      Ende
    Ende
  Ende
Ende

Algorithmus 6.1 : Algorithmus der Domänenbewertungseinheit

erscheinen. Dies ist vor allem kritisch, da eine Reihe von Worten aus der Domäne Double-Metaphon-Schlüssel besitzen, die nur aus einem Zeichen bestehen. Die Jaro-Winkler-Distanz hat das umgekehrte Problem, bei langen Eingaben auch relativ große Änderungen zuzulassen. Demgegenüber ist die Überbewertung des ersten Buchstabens hier ein Vorteil, um auch bei sehr kurzen Schlüsseln verwertbare Distanzwerte zu erhalten (vor allem bei Schlüsseln mit 1 bis 2 Zeichen).

Beispielsweise haben die Worte „show“ und „go“ die Double-Metaphone Schlüssel „x“ und „xf“ beziehungsweise „k“ und „k“. Die Levensthein-Distanz alleine ist hier ungenügend, da diese für die beiden ersten Schlüssel 1 beträgt und die Worte somit als sehr ähnlich eingestuft werden würde. Hier ist das Heranziehen der Jaro-Winkler-Distanz hilfreich, die hier 0 ist und somit die Schlüssel als sehr unterschiedlich erkennt.

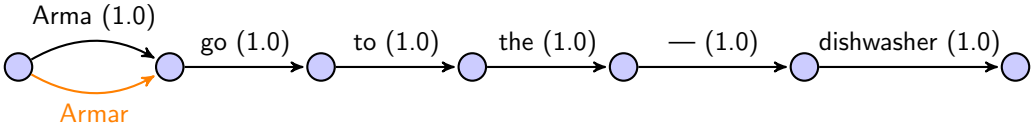

Abbildung 6.11: Verwirrungsnetzwerk nach der Domänenbewertungseinheit

Google

Das Google-Bewertungsmodul bildet Wortpaare und bewertet diese. Hierbei werden alle möglichen Kombinationen von Worten zweier Alternativengruppen betrachtet, sofern es sich bei diesen um Verben, Adjektive oder Substantive handelt. Außerdem wird sichergestellt, dass die betroffenen Worte keinen Befehl zwischen sich haben, also zum selben

oder zu benachbarten Befehlen gehören. Um den Arbeitsaufwand weiter einzuschränken, werden zudem Adjektive aus Alternativengruppen ignoriert, falls gilt $|A| * |O| > 8$, wobei $|A|$ die Anzahl an Alternativen dieser Alternativengruppe und $|O|$ die zum Vergleich in Frage kommenden anderen Worte sind. All diese Einschränkungen dienen der Reduktion der nötigen Suchanfragen an Google, da diese mit täglichen Obergrenzen oder Kosten verbunden sind. Dennoch wurde versucht, eine Balance zu finden zwischen relevanten Daten und Anfragenanzahl, weswegen Adjektive beispielsweise nicht komplett ignoriert werden.

Sind zwei Worte a und b gefunden, so gilt für die Bewertung Gleichung 6.1, wobei bei Suchanfragen nach mehreren Wörtern diese durch Leerzeichen getrennt werden.

$$S_a = \frac{R(a,b)}{R(a)} \tag{6.1}$$

$$S_b = \frac{R(a,b)}{R(b)} \tag{6.2}$$

$$\text{mit: } S_w = Bewertung\ von\ w \tag{6.3}$$

$$R(x,y,...) = Suchresultatanzahl\ f\ddot{u}r\ x,y,... \tag{6.4}$$

Beispielsweise gab Google für die Suchanfragen nach „orange“ 695000000, nach „ juice“ 30400000 und nach „orange juice“ 27600000 Suchergebnisse zurück. Dies würde dementsprechend zu einer Bewertung von 0.03971 für „orange“ und 0.9079 für „ juice“ resultieren.

WordNet

Das WordNet Bewertungsmodul liefert in dieser Implementierung nicht nur eine Bewertung, sondern mehrere auf *WordNet* basierende Maße (siehe Abschnitt 3.2). Diese errechnen alle einen eigenen Wert, der als Bewertung eines eigenständigen Bewertungsmoduls angesehen wird. Die Bewertung folgt dabei den in Abbildung 6.12 dargestellten Schritten:

1. **Wortartenabgleich**
   Beim Wortartenabgleich werden die Wortarten aller Worte des Verwirrungsnetzwerkes mit den möglichen Wortarten in *WordNet* abgeglichen. Dies erfolgt, da manche Worte keine eindeutige Wortart in WordNet haben, beispielsweise kann das Wort „walk“ nach WordNet sowohl ein Nomen als auch ein Verb sein, die Annotierung durch das Sprachverarbeitungswerkzeug kann aber ergeben, dass an dieser Stelle im Satz „walk“ immer ein Verb sein muss. Geben sowohl das Sprachverarbeitungswerkzeug als auch *WordNet* mehrere Wortarten für ein Wort an, so werden alle diese Wortarten im weiteren Verlauf in Betracht gezogen.

   Gesondert anzumerken ist hierbei, dass WordNet nur vier Wortarten unterscheidet: Verben, Nomen, Adjektive und Adverbien. Das verwendete Sprachverarbeitungswerkzeug unterscheidet allerdings unter weit mehr Wortarten wie beispielsweise Partikel, Kardinalzahl, Modalverb aber auch Verb in Vergangenheitsform oder Verb in Zukunftsform. Aus diesem Grund wird wenn möglich eine Angleichung durchgeführt. (Der Quelltext zur entsprechenden Methode findet sich in Quelltextausschnitt 8.1.)

2. **Wortartgruppenbestimmung**
   Das *WordNet*-Bewertungsmodul ist aufgrund der hohen Anzahl nötiger Berechnungen sehr langsam, weshalb an dieser Stelle versucht wird den Arbeitsaufwand zu verringern, indem Bewertungskandidaten ausgeschlossen werden, von denen keine verwertbaren Ergebnisse zu erwarten sind. Deshalb werden in diesem Schritt die Alternativengruppen in Wortartgruppen eingeteilt. In jeder Wortartgruppe findet sich jede Alternativengruppe, deren Alternativen alle die entsprechende Wortart annehmen können. Besteht beispielsweise eine Alternativengruppe aus den Alternativen

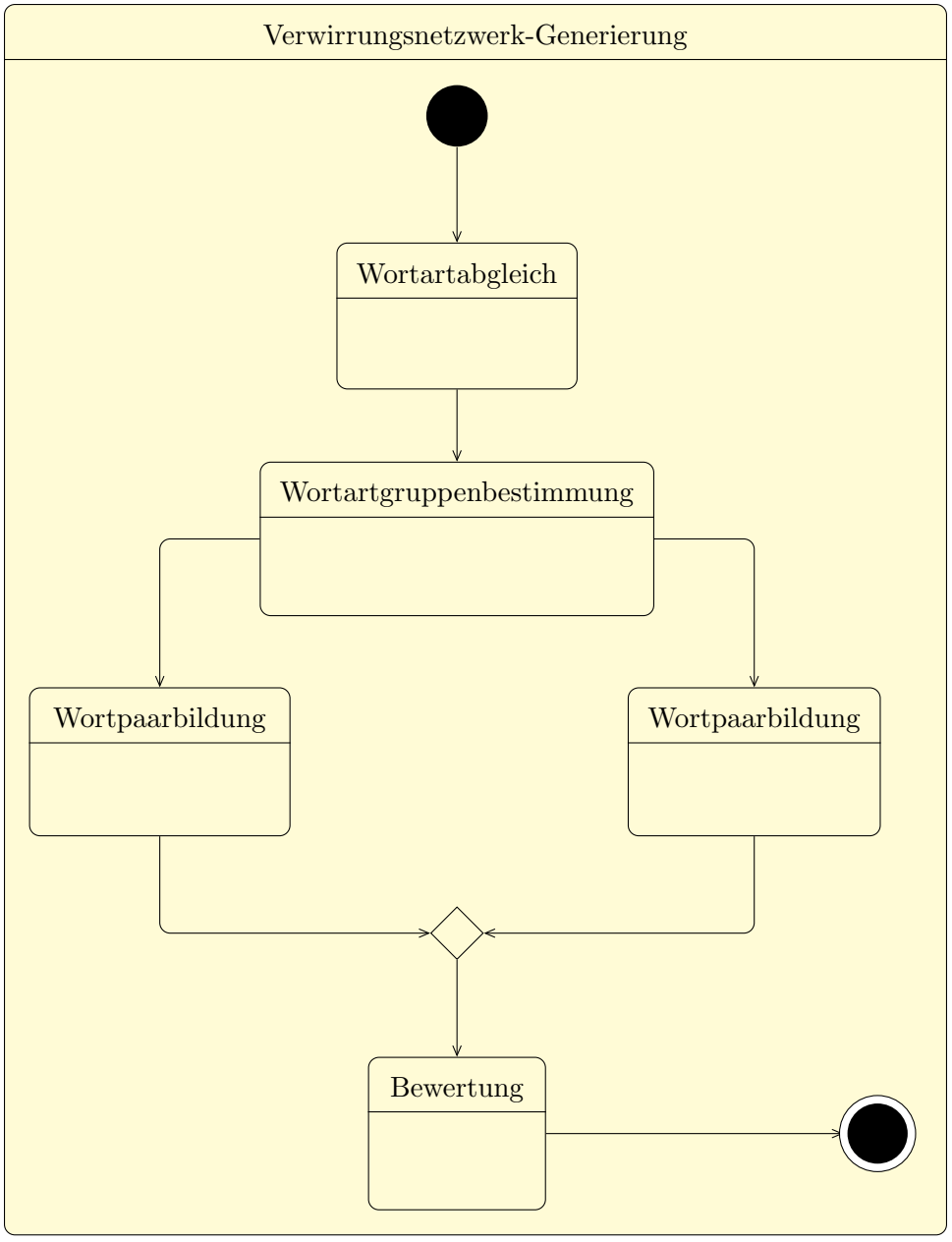

Abbildung 6.12: Verwirrungsnetzwerk-Generierung

„walk“ und „talk“ und das Sprachverarbeitungswerkzeug als auch *WordNet* erkennen Beide als mögliches Nomen oder Verb, so wird diese Alternativengruppe sowohl der Nomengruppe als auch der Verbgruppe hinzugefügt. Wäre beispielsweise „walk“ nur als Verb erkannt durch das Sprachverarbeitungswerkzeug, so würde die Alternativengruppe nur der Verbgruppe hinzugefügt werden.

Dieser Schritt verfolgt den Gedanken, dass wenn an einer Stelle Worte stehen, die sich in der Wortart nicht überschneiden, die Analyse mit WordNet wahrscheinlich auch nicht mehr in der Lage ist, die richtige Alternative auszuwählen.

Zusätzlich zur Beschränkung auf Wortartgruppen gibt es keine Wortartengruppe für Adverbien. Diese Entscheidung liegt darin begründet, dass die Abstandsmaße von WordNet nur zwischen Worten identischer Wordart funktionieren. Da in dem verwendeten Korpus nie mehrere Adverbien in Verbindung mit einem einzigen Verb stehen und der Zusammenhang zwischen Adverbien unterschiedlicher Verben als wenig zuträglich wurde diese alternative komplett außen vorgelassen.

3. **Wordpaarbildung**
Um aus den Ähnlichkeitsmaßen aus Abschnitt 3.2 einen Wert für einzelne Worte zu berechnen, müssen Wortpaare gebildet werden, für die eine Ähnlichkeit bestimmt werden kann. Hierbei wird zwischen zwei Möglichkeiten unterschieden: Bewertung von Wortpaaren gleicher Wortart und Bewertung von Wortpaaren unterschiedlicher Wortart. Diese Unterscheidung muss erfolgen, da alle außer den auf Glossen basierenden (siehe Abschnitt 3.2) Ähnlichkeitsmaße keine Ähnlichkeit zwischen Worten unterschiedlicher Wortart bestimmen können. Von den verwendeten die Maßen sind nur *Lesk* und *VectorPairs* zur Bestimmung der Ähnlichkeit von Wörtern unterschiedlicher Wortart geeignet.

   Die Art der Wortpaarbildung geschieht demzufolge auf zwei verschiedene Weisen:

   a) **Gleiche Wortarten:** Für die Ähnlichkeitsmaße, die gleiche Wortarten benötigen, werden Paare innerhalb aller Wortartgruppen gebildet. Aus diesen werden dann alle Wortpaare bestimmt, die zu benachbarten oder gleichen Befehlen gehören. Ist die Befehlszugehörigkeit nicht eindeutig (weil das Sprachverarbeitungswerkzeug mehrere Ergebnisse für eines der Worte geliefert hat), werden die Befehle der Eingabe durchnummeriert und dann der jeweils höchste Wert für die Befehlszugehörigkeit verwendet.

   b) **Unterschiedliche Wortarten:** Die Wortpaarbildung zwischen Wortartgruppen wurde auf Nomen und Adjektive beschränkt, da hier die Paarung am eindeutigsten ist. Im Detail bedeutet dies, dass ein Paar aus einem Adjektiv und einem Nomen gebildet wird, sofern zwischen dem Adjektiv und dem Nomen keine anderen Worte oder nur Adjektive stehen. Die Entscheidung, welche Worte an einer Stelle im Verwirrungsnetzwerk stehen, wird dabei anhand der Wortartgruppen getroffen.

   Zur Verdeutlichung der beiden Arten der Wortgruppenbildung sind in Tabelle 6.1 die Wortpaare zu dem in Abbildung 6.13 dargestellten Verwirrungsnetzwerk aufgelistet. Die Knoten in diesem Verwirrungsnetzwerk folgen dabei dem Format `Wortart Wort - Befehl`.

4. **Bewertung**
Im Bewertungsschritt wird die Bewertung für jedes Bewertungsmaß gleich der Ähnlichkeit nach dem entsprechenden Ähnlichkeitsmaß gesetzt. Hierfür kommt das das Perl-Werkzeug *WordNet::Similarity* zum Einsatz [PPM04]. Mit dessen Hilfe können

Tabelle 6.1: Wortpaare

Art der Wortpaarbildung	Wortpaare
Gleiche Wortarten	A B
	A C
	X Z
	Y Z
	B D
	C B
Unterschiedliche Wortarten	X B
	X C
	Y B
	Y C
	Z B
	Z C

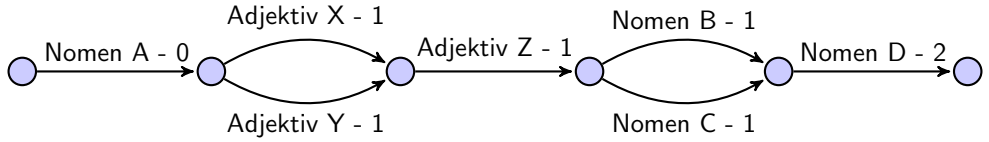

Abbildung 6.13: Beispielnetzwerk zur Wortpaarbildung

alle in Abschnitt 3.2 vorgestellten Ähnlichkeitsmaße bestimmt werden, wobei Abstandsmaße in Ähnlichkeitsmaße übersetzt werden (Ähnlichkeit = Abstand^{-1}). Von den in *WordNet::Similarity* enthaltenen Maßen werden hier *Lesk*, *Jcn*, *Lin*, *Res*, *Hso*, *Lch* und das *VectorPairs* Maß verwendet. Verwendet werden immer alle anwendbaren Maße. Dies bedeutet, dass bei Adjektiv-Nomen-Paaren nur die Maße *Lesk* und *VectorPairs* zum Einsatz kommen, da nur diese für Wortpaare unterschiedlichen Typs verwendet werden können. Für Bewertungen von Wortpaaren identischen Worttyps werden dementsprechend alle Ähnlichkeitsmaße verwendet. Die Ähnlichkeitsmaße sind allerdings immer nur für ein Synset-Paar definiert, weswegen hier die Ähnlichkeit aller Synsets der beiden Worte bestimmt wird und dann der höchste Wert als Bewertung gesetzt wird. Um einen Wert zwischen nur zwei Synsets als Bewertung zu verwenden, müsste die Bedeutung des Wortes an der entsprechenden Stelle (und somit das Synset) bekannt sein.

6.2.2.4 Ausgabenerstellung

Sind alle Bewertungen der Bewertungsmodule für die Alternativen des Verwirrungsnetzwerk berechnet, kann aus diesen eine neue Ausgabehypothese abgeleitet werden. Hierzu wird aus den Bewertungen der Module eine einzige Bewertung pro Alternative errechnet, danach wird die mit der höchsten Bewertung ausgewählt.

Da manche Bewertungseinheiten N-Gramme bewerten und nicht einzelne Alternativen, kann es vorkommen, dass eine Alternative mehrere Bewertungen einer Bewertungseinheit erhält. Werden beispielsweise die 2-Gramme „bring Saft“ und „bring Säfte“ aus dem Verwirrungsnetzwerksausschnitt in Abbildung 6.14 durch das gleiche Bewertungsmodul bewertet, so resultieren zwei Bewertungen für die Alternative „bring“.

Zur Weiterverarbeitung der Bewertungen müssen die Mehrfachbewertungen zu einer Bewertung zusammengeführt werden. Hierfür gibt es mehrere Alternativen, von denen ein paar implementiert wurden. Die Verschmelzung findet dabei immer in den folgenden drei Schritten statt:

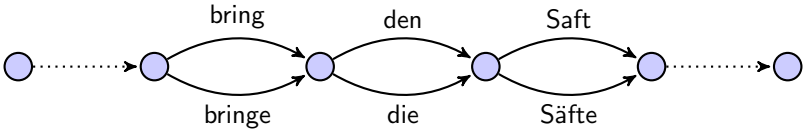

Abbildung 6.14: Beispiel zur Mehrfachbewertung

1. **Reduktion:** Im Reduktion-Schritt werden die Bewertungen jeder Alternative auf eine Bewertung pro Bewertungsmodul reduziert, wobei jedes Ähnlichkeitsmaß des *WordNet*-Bewertungsmoduls als eine Bewertung zählt. Hierfür wurden zwei Alternativen implementiert:

   a) **Durchschnitt:** Bildet den Durchschnittswert aller Bewertungen des gleichen Bewertungsmoduls.

   b) **Höchster Wert:** Wählt den höchsten aller Bewertungen eines Bewertungsmoduls.

2. **Skalierung:** In diesem Schritt werden die Bewertungen $b_i(x)$ jedes Bewertungsmoduls mit einem für dieses Bewertungsmodul spezifischen Gewicht g_i und Exponent e_i multipliziert beziehungsweise potenziert, so dass Gleichung 6.5 gilt.

$$b_i^\star(x) = (g_i * b_i(x))^{e_i} \tag{6.5}$$

$$\text{mit: Skalierte Bewertung } b_i^\star(x) \tag{6.6}$$

$$\text{Alternative } x \in X \text{ Menge aller Alternativen} \tag{6.7}$$

$$\text{Bewertungsmodul } i \in I \text{ Menge aller Bewertungsmodule} \tag{6.8}$$

   Es werden hier Exponente wurden hier verwendet um Bewertungen mit nicht linearem Einfluss zu erfassen.

3. **Fusion:** Im Fusionschritt werden die Bewertungen der unterschiedlichen Bewertungseinheiten zu einem Wert fusioniert. Auch hier wurden zwei Ansätze implementiert:

   a) **Summe:** Bildet die Summe aller Bewertungen (siehe Gleichung 6.9).

$$\overline{b}_s(x) = \sum_I b_i^\star(x) \tag{6.9}$$

$$\text{mit: Gesamtbewertung } \overline{b}_s(x) \tag{6.10}$$

   b) **Durchschnitt:** Bildet den Durchschnitt aller Bewertungen (siehe Gleichung 6.11).

$$\overline{b}_d(x) = \frac{\sum_I b_i^\star(x)}{k} \tag{6.11}$$

$$\text{mit: Gesamtbewertung } \overline{b}_d(x) \tag{6.12}$$

$$\text{Anzahl existierende Bewertungen } k \tag{6.13}$$

Nach Abschluss aller dieser Schritte und sofern die Exponenten und Gewichte richtig gewählt wurden, kann aus dem Beispielverwirrungsnetzwerk aus Abbildung 6.11 die Ausgabe „Armar go to the dishwasher“ abgeleitet werden.

6.3 Optimierung und Auswertung

Die Wahl der Gewichte und Exponenten, die in Abschnitt 6.2.2.4 verwendet werden, ist ausschlaggebend für die Qualität der Ausgabe insbesondere im Bezug auf die Wortfehlerrate. Aus diesem Grund wurde als weiteres Werkzeug *ReviseAnalyser* entworfen, um möglichst optimale Werte für die Gewichte und Exponenten zu finden, so dass die Wortfehlerrate der neu erstellten Ausgabehypothese minimal wird.

Bei *ReviseAnalyser* handelt es sich um ein Kommandozeilenwerkzeug, dass in Verbindung mit einem Korpus bestehend aus Audiodateien sowie ihren Referenztranskriptionen dazu verwendet werden kann, die Wortfehlerrate der eingesetzten Spracherkenner zu analysieren sowie Gewichte und Exponenten zu finden, die die Wortfehlerrate versuchen zu minimieren. Zwischenergebnisse und die für manche Operationen nötigen Referenztranskriptionen werden in der SQLite-Datenbank `wordscore.db` abgelegt. Zusätzlich können mit Hilfe von *ReviseAnalyser* die hier verwendeten Spracherkenner evaluiert werden.

Das die Bestimmung der Gewichte und Exponenten wird hier Optimierung genannt und kann auf zwei Weisen erreicht werden. Bei beiden der folgenden Methoden werden die Gewichte und Faktoren als Funktionsargumente und die Wortfehlerrate als Funktionswert angesehen.

- **Differenzialevolution:** Die Differenzialevolution ist ein evolutionärer Algorithmus von Storn et. al [SP97] zur Optimierung eines Funktionswerts, ohne dass die Funktion differenzierbar sein muss.

  Bei der Differenzialevolution werden, durch iterative Kombination von potentiellen optimalen Lösungen, bei jeder Iteration neue Lösungen ermittelt. Hierbei ersetzten bessere Lösungskandidaten in jedem Schritt vorher gefundene schlechtere Kandidaten. Die Startlösungen werden zufällig Bestimmt. Dies bedeutet das die Differenzialevolution heuristischer Natur ist und deshalb nicht zwingend das globale Optimum findet.

- **Pegasos:** Pegasos ist ein von Shalev-Shwartz et al. [SSSSC11] entwickelter Algorithmus zur Funktionsoptimierung. Dieser Algorithmus zeichnet sich vor allem durch seine geringe Berechnungsdauer aus.

  Pegasos ist ein Klassifikator, kann aber auch als Ranglernverfahren eingesetzt werden. Hierzu wird für alle Verwirrungsnetzwerke zu jeder Alternativengruppe die korrekte Alternative bestimmt. Anschließend können mit Hilfe von Pegasos Gewichte und Faktoren bestimmt werden die der Klassifikation entstammen.

Die Argumente dieses Werkzeuges sowie deren Funktionsweise sind im Folgenden dargestellt, wobei Argumente kursiv erscheinen. Dabei umfasst *ReviseAnalyser* alle, für die Erstellung der Evaluation in Kapitel 7 dieser Arbeit, nötigen Funktionen.

- **`-g`:** Diese Option startet die Generierung und Bewertung von Verwirrungsnetzwerken aus den mit `-s` angegebenen Sprachdateien. Diese Verwirrungsnetzwerke werden dann mithilfe von *Gson* [Goo16b] serialisiert und in der Datenbank abgelegt. Diese können mit den Optionen `-o`, `-t` oder `-u` weiter ausgewertet werden. Wird der Aufruf dieser Methode abgebrochen, so bleiben fertig generierte Verwirrungsnetzwerke in der Datenbank bestehen. Die verwendeten Spracherkenner sind Google und IBM-Watson, von denen jeweils die 5-Best verwendet werden.

- **`-s` *Pfad*:** Setzt den Pfad zu dem Ordner, der die zu verarbeitenden Sprachdateien enthält. Verarbeitet werden alle Dateien mit den Dateiendungen *flac* oder *wav*.

- **`-c`:** Löscht die mit `-g` erstellten Verwirrungsnetzwerke.

- `-e:` Das Google-Bewertungsmodul ist standardmäßig nicht aktiviert, diese Option aktiviert dieses bei der Generierung der Verwirrungsnetzwerke mit `-g`.
- `-b:` Berechnet die Wortfehlerrate, TERp und Bleu (siehe Abschnitt 7.2.2) für die Haupthypothese der Spracherkenner Google und IBM-Watson. Da Google nicht immer Ausgaben zurück gibt, werden für diesen Fall zwei Ergebnisse ausgegeben: Einen, in dem fehlende Ausgaben ignoriert werden und die Wortfehlerrate für die restlichen Ausgaben bewertet wird, sowie einen, in dem fehlende Ausgaben als leere Ausgabe interpretiert werden.

  Zusätzlich werden verschiedene Kombinationen von Google, IBM-Watson sowie Revise ausgewertet, die in Abbildung 6.15 dargestellt sind. Von Oben nach unten umfasst diese Ausgabe die folgenden Informationen:

  1. Wortfehlerrate, TERp und Bleu Google: Die vorderen Werte ignorieren fehlende Ausgaben; die Werte in Klammern sehen diese als leere Ausgaben an.
  2. Wortfehlerrate, TERp und Bleu IBM Watson.
  3. Wortfehlerrate, TERp und Bleu für einen kombinierten Spracherkenner, der die Google Ausgabe verwendet, sofern diese existiert. Existiert diese nicht, wird die erste Nebenhypothese von Google verwendet. Sollte diese allerdings auch fehlen, wird die Haupthypothese von IBM Watson verwendet.
  4. Wortfehlerrate, TERp und Bleu für Google, wenn aus den 5-Best ein Verwirrungsnetzwerk gebildet und dann die jeweilige Alternative mit der höchsten Übergangswahrscheinlichkeit ausgewählt wird.
  5. Wie 4. allerdings für Watson.
  6. Verwendet den Spracherkenner aus 3. für eine Analyse wie in 4.
  7. Bildet Verwirrungsnetzwerke aus den 5-Best von Google und Watson und bewertet diese wie in 4.
  8. Wie 7., allerdings werden die Konfidenzen vor IBM Watson mit 0.7 multipliziert.

- `-o:` Versucht, Gewichte und Exponenten für die Bewertungsmodule von Revise zu finden, so dass die Wortfehlerrate minimiert wird. Hier wird die Differenzialevolutionsmethode von *Scipy* [JOPo01] verwendet. Scipy ist eine Pythonbibliothek, die eine Reihe von wissenschaftlichen Werkzeugen bereitstellt.

  Diese Option verwendet die *Höchster Wert* Reduktion und *Durschnitt* Fusion von Revise (siehe Abschnitt 6.2.2.4).

- `-t:` Gibt die Wortfehlerrate aus, die resultiert wenn aus den in `-g` generierten Verwirrungsnetzwerke eine Ausgabe erstellt wird, nur anhand der Übergangswahrscheinlichkeiten der Alternativen. Dieser Wert weicht generell ab von den mit `-b` berechneten, da bei `-b` im Gegensatz zu `-g` keine Lemmatisierung und Wortverschmelzung stattfindet.

  Diese Option verwendet die *Höchster Wert* Reduktion und *Durchschnitt* Fusion von Revise.

- `-u:` Identisch zu `-o` allerdings wird hier der Pegasos-Lernalgorithmus angewendet. Dieser Lernalgorithmus wurde mithilfe von *Sofia-ML* [Scu10] eingebunden, einem Werkzeug zum Training von Rängen.

  Diese Option verwendet die *Höchster Wert* Reduktion und *Summe* Fusion von Revise (siehe Abschnitt 6.2.2.4).

- `-k` ***num*:** Nur in Verbindung mit `-o` oder *-u* verwendbar. Wechselt die Optimierung in den Kreuzvalidierungsmodus. Dies bedeutet, dass die Verwirrungsnetzwerke in *num* Gruppen äquivalenter Größe geteilt werden. Jede dieser Gruppen wird dann durchlaufend als Testgruppe verwendet. Alle anderen Verwirrungsnetzwerke der Trainingsgruppe werden für eine Optimierung verwendet, die der aus Option `-o` gleicht. Anschließend werden die Wortfehler für die Testgruppe mit den für die Trainingsgruppe berechneten Gewichten und Exponenten ermittelt.
- `-f` ***query1;query2;...*** Akzeptiert eine durch Strichkomma getrennte Liste von SQL-Abfragen, die verwendet werden können, um die mit `-u` und `-o` analysierten Verwirrungsnetzwerke zu beschränken. Beispielsweise beschränkt „length(speech_file) < 20“ die Analyse auf Verwirrungsnetzwerke, deren Quellsprachdatei einen Namen von maximal 19 Zeichen Länge hat. Das Schema der relevanten Tabelle ist in Tabelle 8.1 abgebildet.
- `-w` ***fac*** Nur mit `-g` verwendbar. Multipliziert die Konfidenzen des IBM-Watson-Spracherkenners vor der Generierung der Verwirrungsnetzwerke mit *fac*.

Baseline for Google -> wer: 0.18929597701149425, terp: 0.18893678160919544, bleu: 0.6926643081144135 (real -> wer: 0.2955680399500624, terp: 0.2952559300873908, bleu: 0.6018656160394903)
Baseline for Watson -> wer: 0.27340823970037437, terp: 0.2715355805243446, bleu: 0.605956245711492
Baseline for Google or Watson -> wer: 0.21379525593008739, terp: 0.21317103620474406, bleu: 0.6642584630379552
Baseline for Google + CN -> wer: 0.1903735632183908, terp: 0.19001436781609196, bleu: 0.6894730959218355 (real -> wer: 0.29650436953807735, terp: 0.29619225967540586, bleu: 0.5990927275425683)
Baseline for Watson + CN -> wer: 0.27902621722846443, terp: 0.2771535580524345, bleu: 0.5917113941352744
Baseline for Google or Watson + CN -> wer: 0.2166042446941324, terp: 0.21598002496878901, bleu: 0.6576739828516402
Baseline for Google and Watson + CN -> wer: 0.2075530586766542, terp: 0.20724094881398245, bleu: 0.6696439075947133
Baseline for Google and 0.7 Watson + CN -> wer: 0.2078651685393258, terp: 0.2075530586766542, bleu: 0.6696992831577242

Abbildung 6.15: Options -b Beispielausgabe

6.4 Implementierungsbesonderheiten

Für die Erstellung dieser Arbeit mussten bei der Implementierung einige praktische Lösungen für auftretende Probleme gefunden werden. Diese Probleme und deren Lösungen werden in diesem Abschnitt behandelt.

6.4.1 Generierung der Verwirrungsnetzwerke

Die Übergangswahrscheinlichkeiten, die bei der Verwirrungsnetzwerkgenerierung errechnet werden, hängen im Wesentlichen von zwei Faktoren ab: Der Konfidenz für eine Hypothese sowie der Ähnlichkeit der Hypothesen.

Diese Tatsache bringt bei Google und Watson mehrere Probleme: Das Erste Problem ist, dass die Spracherkenner nicht immer die gleiche Anzahl an Hypothesen zurückgeben. Dies ist deshalb problematisch, da die Ausgabehypothesen der Spracherkenner meist sehr

ähnlich sind, zwischen den Spracherkennern aber größere Unterschiede auftreten können. Wenn keine Anpassung stattfindet hat dies zur Folge, dass die Ausgaben des Spracherkenners mit der größeren Anzahl an Ausgaben im Verwirrungsnetzwerk überbewertet werden. Um diese Problematik zu vermeiden, wird die Haupthypothese des Spracherkenners mit weniger Hypothesen so oft dupliziert, dass beide Spracherkenner die gleiche Anzahl an Hypothesen besitzen, wobei die Obergrenze für die Hypothesenzahl auf 5 gesetzt wurde, um diesen Schritt möglichst einzusparen.

Ein weiteres Problem ist, dass beide Spracherkenner nur für ihre Haupthypothese eine Konfidenz ausgeben, aber eine Konfidenz für alle Hypothesen benötigt wird. Um dies zu beseitigen, wird die Konfidenz einer Hypothese ohne Konfidenz auf den Wert der niedrigsten Konfidenz unter den Hypothesen gesetzt. Dieses Verfahren ist relativ willkürlich, wurde aber gewählt, da es verwertbare Verwirrungsnetzwerke zu liefern scheint.

6.4.2 Zwischenspeicherung

Die in diesem Abschnitt vorgestellten Werkzeuge erzeugen eine große Datenmenge und müssen eine Vielzahl an Anfragen an Spracherkenner und Googles Suchmaschine senden. Das hat zur Folge, dass eventuell hohe Kosten anfallen oder die Verarbeitungszeit sehr lange ist, sollten keine Ergebnisse zwischengespeichert werden.

Aus diesen Gründen werden die Ergebnisse der Anfragen an die Spracherkenner, alle Anfragen an die Google-Suchmaschine, alle Ähnlichkeitswerte für das *WordNet*-Bewertungsmodul sowie alle anderen Anfragen an *WordNet* zwischengespeichert.

7. Evaluation

Ob, beziehungsweise wie gut die in Kapitel 5 beschriebenen und in Kapitel 6 implementierten Methoden zur Erfüllung der ihres Ziels geeignet sind, wird in diesem Kapitel beschrieben. Zur Erinnerung: Das Ziel dieser Arbeit war die Aufbereitung von Spracherkennerausgaben, wobei die Implementierung auf die Verbesserung der Wortfehlerrate beschränkt wurde.

7.1 Referenzspracherkenner

Neben den absoluten Daten ist vor allem ein Vergleich mit dem derzeitigen besten Spracherkenner interessant. Die Bestimmung des besten Spracherkenners bezüglich der Wortfehlerrate wurde schon von Paskaran in seiner Arbeit zur „Evaluation unterschiedlicher Spracherkennungssysteme in der Domäne Humanoide Robotik“ durchgeführt [Pas15]. Dabei hat sich herausgestellt, dass für diese Domäne der Spracherkenner von Google die besten Ergebnisse erzielt, weswegen dieser hier als Gold Standart angesehen wird.

7.2 Vergleichsgrundlage

Neben einem Referenzspracherkenner werden zur Evaluation der Daten noch ein Korpus bestehend aus Sprachdaten sowie deren Transkriptionen benötigt. Außerdem muss sich auf eine oder mehrere Metriken verständigt werden, anhand derer die Ausgaben der Spracherkenner verglichen werden können.

7.2.1 Korpus

Als Datengrundlage wird der in Abschnitt 4.3 beschriebene Korpus verwendet. Dieser enthält Sprachdateien und die zugehörigen Referenztranskriptionen für einige Szenarien aus der Domäne der Roboterprogrammierung. An den Referenztranskriptionen wurden allerdings einige Dinge verändert, um die Evaluation besser verständlich zu machen:

1. Alle Transkriptionen wurden auf amerikanisches Englisch geändert, da die Spracherkenner die amerikanische Schreibweise der Worte ausgeben.
2. Es wurden alle Verzögerungslaute entfernt. Diese Maßname wurde durchgeführt, da die verwendeten Spracherkenner keine Verzögerungslaute ausgeben. Deren Inklusion würde die Wortfehlerraten erhöhen, ohne die Informationsmenge der Transkriptionen zu steigern.

3. Es wurden einige Schreibfehler entfernt und Schreibweisen vereinheitlicht zum Beispiel „dish washer“ → „dishwasher“.

Auch wenn diese Änderungen die Wortfehlerrate von den Spracherkennern sowie von *Revise* auf gleiche Weise verschlechtern, sind diese dennoch wichtig, da sonst die relative Änderung der Wortfehlerrate verzerrt wird.

7.2.2 Metriken

Das Hauptziel dieser Arbeit war die Verbesserung der Wortfehlerrate. Somit ist es naheliegend, diese als Vergleichsgrundlage der Spracherkennerausgaben zu verwenden, allerdings gibt es noch andere Metriken, deren Betrachtung auch interessant ist. Neben der Wortfehlerrate können beispielsweise Metriken aus dem Bereich der automatischen Übersetzung eingesetzt werden, wenn man die automatische Spracherkennung als Art der Übersetzung ansieht, die die Aufnahme als Quellsprache und die Referenztranskription als Zielsprache auffasst. Zwei solcher Metriken werden hier angewandt:

- **TERp:** Die Übersetzungsfehlerrate (engl. Translation Error Rate-Plus) kurz TERp wurde von Snover et al. entworfen und weiterentwickelt [SDS+06, SMDS09]. Sie ähnelt der Wortfehlerrate, allerdings können mehrere Referenztexte verwendet werden. Zusätzlich werden bei TERp Synonyme und abweichende Flexionen akzeptiert. Diese Metrik eignet sich gut für die Spracherkennung, da bestimmt werden kann, ob die richtigen Worte erkennt wurden, unabhängig von deren Flexion.
- **Bleu:** Die Bleu Metrik (von „**Bi**-**L**ingual **E**valuation **U**nderstudy“) ist eine N-Gramm-Metrik [PRWZ02]. Bewertet wird hier welcher Anteil der N-Gramme richtig erkannt wurde, somit sind höhere Werte für diese Metrik besser. Für den 1-Gramm Fall bedeutet dies beispielsweise, dass der Satz „Saft Orangen Der“ zu dem Referenzsatz „Der Orangen Saft“ einen Bleu-Wert von 1.0 erreicht (zum Vergleich wäre die Wortfehlerrate $\frac{2}{3}$). Diese Metrik ist somit vor allem dann interessant, wenn sichergestellt werden soll, dass die richtigen N-Gramme erkannt werden, unabhängig von deren relativer Lage.

Sowohl diese Metriken als auch die Wortfehlerrate haben allerdings das Problem, dass sie für jeden Eintrag des Korpus getrennt berechnet werden müssen und somit für jeden Eintrag getrennte Ergebnisse liefern. Da sich die Vielzahl an Werten, die dies erzeugen würde nur schwer zu verstehen sind, wird hier immer der Durchschnittswert der jeweiligen Metrik auf dem verwendet. Ein einfacher Durchschnitt würde allerdings das Ergebnis verzerren, weswegen der Durchschnitt mit der Länge des Referenzsatzes gewichtet wird.

7.2.3 Konfiguration der Werkzeuge

Da die implementierten Werkzeuge modular und konfigurierbar sind, muss vor der Evaluation dieser sich auf eine Konfiguration von Modulen und Einstellung geeinigt werden. Alle folgenden Konfigurationsentscheidungen sind fest in der Implementierung von *ReviseAnalyser* verankert.

7.2.4 MultiASR

MultiASR verwendet die Google und IBM-Watson Spracherkenner, wobei für die Evaluation eine eigene Implementierung der IMultiASR Schnittstelle geschrieben wurde. Diese verwendet zwei Instanzen der in Abschnitt 6.1 beschriebenen Implementierung von IMultiASR, um die Ausgaben von Google und IBM-Watson getrennt voneinander zu erhalten. Die Gründe hierfür sind in Abschnitt 6.4.1 dargelegt.

7.2.5 ConfusionNetworkBuilder

ConfusionNetworkBuilder kann mit einem Faktor konfiguriert werden, mit dem die niedrigste Konfidenz unter allen Eingabehypothesen multipliziert werden soll, bevor diese als Konfidenz für eine Hypothese ohne Konfidenz verwendet wird. Dieser Faktor wurde für die Evaluation wurde aus den in Abschnitt 6.4.1 beschriebenen Gründen auf 1 gesetzt.

7.2.6 Revise

Für Revise gibt es einen großen Konfigurationsspielraum, da neben der Wahl der in Abschnitt 6.2.2.4 beschriebenen Bewertungsverschmelzung die Ausgabe stark von den gewählten Vorverarbeitungs- und Bewertungsmodulen abhängt. Verändert ein Modul das Verwirrungsnetzwerk, so ist selbst die Registrierungsreihenfolge dieser Module ergebnisrelevant.

Für diese Evaluation werden alle in Abschnitt 6.2.2 beschrieben Module verwendet, bis auf das Google Bewertungsmodul. Dieses wurde nicht eingebunden, da die große Menge an Suchanfragen mit hohen Kosten verbunden ist.

Die Registierungsreihenfolge ist im Folgenden aufgelistet, wobei nur Module explizit gelistet sind, die das Verwirrungsnetzwerk verändern können.

1. Lemmatisierungsmodul
2. Wortverschmelzungsmodul
3. Domain-Bewertungsmodul
4. Restliche Bewertungsmodule

7.3 Evaluationsergebnisse

Die Evaluationsergebnisse werden in diesem Abschnitt vorgestellt. Hierbei wurden mehrere Teilmengen des Korpus, als auch der Korpus als Ganzes als Datengrundlage verwendet. Diese Unterscheidung ist, wie später offensichtlich werden wird, sinnvoll, da das Verhalten des Systems stark abhängig von seinen Eingabedaten ist.

Die Ergebnisse der Evaluation hängen stark von den Exponenten und Faktoren ab, die für die Bewertungen der Alternativen der Verwirrungsnetzwerke gewählt wurden. Es müssen also ein möglichst guter Exponent und Faktor für jede Bewertung gefunden werden. Allerdings kann diese Optimierung nicht auf der gleichen Teilmenge des Korpus stattfinden, die auch zur Evaluation verwendet wird, da sonst keine allgemein gültigen Ergebnisse gefunden werden können. Aus diesem Grund muss der Korpus in zwei Teilmengen geteilt werden, einer, auf der die Optimierung durchgeführt wird und einer weiteren, auf der die Ergebnisse überprüft werden. Diese Teilmengen werden im weitern Verlauf als Trainingsmenge und Testmenge bezeichnet.

Da der Korpus aber sehr klein ist, wird er hier nicht in zwei Mengen geteilt, sondern es wird ein Verfahren Namens Kreuzvalidierung angewendet. Bei diesem wird der Korpus in k Teilmengen äquivalenter Größe geteilt.[1] Anschließend wird reihum eine der Teilmengen als Testmenge verwendet und der Rest als Trainingsmenge. Auf diese Weise kann der ganze Korpus als Testmenge verwendet werden, ohne die Ergebnisse zu verfälschen. Für diese Evaluation wurde $k = 10$ gewählt.

[1] Sollte der Korpus keine durch k ohne Rest teilbare Menge an Einträgen enthalten, wird der Rest so aufgeteilt, dass die Menge mit den meisten Einträgen maximal 1 Eintrag größer ist als die mit den wenigsten.

Als Vergleichsgrundlage dient der Gold-Standart Spracherkenner von Google. Da dieser allerdings Unzuverlässigkeitsprobleme hat, wird zusätzlich die Kombination aus dem Spracherkenner von Google mit dem IBM Watson Spracherkenner herangezogen. Bei dieser wird die Haupthypothese von IBM Watson verwendet, sofern keine Hypothesen von Google vorliegen.

7.3.1 Korpus komplett

Die Evaluationsergebnisse für den kompletten Korpus sind in Tabelle 7.1 dargestellt. Aufgelistet sind sowohl Wortfehlerrate (WER) als auch TERp und Bleu für verschiedene Spracherkenner, wobei der Begriff Spracherkenner hier verschiedene Weiterverarbeitungsmethoden der Spracherkennerrohausgaben, inklusive von dem hier implementierten *Revise*, miteinschließt.

In den aufgelisteten Werten wurden fehlende Spracherkennerausgaben als leere Ausgaben interpretiert. Eventuell Werte in Klammern geben den entsprechenden Wert an, wenn fehlende Ausgaben ignoriert werden. Allerdings wird der nicht bereinigte Wert als Referenzwert angenommen, da dies der in der Praxis relevante Wert ist.

Die Spracherkenner sowie deren verwendete Daten im Einzelnen sind die Folgenden:

1. **Google:** Haupthypothese des Google Spracherkenners; existiert keine Haupthypothese aber eine Nebenhypothese, so wurde diese verwendet (dieser Umstand tritt nur bei diesem Spracherkenner auf).
2. **IBM Watson:** Haupthypothese des IBM Watson Spracherkenners
3. **Google oder Watson:** Wie 1., sofern eine Ausgabe existiert, sonst wird stattdessen 2. verwendet.
4. **Google CN:** Aus den 5-best der Google Ausgaben wird ein Verwirrungsnetzwerk gebildet. Aus diesem werden die Alternativen mit der jeweils höchsten Übergangswahrscheinlichkeit gewählt, um eine Hypothese zu bilden, die hier bewertet wurde.
5. **Watson CN:** Wie 4. mit dem IBM-Watson Spracherkenner als Datengrundlage.
6. **Google oder Watson CN:** Wie 4. mit dem Spracherkenner aus 3. als Datengrundlage.
7. **Google und Watson CN:** Wie 4., allerdings werden die 5-best von dem Google und IBM-Watson Spracherkenner zur Erstellung des Verwirrungsnetzwerks verwendet.
8. **Revise CN:** Wie 4., allerdings nach der Verarbeitung von Revise, insbesondere nach der Lemmatisierung und Wortverschmelzung.
9. **Revise Alle:** Hier werden die Verwirrungsnetzwerke anhand aller Bewertungen ausgewertet, wobei die Optimierten Exponenten und Gewichte zur Alternativenauswahl herangezogen werden. Zur Ermittlung dieser Gewichte wurden mit Hilfe von Differenzialevolution ermittelt (*ReviseAnalyser* Option `-o`).
10. **Revise ohne Lesk und Lch:** Da für die Bewertungen durch Lesk und Lch die Exponenten oft negativ und die Gewichte oft null sind ist anzunehmen, dass diese kaum Einfluss auf die Ergebnisse haben. Um diese Vermutung zu bestätigen, wurde die Evaluation ohne diese Bewertungen wiederholt.
11. **Revise Alle Sofia-ML** Wie 9. Allerdings wurden zur Optimierung der Pegasos-Algorithmus verwendet (*ReviseAnalyser* Option `-u`).
12. **Revise Sofia-ML:** Wie 10. unter Ausschluss der Bewertungen durch Lesk und Lch.

Tabelle 7.1: Evaluationsergebnisse für den kompletten Korpus

Spracherkenner	WER	TERp	Bleu
1. Google	**0.2956 (0.1893)**	**0.2953 (0.1889)**	**0.6019 (0.6927)**
2. IBM Watson	0.2734	0.2715	0.6060
3. Google oder Watson	**0.2138**	**0.2132**	**0.6643**
4. Google CN	0.2965 (0.1904)	0.2962 (0.1900)	0.5991 (0.6895)
5. Watson CN	0.2790	0.2772	0.5917
6. Google oder Watson CN	0.2166	0.2160	0.6577
7. Google und Watson CN	0.2076	0.2072	0.6696
8. Revise CN	0.2047	0.2044	0.6746
9. Revise Alle	0.2029	0.2026	0.6785
10. Revise Ohne Lesk und Lch	**0.2035**	**0.2032**	**0.6774**
11. Revise Alle Sofia-ML	0.2172	0.2169	0.6553
12. Revise Sofia-ML	0.2154	0.2150	0.6537

Für den kompletten Korpus zeigt sich, das mit Hilfe von Revise ein besseres Ergebnis erzielen lässt als mit allen anderen Methoden. Wie gut die Verbesserung ist, hängt allerdings ab von den verwendeten Bewertungsmodulen, der ermittelten Gewichte und Faktoren, als auch der Art und Weise, wie die Werte kombiniert werden. Dies ist besonders deutlich beim vergleich zwischen 9., 10., 11. und 12., da alle diese Werte die gleiche Datengrundlage verwenden und sich nur in der Alternativenauswahl unterscheiden.

Insgesamt konnte eine Verbesserung im Vergleich zu den Rohdaten der Spracherkenner erzielt werden, im Bezug auf alle drei Metriken. Aus den sehr ähnlichen WER und TERp-Werten jedes Spracherkenners lässt sich hier erkennen, dass Wortfehler nur sehr selten aus der falschen Wortform resultieren. Aus den Bleu Werten hingegen lässt sich ablesen, dass nicht nur die Wortfehlerrate verbessert, sondern auch eine größere Anzahl an N-Grammen richtig erkannt werden konnten.

Die Verbesserungen an sich sind allerdings recht klein, vor allem wenn 7. und 8. betrachtet werden, weswegen sich der unverhältnismäßig große Arbeitsaufwand nicht rechtfertigen lässt (siehe Abschnitt 7.3.7). Insbesondere war eine Verbesserung gegenüber dem Google-Spracherkenner unter Ausschluss von fehlenden Ausgaben nicht möglich.

Interessant an diesem Ergebnis ist dennoch, dass die Kombination der Spracherkenner über Verwirrungsnetzwerke (7.) bessere Ergebnisse liefert als 3., obwohl in 3. der deutlich schlechtere IBM-Watson Spracherkenner so weit wie möglich vermieden wird, während in 7. seine Ergebnisse Teil jedes Verwirrungsnetzwerks sind. Außerdem bestätigt sich die Vermutung, dass die Bewertungen von Lesk und Lch keinen oder nur einen sehr kleinen Einfluss auf die Ergebnisse haben, da die Werte von 9. und 10. kaum voneinander abweichen.

7.3.2 Korpus professionelles Mikrophon

Betrachtet man nur Aufnahmen, die mit einem professionellen Mikrophon aufgenommen wurden, ergeben sich die in Tabelle 7.2 aufgelisteten Ergebnisse. Diese Unterscheidung ist sinnvoll, da wie Paskaran in seiner Arbeit gezeigt hat, die Spracherkennerausgaben generell besser sind, wenn die Eingabequalität höher ist.

Die Werte an sich zeigen ein ähnliches Bild wie im vorherigen Abschnitt, mit dem einzigen Unterschied, dass die meisten Werte etwas besser sind. Dies ist darin begründet, dass die Durchschnittswortfehlerrate der Spracherkennerausgaben nicht, oder nicht viel besser ist im Vergleich zu den Ergebnissen des ganzen Korpus. Dies ist hauptsächlich den Szenarien

Tabelle 7.2: Evaluationsergebnisse für das bessere Mikrophon

Spracherkenner	WER	TERp	Bleu
1. Google	**0.2893 (0.1763)**	**0.2889 (0.1758)**	**0.6112 (0.7084)**
2. IBM Watson	0.2565	0.2549	0.6138
3. Google oder Watson	**0.1947**	**0.1943**	**0.6840**
4. Google CN	0.2873 (0.1740)	0.2869 (0.1735)	0.6127 (0.7101)
5. Watson CN	0.2647	0.2631	0.5949
6. Google oder Watson CN	0.1959	0.1955	0.6804
7. Google und Watson CN	0.1873	0.1873	0.6881
8. Revise CN	0.1841	0.1841	0.6949
9. Revise Alle	0.1822	0.1822	0.6967
10. Revise ohne Lch und Lesk	**0.1806**	**0.1806**	**0.6998**
11. Revise Alle Sofia-ML	0.2005	0.2005	0.6682
12. Revise Sofia-ML	0.2009	0.2009	0.6614

4 und 5 zuzuschreiben, da hier sowohl Google als auch IBM-Watson sehr schlechte Wortfehlerraten aufweisen. Im Einzelnen sind diese 0.2842 (0.1894) bzw. 0.2590 für Szenario 4 und 0.4176 (0.1755) bzw. 0.3393 für Szenario 5.

7.3.3 Korpus Szenario 1–3 – professionelles Mikrophon

Werden zusätzlich die Szenarien 4 und 5 aus der Analyse ausgeschlossen, zeigt sich das Bild aus Tabelle 7.3. Hier konnten deutlich bessere Ergebnisse erzielt werden, so dass selbst die Ausgabe von Google unter Ausschluss der fehlenden Ausgaben übertroffen werden konnte. Wie sich deutlich erkennen lässt, liegen hier die Wortfehlerraten für Google und Watson deutlich unter denen für Szenario 4 und 5. Vor allem die Differenz zwischen den Google-Werten mit und ohne Ausschluss von fehlenden Ausgaben ist deutlich geringer als bei Szenario 4 und 5. Dies ist darin begründet, dass der Google-Spracherkenner für die Aufnahmen zu Szenario 4 und 5 oft keine Ausgabe liefert und somit oft nur IBM-Watson als Datengrundlage zur Verfügung steht. Die Ursache hierfür ist wahrscheinlich die extreme Varianz in der Sprachlautstärke dieser Aufnahmen.

Tabelle 7.3: Evaluationsergebnisse für das professionelle Mikrophon für Szenario 1–3

Spracherkenner	WER	TERp	Bleu
1. Google	**0.2347 (0.1715)**	**0.2347 (0.1715)**	**0.6614 (0.7161)**
2. IBM Watson	0.2189	0.2189	0.6498
3. Google oder Watson	**0.1807**	**0.1807**	**0.7012**
4. Google CN	0.2333 (0.1699)	0.2333 (0.1699)	0.6600 (0.7146)
5. Watson CN	0.2210	0.2210	0.6379
6. Google oder Watson CN	0.1785	0.1785	0.7011
7. Google und Watson CN	0.1656	0.1656	0.7079
8. Revise CN	0.1620	0.1620	0.7161
9. Revise Alle	0.1584	0.1584	0.7231
10. Revise ohne Lch und Lesk	**0.1569**	**0.1569**	**0.7261**
11. Revise Alle Sofia-ML	0.1605	0.1605	0.7186
12. Revise Sofia-ML	0.1641	0.1641	0.7098

Der Grund hierfür lässt sich wegen der Komplexität der Entscheidungsfindung von *Revise* nicht zweifelsfrei klären. Allerdings lässt sich hier im Zusammenhang mit den vorangegangen Ergebnissen ein Trend erkennen, dass die Ergebnisse von Revise bei besseren Eingabedaten größere Verbesserungen aufweisen. Dies wird vor allem dann ersichtlich, wenn die

Daten mit dem Rest des Korpus verglichen werden, die in Tabelle 7.4 dargestellt sind. Bei diesen konnte zwischen 3. und 10. keine Verbesserung der Wortfehlerrate erzielt werden. Lediglich eine kleine Verbesserung des Bleu Werts konnte erreicht werden, was auf eine höhere Anzahl an richtig erkannten N-Grammen hindeutet, allerdings ist der Korpus zu klein, um hier definitive Angaben zu machen.

Tabelle 7.4: Evaluationsergebnisse für Szenario 4 & 5 sowie Aufnahmen mit unbekanntem Mikrophon

Spracherkenner	WER	TERp	Bleu
1. Google	**0.3421 (0.2045)**	**0.3416 (0.2039)**	**0.5563 (0.6727)**
2. IBM Watson	0.3152	0.3118	0.5724
3. Google oder Watson	**0.2391**	**0.2380**	**0.6360**
4. Google CN	0.3449 (0.2079)	0.3444 (0.2072)	0.5525 (0.6680)
5. Watson CN	0.3234	0.3201	0.5563
6. Google oder Watson CN	0.2457	0.2446	0.6244
7. Google und Watson CN	0.2397	0.2391	0.6403
8. Revise CN	0.2375	0.2369	0.6429
9. Revise Alle	0.2391	0.2386	0.6389
10. Revise ohne Lch und Lesk	**0.2391**	**0.2385**	**0.6423**
11. Revise Alle Sofia-ML	0.2523	0.2518	0.6181
12. Revise Sofia-ML	0.2490	0.2485	0.6120

7.3.4 Gewichte und Exponenten

Da hier eine Kreuzvalidierung und somit nicht nur eine einzige Optimierung der Gewichte und Exponenten geschehen ist, kann keine Aussage darüber getroffen werden, welche Gewichte und Exponenten im Allgemeinen die Optimalen sind. Allerdings lässt sich bei diesen Werten ein Muster erkennen, da diese oft ähnliche Werte einnehmen.

Im Einzelnen bedeutet dies, dass die Bewertungen für das Domänenbewertungsmodul und das Verwirrungsnetzwerkbewertungsmodul regelmäßig die höchsten Gewichte und Exponenten zugewiesen bekommen. Dahinter folgen in der Regel Lin und Jcn, während die anderen Bewertungsmodule eine deutlich größere Varianz aufweisen, allerdings oft sehr kleine oder sogar negative Exponenten aufweisen. Dies bedeutet aber nicht unbedingt, dass diese Werte keinen Einfluss auf die Resultate der Optimierung haben. Denn wird der Durchschnittswert aller verfügbaren Bewertungen gebildet, kann das Vorhandensein einer Bewertung die Alternativenauswahl beeinflussen, auch wenn die Bewertung effektiv null ist. Dies ist vor allem in Hinblick auf die Vielzahl der *WordNet* Bewertungen relevant, da diese immer gehäuft auftreten. So kommt es vor, dass manche Alternativen nur eine Bewertung haben, während andere Alternativen viele verschiedene Bewertungen aufweisen.

7.3.5 Malus für IBM-Watson

Da IBM-Watson hier immer eine schlechtere Wortfehlerrate erreicht als Google (sofern hier Ausgaben vorliegen), ist es naheliegend, dass die N-Bestvon IBM-Watson in einem geringeren Maß in die Erstellung der Verwirrungsnetzwerke einfließen sollte als die von Google. Um dies zu bewerkstelligen, wurden die Konfidenzen von IBM-Watson vor der Verwirrungsnetzwerkerstellung mit 0.7 skaliert (*ReviseAnalyser* Option `-w` 0.7). Dieser Wert wurde gewählt, da er etwa den Unterschied zwischen Google und IBM-Watson in der Wortfehlerrate für den ganzen Korpus widerspiegelt ($0.2565 * 0.7 \approx 0.1763$).

Es hat sich allerdings gezeigt, dass dieser Ansatz nicht zu einer Verbesserung, sondern zu einer kleinen Verschlechterung der Ergebnisse führt, wie in Tabelle 7.5 dargestellt ist,

wobei die Daten der Spracherkenner (1.-6.), da diese den obigen Daten entsprechen[2]. Dies ist auf eine Verschlechterung der Verwirrungsnetzwerke zurückzuführen, da auch 7. leicht schlechter geworden ist.

Tabelle 7.5: Evaluationsergebnisse für den kompletten Korpus mit einem IBM-Watson Malus

Spracherkenner	WER	TERp	Bleu
7. Google und Watson	0.2079	0.2076	0.6697
10. Revise Ohne Lesk und Lch	**0.2041**	**0.2038**	**0.6773**

7.3.6 Google Bewertungsmodul

Das Google Bewertungsmodul wurde Aufgrund der damit verbundenen Kosten von dieser Evaluation ausgeschlossen. Allerdings wurden im Verlauf der Erstellung dieser Arbeit, mit Hilfe des kostenlosen Täglichen Suchkontingents, genug Daten gesammelt um zumindest einen Teil der Verwirrungsnetzwerke des Korpus zu bewerten. Die 23 Verwirrungsnetzwerke die bewertet werden konnten entstammen alle den Aufnahmen mit unbekanntem Mikrophon. Nach Analyse dieses Korpusauschnitts mit und ohne dem Google Bewertungsmodul resultieren hier die in Tabelle 7.6 dargestellten Daten. Hier konnte keine Verbesserung erzielt werden, die Werte wurden sogar schlechter zwischen 10. und 10b.. Die Verschlechterung ist Syndrom des Umstandes das die errechneten Gewichte und Faktoren sind für Google praktisch immer 0, so dass der Wert an sich nicht in die Bewertung einfließt. Allerdings wird durch die Bildung des Durchschnitts aller Bewertungen im weiteren Verlauf die Gesamtbewertung der Alternativen verzerrt.

Tabelle 7.6: Evaluationsergebnisse mit Google Bewertungsmodul

Spracherkenner	WER	TERp	Bleu
1. Google	**0.3634 (0.2343)**	**0.3634 (0.2343)**	**0.5273 (0.6343)**
2. IBM Watson	0.3895	0.3837	0.5260
3. Google oder Watson	**0.3110**	**0.3052**	**0.5654**
4. Google CN	0.3750 (0.2483)	0.3750 (0.2483)	0.5110 (0.6146)
5. Watson CN	0.3953	0.3895	0.5206
6. Google oder Watson CN	0.3227	0.3169	0.5490
7. Google und Watson CN	0.3052	0.2994	0.5721
8. Revise CN	0.3081	0.2023	0.5671
9. Revise Alle	**0.3081**	**0.3052**	**0.5731**
9b. Revise Alle + Google	**0.3140**	**0.3081**	**0.5676**

7.3.7 Laufzeit

Ziel das Parse Projektes, zu dem auch diese Arbeit gehört, ist die Programmierung von Robotern mithilfe von Sprache. Demzufolge muss das gesamte System in annehmbarer Zeit Ergebnisse liefern, da sonst keine Interaktion bei der Programmierung möglich ist. Welche Zeitspannen annehmbar sind wird hier nicht bestimmt, sondern es wird angenommen, dass die Zeitspannen im Sekundenbereich liegen sollten. Die in diesem Kapitel vorgestellten Werte wurden auf einem 2013 MacBook Pro mit 16GB Ram, einem 2,6GHz Intel Core i5 sowie einer SSD erstellt. Die Datenrate der verwendeten Internetverbindung betrug 16MBit Downstream und 1MBit Upstream.

[2]Die Daten für die restlichen Unterteilungen des Korpus finden sich in Tabelle 8.3.

Wie lange die Auswertung einer Sprachdatei mithilfe von *MultiASR*, *ConfusionNetworkBuilder* und *Revise* genau dauert, lässt sich allerdings nicht exakt bestimmen, da diese stark von der Sprachdatei abhängt.

Die Spracherkennung an sich mit *MultiASR* hat im Wesentlichen die gleiche Geschwindigkeit, wie die direkte Verwendung der entsprechenden Schnittstellen, wobei die Umwandlung der Audiodateien einige Sekunden beansprucht. Der größte Zeitanteil wird allerdings vom Hochladen der Audiodatei und der Bearbeitungszeit des Spracherkenners beansprucht. Auch das anschließende Erstellen der Verwirrungsnetzwerke mit Hilfe von *ConfusionNetworkBuilder* bewegt sich im Sekundenbereich.

Die Verarbeitung mit *Revise* kann allerdings deutlich länger dauern. Wie in Abschnitt 6.4.2 werden derzeit alle Ergebnisse des *WordNet*-Bewertungsmoduls zwischengespeichert, da diese extrem lange dauern können. Dabei sind 10 Minuten Laufzeit für die Bewertung eines Wortpaares kein Einzelfall. Sind allerdings alle diese Werte vorberechnet, was durchaus möglich ist da *WordNet* sich nur langsam ändert, kann auch dieser Teil der Analyse im Sekundenbereich abgeschlossen werden. Die Gesamtverarbeitungszeit aller Sprachdaten des Korpus beläuft sich, wenn alle Ergebnisse vorberechnet sind und die Dauer der Spracherkennung ausgeklammert wird[3], auf etwa 30 Minuten, also etwa 20 Sekunden pro Sprachdatei.

Die Bestimmung von Gewichten und Faktoren durch *ReviseAnalyser* ist auch extrem langsam und benötigt für den ganzen Korpus etwa 6 Stunden auf der genannten Maschine. Da dieser Arbeitsschritt aber in der Praxis nur einmal durchgeführt werden muss, ist eine Laufzeit hier akzeptabel, allerdings erschwert die Laufzeit die Validierung von Änderungen an der Methodik von Revise. Problematisch ist dies vor allem, da weitere Bewertungsmodule die Laufzeit zusätzlich erhöhen würden.

7.4 Fazit

Die hier vorgestellten Werkzeuge (10.) verbessern die Wortfehlerrate im Vergleich zum Gold Standart Spracherkenner (1.). Es konnte für diesen Korpus eine Verbesserung von bis zu 30% erreicht werden. Außerdem lässt sich eine Schrittweise Verbesserung in jedem der Arbeitschritte von Revise erkennen. So ist eine Verbesserung von den Ausgaben beider Spracherkenner (1. und 2.) zu der Kombination dieser mit Hilfe von Verwirrungsnetzwerken (7.) über die Aufbereiteten Verwirrungsnetzwerke durch Revise (8.) bis hin zum Resultat der Neubewertung zu erkennen (9.). Anzumerken ist hier das sich die Spracherkennerausgaben und somit die Ergebnisse von Revise schnell ändern können. Beispielsweise konnte, für die Aufnahmen zu Szenario 1–3 mit dem professionellen Mikrophon, zwischenzeitlich eine Wortfehlerrate von 23.34% mit Hilfe des Google Spracherkenners erreicht werden, im Vergleich zum jetzigen Wert von 23.47%. Dies Wert wurde noch vor der Entfernung der Unstetigkeiten aus dem Korpus ermittelt — die tatsächliche Wortfehlerrate müsste also noch geringer gewesen sein.

Allerdings ist auch offensichtlich das nicht alle Bewertungsmodule zur Erreichung dieser Werte relevant waren (siehe 9. und 10.). Zudem zeigt sich am Beispiel von Google das die derzeitige Alternativenauswahl zwar funktioniert, aber bei einer größeren Bewertungsmodulzahl wahrscheinlich aufhören wird zu funktionieren.

Zudem muss das Laufzeitproblem behoben werden, bevor *Revise* als Teil eines Systems zur Programmierung von Robotern eingesetzt werden kann. Durch Vorberechnung aller Abstände in *WordNet* und moderner Hardware sollte dies erreichbar sein.

[3] Dieser Wert kann ohne Verwendung eines anderen Spracherkenners, oder schnellerer Internetverbindung nicht beeinflusst werden

8. Zusammenfassung und Ausblick

Diese Arbeit befasste sich mit der Aufbereitung von Spracherkennerausgaben, insbesondere der Verbesserung der Wortfehlerrate. Zu diesem Zweck wurden Spracherkenner ausgewählt, deren Ausgaben verbessert werden sollten. Hierzu wurden typische Fehler dieser Spracherkenner im Kontext des verwendeten Korpus aufgezeigt.

Um diese Fehler zu beheben, wurde ein Ansatz vorgestellt, in dem die Ausgaben dieser Spracherkenner zusammen dazu verwendet werden, eine einzige, bessere Ausgabe zu erzeugen. Zur Verbindung der verschiedenen Ausgaben wurden Verwirrungsnetzwerke verwendet. Diese wurden dann als Grundlage für eine Neubewertung der Ausgaben der Spracherkenner herangezogen. Die Bewertungen beruhen auf der Verwendung externer Datenquellen wie *WordNet* und einer Ontologie des Roboters und seiner Umgebung.

In der Evaluation konnte gezeigt werden, dass sich anhand dieser Bewertungen neue Ausgaben erzeugen lassen, die den Spracherkennerausgaben im Bezug auf die Wortfehlerrate überlegen sind. Hierbei wurde aufgezeigt, dass der Erfolg dieser Methodik stark von den verwendeten Rohdaten abhängt und im besten Fall eine relative Verbesserung von etwa 10% erreicht werden kann im Vergleich zur Rohausgabe beider Spracherkenner.

Im weiteren Verlauf können weitere Schritte in diesem Bereich erfolgen. Dies kann zum einen durch den Entwurf und die Implementierung weiterer Bewertungsmodule geschehen, wobei einige Ansätze schon in dieser Arbeit vorgestellt wurden. Zum anderen kann eine neue Methode zur Alternativenauswahl gefunden werden. Vor allem die ungleiche Verteilung der Anzahl an Bewertungen pro Alternative erschwert derzeit eine aussagekräftige Bestimmung der Gewichte und Exponenten der verwendeten Bewertungsmodule. Hier wäre es sinnvoll, ein Ranglernverfahren zu finden, das gegenüber dieser ungleichen Verteilung nicht, oder weniger anfällig ist.

Vor der praktischen Anwendung des hier implementierten Systems muss zudem eine Lösung für Laufzeitprobleme des *WordNet* Bewertungsmoduls gefunden werden. Hier sind mehrere Schritte denkbar: Durch Entfernung von Maßen, die nicht zur Verbesserung der Wortfehlerrate beitragen, kann definitiv die Laufzeit gesenkt werden; in Betracht kommen hier vor allem Lch und Lesk. Andererseits können die Abstandsmaße auch vorberechnet werden, was die Auswertung erheblich beschleunigen würde.

Literaturverzeichnis

[Ant09] Anthropomorphic Software LLC: *Anthropomorphic Software*. `http://www.amorphics.com/index.html`. Version: 2009 (zitiert auf Seite 6).

[ARA+06] Asfour, T. ; Regenstein, K. ; Azad, P. ; Schroder, J. ; Bierbaum, A. ; Vahrenkamp, N. ; Dillmann, R.: ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control, IEEE, Dezember 2006. – ISBN 978–1–4244–0199–4 978–1–4244–0200–7, 169–175 (zitiert auf Seite 16).

[BBR01] Bangalore, Srinivas ; Bordel, German ; Riccardi, Giuseppe: Computing consensus translation from multiple machine translation systems, IEEE, 2001. – ISBN 0–7803–7343–X, S. 351–354 (zitiert auf Seite 9).

[Bes95] Best, Catherine T.: Learning to perceive the sound pattern of English. In: *Advances in infancy research* 9 (1995), S. 217–217. – ISSN 0732–9598 (zitiert auf Seite 4).

[BP02] Banerjee, Satanjeev ; Pedersen, Ted: An adapted Lesk algorithm for word sense disambiguation using WordNet. In: *Computational linguistics and intelligent text processing*. Springer, 2002. – ISBN 3–540–43219–1, S. 136–145 (zitiert auf Seite 10).

[Dev15] Developer.att.com: AT&T Speech API | AT&T Developer. (2015). `http://developer.att.com/apis/speech` (zitiert auf Seite 15).

[FFm16] FFmpeg: *FFmpeg*. `http://www.ffmpeg.org`. Version: 2016 (zitiert auf Seite 32).

[Fis97] Fiscus, Jonathan G.: A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In: *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, IEEE, 1997, S. 347–354 (zitiert auf Seite 9).

[Fra15] Framenet.icsi.berkeley.edu: FrameNet. (2015). `https://framenet.icsi.berkeley.edu/fndrupal/` (zitiert auf Seite 27).

[Goo16a] Google Inc.: *Google Custom Search*. `https://cse.google.com/`. Version: 2016 (zitiert auf Seite 28).

[Goo16b] Google Inc.: *Gson*. `https://github.com/google/gson`. Version: 2016 (zitiert auf Seite 48).

[Gü15] 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 4 und 16).

[HFH+09] Hall, Mark ; Frank, Eibe ; Holmes, Geoffrey ; Pfahringer, Bernhard ; Reutemann, Peter ; Witten, Ian H.: The WEKA data mining software: an

update. In: *ACM SIGKDD explorations newsletter* 11 (2009), Nr. 1, S. 10–18. – ISSN 1931–0145 (zitiert auf Seite 29).

[HS05] Honal, Matthias ; Schultz, Tanja: Automatic Disfluency Removal on Recognized Spontaneous Speech-Rapid Adaptation to Speaker Dependent Disfluencies, 2005, S. 969–972 (zitiert auf Seite 10).

[HSO98] Hirst, Graeme ; St-Onge, David: Lexical chains as representations of context for the detection and correction of malapropisms. In: *WordNet: An electronic lexical database* 305 (1998), S. 305–332 (zitiert auf Seite 10).

[IBM16] IBM: *Speech to Text | IBM Watson Developer Cloud*. `https://www.ibm.com/watson/developercloud/speech-to-text.html`. Version: 2016 (zitiert auf Seite 15).

[JC97] Jiang, Jay J. ; Conrath, David W.: Semantic similarity based on corpus statistics and lexical taxonomy. In: *arXiv preprint cmp-lg/9709008* (1997) (zitiert auf Seite 10).

[JM09] Jurafsky, Daniel ; Martin, James H.: *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., 2009. – ISBN 0–13–187321–0 (zitiert auf den Seiten 3, 4 und 7).

[JOPo01] Jones, Eric ; Oliphant, Travis ; Peterson, Pearu ; others: *SciPy: Open source scientific tools for Python*. 2001 `http://www.scipy.org/`. – [Online; accessed 2016-07-12] (zitiert auf Seite 49).

[KL03] Keller, Frank ; Lapata, Mirella: Using the web to obtain frequencies for unseen bigrams. In: *Computational linguistics* 29 (2003), Nr. 3, S. 459–484 (zitiert auf Seite 11).

[Koc15] Kocybik, Markus: *Projektion von gesprochener Sprache auf eine Handlungsrepräsentation*, KIT, Diss., 2015 (zitiert auf den Seiten 1, 3 und 39).

[LC98] Leacock, Claudia ; Chodorow, Martin: Combining Local Context and WordNet Similarity for Word Sense Identification. Version: 1998. `https://books.google.de/books?id=Rehu8OOzMIMC`. In: Fellbaum, Christiane (Hrsg.): *WordNet: An Electronic Lexical Database*. MIT Press, 1998. – ISBN 978–0–262–06197–1 (zitiert auf Seite 10).

[Lev66] Levenshtein, Vladimir I.: Binary codes capable of correcting deletions, insertions and reversals. In: *Soviet physics doklady* Bd. 10, 1966, S. 707 (zitiert auf Seite 41).

[Lin98] Lin, Dekang: An information-theoretic definition of similarity, 1998, S. 296–304 (zitiert auf Seite 10).

[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: *Audio, Speech, and Language Processing, IEEE Transactions on* 14 (2006), Nr. 5, S. 1526–1540. – ISSN 1558–7916 (zitiert auf Seite 10).

[MBS00] Mangu, Lidia ; Brill, Eric ; Stolcke, Andreas: Finding consensus in speech recognition: word error minimization and other applications of confusion networks. In: *Computer Speech & Language* 14 (2000), Nr. 4, S. 373–400. – ISSN 0885–2308 (zitiert auf den Seiten 9 und 23).

[Mil95] Miller, George A.: WordNet: a lexical database for English. In: *Communications of the ACM* 38 (1995), Nr. 11, S. 39–41. – ISSN 0001–0782 (zitiert auf Seite 6).

[MSA+11] Michel, Jean-Baptiste ; Shen, Yuan K. ; Aiden, Aviva P. ; Veres, Adrian ; Gray, Matthew K. ; Pickett, Joseph P. ; Hoiberg, Dale ; Clancy, Dan ; Norvig, Peter ; Orwant, Jon ; others: Quantitative analysis of culture using millions of digitized books. In: *science* 331 (2011), Nr. 6014, S. 176–182 (zitiert auf Seite 28).

[MUN06] Matusov, Evgeny ; Ueffing, Nicola ; Ney, Hermann: Computing Consensus Translation for Multiple Machine Translation Systems Using Enhanced Hypothesis Alignment, 2006 (zitiert auf Seite 9).

[NH05] Nakov, Preslav ; Hearst, Marti: A study of using search engine page hits as a proxy for n-gram frequencies, Citeseer, 2005 (zitiert auf Seite 11).

[Ora16] Oracle: *Introduction to the Service Provider Interfaces*. `https://docs.oracle.com/javase/tutorial/sound/SPI-intro.html`. Version: 2016 (zitiert auf Seite 32).

[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 den Seiten 4, 16 und 53).

[Pat03] Patwardhan, Siddharth: *Incorporating Dictionary and Corpus Information into a Context Vector Measure of Semantic Relatedness*, University of Minnesota, Masters's Thesis, 2003 (zitiert auf Seite 10).

[Phi00] Philips, Lawrence: The Double Metaphone Search Algorithm. In: *C/C++ Users Journal* (2000), Nr. June 2000 (zitiert auf Seite 6).

[PPM04] Pedersen, Ted ; Patwardhan, Siddharth ; Michelizzi, Jason: WordNet:: Similarity: measuring the relatedness of concepts, Association for Computational Linguistics, 2004, S. 38–41 (zitiert auf Seite 45).

[Pri10] Princeton University: *WordNet*. `http://wordnet.princeton.edu`. Version: 2010 (zitiert auf Seite 6).

[PRWZ02] Papineni, Kishore ; Roukos, Salim ; Ward, Todd ; Zhu, Wei-Jing: BLEU: a method for automatic evaluation of machine translation, Association for Computational Linguistics, 2002, S. 311–318 (zitiert auf Seite 54).

[Ren16] Rennie, Jason: *WordNet Perl Module*. `http://qwone.com/~jason/WordNet/`. Version: 2016 (zitiert auf Seite 39).

[Res95] Resnik, Philip: Using information content to evaluate semantic similarity in a taxonomy. In: *arXiv preprint cmp-lg/9511007* (1995) (zitiert auf Seite 10).

[Scu10] Sculley, David: Combined regression and ranking. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2010, S. 979–988 (zitiert auf Seite 49).

[SDS+06] Snover, Matthew ; Dorr, Bonnie ; Schwartz, Richard ; Micciulla, Linnea ; Makhoul, John: A study of translation edit rate with targeted human annotation, 2006, S. 223–231 (zitiert auf Seite 54).

[Shr94] Shriberg, Elizabeth E.: *Preliminaries to a theory of speech disfluencies*, Citeseer, Diss., 1994 (zitiert auf Seite 5).

[SMDS09] Snover, Matthew G. ; Madnani, Nitin ; Dorr, Bonnie ; Schwartz, Richard: TER-Plus: paraphrase, semantic, and alignment enhancements to Translation Edit Rate. In: *Machine Translation* 23 (2009), Nr. 2-3, S. 117–127. – ISSN 0922–6567 (zitiert auf Seite 54).

[SP97] Storn, Rainer ; Price, Kenneth: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. In: *Journal of global optimization* 11 (1997), Nr. 4, S. 341–359 (zitiert auf Seite 48).

[SSSSC11] Shalev-Shwartz, Shai ; Singer, Yoram ; Srebro, Nathan ; Cotter, Andrew: Pegasos: Primal estimated sub-gradient solver for svm. In: *Mathematical programming* 127 (2011), Nr. 1, S. 3–30 (zitiert auf Seite 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 4 und 16).

[Sto02] Stolcke, Andreas: SRILM-an extensible language modeling toolkit, 2002 (zitiert auf Seite 35).

[SW16] Shires, Glen ; Wennborg, Hans: *Web Speech API Specification*. `https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html`. Version: 2016 (zitiert auf Seite 15).

[Win06] Winkler, William E.: Overview of record linkage and current research directions. In: *Bureau of the Census*, Citeseer, 2006 (zitiert auf Seite 41).

[WP94] Wu, Zhibiao ; Palmer, Martha: Verbs semantics and lexical selection, Association for Computational Linguistics, 1994, S. 133–138 (zitiert auf Seite 10).

[YD14] Yu, D. ; Deng, L.: *Automatic Speech Recognition: A Deep Learning Approach*. Springer London, 2014 `https://books.google.de/books?id=rUBTBQAAQBAJ`. – ISBN 978–1–4471–5779–3 (zitiert auf den Seiten xi, 3, 4 und 13).

Anhang

Quelltextausschnitt 8.1: Quelltext zur Wordartableichung

```
private static WordNetQuery.Type getWordNetType(POSTag posTag) {
    WordNetQuery.Type type;
    switch (posTag) {
        case ADJECTIVE:
        case ADJECTIVE_COMPARATIVE:
        case ADJECTIVE_SUPERLATIVE:
            type = WordNetQuery.Type.ADJECTIVE;
            break;
        case ADVERB:
        case ADVERB_COMPARATIVE:
        case ADVERB_SUPERLATIVE:
        case ADVERB_WH:
            type = WordNetQuery.Type.ADVERB;
            break;
        case CARDINAL_NUMBER:
            type = WordNetQuery.Type.NOUN;
            break;
        case INTERJECTION:
            type = WordNetQuery.Type.NOUN;
            break;
        case NOUN:
        case NOUN_PLURAL:
        case NOUN_PROPER_PLURAL:
        case NOUN_PROPER_SINGULAR:
            type = WordNetQuery.Type.NOUN;
            break;
        case PARTICLE:
            type = WordNetQuery.Type.ADVERB;
            break;
        case VERB:
        case VERB_PARTICIPLE_PAST:
        case VERB_PARTICIPLE_PRESENT:
        case VERB_PAST_TENSE:
        case VERB_SINGULAR_PRESENT_NONTHIRD_PERSON:
        case VERB_SINGULAR_PRESENT_THIRD_PERSON:
            type = WordNetQuery.Type.VERB;
            break;
        default:
```

```
            type = null;
            break;
//Don't exist in WordNet
//case CLOSER:
//case COMMA:
//case COLON:
//case CONJUNCTION_COORDINATING:
//case CONJUNCTION_SUBORDINATING:
//case DETERMINER:
//case DETERMINER_WH:
//case PREDETERMINER:
//case EXISTENTIAL_THERE:
//case FOREIGN_WORD:
//case LEFT_PAREN:
//case RIGHT_PAREN:
//case LIST_ITEM_MARKER:
//case POSSESSIVE_ENDING:
//case PRONOUN_PERSONAL:
//case PRONOUN_POSSESSIVE:
//case PRONOUN_POSSESSIVE_WH:
//case PRONOUN_WH:
//case SYMBOL:
//case TO:
//case VERB_MODAL:
    }
    return type;
}
```

Tabelle 8.1: Schema Sprachdaten

Spalte	Typ	Bedeutung
id	INTEGER	Identifikationsnummer (Primärschlüssel)
speaker_id	int(11)	Identifikationsnummer des Sprechers
speech_file	text	Dateiname der zugehörigen Sprachdatei
ref_txt	text	Referenztext
date	varchar(16)	Aufnahmedatum
location	varchar(128)	Ort der Aufnahme
microphone	varchar(256)	Mikrophonname
szenario_id	int(11)	Szenarioidentifikationsnummer
recorder	varchar(128)	Ersteller der Aufnahme
notes	varchar(128)	Notizen

Tabelle 8.2: Pfade durch das Verwirrungsnetzwerk aus Abbildung 6.7

Pfad	Wahrscheinlichkeitssumme
1-a-x	2.1
2-a-x	1.9
1-b-x	1.7
2-b-x	1.5
1-a-y	1.5
2-a-y	1.3
1-b-y	1.1
2-b-y	0.9

Tabelle 8.3: Evaluationsergebnisse mit einem IBM-Watson Malus

Spracherkenner	WER	TERp	Bleu
Professionelles Mikrophon			
7. Google und Watson	0.1880	0.1880	0.6881
10. Revise Ohne Lesk und Lch	**0.1826**	**0.1826**	**0.6975**
Szenario 1–3 – Professionelles Mikrophon			
7. Google und Watson	0.1656	0.1656	0.7079
10. Revise Ohne Lesk und Lch	**0.1605**	**0.1605**	**0.7205**
Szenario 4–5 & unbekanntes Mikrophon			
7. Google und Watson	0.2402	0.2397	0.6405
10. Revise Ohne Lesk und Lch	**0.2424**	**0.2419**	**0.6356**

- Thing
  - Method
    - bring
    - open
    - come
    - empty
    - fill
    - get
    - go
    - hand
    - open
    - pour
    - put
    - show
  - Object
    - cup
    - cupboard
    - dishwasher
    - door
    - fridge
    - juice
    - microwave
    - popcorn
    - table
    - television
    - tv
    - vodka
  - Robot
    - Armar

Abbildung 8.1: Domänenontolgie