Karlsruhe Institute of Technology

Entwicklung eines Dialogagenten für dialogbasierte Programmierung

Masterarbeit
von

Mario Schlereth

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: 13.10.2016 – 12.04.2017

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, 12.04.2017

. . . . . .

(**Mario Schlereth**)

Kurzfassung

Um die natürliche Sprache als gängigstes Kommunikationsmittel des Menschen auch für die Programmierung einsetzen zu können, wurde die Rahmenarchitektur PARSE erstellt. Diese versucht aus eingesprochenen Instruktionen ausführbaren Quellcode zu generieren. Da das Verständnis natürlicher Sprache eine komplexe Aufgabe darstellt, gelingt es PARSE in der Regel nicht, eine Nutzeraussage komplett zu verstehen. In solchen Fällen soll, mit Hilfe des in dieser Arbeit erstellten Dialogagenten, der Nutzer nach dessen Intention gefragt werden, um mit der Antwort die Probleme zu korrigieren. Hierzu behandelt der Dialogagent Fehler des Spracherkenners, des Koreferenzauflösers und des Bedinungsanalysierers. Für die Erweiterbarkeit des Dialogagenten auf weitere Fehlerklassen liegt dessen wichtigster Komponente, dem Dialogmanager, das Entwurfsmuster Zuständigkeitskette zugrunde. Die Evaluation hat gezeigt, dass der Dialogagent zwischen 22 und 55 Prozent der Probleme des Spracherkenners beseitigen kann, abhängig von den Englischkenntnissen der Nutzer und den zugrundeliegenden Problemen. Koreferenzen können in der Hälfte der Fälle korrekt aufgelöst werden. Die Evaluation des Bedingungsanalysierers lieferte aufgrund einer unzureichenden Fragestellung keine belastbaren Ergebnisse über die Lösungsrate.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

In diesem Kapitel wird zunächst die Motivation für die Erstellung dieser Abschlussarbeit dargelegt. Anschließend findet eine Beschreibung der Ziele, des anzufertigenden Dialogagenten, statt. Der letzte Teil dieses Kapitels stellt den Aufbau dieser Arbeit vor.

1.1. Motivation

Für die Interaktion zwischen Mensch und Maschine werden heutzutage vor allem druck- und berührungsempfindliche Schnittstellen eingesetzt. Beispiele hierfür sind Tastaturen, Computermäuse oder berührungsempfindliche Displays. Die am häufigsten verwendete Form der Kommunikation zwischen verschiedenen Menschen ist allerdings die gesprochene Sprache. Um daher die Interaktion für den Menschen mit einer Maschine natürlicher zu gestalten, bietet sich die gesprochene Sprache als Schnittstelle an. Die Kommunikation mit Hilfe natürlicher Sprache ist jedoch eine komplexe Aufgabe und stellt daher Maschinen vor große Herausforderungen. Die Probleme sind zum einen die korrekte Identifikation des gesagten Wortes und zum anderen die Extraktion der Bedeutung aus den einzelnen Aussagen.
Einen Anwendungsfall für die Kommunikation zwischen Mensch und Maschine mit Hilfe natürlicher Sprache stellt die dialogbasierte Programmierung dar. Eine Rahmenarchitektur, die die dialogbasierte Programmierung ermöglichen soll, ist PARSE. Diese versucht durch die Kombination einfacher Basisbefehle komplizierte Aufgaben zu lösen. Ein Beispiel hierfür ist es einem Haushaltsroboter beizubringen, welche Einzelschritte dieser auszuführen hat, um die Wäsche aus der Waschmaschine zu nehmen und in den Wäschetrockner zu legen.
PARSE versucht mit Hilfe eines Spracherkenners die gesagten Wörter zu erhalten und diese in den darauffolgenden Schritten mit Semantik anzureichern. Hierfür besitzt die Rahmenarchitektur verschiedene Komponenten, unter anderem für die Auflösung von Mehrdeutigkeiten, die Bereitstellung von Kontext, die Identifikation von Bedingungen oder die Auflösung von Koreferenzen. Trotzdem gelingt es PARSE in der Regel nicht die Nutzeraussage vollständig zu verstehen.
Ein wesentlicher Grund hierfür ist, dass der Spracherkenner meist nicht alle Wörter richtig erkennt. Dies erschwert den nachfolgenden Bearbeitungsstufen eine korrekte Interpretation der Nutzeraussage. Darüber hinaus können auch die weiteren Komponenten von PARSE neue Fehler durch die Annotation falscher semantischer Bedeutungen erzeugen. Beispiele hierfür sind falsch interpretierte Bedingungen oder die falsche Auflösung eines mehrdeutigen Wortes.

In solchen Fällen bietet sich der Einsatz eines Dialogagenten an, mit dessen Hilfe der Anwender nach seiner tatsächlichen Intention gefragt werden kann. Daneben können die Fragen des Dialogagenten auch zur Korrektur falsch verstandener Wörter eingesetzt werden. Mit Hilfe dieser Fragen sollen bestehende Verständnisprobleme bei der Verarbeitung der ursprünglichen Nutzeraussage beseitigt werden. Gelingt dies, so kann PARSE seine Programmieraufgabe ausführen und das System hat eine neue Aufgabe erlernt.

1.2. Ziele

Das Ziel dieser Abschlussarbeit ist somit die Erstellung eines Dialogagenten für PARSE. Dieser soll Probleme verschiedener Komponenten der Rahmenarchitektur lösen. Der Dialogagent soll zudem eine Architektur aufweisen, die es erlaubt, möglichst ohne Eingriffe in den bestehenden Quellcode neue Komponenten für die Bearbeitung von Fehlern einzubinden. Eine solche Struktur soll zum einen die Erweiterbarkeit des Dialogagenten fördern und zum anderen den Austausch bestehender Komponenten erleichtern. Da PARSE im Kontext der dialogbasierten Programmierung für verschiedene Aufgaben in unterschiedlichen Domänen eingesetzt werden kann, ist ein weiteres wesentliches Ziel des Dialogagenten, dass dieser domänenunabhängig funktioniert. Aufgrund der Tatsache, dass der Dialogagent erst dann aktiv werden soll, wenn die anderen Komponenten der Rahmenarchitektur nicht mehr weiterwissen, kann der Agent als letztes Glied in einer Kette von Bearbeitungsschritten angesehen werden. Dadurch existiert keine weitere Komponente die dessen Resultate überprüft. Somit ist ein essentielles Ziel des Dialogagenten, dass dieser möglichst keine neuen Fehler generiert.

1.3. Aufbau der Arbeit

Dieser Abschnitt beschreibt die Struktur der vorliegenden Abschlussarbeit. Zunächst werden in Kapitel 2 die theoretischen Grundlagen für das Verständnis der weiteren Kapitel erläutert. Anschließend werden in Kapitel 3 die verwandten Arbeiten vorgestellt. In Kapitel 4 findet eine Beschreibung von PARSE statt. Hierbei wird auch auf die einzelnen Komponenten dieser Rahmenarchitektur eingegangen. Kapitel 5 umfasst insbesondere die Analyse von PARSE sowie den Entwurf des daraus abgeleiteten Dialogagenten. In diesem Zusammenhang werden die zu bearbeitenden Problemstellungen und die dafür vorgesehenen Lösungsansätze vorgestellt. Die Implementierung dieser Lösungsansätze wird in Kapitel 6 beschrieben. Die Ergebnisse der im Rahmen dieser Abschlussarbeit durchgeführten Laborstudie werden in Kapitel 7 erläutert. Abschließend findet in Kapitel 8 eine Zusammenfassung der Erkenntnisse statt und es wird ein kurzer Ausblick gegeben.

2. Grundlagen

Die für das Verständnis dieser Abschlussarbeit notwendigen Grundlagen werden in diesem Kapitel vorgestellt. Zunächst findet eine Beschreibung der wichtigsten sprachwissenschaftlichen Grundbegriffe statt. Danach werden die verschiedenen Arten der Dialoginitiative sowie der Dialogführung betrachtet. Abschließend werden der Aufbau und die Komponenten eines Dialogsystems erläutert.

2.1. Sprachwissenschaftliche Grundbegriffe

Dieses Unterkapitel stellt die sprachwissenschaftlichen Grundbegriffe vor, die für das Verständnis dieser Abschlussarbeit relevant sind.

2.1.1. Konstituente

In dieser Arbeit steht eine Konstituente für eine Phrase, also eine Gruppe zusammengehörender Wörter. Beispiele hierfür sind Nominalphrasen, Sequenzen von Wörtern, die ein Nomen und dazugehörende weitere Wörter, z. B. einen Artikel, enthalten oder Verbalphrasen, welche aus einem Verb und mindestens der ersten darauffolgenden Phrase bestehen.[JM09, S. 420ff.]

2.1.2. Entität

Eine Entität ist ein Oberbegriff für abstrakt oder konkret existierende Dinge, z. B. eine grüne Tasse oder eine fiktive Person. Entscheidend ist in diesem Zusammenhang, dass die Objekte existieren können.[Hey16]

2.1.3. Semantische Rolle

Um die Bedeutung eines Satzes zu verstehen, ist es hilfreich den Prädikaten die Konstituenten zuzuordnen, die eine semantische Bedeutung im Kontext dieser Prädikate besitzen. Die semantische Bedeutung einer Konstituente in Bezug auf ein bestimmtes Prädikat wird als semantische Rolle dieser Konstituente bezeichnet.[JM09, S. 704] Da im Kontext dieser Abschlussarbeit gesprochene Sprache verarbeitet wird, kann nicht auf Sätze zurückgegriffen werden. Stattdessen liegt eine Menge an Instruktionen vor. Diese Instruktionen bestehen aus Konstituenten und Prädikaten. Daher kann die für Sätze beschriebene Vorgehensweise auf die vorliegenden Instruktionen übertragen werden.

2.1.4. Syntaktisches Zerteilen

Ein Vorverarbeitungsschritt in der Sprachverarbeitung ist das syntaktische Zerteilen (engl. parsing). Hierbei geht es um die Erkennung von Sätzen aus dem Eingabewortstrom und der Zuweisung einer syntaktischen Struktur zu diesen Sätzen. In der Regel ist diese Struktur ein Baum, welcher den Wörtern Wortarten zuordnet und diese Wortarten immer weiter mit Hilfe von Konstituenten zusammenfasst, bis der komplette Satz in einem Startsymbol, der sogenannten Wurzel, zusammengeführt ist.[JM09, S. 461f.]

2.1.5. Korpus

Ein Korpus ist eine computerlesbare Sammlung von Texten oder Sprachaufnahmen[JM09, S. 119]. Im Kontext dieser Abschlussarbeit handelt es sich stets um eine Sammlung von Sprachaufnahmen.

2.1.6. Ontologie

Wissen kann in Form von Kategorien, denen Dinge oder Personen zugeordnet werden, repräsentiert werden. Die Elemente einer Kategorie können auch Beziehungen untereinander besitzen. Eine Menge von Kategorien bzw. Konzepten, die hierarchisch strukturiert ist und eine Domäne beschreibt, wird als Ontologie bezeichnet.[JM09, S. 606]

2.1.7. Dialogstrategie

Die Dialogstrategie steuert die Gesprächsführung eines Dialogmanagers. Hierfür wird der aktuelle Zustand des Gesprächs inklusive der letzten Nutzeraussage betrachtet und anschließend mit Hilfe einer Entscheidungslogik, der Dialogstrategie, über den nächsten auszuführenden Schritt entschieden.[MRGRD14, S. 344]

2.2. Dialoginitiative

In diesem Abschnitt werden die drei existierenden Arten von Dialoginitiativen vorgestellt. Zudem wird jeweils ausgeführt in welchem Kontext die einzelnen Arten effektiv eingesetzt werden können.

2.2.1. Systemgeführte Dialoginitiative

Hierbei führt das Dialogsystem das Gespräch. Das bedeutet, das System gibt die Reihenfolge der Gesprächspunkte vor und für den Anwender besteht keine Möglichkeit darauf Einfluss zu nehmen. Dieses Konzept bietet sich an, wenn eine Reihe von Informationen vom Nutzer abgefragt werden soll.[DB01] Ein Vorteil dieser Art der Dialoginitiative ist, dass dem System stets bekannt ist, auf welche Frage der Nutzer gerade antwortet. Allerdings kann dies abhängig von der Aufgabenstellung auch ein Nachteil sein, denn der Anwender kann nur auf die aktuell gestellte Frage antworten und nicht, z. B. durch eine umfangreichere Antwort, auf mehrere Fragen gleichzeitig antworten. Ein konkreter Anwendungsfall für die systemgeführte Dialoginitiative wäre die Abfrage eines Benutzernamens sowie des zugehörigen Passworts.[JM09, S. 864f.]

2.2.2. Nutzergeführte Dialoginitiative

Kontrolliert der Nutzer den Gesprächsfluss, d. h. das System reagiert nur auf die Aussagen des Anwenders und stellt keine eigenen Fragen, wird von einer nutzergeführten Dialoginitiative gesprochen. Dieses Konzept ist vorteilhaft, wenn die Anwendung von erfahrenen Nutzern bedient werden soll, die das System soweit kennen, dass sie den Dialog selbst steuern können. Nachteilig ist die nutzergeführte Dialoginitiative für unerfahrene Anwender,

da diese meist nicht in der Lage sind, dem System Anweisungen zu geben, mit denen es etwas Sinnvolles anfangen kann.[DB01] Ein Anwendungsbereich für diese Art der Dialoginitiative sind zustandslose Datenbankabfragesysteme. Dabei stellt der Nutzer dem System einzelne Fragen, welche in SQL-Datenbankfragen überführt werden. Anschließend werden dem Anwender die Ergebnisse ausgegeben.[JM09, S. 864]

2.2.3. Gemischte Dialoginitiative

Bei dieser Art des Dialogs übernehmen abwechselnd das System und der Nutzer die Gesprächsführung. Dieses Vorgehen stellt für viele Anwendungen das gewünschte Gesprächsverhalten dar, da das System den Anwender meist führen soll, um Informationen zu sammeln, der Nutzer aber gleichzeitig in der Lage sein soll, Fehler zu korrigieren oder selbst Informationen zu erfragen.[DB01] Ein Vorteil der gemischten Dialoginitiative ist, dass mit einer ausführlichen Antwort mehrere Fragen auf einmal beantwortet werden können. Allerdings müssen meist viele Regeln definiert werden, um dies zu ermöglichen, was eine Schwäche dieses Ansatzes darstellt. Anwendung findet die gemischte Dialoginitiative u. a. in Flugbuchungssystemen.[JM09, S. 865f.]

2.3. Dialogführung

Dieser Abschnitt stellt zwei verschiedene Arten der Dialogführung vor.

2.3.1. Gelenkter Dialog

Bei diesem Konzept werden die Antwortmöglichkeiten des Nutzers eingeschränkt. Dies kann auf zwei unterschiedliche Arten erreicht werden. Zum einen indem dem Nutzer eine Alternativfrage gestellt wird, z. B. „Wollen sie heute Mittag Pizza, Spaghetti oder Currywurst essen?“, und zum anderen durch eine ja/nein-Frage, z. B. „Wollen sie heute Mittag wie üblich einen Salatteller essen?“.[Sch04][TD11, S. 175] Eine ja/nein-Frage kann hierbei als Spezialform der Alternativfrage angesehen werden, bei der nur eine Alternative verifiziert wird.

2.3.2. Offener Dialog

Der offene Dialog erlaubt dem Nutzer eine Antwort die keinen Einschränkungen unterliegt. Diese Art der Dialogführung wird in der Regel dann eingesetzt, wenn viele Antwortmöglichkeiten existieren und eine Aufzählung aller Alternativen ungeeignet ist.[TD11, S. 175]

2.4. Dialogsystem

Ein rechnerbasiertes System wird als Dialogsystem bezeichnet, wenn es in der Lage ist mit Menschen durch gesprochene Sprache zu interagieren. Hierbei werden meist die folgenden zwei grundlegenden Annahmen getroffen. Der Dialog wird nur zwischen zwei Teilnehmern geführt und alle Interaktionen finden der Reihe nach statt. Letzteres bedeutet, dass das Gespräch in Form von Zyklen abläuft. Ein Teilnehmer sagt etwas während sein Gegenüber zuhört, anschließend interpretiert dieser das Gesagte, trifft eine Entscheidung wie er zu erwidern gedenkt und antwortet. Im Anschluss findet der gleiche Prozess mit vertauschten Rollen statt und danach kann der Zyklus von Neuem beginnen.[Tho09, S. 1ff.]
Um dieses Verhalten zu ermöglichen, benötigt ein Dialogsystem die in Abbildung 2.1 vorgestellten Komponenten. Die Aufgaben und Funktionsweisen der einzelnen Komponenten werden in den folgenden Abschnitten erläutert.

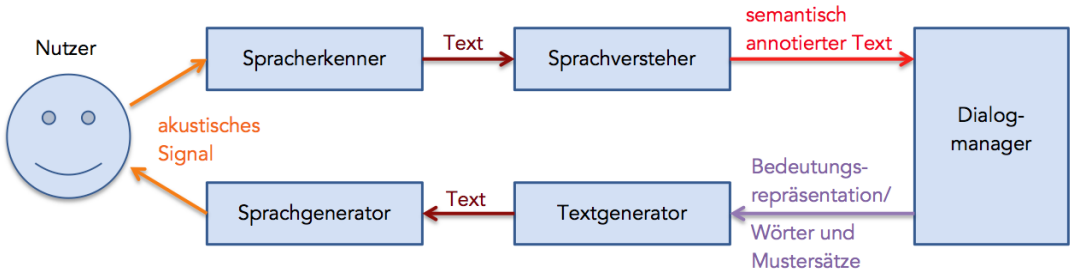


Abbildung 2.1.: Komponenten eines Dialogsystems[Tho09, S. 8]

2.4.1. Spracherkenner

Der erste Arbeitsschritt eines Dialogsystems ist die Übersetzung der Nutzeraussage, beziehungsweise der Tonspur dieser, in Text. Diese Aufgabe wird vom Spracherkenner (engl. automatic speech recognition) übernommen. Gemäß dem aktuellen Stand der Technik wird dies in der Regel mit Hilfe statistischer Modelle durchgeführt. Einer Tonspur wird dabei meist die wahrscheinlichste Sequenz an Wörtern zugeordnet. Allerdings sind moderne Spracherkenner auch dazu in der Lage, eine Liste an Hypothesen auszugeben, die sogenannte N-besten-Liste. In diesem Kontext steht das N für die Anzahl der beigefügten Hypothesen. Jede Hypothese besteht hierbei aus einem Wort und der zugehörigen Wahrscheinlichkeit, dass dieses dem Gesagten auf der Tonspur entspricht. Diese Wahrscheinlichkeit wird auch als Vertrauensniveau (engl. confidence score) bezeichnet. Das Resultat der Spracherkennung hängt jedoch stark von der Schwierigkeit der Aufgabe und der Menge an Trainingsdaten ab. In Fällen, in denen der Nutzer frei Antworten kann und somit die Antwort nicht unbedingt aus einer bestimmten Domäne kommen muss, sind hohe Fehlerraten die Regel.[Tho09, S. 8f.]

2.4.2. Sprachversteher

Für das Sprachverständnis (engl. spoken language understanding), d. h. die semantische Interpretation der Ausgabe des Spracherkenners, ist der Sprachversteher zuständig. Entscheidend ist hierbei nicht die Bedeutung der einzelnen Wörter, sondern welche Intention der Aussage des Anwenders zugrunde liegt. Ob der Nutzer die Aufforderung „Gib mir deine Telefonnummer.“ äußert oder die Frage stellt „Kann ich deine Telefonnummer haben?“ sollte für das Resultat des Spracherkenners keinen Unterschied machen. Das Ergebnis soll in beiden Fällen die Erkenntnis sein, dass der Anwender die Telefonnummer wünscht.[Tho09, S. 9]
Um dies zu erreichen gibt es verschiedene Vorgehensweisen. Eine Möglichkeit sind die sogenannten Sprachakte (engl. speech acts). Dabei werden die Aussagen des Nutzers auf die zugrundeliegenden Aktionen abgebildet und erhalten somit eine semantische Bedeutung. Die Frage nach der Telefonnummer wird zum Beispiel der Aktion „Anfrage(Telefonnummer)“ zugeordnet.[Tho09, S. 9f.]
Eine andere Vorgehensweise zum Verständnis natürlicher Sprache besteht in der Definition semantischer Grammatiken. Mit Hilfe dieser werden die vom Spracherkenner ausgegebenen Wörter mit semantischen Etiketten (engl. labels) versehen. Dadurch lassen sich im nächsten Schritt, z. B. die für eine Flugbuchung notwendigen Informationen, aus den annotierten Sätzen extrahieren. Neben der Definition und Anwendung von Grammatiken, gibt es auch noch statistische Verfahren, um die Wörter mit semantischen Etiketten zu kennzeichnen.[JM09, S. 858ff.]

2.4.3. Dialogmanager

Der Dialogmanager erhält die Ergebnisse des Spracherkenners sowie des Sprachverstehers und erzeugt hieraus und aus dem bisherigen Verlauf des Gesprächs den aktuellen Systemzustand. Aufgrund dieses Zustands trifft der Dialogmanager anschließend die Entscheidung, wie eine adäquate Reaktion auszusehen hat und initiiert die Antworterzeugung durch den Text- und Sprachgenerator. Somit kontrolliert der Dialogmanager die Struktur des Gesprächs.[Tho09, S. 11][JM09, S. 863]
Da der Fokus dieser Abschlussarbeit auf der Erstellung eines Dialogagenten, welcher einen Dialogmanager enthält, liegt, werden die verschiedenen Arten an Dialogmanagern sowie die zugehörigen Architekturen ausführlich in Kapitel 3 dargestellt.

2.4.4. Textgenerator

Um die Ausgabe des Dialogmanagers für den Menschen verständlich zu machen, hat der Textgenerator die Aufgabe, die syntaktische Struktur der Antwort sowie die zu verwendenden Wörter festzulegen. Hierfür existieren zwei verschiedene Ansätze. Eine Möglichkeit ist die Verwendung von vordefinierten Mustersätzen mit Platzhaltern. Diese Sätze geben bereits die Syntax der Antwort vor und der Bezug zum aktuellen Kontext wird durch das Einfügen passender Wörter in die Platzhalter erreicht. Die einzusetzenden Wörter sind bereits in der Ausgabe des Dialogmanagers enthalten und können somit direkt übernommen werden. Der folgende Mustersatz verdeutlicht das Prinzip: „Um welche Uhrzeit wollen Sie in STADT_PLATZHALTER abfliegen?“ STADT_PLATZHALTER kann hierbei durch jede beliebige Stadt z. B. München ersetzt werden, sodass die Frage am Ende lautet: „Um welche Uhrzeit wollen Sie in München abfliegen?“ Beim zweiten Ansatz gibt der Dialogmanager keine Wörter vor, sondern die Bedeutung der Ausgabe. Der Textgenerator erzeugt hieraus anschließend Sätze und gibt diese dann an den Sprachgenerator weiter.[JM09, S. 861f.]

2.4.5. Sprachgenerator

Der Sprachgenerator (engl. speech syntheszier) hat die Aufgabe, die vom Textgenerator erzeugten Texte in akustische Signale zu übersetzen. Dies erfolgt in zwei Schritten. Zunächst wird der Text in eine phonetische Repräsentation überführt und anschließend wird diese in Form von hörbaren Schallwellen ausgegeben.[JM09, S. 283f.]

3. Verwandte Arbeiten

In diesem Kapitel werden verschiedene Arten von Dialogmanagern vorgestellt. Hierfür werden zuerst die entsprechenden Charakteristiken aufgezeigt und anschließend jeweils zwei Realisierungen beschrieben. Es wird darüber hinaus grundsätzlich zwischen regelbasierten und statistischen Dialogmanagern unterschieden, deren Eigenheiten zu Beginn des entsprechenden Unterkapitels ausgeführt werden.

3.1. Regelbasierte Dialogmanager

Für alle regelbasierten Dialogmanager wird die Gesprächsstrategie händisch erstellt. Dafür sind in der Regel Experten notwendig, die sich in der entsprechenden Domäne auskennen. Ein Vorteil dieser Vorgehensweise ist, dass die Systemausgabe in der Regel vorhersagbar ist[TD11, S. 183]. Allerdings ist keines dieser Systeme in der Lage, mit Unsicherheit bezüglich der Spracheingabe umzugehen[Tho09, S. 18].
Im Folgenden werden nun vier verschiedene Konzepte regelbasierter Dialogmanager vorgestellt. Hierbei steigt die Komplexität des zugrundeliegenden Anwendungsfalls von Typ zu Typ an.

3.1.1. Graphbasierte Dialogmanager

Bei dieser Art von Dialogmanagern wird der Gesprächsablauf in Form eines gerichteten Graphen modelliert, der zudem die Eigenschaften eines endlichen Automaten erfüllt. Die Knoten stehen in diesem Fall für die Dialogaktivitäten und die Kanten entsprechen den Bedingungen für den Zustandsübergang. Aktivitäten wären zum Beispiel das Ausgeben eines Textes mit Hilfe eines Sprachgenerators, die Aktivierung des Spracherkennermoduls, das Setzen von Variablen oder die Abfrage externer Datenquellen.[TD11, S. 181] Eine Bedingung, um in den nächsten Zustand zu gelangen, könnte sein, dass ein Wort mit einer gewissen Konfidenz erkannt werden muss, bevor es vom System akzeptiert wird.
Ein Nachteil dieser Dialogmanager ist, dass aufgrund der umfangreichen Graphen und der damit einhergehenden Komplexität der Einsatz von Überzeugungssystemen oder von gemischten Dialoginitiativen in der Regel nicht realisierbar ist. Daher wird bei dieser Art von Dialogmanagern meist der gelenkte Dialog eingesetzt. In einigen Fällen wird auch auf den offenen Dialog zurückgegriffen.[TD11, S. 181]
Der graphbasierte Dialogmanager eignet sich besonders für Aufgaben, bei denen der Anwender nur eingeschränkte Antwortmöglichkeiten besitzt. Dies sind zum Beispiel einfache Banktransaktionen, Fußballergebnis- oder Wetterabfragen[McT98]. Ungeeignet ist diese

Art des Dialogmanagements für Anwendungen, bei denen der Nutzer die Initiative übernehmen kann oder den Dialog führt.[Tho09, S. 17]

Graphbasierter Dialogmanager von McTear

McTear[McT98] beschreibt einen an der University of Ulster entwickelten graphbasierten Dialogmanager, dessen Aufgabe die Ermittlung der Vor- und Nachnamen von Nutzern eines Verzeichnisabfragesystems ist. Die Maximierung der Anzahl erfolgreicher Interaktionen steht hierbei im Vordergrund, auch wenn dies einen höheren Kommunikationsaufwand zwischen dem System und dem Anwender bedeutet. Daher sind bei diesem Dialogmanager für die vollständige Namenserfassung zwischen vier und 30 Interaktionen vorgesehen, statt einer einzigen bei alternativen Ansätzen. Die Fragen des Systems sind in einem Graphen, bestehend aus Wenn-Dann-Bedingungen, enthalten und werden mit zunehmender Entfernung zur Wurzel immer spezifischer. Zum Beispiel wird bei einer nicht verstandenen Eingabe des Vornamens der Nutzer im nächsten Schritt aufgefordert, seinen Vornamen zu buchstabieren. Falls der Dialogmanager nach 30 Interaktionen immer noch keine Lösung gefunden hat, bricht das System die Namensermittlung ergebnislos ab. Eine Nutzereingabe wird als erfolgreich angesehen und somit gespeichert, sobald ein vorher festgelegter Konfidenzwert erreicht beziehungsweise überschritten wird.
Der Ansatz von McTear eignet sich besonders für Anwendungsfälle, in denen nur nach einer kleinen Anzahl an Informationen gefragt wird, da ansonsten die Komplexität des Graphen aufgrund der vielen Zustände schwer handhabbar ist. Die Vorgehensweise von McTear zeigt zudem, wie trotz einer systemgeführten Dialoginitiative ein nutzerfreundliches Frageverhalten erstellt werden kann. Hierfür wird zunächst ohne Hilfestellung nach der gesuchten Information gefragt. Falls eine Antwort nicht befriedigend war, wird stets etwas detaillierter nachgefragt. Dadurch können erfahrene Nutzer eine Aufgabe schnell bearbeiten und unerfahrenen Anwendern wird eine Hilfestellung gegeben.

ETUDE

Der graphbasierte Dialogmanager ETUDE von Pieraccini et al.[PCD+01] versucht mit Hilfe der Integration von universellen Schnittstellenbefehlen die gemischte Dialoginitiative zu ermöglichen. Beispiele für derartige Befehle sind „wiederhole“, „gehe zurück“, „mache rückgängig“ oder „starte von Neuem“. Um „gehe zurück“ zu realisieren, müssen die Knoten, die diesen Befehl zulassen, zum einen die hierfür notwendige Funktionalität implementieren und zum anderen wissen, zu welchem Knoten im Graphen sie zurück gehen sollen. Hierfür ist ein Stapel mit allen bisher durchlaufenen Knoten erforderlich, der bei jeder Aktion aktualisiert wird. Ähnliche Anforderungen müssen für die anderen Befehle implementiert werden.
Darüber hinaus besitzt ETUDE noch zwei weitere Methoden zur Realisierung der gemischten Dialoginitiative. Mit einem „gehe zu“-Aufruf kann der Anwender den aktuellen Dialog verlassen und an eine beliebige Stelle im Graphen springen. Das Ziel ist in diesem Fall, das Gespräch mit dem aufgerufenen Knoten weiterzuführen und nicht zum ursprünglichen Knoten zurückzukehren. Bei der zweiten Methode findet nach der Ausführung eines Unterdialogs wieder eine Rückkehr zum aufrufenden Knoten statt und das Gespräch wird an der ursprünglichen Stelle fortgesetzt.
Bei ETUDE wird die Nutzerfreundlichkeit durch den Einsatz einer gemischten Dialoginitiative erreicht. Hierzu werden Befehle erlaubt, mit deren Hilfe der Anwender u. a. Fragen wiederholen oder Fehler korrigieren kann. Ein Nachteil ist in diesem Zusammenhang allerdings, dass diese Befehle weitere Zustandsübergänge im Graphen darstellen und somit die Komplexität erhöhen. Daher ist auch dieser Ansatz nur auf Anwendungsfälle übertragbar, die sehr wenige Informationen abfragen.

3.1.2. Formularfüllende Dialogmanager

Bei diesem Ansatz repräsentiert das Dialogsystem oder Teile davon ein Formular, welches dynamisch mit Hilfe des Nutzers und weiterer Informationsquellen gefüllt wird. Dadurch können mit einer Antwort des Nutzers mehrere Lücken im Formular auf einmal ausgefüllt werden. Zudem ermöglicht diese Vorgehensweise den Einsatz von gemischten Dialoginitiativen. Darüber hinaus priorisiert das System die noch zu füllenden Lücken und stellt somit die Frage, die den höchsten Nutzen stiftet.[TD11, S. 182] Ein Nachteil von formularfüllenden Dialogmanagern sind die fest vorgegebenen Lücken, welche keine dynamisch erweiterbare Interaktion, wie z. B. die flexible Urlaubsplanung mit mehreren Zwischenstopps, erlauben[RX99].

Mercury

Auch diese Art von Dialogmanagern kann durch endliche Automaten realisiert werden. Allerdings ist dies sehr kompliziert und ineffizient. Daher wurde für das Flugbuchungssystem Mercury von Seneff und Polifroni[SP00] ein Ansatz mit geordneten Regeln gewählt, d. h. die Regeln werden in einer vorgegebenen Reihenfolge durchlaufen. Hierbei wird der Zustand des Systems durch eine Menge an Variablen ausgedrückt, die leer oder gefüllt sein können. Operationen können die leeren Variablen zu einem Formular hinzufügen, welches dann dem Nutzer, in Form einer Frage in natürlicher Sprache, übergeben wird. Fehlen zum Beispiel noch Informationen über die Abflugzeit und die gewünschte Fluggesellschaft, werden diese beiden Variablen dem Formular hinzugefügt und es könnte folgende Frage gestellt werden: „Bitte geben Sie eine Abflugzeit oder die Fluggesellschaft an, mit der Sie fliegen möchten?“ In diesem Fall kann der Nutzer auch auf beide Frageteile gleichzeitig antworten und das System füllt dann beide Variablen mit Werten.
Mercury verwendet über 200 Regeln um seine Funktionalität zu gewährleisten. Hiervon sind lediglich neun für die Fragestellung nach fehlenden Informationen notwendig. Die anderen Regeln werden für die weiteren Aufgaben des Systems benötigt. Allein neun Regeln sind für die Nutzeranmeldung erforderlich, elf für Interaktionen wie Hilfestellungen, Wiederholungen oder Entschuldigungen für nicht zur Verfügung stehende Systemkomponenten, weitere um zu bestimmen, ob der aktuelle Nutzerwunsch im Kontext seiner vorhergehenden Aussagen plausibel ist. Mit 26 Regeln benötigt die Ermittlung der besten Flüge nach einer Flugdatenbankabfrage die meisten Regeln für eine einzelne Aufgabe. Weitere Regeln sind für die Vorbereitung von Nutzerinformationen und Datenbankabfragen, für die Preisberechnung sowie das Versenden der E-Mails mit den Reisedaten erforderlich.
Ein Vorteil von Mercury ist, dass der Nutzer mehrere Fragen gleichzeitig bearbeiten kann. Dadurch können erfahrene Anwender eine Flugbuchung schneller abschließen, als wenn sie jede Information einzeln einsprechen müssten. Ein Nachteil von Mercury ist allerdings die Domänenabhängigkeit. Vor diesem Hintergrund liefert die Information, dass Mercury 73 Prozent der Anfragen korrekt verarbeiten konnte, wenig Aufschluss darüber, ob diese Vorgehensweise auch in anderen Anwendungsfällen funktioniert. Denn dafür müssten die Regeln des Systems neu formuliert werden, wodurch die Evaluation ihre Aussagekraft verliert.

POSSDM

Ein weiterer formularfüllender Dialogmanager ist POSSDM (POSTECH Situation-Based Dialogue Manager) von Lee et al.[LJE+06]. Mit Hilfe dieses Dialogmanagers kann sich der Nutzer in natürlicher Sprache über das Fernsehprogramm informieren. Um dies zu erreichen verwendet POSSDM drei verschiedene Arten von Regeln. Die Situations-Handlungs-Regeln greifen, nach erfolgreicher Kommunikation, auf externe Datenquellen zurück, um dem Nutzer die gewünschten Informationen zur Verfügung zu stellen. Gelingt dies für ein Fernsehprogramm nicht, werden dem Anwender alternative Programme mit Hilfe der

zweiten Regelkategorie zur Auswahl gestellt. Die letzte Regelart entscheidet, ob es sich bei einer Nutzerinitiative um einen neuen Dialog mit einem neuen Kontext handelt oder nicht. Da ein effektives Dialogmanagement eine Menge an Regeln erfordert und dies sehr zeitaufwändig ist, werden bei POSSDM für viele Aussagen vordefinierte Beispieldialoge verwendet. Diese wurden aus einem Korpus, bestehend aus 380 Nutzeraussagen, extrahiert. Für den Fall, dass kein passender Beispieldialog zur Verfügung steht, existieren noch manuell erstellte Regeln, um dem Anwender einen alternativen Vorschlag zu unterbreiten.
Eine Stärke von POSSDM ist der Umgang mit Fehlern. Denn steht ein Fernsehprogramm nicht zur Verfügung, wird dem Nutzer eine Alternative angeboten, statt seine Anfrage zurückzuweisen. Allerdings beträgt die Nutzerzufriedenheit nur 75 Prozent bei gesprochenen Spracheingaben. Dies scheint vor allem dem Spracherkenner geschuldet, denn bei beschriebenen Eingaben liegt dieser Wert bei 93 Prozent. Eine Schwäche von POSSDM ist, dass es auf Beispieldialogen basiert, die von einem Domänenexperten erstellt werden müssen.

3.1.3. Agendabasierte Dialogmanager

Da Gespräche oft aus zusammengesetzten Teildialogen bestehen, enthalten formularfüllende Dialogmanager bei komplexeren Anwendungen teilweise Lücken, die für den aktuellen Kontext nicht relevant sind. Diesem Problem wird mit agendabasierten Dialogmanagern begegnet, bei denen das Gespräch aus einer Agenda besteht, welche nur die benötigten Teildialoge einbindet.[Tho09, S. 17] Dieses Modell ermöglicht die permanente Informationsgewinnung im Verlauf des Dialogs und erlaubt gleichzeitig den Wechsel zwischen verschiedenen zu bearbeitenden Aufgaben während des Gesprächs[TD11, S. 182].

Agendabasierter Dialogmanager von Rudnicky und Xu
Um den Problemen von formularfüllenden Dialogmanagern zu begegnen, haben Rudnicky und Xu[RX99] einen agendabasierten Dialogmanager entworfen. Dieser besteht aus einem Baum mit Knoten, welche jeweils eine Art Agenten repräsentieren, deren Aufgabe es ist, Informationen zu sammeln. In diesem Kontext kann ein Agent als ein kurzes Formular angesehen werden, welches z. B. nur eine Lücke für die Abflugzeit enthält. Um die gewünschte Information zu bekommen, kann der Agent selbst Fragen ausgeben und die für ihn relevanten Informationen aus einer Nutzerantwort extrahieren und damit seine Lücken füllen.
Die Aussage eines Nutzers wird beginnend mit der Wurzel, von links nach rechts und zuerst in die Tiefe gehend durch den Baum geleitet. Sobald ein Agent eine Information verwendet, wird diese für die nachfolgenden Knoten blockiert. Ein Agent kann sich zudem selbst zur Wurzel ernennen und wandert somit an die Spitze des Baumes, der Agenda, und wird dadurch beim nächsten Durchlauf zuerst aufgerufen. Mit Hilfe dieser Vorgehensweise sollen am Ende des Dialogs die Lücken aller Agenten gefüllt sein und die Nutzerintention vollständig im Baum enthalten sein. Damit zudem z. B. eine flexible Urlaubsplanung erfolgen kann, muss der Baum dynamisch erweiterbar sein. Hierfür wurde eine Bibliothek von Unterbäumen angelegt, mit deren Hilfe der Baum, aufgrund von entsprechenden Nutzereingaben, um weitere Unterbäume erweitert werden kann.
Ein Vorteil dieses Ansatzes besteht darin, dass die Anzahl der Lücken dynamisch je nach Umfang des Kundenwunsches veränderbar ist. Dadurch ist das System einfach zu erweitern, ohne die Struktur des Dialogmanagers verändern zu müssen. Allerdings ist das System domänenabhängig implementiert und die Autoren wissen selbst nicht, ob dieser Ansatz auf komplexere Anwendungen erfolgreich übertragbar ist.

RavenClaw
RavenClaw ist eine Rahmenarchitektur von Bohus und Rudnicky[BR03] mit der sich agendabasierte Dialogmanager generieren lassen. Ein hiermit erstellter Dialogmanager besitzt

eine Zweischichtenarchitektur. Die eine Schicht besteht aus einer domänenunabhängigen Dialogmaschine enhält die grundlegenden Unterhaltungsstrategien. Die zweite darunter liegenden Schicht, steuert die domänenspezifischen Dialogspezifikationen. Diese entsprechen einem Baum bestehend aus Dialogagenten und Dialogagenturen. Erstere entsprechen hierbei den Blättern und letztere den inneren Knoten, welche die darunterliegenden Knoten, basierend auf einer logischen und zeitlichen Struktur, steuern. Es existieren vier verschiedene Arten von Blättern. Die Informationsblätter können etwas ausgeben. Mit Abfrageblättern kann man Informationen erbitten. Erwartungsblätter erwarten auch Informationen, aber ohne vorher explizit danach zu fragen. Mit den Domänenoperationsblättern können domänenspezifische Aktionen ausgeführt werden.
Mit Hilfe der Dialogmaschine wird der erstellte Baum durchlaufen. Hierbei wird das gleiche Prinzip, wie in dem bereits vorgestellten agendabasierten Dialogmanager von Rudnicky und Xu, angewendet. Daher sind auch die Bäume der Dialogmanager, die auf RavenClaw basieren, dynamisch erweiterbar.
Aufgrund der Tatsache, dass RavenClaw eine Rahmenarchitektur ist, lässt sich dieses Konzept gut auf andere Anwendungsfälle übertragen. Ein weiterer Vorteil ist die Flexibilität der Rahmenarchitektur bezüglich der Dialoginitiative. Rudnicky und Xu haben den Einsatz von RavenClaw in fünf Werkzeugen kurz beschrieben. Hierbei zeigt sich, dass die Rahmenarchitektur für die systemgeführte, nutzergeführte und gemischte Dialoginitiative eingesetzt werden kann.

3.1.4. Schließende Dialogmanager

Bei dieser Art von Dialogmanagern wird eine Menge logischer Axiome angelegt, welche die Prinzipien des rationalen Verhaltens, der Kommunikation und Kooperation enthalten[TD11, S. 182f.]. Mit Hilfe dieser Axiome werden während des Gesprächsverlaufs Schlussfolgerungen aus dem Gesagten gezogen, um die Interaktion zwischen Mensch und Maschine natürlicher zu gestalten. Nachteilig ist in diesem Kontext die aufwändige Erstellung der Schlussfolgerungsregeln, die auch im Zusammenspiel konsistent sein müssen. Der Vorteil ist, dass dadurch die Anzahl der Interaktionen zwischen den Gesprächspartnern verringert werden kann, was zu einem flüssigeren Dialogablauf führt.

FLoReS
Ein Beispiel für einen schließenden Dialogmanager ist FLoReS (Forward Looking, Reward Seeking) von Morbini et al.[MDS+12]. Als Grundlage dienen Informationszustände, die den bisherigen Gesprächsverlauf und relevante Informationen für den weiteren Dialog enthalten. Diese Informationszustände können durch schließende Regeln und Ereignisse, z. B. Nutzereingaben oder Systemaktionen, verändert werden. In diesem Kontext stehen schließende Regeln für eine Menge vorher definierter Schlussfolgerungen, z. B. wenn ein Nutzer Albträume hat, können daraus Schlafprobleme gefolgert werden, um zusätzliches Wissen zu generieren.
Entsprechend der Dialogstrategie werden bei FLoReS zunächst alle Ereignisse ausgewertet und die Informationszustände gemäß den hieraus resultierenden Aktionen aktualisiert. Danach verändern die schließenden Regeln die Informationszustände so lange, bis diese einen stabilen Zustand erreicht haben. Damit im nächsten Schritt die Aktion mit dem höchsten Nutzen für den weiteren Gesprächsverlauf ausgeführt wird, muss an dieser Stelle der passende Operator aktiviert werden. Ein Operator steht hierbei für einen Teildialog, welcher einem Baum aus System- und Nutzeraktionen und den dazugehörigen Zuständen entspricht.
Die Stärke von FLoReS ist, dass dieser Dialogmanager aufgrund der schließenden Dialogstrategie stärker auf die tatsächlichen Bedürfnisse des Anwenders eingehen kann, als die bisher vorgestellten Dialogmanager. Da hierfür allerdings sehr viele domänenspezifische

Regeln zum Einsatz kommen, lässt sich FLoReS nicht einfach auf andere Anwendungsfälle übertragen.

ARTIMIS
Der Dialogmanager des Agenten ARTIMIS von Sadek et al.[SBP97] implementiert eine formale Theorie der Interaktion, um Schlussfolgerungen zu ermöglichen. Diese Theorie enthält in einer homogenen logischen Rahmenarchitektur die Prinzipien des rationalen Verhaltens, der Kommunikation und Kooperation in Form einer Menge generischer Axiome. Mit Hilfe dieser Rahmenarchitektur wird das System in die Lage versetzt, automatisch zu schließen und somit Überzeugungen und Intentionen zu erkennen. Dadurch soll dieser Dialogmanager mit komplexen Situationen und unvollständigen Eingaben umgehen können.
ARTIMIS wurde eingesetzt, um telefonisch Informationen über das Wetter und Stellenausschreibungen abzufragen. Um dies zu realisieren wurden, neben der homogenen logischen Rahmenarchitektur, noch semantische Repräsentationen für die relevanten domänenspezifischen Attribute, z. B. geographische Angaben, erstellt.
Der Einsatz der schließenden Dialogstrategie verbessert die Interpretation der Nutzeraussage und stellt somit eine Stärke des Systems dar. Allerdings müssen hierfür die domänenspezifischen Informationen in das zugrundeliegende Regelwerk eingefügt werden. Dies ist sehr nachteilig für die Übertragung des Konzepts auf andere Anwendungsfälle.

3.2. Statistische Dialogmanager

Eine Gemeinsamkeit die alle statistischen Dialogmanager teilen ist, dass sie mit selbstverstärkendem Lernen arbeiten und hierfür Trainingsdaten benötigen. Während bei den MDP-Dialogmanagern lediglich die Dialogstrategie statistisch optimiert wird, verwenden die POMDP-Dialogmanager die statistischen Verfahren darüber hinaus, um mit der Unsicherheit der Nutzereingabe umgehen zu können.[Tho09, S. 20ff.]

3.2.1. MDP-Dialogmanager

Für die regelbasierten Dialogmanager muss die Gesprächsstrategie immer manuell erstellt werden. Dies ist sehr aufwändig, schwierig zu warten und bietet meist ein suboptimales Ergebnis, d. h. es existieren andere Strategien, die die gewünschte Aufgabe mit weniger Interaktionen erledigen könnten. Eine Alternative hierzu besteht darin, ein objektives Maß für die Güte der Dialogstrategie einzuführen und dieses dann automatisch zu optimieren. Dieses Maß entspricht einer Belohnungsfunktion die jedem Tupel, bestehend aus einem Zustand und einer Aktion, einen Wert zuweist. Systeme, die diese Funktion einsetzen und versuchen sie zu optimieren werden als selbstverstärkend lernend bezeichnet. Gilt darüber hinaus noch die Markow-Annahme, d. h. die Wahrscheinlichkeit eines neuen Zustands hängt ausschließlich vom vorhergehenden Zustand und der gewählten Aktion ab, so spricht man von einem Markow-Entscheidungsproblem (engl. MDP Markov Decision Process).[Tho09, S. 20ff.]
Trotz der automatischen Optimierung der Gesprächsstrategie müssen die Zustandsübergänge, bei MDP-Dialogmanagern, weiterhin manuell definiert werden. Dies wird mit zunehmender Dialogkomplexität schwieriger.[Tho09, S. 21]

MDP-Dialogmanager von Levin und Pieraccini
Dieses Konzept der Dialogstrategieoptimierung mit Hilfe von MDP wurde zuerst von Levin und Pieraccini[LP97] vorgestellt. Angewendet wurde dieser Ansatz hierbei auf einen formularfüllenden Dialogmanager und AMICA[PLE97], einem Forschungsdialogsystem für

gesprochene Sprache mit einer gemischten Dialoginitiative. Die Realisierung eines MDP-Dialogmanagers wird an dieser Stelle beispielhaft anhand des formularfüllenden Dialogmanagers veranschaulicht.
Der Zustandsraum besteht aus den Lücken des Formulars, die entweder den Zustand gefüllt oder nicht gefüllt besitzen. Die Menge der Aktionen umfasst die Aktionen, die einzelne Fragen (z. B. nach dem gewünschten Monat) oder zusammengesetzte Fragen (z. B. nach dem Datum, enthält Tag, Monat und Jahr) repräsentieren und die Aktion, welche den Dialog beendet und das Formular einreicht. Die Zustandsübergänge stehen für das Füllen der entsprechenden Lücken nach einer Nutzerantwort. Die Kostenfunktion entspricht der Summe der Anzahl der Fragen an den Nutzer, der nicht gefüllten und falsch gefüllten Lücken, multipliziert mit den jeweiligen Kostenparametern. Zusammengesetzte Fragen führen zu einer schnellen Abarbeitung des Formulars, was sich positiv auf die Kosten auswirkt. Da dadurch die Antworten allerdings komplizierter werden und somit der Spracherkenner mehr Fehler produzieren kann, was zu nicht oder falsch gefüllten Lücken führt, repräsentiert die Kostenfunktion ein Optimierungsproblem, mit dessen Hilfe eine optimale Dialogstrategie gefunden werden kann.
Durch die Optimierung der Dialogstrategie wird die Anzahl der Fragen verringert, die dem Anwender gestellt werden müssen. Dies ist ein wesentlicher Vorteil dieses Dialogmanagers. Der Nutzen dieser Vorgehensweise zeigt sich aber vor allem bei Anwendungen, die viele Informationen benötigen. Aufgrund der Komplexität der Optimierung der Dialogstrategie ist dieser Ansatz in solchen Fällen allerdings in der Regel nicht einsetzbar.

MDP-Dialogmanager von Denecke et al.
Ein Problem von MDP-Dialogmanagern besteht darin, dass der Zustands- und Aktionsraum im Verhältnis zu den Trainingsdaten sehr groß ist und daher die Kostenfunktion meist nicht gegen die optimale Strategie konvergiert. Um diesem Problem zu begegnen haben Denecke et al.[DDN05] einen Ansatz entwickelt, der die Kostenfunktion nur über die Menge der häufig besuchten Zustände optimiert. Hierfür wird das System zunächst trainiert und anschließend werden aus der Verteilung der besuchten Zustände die 50 häufigsten herausgenommen, über welche dann optimiert wird. Die Werte der Kostenfunktion für die Zustände, die nicht in dieser Menge enthalten sind, werden anhand von berücksichtigten Zuständen, die „ähnlich“ zu den jeweils unberücksichtigten Zuständen sind, näherungsweise bestimmt.
Um die für diese Vorgehensweise notwendige Abstraktion zu erreichen, müssen die Kostenfunktion und die Aktionen abstrahiert werden, d. h. ähnliche Aktionen werden durch eine übergeordnete Aktion dargestellt. Dies kann zu Problemen führen, wenn sich die zusammengefassten Aktionen in ihrer Charakteristik nicht genug ähneln. Dadurch repräsentiert die abstrahierte Kostenfunktion nicht mehr ausreichend die tatsächliche Kostenfunktion und somit führt die Optimierung der abstrahierten Kostenfunktion zu keiner optimalen Strategie. Eine Lösung hierfür wäre die Einführung weiterer Zustandsvariablen, um die unterschiedlichen Aktionen darstellen zu können.
Eine Stärke dieses Dialogmanagers ist, dass der Zustandsraum für die Optimierung der Dialogstrategie eingeschränkt wird, wodurch dieser Ansatz auch in komplexeren Anwendungsfällen einsetzbar ist. Dennoch muss für jedes Einsatzgebiet die Dialogstrategie erneut mit Hilfe von Trainingsdaten optimiert werden. Dies stellt einen enormen Aufwand dar und ist daher eine Schwäche des Dialogmanagers.

3.2.2. POMDP-Dialogmanager

Da sowohl die Spracherkenner als auch die semantischen Annotierer hohe Fehlerraten aufweisen, besteht für einen Dialogmanager die Notwendigkeit mit Unsicherheit umgehen zu können. Bei den bisher vorgestellten Ansätzen wurde diese Herausforderung durch zu-

sätzliche Zustände, welche für die unsicheren Situationen stehen gelöst. Ein Vorteil dieser Vorgehensweise ist die einfache Integration der zusätzlichen Zustände in die bereits bestehende Rahmenarchitektur. Der Nachteil ist allerdings, dass sich die Bereitstellung einer grundsätzlichen Definition der Zustände der Unsicherheit als schwierig erweist.[Tho09, S. 18]
Ein alternativer Ansatz der dieses Problem beseitigt, besteht in der Erstellung eines Modells, welches die Unsicherheit direkt in die Zustände integriert. Hierfür wird die Umwelt des Systems durch partiell beobachtbare Zufallsvariablen definiert, deren Zustand aus den Beobachtungen geschlossen werden muss. Die den Zufallsvariablen zugrunde liegende Wahrscheinlichkeitsverteilung bietet hierbei eine stichhaltige Repräsentation der Unsicherheit.[Tho09, S. 18]
Die Kombination aus diesem Ansatz und dem Markow-Entscheidungsproblem mit seinem selbstverstärkenden Lernen wird als POMDP (Partially Observable Markov Decision Process) bezeichnet. Allerdings liegt diesem Modell ein leicht abgewandelter Ansatz des Lernens zugrunde. Hierbei wird die Belohnungsfunktion über die Umweltzustände und nicht über die Systemzustände definiert. Dadurch soll das Ergebnis anhand dessen optimiert werden, was der Nutzer tatsächlich haben möchte, statt wie bisher, anhand dessen was das System glaubt, das er haben möchte.[Tho09, S. 22]
Allerdings repräsentieren die Systemzustände nun Wahrscheinlichkeiten und entsprechen somit stetigen Variablen. Dadurch entsteht eine hohe Berechnungskomplexität, was dazu führt, dass ein großer Teil der Forschung darauf verwendet wird, den Berechnungsaufwand zu reduzieren, um Systeme mit komplexeren Dialogen zu ermöglichen. Ein weiteres Problem von POMDP-Dialogmanagern ist die Gestaltung der Modellparameter. Da die Beobachtungs- und Übergangswahrscheinlichkeiten einen großen Einfluss auf die Entwicklung der Systemzustände haben, müssen diese Wahrscheinlichkeiten passend gewählt werden.[Tho09, S. 23]

POMDP-Dialogmanager von Roy et al.
Der POMDP-Dialogmanager von Roy et al.[RPT00] arbeitet mit einem Überzeugungsraum. Jede Überzeugung besteht hierbei aus einer Wahrscheinlichkeitsverteilung über die Menge der Zustände, welche die Wahrscheinlichkeit repräsentiert, dass sich der Nutzer in diesen Zuständen befindet. Die initiale Überzeugung wird bei jeder Nutzerbeobachtung aktualisiert. Da es allerdings aufgrund des hohen Rechenaufwands unmöglich ist, eine optimale Dialogstrategie für nicht triviale Gespräche zu bestimmten, wird mit einer angenähert-besten Strategie gearbeitet. Unter der Annahme, dass Unsicherheit in der Regel domänenspezifisch und lokal ist, wird hierfür der Überzeugungsraum vereinfacht. Die Wahrscheinlichkeit, dass ein Dialogsystem die Befehle das Fernsehprogramm zu wechseln oder Kaffee zu holen deutlich einfacher unterscheiden kann als die Fernsehsender ABC und NBC auf die es umschalten soll, führt zu unterschiedlich hohen Unsicherheiten, was für die Überzeugungsraumvereinfachung ausgenutzt werden kann. Mit dieser Annahme ist es nun möglich eine optimale Dialogstrategie zu berechnen.
Eine Stärke dieses Dialogmanagers ist, dass dieser Unsicherheiten bezüglich der Nutzeraussage direkt für die Optimierung der Dialogstrategie verwendet. Die Evaluation zeigt, dass dieser Ansatz bessere Resultate liefert, als die verglichenen MDP-Dialogmanager. Dennoch sind auch für diesen Dialogmanager domänspezifische Trainingsdaten erforderlich, was eine schnelle Übertragung der Vorgehensweise auf andere Anwendungsfälle erschwert.

POMDP-Dialogmanager von Thomson et al.
Ein anderer Ansatz zur Verringerung der Berechnungskomplexität von POMDP-Dialogmanagern, wird von Thomson et al.[TSY08] vorgeschlagen. Maßnahmen zur Reduzierung des Rechenaufwands sind die Faktorisierung des Systemzustandes gemäß den zu füllen-

den Lücken und die folgenden Annahmen: Die Nutzeraktion hängt ausschließlich von der letzten Systemaktion und der Nutzerintention für die aktuelle Lücke ab. Die Vergangenheit einer Lücke beruht auf der vorhergehenden Vergangenheit dieser Lücke und der gegenwärtigen Nutzeraktion. Die Intentionen eines Nutzers für eine Lücke hängen von den vorhergehenden Werten der Lücke, der letzten Systemaktion und einer Kombination weiterer Nutzerintentionen für andere Lücken ab. Diese Kombination lässt sich in Form eines Bayesschen Netzwerks darstellen.
Um das Lernen von Dialogstrategien mit Hilfe von Bayesschen Netzen zu vereinfachen, werden einzelne Aktionen zu einer Aktion zusammengefasst. Ein Beispiel hierfür ist die Verifizierung eines Lückeninhalts. Das System sollte stets die Alternative für eine Lücke bestätigen lassen, welche die höchste Wahrscheinlichkeit hat, in diese Lücke zu passen, alles andere wäre nicht optimal. Somit ist eine Betrachtung der weiteren Alternativen überflüssig und die dafür vorgesehenen Aktionen werden nicht benötigt. Daher können alle Verifizierungsaktionen für eine Lücke zu einer einzigen Aktion zusammengefasst werden. Die Evaluation dieses Dialogmanagers zeigt, dass die selbstlernende Dialogstrategie bessere Resultate als bisherige Ansätze liefert, selbst wenn beide auf die gleichen Informationen zurückgreifen. Ein Nachteil der Ergebnisse ist allerdings, dass diese lediglich aus Simulationen stammen. Daher sind die Resultate nur eingeschränkt aussagekräftig, da sich Menschen nicht vorhersehbar verhalten.

4. PARSE

In die Rahmenarchitektur PARSE (Programming ARchitecture for Spoken Explanations)[1] soll der Dialogagent eingebunden werden, der in dieser Abschlussarbeit erstellt wird. In diesem Kapitel werden das Konzept, die Architektur sowie die einzelnen Komponenten von PARSE vorgestellt. Zudem wird ein Anwendungsfall beschrieben und der Umfang sowie die Entstehung des vorhandenen Korpuses erläutert.

4.1. Konzept

Bei PARSE handelt es sich um ein agentenbasiertes System, welches die Programmierung mit Hilfe natürlicher Sprache ermöglichen soll. Hierbei lösen die Agenten einzelne Aufgaben, z. B. die Koreferenzauflösung oder die Kontextanalyse. Da die Agenten unabhängig voneinander agieren, können sowohl regelbasierte als auch statistische Ansätze verwendet werden. Der Einsatz evaluationsgetriebener Entwicklung erlaubt die schnelle Integration und Bewertung der einzelnen Agenten. Evaluationsgetriebene Entwicklung bedeutet in diesem Kontext, die kontinuierliche Bewertung des Systems beziehungsweise einzelner Teile hiervon, um den Fortschritt der Entwicklung sicherzustellen. Für das Erreichen von tiefergehendem Sprachverständnis werden verschiedene Wissensdatenbanken eingebunden. Das Wissen des jeweiligen Anwendungsbereichs wird in Form einer Ontologie modelliert und ermöglicht dadurch die Nutzung des Domänenwissens für das Sprachverständnis. Durch diese leicht austauschbaren Ontologien soll PARSE möglichst domänenunabhängig funktionieren. Dies hat den Vorteil, dass bei einem Wechsel des Anwendungsbereichs der Analyseteil des Systems nicht neu implementiert werden muss.[WT15]

4.2. Architektur

Die Architektur von PARSE basiert auf einem graphbasierten Datenspeicher, vgl. Abbildung 4.1. Der Graph repräsentiert hierbei die Spracheingabe und agiert gleichzeitig als Speichermedium für die Agenten (rot dargestellt). Ein Knoten enthält in diesem Zusammenhang ein Wort sowie dessen Attribute. Eine Kante repräsentiert die Beziehung zwischen zwei Knoten. Der Graph kann zudem als Schnittstelle zwischen den einzelnen Agenten angesehen werden, da die Agenten die Ergebnisse der anderen Agenten lediglich als Transformation des Graphen erhalten. Die Eingabeerkennung wird bei PARSE in Form

[1] PARSE https://ps.ipd.kit.edu/180_422.php abgerufen am 03.01.2017

von Modulen gekapselt, was eine einfache Integration verschiedener Vorverarbeitungsstufen (blau dargestellt) ermöglicht. Zur Einbindung von domänenspezifischen Ontologien oder Weltwissen (beides grün eingezeichnet) stellt diese Rahmenarchitektur Schnittstellen bereit.[WT15] In den Nachverarbeitungsstufen (grau eingezeichnet) soll aus dem erzeugten Graph Quellcode generiert werden.

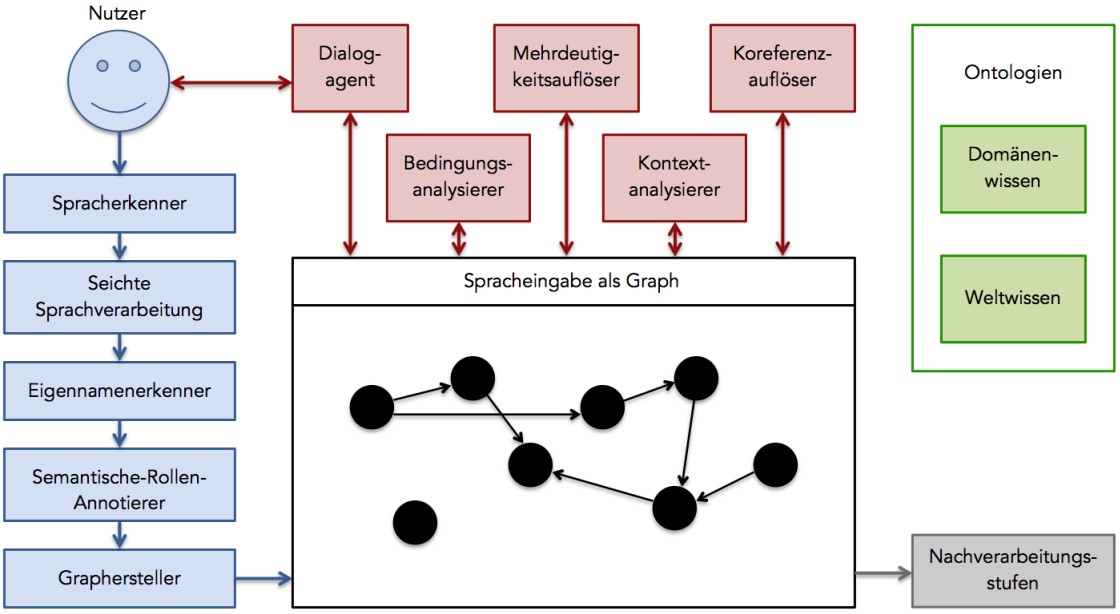


Abbildung 4.1.: Architektur von PARSE

4.3. Vorverarbeitungsstufen

In diesem Unterkapitel werden die aktuell von PARSE eingesetzten Vorverarbeitungsstufen vorgestellt und deren Funktionsweise kurz erläutert, sofern dies noch nicht an einer anderen Stelle geschehen ist.

4.3.1. Spracherkenner

Der Spracherkenner ist die erste Vorverarbeitungsstufe und übersetzt akustische Signale in Text, d. h. die Spracheingabe wird in eine Liste bestehend aus Token überführt. Eine Beschreibung des Spracherkenners ist bereits in Abschnitt 2.4.1 erfolgt. PARSE ermöglicht das Einbinden mehrerer Spracherkenner und kann somit das Resultat der Spracherkennung durch die Kombination der einzelnen Ergebnisse verbessern.

4.3.2. Seichte Sprachverarbeitung

Da sich die gesprochene Sprache nicht an die grammatikalischen Regeln der geschriebenen Sprache hält, können keine sinnvollen Syntaxbäume und Abhängigkeitsgraphen mit Hilfe des syntaktischen Zerteilens, vgl. Abschnitt 2.1.4, erstellt werden. Für diesen Anwendungsfall ist es daher notwendig, auf das seichte Zerteilen (engl. shallow parsing) zurückzugreifen, bei dem der Eingabestrom nicht in Sätze sondern in Instruktionen aufgeteilt wird. Die dabei verwendete Methode zum Aufteilen der Eingabe wird als Phrasenerkennung (engl. chunking) bezeichnet. Diese Methode identifiziert und klassifiziert die flachen, d. h. nicht überlappenden Konstituenten der Eingabe. Allerdings beschränkt sich die Phrasenerkennung auf die Identifikation der Nominal-, Verbal-, Adjektiv- und Präpositionalphrasen.[JM09, S. 484f.] Für die Phrasenerkennung ist es erforderlich, dass

jedem Token bereits die entsprechende Wortart zugeordnet wurde[Koc15]. Daher wird dieser Schritt beim seichten Zerteilen zuerst durchgeführt.
Die Vorverarbeitungsstufe von PARSE, die für das seichte Zerteilen zuständig ist, wird im Kontext dieser Rahmenarchitektur Seichte Sprachverarbeitung (engl. shallow natural language processing) genannt.

4.3.3. Eigennamenerkenner

Mit Hilfe der Eigennamenerkennung (engl. named entity recognition) sollen Benennungen in einem Text aufgespürt und klassifiziert werden. In der Regel handelt es sich um Namen für Orte, Personen oder Organisationen. Generell kann aber auch nach domänenspezifischen Namen z. B. von Proteinen oder Genen gesucht werden.[JM09, S. 759f.]
Der Eigennamenerkenner von PARSE sucht neben den Kategorien für Personen, Orte und Organisationen noch nach sonstigen Namen (engl. miscellaneous names). Unter sonstige Namen fallen in diesem Zusammenhang z. B. Veranstaltungen oder in der englischen Sprache Adjektive wie „Italian“. Somit sind die Entitäten dieser Kategorie vielfältiger Natur.[SD03]

4.3.4. Semantische-Rollen-Annotierer

Beim Annotieren semantischer Rollen (engl. semantic role labeling) wird gemäß der Aussage des Prädikats jeder Satzkonstituente ihre semantische Rolle zugeordnet[JM09, S. 704]. Für jedes in einem Satz vorkommende Prädikat ergänzt der Semantische-Rollen-Annotierer von PARSE hierfür die Token des entsprechenden Satzes um ein Attribut. Dieses enthält die sogenannte Argumentnummer des jeweiligen Tokens. Jedem annotierten Prädikat werden nun die referenzierten semantischen Rollen hinzugefügt. Die Argumentnummern bilden in diesem Kontext die Token auf die entsprechende semantische Rolle des Prädikats ab. Normalerweise benötigen Semantische-Rollen-Annotierer Sätze als Eingabe. Da PARSE allerdings natürliche Sprache verarbeitet und somit keine Satzzeichen vorhanden sind, liegen keine Sätze vor, denen ein Prädikat eindeutig zugewiesen werden kann. Als Alternative zu den Sätzen werden die Prädikate daher den von der Seichten Sprachverarbeitung definierten Instruktionen zugeordnet und anschließend mit semantischen Rollen annotiert.[Hey16]

4.3.5. Graphersteller

Die Token in der vom Spracherkenner erzeugten Liste werden in den Vorverarbeitungsstufen, Seichte Sprachverarbeitung, Eigennamenerkennung und dem Semantische-Rollen-Annotierer mit zusätzlichen Informationen versehen. Aus dieser Liste generiert nun der Graphersteller einen Graphen. Hierbei werden die einzelnen Token durch Knoten sowie die Zeiger zwischen den Listenelementen durch Kanten repräsentiert.

4.4. Agenten

Dieses Unterkapitel stellt die derzeit in PARSE verfügbaren Agenten sowie deren jeweilige Funktion vor. Auf eine Beschreibung des Dialogagenten wird an dieser Stelle verzichtet, da dieser in den folgenden Kapiteln ausführlich behandelt wird.

4.4.1. Mehrdeutigkeitsauflöser

Falls ein Wort mehrere Bedeutungen besitzen kann, soll mit Hilfe der Mehrdeutigkeitsauflösung (engl. word sense disambiguation) die aktuelle Bedeutung ermittelt werden. Um dieses Ziel zu erreichen gibt es verschiedene Vorgehensweisen. Eine Möglichkeit ist die Bedeutung eines Wortes anhand des umgebenden Kontextes zu bestimmten.[JM09, S. 671f.]
Bei PARSE wird ein anderer Ansatz gewählt. Dieser ordnet den mehrdeutigen Wörtern die Bedeutung zu, die diese am häufigsten besitzen.

4.4.2. Kontextanalysierer

Der Kontextanalysierer ergänzt die im Graphen enthaltene Aussage mit acht verschiedenen Arten an Kontext. Zunächst werden die Entitäten bestimmt und mit Informationen versehen, die diese näher beschreiben. Einer als „rote Tasse“ bezeichneten Tasse wird zum Beispiel die Eigenschaft beigefügt, dass sie rot ist. Im nächsten Schritt wird versucht, diese Entitäten einem Konzept zuzuordnen, z. B. wird der „große Kühlschrank“ dem Konzept „Kühlschrank“ zugewiesen. Falls vorhanden werden diese Konzepte anschließend mit ihren Überkonzepten gekennzeichnet. Im Beispiel mit dem „Kühlschrank“ wäre das Überkonzept „Haushaltsgerät“. Zudem können Konzepte auch auf andere Konzepte verweisen und repräsentieren dadurch Teil-Ganzes-Beziehungen. Daneben werden die Aktionen mit Zusatzinformationen, z. B. den beteiligten Entitäten, versehen. Die nächste Art an Kontext sind die Zustände der Entitäten, z. B. kann eine Türe offen oder geschlossen sein. In diesem Zusammenhang gibt es noch eine weitere Art an Kontext, den Zustandsübergang. Dieser beschreibt die Transformation des Zustands einer Entität mit Hilfe einer Aktion. Zum Beispiel ändert sich mit der Aktion „Türe schließen“ der Zustand einer Türe von „offen“ in „geschlossen“. Die achte Art an Kontext stellt die Ortsbeziehung dar. Diese beschreibt die Position einer Entität in Abhängigkeit von anderen Entitäten. Die folgenden Anweisung enthält eine Ortsbeziehung zwischen der „Wasserflasche“ und dem „Tisch“: „Bitte reiche mir die Wasserflasche, die auf dem Tisch steht.“[Hey16]

4.4.3. Koreferenzauflöser

In dem Satz „Michelle hat den Führerschein bestanden und sie fährt jetzt mit dem Auto.“ beziehen sich sowohl *Michelle* als auch *sie* auf die Entität „Michelle“. Verweisen zwei Wörter auf dieselbe Entität, werden diese Wörter als koreferierend bezeichnet. Wird nun ein ganzer Text statt eines einzelnen Satzes betrachtet, kann es sein, dass das Wort *sie* öfter vorkommt und sich hierbei auf unterschiedliche Personen bezieht. Daher ist es notwendig Koreferenzen aufzulösen, um die einzelnen Wörter den jeweiligen Entitäten zuordnen zu können.[JM09, S. 729f.] Diese Aufgabe wird bei PARSE vom Koreferenzauflöser übernommen.

4.4.4. Bedingungsanalysierer

Die Aufgabe des Bedingungsanalysierers ist das Erkennen von bedingten Anweisungen im Graphen. Hierfür muss der Agent WENN-DANN-SONST-Bedingungen in den Eingabesätzen erkennen und anschließend die entsprechende Anweisungsstruktur extrahieren. Der folgende Beispielsatz soll die Vorgehensweise verdeutlichen: „Wenn die Wäsche trocken ist, dann lege sie zusammen, andernfalls tue sie in den Trockner.“ Der erste Teil des Satzes beschreibt die Bedingung. Falls diese erfüllt wurde, soll die zwischen den Kommas stehende Anweisung ausgeführt werden. Die vorzunehmende Handlung, für den Fall, dass die Bedingung nicht erfüllt ist, steht im letzten Teil des Satzes.[Ste16] Das Erkennen von WENN-DANN-SONST-Bedingungen in natürlicher Sprache ist essentiell, um PARSE in die Lage zu versetzten, mit Hilfe von Spracheingaben zu programmieren.

4.5. Anwendungsfall Haushaltsroboter

Eine Anwendung für PARSE stellt die sprachbasierte Programmierung des Haushaltsroboters ARMAR-III[ARA+06] dar. Dieser ist mit einer großen Menge an Basisfähigkeiten ausgestattet, z. B. fahren, Gegenstände aufnehmen, etc. Um ARMAR-III komplexere Fähigkeiten beizubringen, kann der Nutzer dem Haushaltsroboter mit Hilfe von PARSE Befehle, bestehend aus Kombinationen von Basisfähigkeiten, erteilen, welche anschließend zu einer neuen Fähigkeit kombiniert werden. Ein Beispiel wäre der neue Befehl „Orangensaft

aus dem Kühlschrank holen“. Dieser besteht aus den einzelnen Anweisungen, „fahre zum Kühlschrank“, „öffne diesen“, „nimm den Orangensaft“, „schließe den Kühlschrank“, „fahre zu mir zurück“ und „gib mir den Orangensaft“. Damit derartige Anweisungen von PARSE verstanden werden können, muss dieses System in der Lage sein, längere und komplexere Spracheingaben zu verarbeiten, als viele andere Sprachverarbeitungssysteme.[WT15]

4.6. Korpus von PARSE

Um die einzelnen Vorverarbeitungsstufen und Agenten von PARSE zu evaluieren, wurden bisher sieben unterschiedliche Szenarien von verschiedenen Nutzern eingesprochen. Diese hatten alle das Ziel, dem Haushaltsroboter ARMAR-III, vgl. Unterkapitel 4.5, eine neue Aufgabe beizubringen und simulierten somit einen Anwendungsfall der dialogbasierten Programmierung. Die folgende Aufgabenstellung ist ein Ausschnitt aus einem dieser sieben Szenarien: „In this scene you want the robot to prepare your meal. Therefore he should take any of the plates, which are in the dishwasher. Give the plate a wash and put the instant meal from the fridge on the plate. Then warm it in the microwave. Afterwards he should put the plate on the table.“[Hey16] Die Anwender sollten nun ARMAR-III in eigenen Worten erklären, wie dieser die Aufgabe als Kombination seiner Basisfähigkeiten lösen kann. Durch die Aufzeichnung dieser Nutzeranweisungen entstand ein Sprachkorpus für PARSE, der aus 172 Audiodateien besteht. Diese sind in der Regel kürzer als eine Minute. Allerdings umfasst dieser Korpus keine Dialoge, sondern lediglich die initialen Handlungsanweisungen.

5. Analyse und Entwurf

Zunächst werden in diesem Kapitel die generellen Herausforderungen beim Verständnis natürlicher Sprache erörtert. Hierbei findet bereits eine Vorstellung allgemeiner Lösungsansätze statt. Diese werden anschließend auf PARSE übertragen. Dem folgt eine Beschreibung der Aufgabenstellung des, in dieser Abschlussarbeit zu erstellenden, Dialogagenten. Im Anschluss an die Abgrenzung des Dialogagenten vom Dialogmanager, findet die Diskussion der verwandten Arbeiten statt. Darauf folgt die Analyse von PARSE. Nach der Auswahl der zu bearbeitenden Fehlerklassen wird dieses Kapitel mit den Entwürfen für den Dialogagenten und den Dialogmanager abgeschlossen.

5.1. Verarbeitung natürlicher Sprache

Für den Menschen ist die natürlichste Art der Kommunikation die gesprochene Sprache. Nachdem er die Fähigkeit des Sprechens im Kleinkindalter erworben hat, besitzt er diese bis zu seinem Tod. Aus diesem Grund verfügen technische Geräte, die in der Lage sind mit dem Menschen in Form von natürlicher Sprache zu kommunizieren, über eine in allen Lebensbereichen des Menschen einsetzbare Schnittstelle. Dennoch nutzen viele Geräte diese Form der Kommunikation nicht. Dies kann nur bedeuten, dass es offensichtlich nicht trivial ist, diese Art der Mensch-Maschine-Interaktion einzusetzen. Im folgenden Abschnitt werden die Herausforderungen beim Verständnis gesprochener Sprache verdeutlicht und anschließend findet ein Überblick über mögliche Lösungsansätze statt.

5.1.1. Herausforderungen beim Verständnis natürlicher Sprache

Eine Herausforderung bei der Interaktion mit Hilfe gesprochener Sprache ist, dass der Mensch und die Maschine in der Regel einen sehr unterschiedlichen Wissenstand über ihre gegenwärtige Umgebung besitzen. Der Mensch ist sich normalerweise bewusst wo er sich befindet. Er kann die Dinge in seiner Umgebung sehen, hören, riechen und ertasten. All das kann die Maschine meist nicht. Da sich aber viele Aussagen des Menschen auf seine Umgebung beziehen, ist es für ein Gerät meist nicht nachvollziehbar wovon der Anwender spricht. Dies schränkt den Anwendungsbereich der Mensch-Maschine-Interaktion für die meisten Geräte deutlich ein.
Die nächste Herausforderung besteht im Fehlen einer gemeinsamen Vergangenheit zwischen dem Menschen und der Maschine. Unterhalten sich Menschen untereinander, beziehen sich viele Aussagen auf vergangene Erlebnisse und gemeinsame Erfahrungen. Daher können oft Details weggelassen werden, die den Gesprächspartnern bekannt sind, für das

Verständnis des Gesagten durch Dritte aber notwendig wären. Aufgrund der Tatsache, dass Maschinen in der Regel bei jeder Interaktion mit dem Menschen von Null beginnen, fehlen die Kontextinformationen, die das Gerät und sein Anwender bereits in der Vergangenheit ausgetauscht haben. Daher muss der Maschine alles erneut erklärt werden, was die Kommunikation unnatürlich und somit für den Menschen unbefriedigend macht.
Selbst unabhängig von der Einordnung des Gesagten in die Umgebung oder in die Vergangenheit, stellt die semantische Interpretation menschlicher Aussagen ein Problem dar. Das Gesagte ist teilweise mehrdeutig und bedarf selbst unter Menschen der Nachfrage. In diesem Zusammenhang stellen auch Sprünge im Kontext eine Herausforderung dar. Denn Menschen artikulieren stets ihre Gedanken unabhängig davon, ob dies sich auf das zuvor Gesagte bezieht.
Neben diesen grundsätzlichen sprachlichen Herausforderungen besteht ein weiteres Problem darin, dass die Leistungsfähigkeit der eingesetzten Mikrofone oft nicht ausreicht, um das Gesagte fehlerfrei interpretieren und effektiv von Hintergrundgeräuschen trennen zu können.
Da die eben beschriebenen Herausforderungen nicht isoliert sondern in der Regel kombiniert auftreten, verstärken sie sich gegenseitig. Die vorgestellten Probleme treten zudem nicht nur in Dialogen auf, sondern auch wenn ein Gerät lediglich eine Anweisung verarbeiten soll. Daher sind diese Herausforderungen in allen Anwendungsfällen in denen gesprochene Sprache als Kommunikationsmittel zwischen Mensch und Maschine eingesetzt wird, zu lösen.

5.1.2. Lösungsansätze

Um das Problem des unterschiedlichen Bewusstseins über die gegenwärtige Umgebung zu beseitigen, muss die Maschine mit Sensoren ausgestattet werden, die in der Lage sind, die Umgebung wahrzunehmen. Diese Sensordaten müssen anschließend von der Maschine sinnvoll interpretiert werden. Hierfür ist das Einbinden von Weltwissen und domänenspezifischen Ontologien notwendig.
Zum Aufbau einer gemeinsamen Vergangenheit zwischen dem Gerät und dem Anwender ist es notwendig, dass die Gesprächsdaten aus den vorangegangen Unterhaltungen und die daraus gewonnenen Informationen für jede weitere Interaktion zur Verfügung stehen, um dieses Wissen zur Interpretation der Nutzeraussage einsetzen zu können. Dadurch können auch Gewohnheiten des Anwenders identifiziert und für die Sprachverarbeitung eingesetzt werden.
Zur Verbesserung der semantischen Interpretation können die hierfür eingesetzten Werkzeuge mit Trainingsdaten dahingehend verbessert werden, dass sie Mehrdeutigkeiten und weitere in diesem Kontext auftretende Probleme erkennen und versuchen diese aufzulösen. Nichtsdestotrotz gelingt dies nicht in allen Fällen, da manche Aussagen tatsächlich mehrdeutig sind. Aus diesem Grund, sollte auch die Maschine in die Lage versetzt werden, Rückfragen stellen zu können, damit sie im Zweifelsfall die Semantik korrekt interpretiert. Hierfür bietet sich die Integration eines Dialogagenten an.
Das technische Problem der ungeeigneten Mikrofone, lässt sich durch hochwertigeres Equipment beseitigen. In diesem Kontext kann der Einsatz von Stimmerkennungssoftware die Entfernung von Hintergrundgeräuschen verbessern, da die Software die Tonspur des Nutzers erkennen und somit soweit möglich herausfiltern kann.
Da sich die Herausforderungen aus Abschnitt 5.1.1 gegenseitig verstärken, kann die Interpretation gesprochener Sprache nur durch einen ganzheitlichen Ansatz gelingen, der versucht, alle beschriebenen Lösungsansätze zu kombinieren. Allerdings können in der Regel, besonders bei längeren Eingaben, aufgrund der großen Anzahl an Fehlerquellen auch mit einem umfassenden Ansatz, nicht alle Probleme beseitigt werden. Daher ist es notwendig eine weitere Möglichkeit bereitzustellen, um Fehlinterpretation zu verhindern oder zu korrigieren. Hierfür bietet sich ein Dialogagent an. Dieser versetzt die Maschine in die

Lage, neben der bereits erwähnten Möglichkeit von Rückfragen zur semantischen Interpretation, den Menschen bezüglich schlecht verstandener Wörter, der Umweltsituation oder vergangener Ereignisse zu befragen.

5.1.3. Übertragung der Lösungsansätze auf PARSE

Die Rahmenarchitektur PARSE verfolgt das Ziel mit Hilfe von verbalen Instruktionen ausführbaren Quellcode zu erstellen. Hierdurch soll der Anwender in die Lage versetzt werden, komplizierte Aufgaben durch eine Kombination von Basisfähigkeiten zu lösen. Im Folgenden wird erläutert, ob und wie die in Abschnitt 5.1.2 vorgestellten Lösungsansätze für des Verständnis gesprochener Sprache auf PARSE übertragen werden können.
Die Rahmenarchitektur besitzt derzeit keine Möglichkeit, Sensordaten über die Umgebung zu sammeln. Somit kann PARSE die Umgebung lediglich anhand der Nutzeraussage interpretieren. Hierzu muss die Spracheingabe mit Welt- und Domänenwissen angereichert werden. Dafür wird bei der Rahmenarchitektur der Kontextanalysierer eingesetzt.
Gegenwärtig bietet PARSE keine Funktionalität an, die es erlaubt Gesprächsdaten vergangener Interaktionen für die aktuelle Kommunikation zu Verwenden. Dies stellt eine erhebliche Einschränkung der Möglichkeiten der Rahmenarchitektur dar. Der Einsatz eines Dialogagenten erlaubt, zumindest innerhalb eines Gesprächs die vorangegangenen Aussagen zu berücksichtigen. Als Speichermedium kann in diesem Fall der Graph von PARSE eingesetzt werden.
Für die semantische Interpretation des Gesagten, jenseits der Umweltinterpretation sowie des Vergangenheitsbezugs, besitzt die Rahmenarchitektur mehrere Komponenten. Bei den Vorverarbeitungsstufen sind dies die seichte Sprachverarbeitung, der Eigennamenerkenner und der Semantische-Rollen-Annotierer. Auch die Agenten Koreferenzauflöser, Mehrdeutigkeitsanalysierer und Bedingungsanalysierer dienen diesem Zweck. Trotz allem ist PARSE nicht immer in der Lage, den Anwender zu verstehen. Daher bietet sich der Einsatz eines Dialogagenten an, um dem System die Möglichkeit zu eröffnen, in kritischen Fällen Rückfragen zu stellen.
Auf das vom Anwender eingesetzte Mikrofon hat PARSE keinen Einfluss.
Zusammenfassend bietet es sich für PARSE an, einen Dialogagenten für die Verifizierung schlecht verstandener Wörter, zur Beschreibung der Umgebung durch den Anwender oder für weitere Fragen zum Verständnis der Nutzeraussage einzusetzen. Daher beschäftigt sich der folgende Teil dieses Kapitels mit der Analyse, welche Eigenschaften ein solcher Dialogagent besitzen sollte und wie ein geeigneter Entwurf dieses Agenten aussieht.

5.1.4. Aufgabenstellung, Fehlerklasse und Indikator

Die Architektur von PARSE sowie dessen Komponenten wurden bereits in Kapitel 4 vorgestellt. Hierbei wurden auch die jeweiligen Aufgaben der Vorverarbeitungsstufen und Agenten beschrieben, welche im Weiteren als Aufgabenstellung bezeichnet werden. Während der Bearbeitung einer Aufgabenstellungen können verschiedene Fehler auftreten. Eine Fehlerklasse beschreibt in diesem Kontext die Art des aufgetretenen Fehlers. Allerdings kann ein Fehler in der Regel nicht direkt erkannt werden. Daher sind sogenannte Indikatoren notwendig, die auf die entsprechende Fehlerklasse hinweisen, vgl. Abbildung 5.1. Diese Indikatoren können Konfidenzwerte sein, dass sich Annotationen verschiedener Agenten widersprechen oder das erwartete Annotationen fehlen, z. B. falls eine WENN-Bedingung über keine DANN-Anweisung verfügt. Eine Aufgabenstellung kann in diesem Zusammenhang mehrere Fehlerklassen besitzen, aber eine Fehlerklasse gehört eindeutig zu einer Aufgabenstellung. Dies wird durch die Definition der Fehlerklasse erreicht, die eindeutig eine bestimmte Art von Fehler beschreibt. Wenn die Charakterisierung der Fehlerklasse zu weit gefasst ist und somit auf mehrere Aufgabenstellungen zutreffen könnte, wäre nicht mehr nachvollziehbar, welches Problem im Einzelnen gelöst wird. Daher ist diese Eindeutigkeit

notwendig. Ebenso kann eine Fehlerklasse mehrere Indikatoren besitzen, aber es ist auch möglich, dass kein Indikator für eine Fehlerklasse vorhanden ist. In diesem Fall können die entsprechenden Fehler nicht erkannt werden. Existiert ein Indikator, so weißt dieser auf genau eine Fehlerklasse hin.
Ein Beispiel für eine Aufgabenstellung der Rahmenarchitektur ist der Spracherkenner. Dieser kann Wörter fehlinterpretieren, z. B. „bald“ als „Wald“ deuten. Somit besitzt der Spracherkenner die Fehlerklasse *Wort falsch erkannt*. Ein Indikator für diese Fehlerklasse sind niedrige Konfidenzwerte, welche der Spracherkenner für das Wort „Wald“ geliefert hat, vgl. Abschnitt 5.5.1.

5.2. Aufgabenstellung des Dialogagenten

Für die weitere Analyse und den daraus abgeleiteten Entwurf des Dialogagenten ist es entscheidend, zunächst die Aufgabenstellung des Dialogagenten zu definieren. Um die Programmierung in natürlicher Sprache mit Hilfe von PARSE zu realisieren, muss der Nutzer dem System die auszuführenden Befehle verbal mitteilen können. Aus diesen Anweisungen wird dann mit Hilfe der Rahmenarchitektur, siehe Kapitel 4, ein Graph erstellt, welcher die Aussagen des Anwenders repräsentiert. Jedoch treten bei der Sprachverarbeitung in der Regel an verschiedenen Stellen Fehler auf. Diese werden für PARSE ausführlich im Unterkapitel 5.5 beschrieben. Dadurch ist der Graph allerdings keine korrekte Repräsentation der Anwenderintention und muss daher, für das vollständige Verständnis der Nutzereingabe, von Fehlern befreit werden. Dies kann in Form von Rückfragen mit Hilfe natürlicher Sprache und der anschließenden Integration der Antwort in den Graphen geschehen. Somit wird bei PARSE die Sprachverarbeitung für zwei unterschiedliche Aufgaben verwendet. Einmal zur initialen Eingabe der Programmierungsbefehle und zum anderen zur Korrektur von Fehlern im Graphen. Für die erste Aufgabe sind sehr lange Befehlsketten zu verarbeiten, was eine deutlich höhere Komplexität aufweist, als das Verständnis kurzer Antworten auf Rückfragen. Letzteres benötigt allerdings zusätzlich zur Fähigkeit der Antwortinterpretation noch einen Dialogmanager für die Generierung von Fragen, was bei der Verarbeitung der initialen Eingabe nicht notwendig ist. Aus diesen Gründen werden die beiden Aufgaben bei PARSE getrennt voneinander bearbeitet. Der Fokus des in dieser Arbeit erstellten Dialogagenten liegt auf der Behebung von Fehlern im Graphen durch Rückfragen, sowie der hierfür notwendigen Fragegenerierung und Antwortinterpretation. Aufgrund der beschriebenen Aufgabenstellung ist es zielführend, wenn das System die Dialoginitiative übernimmt.
Darüber hinaus existieren weitere Anforderungen an den Dialogagenten. Zum einen soll dieser domänenunabhängig arbeiten und zum anderen soll der Funktionsumfang des Dialogagenten mit Hilfe einer geeigneten Architektur effizient erweiterbar sein.

5.3. Unterschied zwischen Dialogagent und Dialogmanager

Im Kontext dieser Abschlussarbeit steht ein Agent für eine Komponente, die eine Aufgabenstellung der Sprachverarbeitung lösen kann und hierfür den Graphen als Eingabe benutzt. Beim Dialogagenten besteht die Aufgabenstellung darin, Fehler im Graphen zu beseitigen, vgl. Unterkapitel 5.2. Dafür wird ein Dialog mit dem Nutzer initiiert, um dessen tatsächliche Intention zu erfragen und somit Fehlinterpretationen im Graphen zu korrigieren. Für diese Aufgabe ist ein Dialogmanager erforderlich, der den Zustand des Graphen analysiert. Hierbei werden im Graphen Indikatoren für Fehlerklassen gesucht. Falls mehrere Indikatoren gefunden werden, entscheidet die Dialogstrategie des Dialogmanagers nach welchem vermeintlichen Fehler der Anwender gefragt wird. Anschließend wird die Frage mit Hilfe des Textgenerators erzeugt und durch den Sprachgenerator an den Nutzer ausgegeben. Somit steht der Dialogmanager nur für einen Teil des Dialogagenten, vgl. Abbildung

5.2.
Dennoch werden in Kapitel 3 Dialogmanager statt Dialogagenten bei den verwandten Arbeiten betrachtet. Dies liegt daran, dass die weiteren Komponenten des Dialogagenten jenseits des Dialogmanagers in der Regel nur eingebunden sind und nicht in dieser Arbeit erstellt werden, wie z. B. der Sprachgenerator oder die Grapherzeugung mit Hilfe von PARSE. Eine Ausnahme bildet der Textgenerator, welcher allerdings eine geringe Komplexität aufweist und darüber hinaus entscheidend von der Implementierung des Dialogmanagers abhängt. Daher liegt das Hauptaugenmerk dieser Abschlussarbeit auf der Erstellung der Entscheidungslogik für die Frageprioriserierung - also der Kernaufgabe eines Dialogmanagers.

5.4. Diskussion der verwandten Arbeiten

In Kapitel 3 werden sechs verschiedene Arten an Dialogmanagern mit jeweils zwei Realisierungen vorgestellt. Dennoch ist es nötig für die Beseitigung der Fehler im Graphen von PARSE einen neuen Dialogmanager zu erstellen. Die Gründe hierfür werden im Folgenden erläutert.
Statistische Dialogmanager benötigen, wie in Unterkapitel 3.2 erwähnt, Trainingsdaten, um die Dialogstrategie zu optimieren. POMDP-Dialogmanager erfordern darüber hinaus Trainingsdaten für den Umgang mit Unsicherheiten bei der Spracherkennung. Der Sprachkorpus von PARSE besteht aus 172 Audiodateien, die nur die initiale Nutzereingabe und somit keine Dialoge enthalten. Des Weiteren sind die meisten Spracheingaben kürzer als eine Minute. Dieser Korpus ist daher nicht ausreichend, um einen POMDP-Dialogmanager zu trainieren. Unabhängig davon stehen keine Trainingsdaten für die Optimierung der Dialogstrategie zur Verfügung, da der Korpus keine Dialoge enthält. Dies ist allerdings sowohl für MDP- als auch für POMDP-Dialogmanager essentiell. Aus diesem Grund können derzeit bei PARSE keine statischen Dialogmanager eingesetzt werden.
Formularfüllende Dialogmanager versuchen, durch Fragen an den Nutzer, vorgegebene Lücken in einem Formular dynamisch zu füllen, vgl. Abschnitt 3.1.2. Diese Art von Dialogmanagern eignet sich daher besonders, wenn die benötigten Informationen im Voraus bekannt sind, z. B. bei Flugbuchungssystemen. Bei PARSE ist dies allerdings nicht der Fall, denn die Fehler im Graphen treten erst zur Laufzeit auf und sind unterschiedlicher Natur, d. h. die Indikatoren verweisen auf verschiedene Aufgabenstellungen. Zudem soll der Dialogagent von PARSE pro Aufruf nur einen Fehler lösen, um anschließend den anderen Agenten die Möglichkeit zu geben, mit Hilfe der neuen Information, weitere Fehler im Graphen zu beseitigen. Dadurch wird die Fähigkeit mehrere Aufgaben gleichzeitig zu bearbeiten, was einer Stärke der formularfüllenden Dialogmanager entspricht, obsolet. Aus den genannten Gründen ist diese Art der Dialogmanager ungeeignet für den Einsatz im Dialogagenten von PARSE.
Wie in Abschnitt 3.1.3 erläutert, eignen sich agendabasierte Dialogmanager für Anwendungsfälle, bei denen mehrere Teildialoge zu einem Gespräch verknüpft werden können, z. B. bei einer Urlaubsplanung mit mehreren Komponenten wie Auto mieten, Flüge oder Hotel buchen, etc. Auch hier ist das Ziel die Lücken der verschiedenen Teilformulare zu füllen. Somit eignen sich agendabasierte Dialogmanager aus den gleichen Gründen wie formularfüllende Dialogmanager nicht für den Einsatz in dieser Abschlussarbeit.
Für den Einsatz schließender Dialogmanager ist die Definition einer Menge logischer Axiome erforderlich, vgl Abschnitt 3.1.4. Mit Hilfe dieser Axiome sollen im Gesprächsverlauf Schlussfolgerungen aus dem Gesagten gezogen werden, um den Dialog natürlicher zu gestalten. Abgesehen von dem erheblichen Aufwand, den die Definition derartiger Axiome mit sich bringt, ist der kurze Gesprächsverlauf des Dialogagenten bei PARSE, der in der Regel nur aus einer Frage und einer Antwort bestehen soll, ungeeignet für das Ziehen von Schlussfolgerungen aus dem Gesprächsverlauf. Aus diesen beiden Gründen wird in dieser

Arbeit auf den Einsatz eines schließenden Dialogagenten verzichtet.
Bei den graphbasierten Dialogmanagern ist der Gesprächsverlauf bereits vor der Ausführung in Form eines gerichteten Graphen festgelegt. Von diesem gerichteten Graphen ist der Graph der bei PARSE als Repräsentation der Nutzeraussage eingesetzt wird zu unterscheiden. Graphbasierte Dialogmanager eignen sich vor allem für Anwendungsfälle, in denen die Antwortmöglichkeiten des Nutzers eingeschränkt sind, z. B. Wetterabfragen. Ungeeignet ist diese Form des Dialogmanagements für Aufgaben, bei denen der Anwender die Initiative übernimmt oder den Dialog führt, vgl. Abschnitt 3.1.1.
Da der Dialogagent bei jedem Aufruf, gemäß den entdeckten Indikatoren, andere Fragen stellt und diese dem Anwender im Voraus unbekannt sind, ist es nicht zielführend, dem Nutzer die Initiative zu überlassen. Trotz der stets neuen Fragen, kann bei dem eingesetzten Dialogmanager ein vordefinierter Gesprächsverlauf festgelegt werden, vgl. Unterkapitel 5.8.
Offene Fragen führen zwar auf der einen Seite zu einer natürlicheren Interaktion zwischen Mensch und Maschine, auf der anderen Seite erhöhen diese allerdings die Komplexität der Antwortinterpretation. Da der Fokus dieser Abschlussarbeit darauf liegt, einen Dialogagenten bereitzustellen, der die Fehler im Graphen löst, wird die Benutzerfreundlichkeit als zweites Ziel der effektiven Fehlerbeseitigung untergeordnet. Daher soll der eingesetzte Dialogmanager dem Nutzer in der Regel eingeschränkte Antwortmöglichkeiten anbieten.
Zusammenfassend lässt sich somit das Anforderungsprofil wie folgt beschreiben: Der Dialogagent soll mit Hilfe einer systemgeführten Dialoginitiative dem Nutzer Fragen mit eingeschränkten Antwortmöglichkeiten gemäß einem vordefinierten Gesprächsverlauf stellen und mit Hilfe der Antworten die Fehler im Graphen beseitigen. Hierzu muss der Graph zunächst analysiert werden, um die Fehlerstellen zu extrahieren. Danach muss eine Frage für den Nutzer formuliert und ausgegeben werden. Der nächste Schritt ist die Antwortinterpretation. Abschließend müssen die Erkenntnisse noch in den Graphen integriert werden.
Von allen aufgeführten Arten an Dialogmanagern hat der graphbasierte Dialogmanager die größte Schnittmenge mit dem eben vorgestellten Anforderungsprofil und wird daher in dieser Abschlussarbeit eingesetzt.
Allerdings können die graphbasierten Dialogmanager von McTear und ETUDE, vgl. Abschnitt 3.1.1, nicht in PARSE eingesetzt werden. Zum einen sind diese für andere Anwendungsfälle erstellt worden und zum anderen müssen die Fragen des in dieser Arbeit eingesetzten Dialogmanagers dynamisch erzeugt werden und können daher nicht aus im Voraus festgelegten Fragen bestehen. Aus diesen Gründen wird für PARSE ein neuer Dialogmanager als Teil des Dialogagenten erstellt.

5.5. Analyse von PARSE

In diesem Unterkapitel werden die Aufgabenstellungen der einzelnen Komponenten von PARSE, hinsichtlich möglicher Fehlerklassen und den dazugehörigen Indikatoren, untersucht. Dies ist für den späteren Entwurf des Dialogmanagers notwendig. Durch das zugrundeliegende Konzept können in PARSE Fehler bei der Bearbeitung einer Aufgabenstellung zu Folgefehlern in den weiteren Vorverarbeitungsstufe sowie den Agenten führen und die Probleme dadurch verstärken. Ein Beispiel hierfür wäre ein im Spracherkenner falsch identifiziertes Wort, welches in der Seichten Sprachverarbeitung mit der, für das erkannte Wort, korrekten Wortart annotiert wird, aber das eigentlich gesagte Wort eine andere Wortart besitzt. In einem solchen Fall ist das Ziel das Wort zu korrigieren und nicht an ein falsches Wort die ursprünglich intendierte Wortart zu annotieren. Daher werden Folgefehler bei der Analyse der einzelnen Aufgabenstellungen nicht betrachtet, da dies zu einem enormen Anstieg an Fehlerklassen beziehungsweise deren Indikatoren führen würde und keine Aussagen über die in der jeweils betrachteten Aufgabenstellung auftretenden Fehler liefert.
Stellt PARSE für eine Fehlerklasse keine Indikatoren bereit, findet eine kurze Beschreibung wünschenswerter Indikatoren statt, sofern diese durch einen Dialogagenten sinnvoll,

zur Beseitigung von Fehlern, eingesetzt werden könnten. Eine Auflistung der relevanten Aufgabenstellungen sowie deren Fehlerklassen und den dazugehörigen vorhandenen Indikatoren ist in Tabelle 5.1 dargestellt.

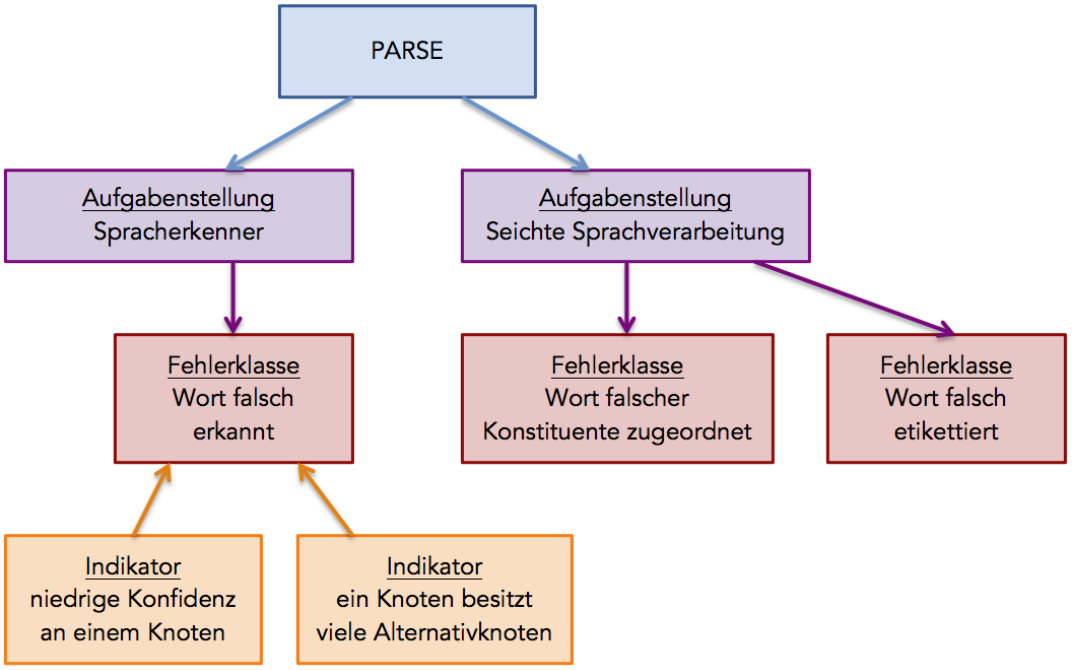


Abbildung 5.1.: Beziehungen zwischen Aufgabenstellung, Fehlerklasse und Indikator für die Aufgabenstellungen Spracherkenner und Seichte Sprachverarbeitung

5.5.1. Spracherkenner

Bei der Aufgabenstellung des Spracherkenners, eine Tonspur in Text zu überführen, kann es zu Fehlinterpretationen einzelner Laute oder deren Kombination kommen. Dadurch können die gesagten Wörter, welche einer Kombination von Lauten entsprechen, von den erkannten Wörtern abweichen. Somit besitzt der Spracherkenner die Fehlerklasse *Wort falsch erkannt*. Um ein derartiges Wort im Graphen von PARSE zu erkennen, können zwei verschiedene Indikatoren herangezogen werden, vgl. Abbildung 5.1. Da jeder Knoten ein Wort repräsentiert, ist eine niedrige Konfidenz des Spracherkenners für einen Knoten ein Hinweis auf ein falsch verstandenes Wort. Ein zweites Indiz sind viele vorgeschlagene Alternativknoten für einen Knoten. Allerdings reicht die Existenz dieser Indikatoren alleine nicht aus, um die Probleme der Spracherkennung im Graphen zu lösen. Denn eine niedrige Konfidenz könnte auch nur bedeuten, dass ein Wort schwer zu erkennen war, aber dennoch richtig verstanden wurde. Das gleiche gilt für das Auftreten vieler Alternativknoten.
Es bestehen zwei Möglichkeiten, Fehler des Spracherkenners mit Hilfe des Dialogagenten zu beseitigen. Zum einen kann der Anwender aufgefordert werden, seine Programmieranleitung erneut komplett einzusprechen und zum anderen können einzelne Knoten durch Rückfragen an den Nutzer verifiziert werden. Die erste Variante kann jedoch neue Fehler an anderer Stelle erzeugen und löst somit das Problem nicht nachhaltig. Die Verifizierung einzelner Knoten kann zu sehr häufigen Rückfragen führen, falls der initiale Graph viele Fehler aufweist. Dennoch ist dies der Ansatz des Dialogagenten von PARSE, da bei dieser Vorgehensweise der Anwender korrekterweise den Eindruck hat, dass seine Anfrage bearbeitet wird. Muss der Nutzer dagegen seine initiale Aussage z. B. fünf Mal wiederholen, hat er diesen Eindruck zu Recht nicht.

5.5.2. Seichte Sprachverarbeitung

Die Aufgaben der Seichten Sprachverarbeitung sind das Annotieren der einzelnen Tokens mit den entsprechenden Wortarten sowie die Erstellung der Konstituenten, vgl. Abschnitt 4.3.2. Daher besitzt diese Vorverarbeitungsstufe die beiden Fehlerklassen *Wort falsch etikettiert* und *Wort falscher Konstituente zugeordnet*, siehe Abbildung 5.1. Etikettieren steht in diesem Zusammenhang für das Annotieren der Wortarten. PARSE stellt allerdings für beide Fehlerklassen keine Indikatoren bereit, da lediglich die Etiketten ohne weitere Informationen annotiert werden. Dadurch können Fehler, die bei der Seichten Sprachverarbeitung gemacht wurden, nicht erkannt und somit auch nicht durch den Dialogagenten mit Hilfe von Rückfragen gelöst werden.
Wünschenswerte Indikatoren zur Identifikation falsch etikettierter Wörter wären, die Konfidenz für die aktuell annotierte Wortart sowie alternative Wortarten. Sofern diese Informationen nicht bereits durch das aktuell eingesetzte Sprachverarbeitungswerkzeug für die Seichte Sprachverarbeitung erzeugt werden, könnten diese aus der Kombination der Ergebnisse verschiedener Sprachverarbeitungswerkzeuge für diese Vorverarbeitungsstufe ermittelt werden. Die gleiche Vorgehensweise kann für die Generierung von Konfidenzen für die Konstituenten oder die Bereitstellung alternativer Konstituenten eingesetzt werden.

5.5.3. Eigennamenerkenner

Der Eigennamenerkenner soll Wörter im Text finden, die für Orte, Personen oder Organisationen stehen und diese dann mit der entsprechenden Kategorie kennzeichnen. Für diese Aufgabenstellung existieren die drei folgenden Fehlerklassen: *Eigenname nicht erkannt*, *Eigenname erkannt aber falscher Kategorie zugeordnet* und *Annotiertes Wort ist kein Eigenname*. Da die Seichte Sprachverarbeitung von PARSE unter anderem bei bestimmten Nomen die Wortart „Eigenname“ annotiert, kann dieses Etikett an einem vom Eigennamenerkenner nicht gekennzeichneten Wort, ein Indikator dafür sein, dass dieses Wort fälschlicherweise nicht als Eigenname erkannt wurde. Der in Abschnitt 4.4.2 eingeführte Kontextanalysierer stellt für die im Text verwendeten Entitäten Kontextinformationen bereit, die sich aus dem Satzzusammenhang ergeben. Diese Information kann mit dem annotierten Eigennamen verglichen werden und falls dieser in diesem Zusammenhang keinen Sinn ergibt, ist dies ein Indikator für die Fehlerklasse *Eigenname erkannt aber falscher Kategorie zugeordnet*. Zum Beispiel könnte in dem Satz „München ist Tabellenführer der Fußballbundesliga.“ dem Wort „München“ die Information hinzugefügt werden, dass es sich um einen Fußballverein handelt. Falls der Eigennamenerkenner „München“ allerdings als Ort gekennzeichnet hat, ist die Kontextinformation ein Indiz dafür, dass der annotierte Eigenname falsch ist. Als Indikator für die dritte Fehlerklasse können erneut die Etiketten der Seichten Sprachverarbeitung dienen. Denn falls die annotierte Wortart nicht „Eigenname“ ist, weist dies darauf hin, dass ein durch den Eigennamenerkenner als Eigenname etikettiertes Wort, fälschlicherweise annotiert wurde.

5.5.4. Semantische-Rollen-Annotierer

Mit Hilfe des Semantische-Rollen-Annotierers werden Instruktionen mit semantischen Rollen versehen, vgl. Abschnitt 4.3.4. Aus dieser Perspektive könnten nicht annotierte Wörter als Fehlerklasse des Semantische-Rollen-Annotierers angesehen werden. Allerdings kann diese Vorverarbeitungsstufe nur Instruktionen etikettierten, deren Satz- bzw. Instruktionsstruktur zu vorher definierten Mustern passt. Daher sind nicht annotierte Textteile in der Regel einem Mangel an Vergleichsmustern, in den Werkzeugen für die Erkennung semantischer Rollen, geschuldet und werden daher in dieser Abschlussarbeit nicht als Fehlerklasse angesehen.
Somit besitzt diese Vorverarbeitungsstufe nur die Fehlerklasse *Konstituente mit falscher*

semantischer Rolle annotiert. Die in den Prädikaten annotierten Rollen besitzen einen Konfidenzwert. Dieser gibt jedoch nicht die Wahrscheinlichkeit an, ob die annotierten Rollen korrekt sind, sondern wie viele verschiedene andere semantische Rollenmuster auch auf diese Instruktionsstruktur gepasst hätten[Hey16]. Existieren zum Beispiel vier verschiedene Rollenmuster, die für eine Aussage passen würden, beträgt die Konfidenz ein Viertel, also eine Aussage geteilt durch vier Möglichkeiten. Daher kann dieser Konfidenzwert nur eingeschränkt als Indikator für die betrachtete Fehlerklasse herangezogen werden. Liegt die Konfidenz bei eins, wird lediglich ein Rollenmuster vorgeschlagen. Das bedeutet aber nicht zwingend, dass die annotierte Rolle korrekt ist, z. B. könnte das Wort gar keine Rolle besitzen und wurde somit fälschlich annotiert.
Um nun dieses Problem des Semantische-Rollen-Annotierers mit Hilfe des Dialogagenten zu lösen, fehlen noch die alternativen Rollenmuster, nach denen gefragt werden könnte, wenn die Konfidenz nicht bei eins liegt. Allerdings ist an dieser Stelle zu berücksichtigen, dass PARSE Menschen, die sich nicht mit Linguistik oder Informatik auskennen, in die Lage versetzen soll, einfache Handlungsinstruktionen zu programmieren. Daher würden auch vorhandene alternative Rollenmuster nicht weiterhelfen, da der Anwender meist weder die aktuellen Rollen, noch die verwendeten Rollennamen kennt und somit eine Frage hierzu nicht adäquat beantworten könnte. Aus diesem Grund wird der Dialogagent im Rahmen dieser Abschlussarbeit nicht zum Lösen der Probleme dieser Vorverarbeitungsstufe eingesetzt.

5.5.5. Graphersteller

Die Aufgabe des Grapherstellers ist die Überführung der vom Spracherkenner erzeugten Liste in einen Graphen und besitzt die Fehlerklasse *Graph fehlerhaft erstellt*. Da dies jedoch kein Problem der Sprachverarbeitung darstellt, sondern ausschließlich von der korrekten Implementierung der Überführungsmethode abhängt und somit im Rahmen der Softwareentwicklung z. B. durch Testklassen behandelt werden sollte, ist es nicht zielführend den Dialogagenten zur Lösung dieser Fehlerklasse einzusetzen.

5.5.6. Mehrdeutigkeitsauflöser

Beim Mehrdeutigkeitsauflöser können die Fehlerklassen *Mehrdeutigkeit nicht aufgelöst* und *Mehrdeutigkeit falsch aufgelöst* auftreten. Falls ein Wort keine Annotation des Mehrdeutigkeitsauflösers besitzt, kann dies zwei Ursachen haben. Entweder das Wort ist nicht mehrdeutig oder dieser Agent hat die Mehrdeutigkeit nicht korrekt aufgelöst. Um Letzteres herauszufinden müsste eine Suche nach Mehrdeutigkeiten für das entsprechende Wort in verschiedenen Lexika und Ontologien durchgeführt werden. Dies würde allerdings die Funktionsweise des Mehrdeutigkeitsauflösers erneut implementieren und ist daher nicht zielführend. Denn falls der Mehrdeutigkeitsauflöser nicht richtig funktioniert, sollte dieser korrigiert oder erweitert werden, anstatt dessen Aufgabe erneut zu bearbeiten. Somit steht für die Fehlerklasse *Mehrdeutigkeit nicht aufgelöst* kein Indikator zur Verfügung. Ein niedriger Konfidenzwert einer zugeordneten Bedeutungserklärung repräsentiert einen Indikator für die Fehlerklasse *Mehrdeutigkeit falsch aufgelöst*.

5.5.7. Kontextanalysierer

Der Kontextanalysierer ordnet den Entitäten und Aktionen der Nutzeraussage verschiedene Arten an Kontext zu, vgl. 4.4.2. In diesem Zusammenhang können zwei mögliche Arten von Fehlern auftreten. Zum einen kann einer Entität oder Aktion kein Kontext zugeordnet sein, obwohl diese einen besitzt. Zum anderen kann einer Aktion oder Entität ein falscher Kontext zugewiesen sein. Diese beiden Probleme können für alle acht Arten an Kontext auftreten. Somit besitzt der Kontextanalysierer 16 Fehlerklassen. Da die Beschreibung jeder

dieser Fehlerklassen stets eine Wiederholung der beiden erwähnten Fehlerarten, bezogen auf einen anderen Kontext, darstellt und im weiteren Verlauf der Arbeit nicht relevant ist, werden an dieser Stelle nur die beiden folgenden abstrakten Fehlerklasse eingeführt. Die Fehlerklasse *Kontext fehlt* repräsentiert den Fall, dass ein Wort eigentlich einen Kontext besitzt, dieser jedoch nicht annotiert wurde. Allerdings kann aus der Tatsache, dass einem Wort keine bestimmte Kontextart zugeordnet ist, nicht abgeleitet werden, dass diese fehlt, denn das Wort könnte tatsächlich keinen entsprechenden Kontext besitzen. Somit existiert für diese Fehlerklasse kein Indikator. In die Fehlerklasse *Kontext falsch annotiert* fallen alle Wörter deren annotierter Kontext nicht korrekt ist. Falls ein Wort mit mehreren Kontextarten annotiert ist und diese sich widersprechen, ist dies ein Indikator für falsch zugeordneten Kontext.

5.5.8. Koreferenzauflöser

Bei der Auflösung von Koreferenzen treten die Fehlerklassen *Koreferenz falsch aufgelöst* und *Koreferenz nicht aufgelöst* auf. Die Koreferenzen werden bei PARSE durch Kanten zwischen Entitäten dargestellt. Diese Entitäten wurden vom Kontextanalysierer in den Graphen eingefügt, vgl. Abschnitt 4.4.2. Als Indikator für die erste Fehlerklasse können die, in den Kanten, annotierten Konfidenzwerte herangezogen werden, die die Wahrscheinlichkeit angeben, dass die vorgenommene Koreferenzauflösung korrekt ist. Da sich Personalpronomen stets auf eine Entität beziehen, stellen nicht koreferierte Personalpronomen einen Indikator für die zweite Fehlerklasse dar.

5.5.9. Bedingungsanalysierer

Die Aufgabe des Bedingungsanalysierers ist das Erkennen von WENN-DANN-SONST-Bedingungen in der Nutzereingabe. Hierbei können folgende Fehlerklassen auftreten: *Bedingung nicht erkannt*, *DANN-Anweisung nicht erkannt* und *SONST-Anweisung nicht erkannt*. Für die erste Fehlerklasse stellt PARSE keine Indikatoren bereit. Das Problem ist an dieser Stelle, dass verschiedene sprachliche Formulierungen eine Bedingung repräsentieren können. Um einen Indikator hierfür zu erhalten, könnten Wörter, die für eine Bedingung stehen mit Konfidenzen versehen werden, die angeben, mit welcher Wahrscheinlichkeit die Wörter im aktuellen Kontext für eine Bedingung stehen. Da dieser Indikator allerdings derzeit nicht von PARSE bereitgestellt wird, kann der Dialogagent nicht zur Lösung der ersten Fehlerklasse eingesetzt werden.
Das Fehlen einer DANN-Anweisung nach einer Bedingung ist ein Indikator für die zweite Fehlerklasse. Äquivalent beschreibt das Fehlen einer SONST-Anweisung nach einer Bedingung einen Indikator für die dritte Fehlerklasse.

5.6. Auswahl der Fehlerklassen

Um gemäß der Aufgabenstellung, siehe Unterkapitel 5.2, die Fehler im Graphen zu beseitigen, müssen zunächst die zu bearbeitenden Fehlerklassen identifiziert werden. Da hierfür Indikatoren nötig sind, welche auf die jeweiligen Probleme hinweisen, werden im Weiteren nur die Fehlerklassen aus Tabelle 5.1 berücksichtigt, die über Indikatoren verfügen. Aus diesen Fehlerklassen werden nun drei ausgewählt, um zu zeigen, dass der in dieser Arbeit vorgestellte Ansatz des Dialogagenten funktioniert. Die Bearbeitung aller Fehlerklassen mit Indikatoren würde über den Umfang dieser Abschlussarbeit hinausgehen. Bei der Auswahl werden vor allem die Fehlerklassen berücksichtigt, die den größten Nutzen für das Verständnis der Anwenderintention bieten.
Da die Qualität aller Vorverarbeitungsstufen und Agenten unmittelbar von der Güte des Spracherkenners abhängt, wird die Fehlerklasse *Wort falsch erkannt* vom Dialogagenten

Tabelle 5.1.: Übersicht der relevanten Aufgabenstellungen, Fehlerklassen und Indikatoren

Aufgabenstellung	**Fehlerklasse**	**Indikator**
Spracherkenner	Wort falsch erkannt	niedrige Konfidenz an einem Knoten
		ein Knoten besitzt viele Alternativknoten
Seichte-Sprachverarbeitung	Wort falscher Konstituente zugeordnet	
	Wort falsch etikettiert	
Eigennamen-erkenner	Eigenname nicht erkannt	Nomen wurde mit der Wortart „Eigenname“ annotiert
	Eigenname erkannt aber falscher Kategorie zugeordnet	Kontextinformation des Wortes
	Annotiertes Wort ist kein Eigenname	Nomen wurde nicht mit der Wortart „Eigenname“ annotiert
Semantische-Rollen-Annotierer	Konstituente mit falscher semantsicher Rolle annotiert	Konfidenzwert für die zugewiesene Rolle
Mehrdeutigkeits-auflöser	Mehrdeutigkeit nicht aufgelöst	
	Mehrdeutigkeit falsch aufgelöst	niedrige Konfidenz der annotierten Bedeutungserklärung
Kontextanalysierer	Kontext fehlt	
	Kontext falsch annotiert	mehrere Kontextarten die sich widersprechen
Koreferenzauflöser	Koreferenz falsch aufgelöst	Konfidenzwert für Koreferenzauflösung
	Koreferenz nicht aufgelöst	Personalpronomen nicht koreferiert
Bedingungs-analysierer	Bedingung nicht erkannt	
	DANN-Anweisung nicht erkannt	fehlen einer DANN-Anweisung nach einer Bedingung
	SONST-Anweisung nicht erkannt	fehlen einer SONST-Anweisung nach einer Bedingung

bearbeitet. Aufgrund der Tatsache, dass PARSE dazu dient die Benutzereingabe in Programme zu überführen, ist die Erkennung von Klassen bzw. Objekten, die häufig durch Eigennamen repräsentiert werden, von großer Bedeutung. Deswegen ist die Lösung der Probleme des Eigennamenerkenners sehr wünschenswert. Allerdings besitzt diese Vorverarbeitungsstufe derzeit noch erhebliches Verbesserungspotenzial, zum Beispiel durch das Einbinden weiteren Weltwissens zur Eigennamenerkennung. Um die Dialoge mit dem Anwender somit nicht unnötig in die Länge zu ziehen, wird derzeit auf die Auflösung der Probleme dieser Vorverarbeitungsstufe mit Hilfe des hier erstellten Sprachagenten verzichtet. Sobald in der Zukunft eine erweiterte Version des Eigennamenerkenners in PARSE eingebunden ist, kann der Dialogagent durchaus zur Lösung der auftretenden Probleme herangezogen werden. Semantische Rollen sind für das Verständnis der Nutzerintention

sehr wichtig. Allerdings ist ein Dialogagent für die Beseitigung der Fehlerklasse *Konstituente mit falscher semantischer Rolle annotiert* eher ungeeignet, da die meisten Anwender mit dem Konzept der semantischen Rolle nicht vertraut sind und somit auch keine Fragen hierzu beantworten können. Das Auflösen von Mehrdeutigkeiten ist ein weiterer Schritt zur Erzeugung von validem Quellcode. Da der Mehrdeutigkeitsauflöser die Wörter mit ihrer häufigsten Bedeutung annotiert, müsste der Anwender bei Rückfragen in der Regel Wörter bestätigen. Dies ist nicht sehr hilfreich für PARSE. Daher wird an dieser Stelle auf die Beseitigung der Fehler des Mehrdeutigkeitsauflösers durch den Dialogagenten verzichtet. Um Probleme des Kontextanalysierers durch Rückfragen auflösen zu können, müssten die Anwender mit den verwendeten Konzepten vertraut sein, damit sie die Fragen korrekt beantworten können. Dies ist in der Regel nicht der Fall. Deswegen wird auf eine Bearbeitung dieser Aufgabenstellung verzichtet. Eine weitere entscheidende Fehlerklasse zur Interpretation der Nutzerintention ist *Koreferenz falsch aufgelöst*. Die folgende Anweisung an einen Haushaltsroboter zeigt beispielhaft die enorme Bedeutung dieser Fehlerklasse: „Tue die Wäsche in die Waschmaschine und hänge sie anschließend auf den Wäscheständer.“ Falls nun die Probleme der eben erwähnten Fehlerklasse des Koreferenzauflösers nicht beseitigt wurden, kann der Anwender anschließend den Haushaltsroboter dabei beobachten, wie dieser versucht die Waschmaschine am Wäscheständer aufzuhängen. Für die dialogbasierte Programmierung ist des Weiteren das Verständnis von Bedingungen und den dazugehörigen Anweisungen entscheidend, um die Spracheingabe in sinnvolle Programme zu überführen. Aus diesem Grund findet eine Bearbeitung der Fehlerklasse *DANN-Anweisung nicht erkannt* in dieser Abschlussarbeit statt. Auf die Beseitigung der Fehlerklasse *SONST-Anweisung nicht erkannt* wird allerdings verzichtet. Dies geschieht vor dem Hintergrund, dass Menschen häufig keine SONST-Anweisung nennen, falls für eine nicht erfüllte Bedingung keine Handlung vorzunehmen ist. Legt man diese Annahme zugrunde, würde die Behandlung dieser Fehlerklasse viele Fragen erzeugen, die in der Regel keinen Zusatznutzen bieten.
Zusammenfassend wird der Dialogagent somit für die Beseitigung von Problemen der folgenden drei Fehlerklassen eingesetzt, *Wort falsch erkannt*, *Koreferenz falsch aufgelöst* und *DANN-Anweisung nicht erkannt*.

5.7. Entwurf des Dialogagenten

Die Aufgabe des Dialogagenten ist die Initiierung eines Gesprächs mit dem Anwender, damit dieser Fragen des Systems beantwortet, vgl. Unterkapitel 5.2. Mit Hilfe der Antworten sollen anschließend die Fehler im Graphen von PARSE beseitigt werden. Zur Führung dieses Gesprächs benötigt der Dialogagent die Komponenten eines Dialogsystems gemäß Unterkapitel 2.4. Somit ist der grundlegende Entwurf des Dialogagenten bereits vorgegeben. Daher werden im Folgenden die einzelnen Komponenten und deren Zusammenspiel näher erläutert. Die komplette Architektur des Dialogagenten ist in Abbildung 5.2 dargestellt. Rot kennzeichnet hierbei die selbst erstellten Komponenten und Blau symbolisiert die eingebundenen Elemente. In der Abbildung sind auch die Komponenten des Dialogagenten dargestellt, die als Schnittstelle zur Umwelt agieren. Dies ist zum einen der Sprachgenerator, dessen Ergebnis als akustisches Signal an den Nutzer ausgegeben wird und zum anderen der Spracherkenner, der die Antwort des Anwenders, beziehungsweise das akustische Signal dessen, als Eingabe erhält. Darüber hinaus analysiert der Dialogmanager noch den Graphen der initialen Spracheingabe des Nutzers und fügt die Erkenntnisse des Dialogagenten in diesen ein. Die Abbildung 5.2 kann als detaillierter Ausschnitt von Abbildung 4.1 angesehen werden.

Um Redundanzen zu vermeiden werden der Spracherkenner und der Sprachversteher aus Abbildung 2.1 durch die Vorverarbeitungsstufen von PARSE realisiert. Obwohl diese eigentlich zur Analyse der initialen Eingabe konzipiert wurden, können sie dennoch ohne

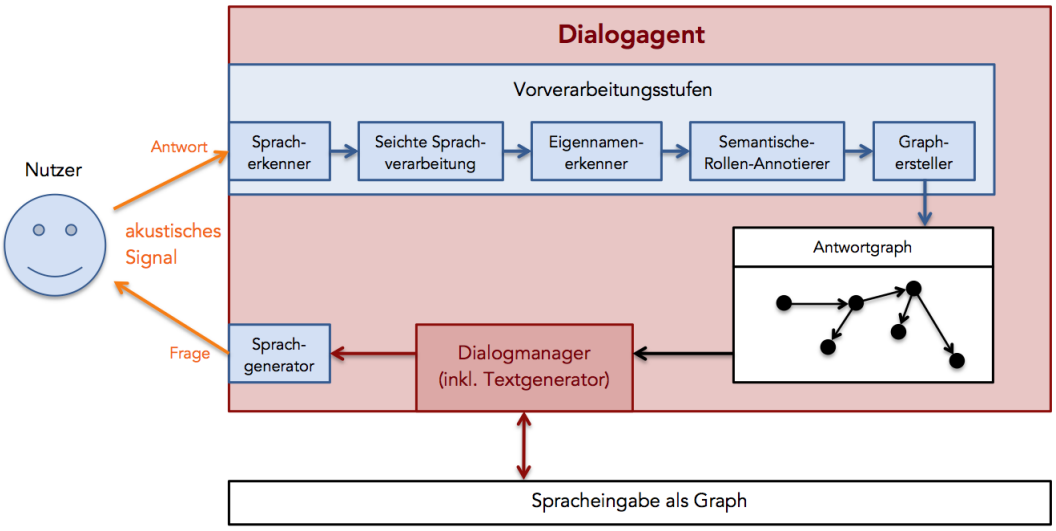


Abbildung 5.2.: Aufbau des Dialogagenten

Abwandlung für das Verständnis und die Interpretation einer Nutzerantwort eingesetzt werden. Dadurch liegt die Antwort des Anwenders genau wie dessen Spracheingabe auch als Graph vor. Dies erlaubt es dem Dialogmanager die gleichen Methoden zur Interpretation der initialen Aussage und der Nutzerantworten einzusetzen, vgl. Abschnitt 5.8.2.
Im Weiteren wird die initiale Eingabe auch als Graph und die Antwort als Antwortgraph bezeichnet. Aufgrund der Tatsache, dass der Dialogagent die Fragen dynamisch im Kontext des jeweiligen Graphen und je nach Fehlerklasse erzeugen soll, siehe Unterkapitel 5.4, bietet es sich an, den Textgenerator in den Dialogmanager zu integrieren. Die innere Struktur des Dialogmanagers wird in Abschnitt 5.8.2 näher erläutert.
Der Sprachgenerator des Dialogagenten wird von einem Drittanbieter eingebunden, da dessen Eigenentwicklung über den Umfang dieser Arbeit hinausgehen würde und diese Komponente nicht im Fokus der wissenschaftlichen Untersuchung dieser Abschlussarbeit steht.

5.8. Entwurf des Dialogmanagers

Der Dialogmanager ist die Komponente des Dialogagenten, welche die Entscheidungslogik für die Interaktion mit dem Nutzer enthält. Da der Dialogagent vor allem aus bereits vorhandenen Elementen besteht, repräsentiert der hier vorgestellte Dialogmanager den wesentlichen wissenschaftlichen Beitrag dieser Abschlussarbeit. Um ein sinnvolles Konzept für den Aufbau dieser Komponente erstellen zu können, muss zunächst das Anforderungsprofil beschrieben werden. Hierfür findet im Folgenden die Beschreibung der vier wesentlichen Aufgaben des Dialogmanagers statt. Danach wird der Aufbau des Dialogmanagers vorgestellt. Im Anschluss daran werden die im Dialogmanager integrierten Fehlerklassen und ihre Dialogstrategien erläutert.

5.8.1. Aufgaben des Dialogmanagers

Als Erstes soll der Dialogmanager den Graphen mit der initialen Nutzereingabe nach Indikatoren durchsuchen, die auf Fehlerklassen hinweisen. Aufgrund der Tatsache, dass diese Abschlussarbeit die Machbarkeit des im Weiteren vorgestellten Konzepts zeigen soll, ist die Anzahl der betrachteten Fehlerklassen auf drei beschränkt, vgl. Unterkapitel 5.6.
Sofern der Dialogmanager einen entsprechenden Indikator identifiziert hat, soll eine Frage generiert werden, mit deren Hilfe der Anwender die aktuelle Situation entweder verifizieren

oder korrigieren kann. Die erzeugte Frage wird anschließend an den Sprachgenerator zur weiteren Verarbeitung weitergeleitet.
Als Nächstes wird die Antwort des Nutzers beim Durchlaufen der Vorverarbeitungsstufen interpretiert und abschließend wird aus dem Ergebnis ein Antwortgraph erstellt, siehe Abbildung 5.2. Die dritte wesentliche Aufgabe des Dialogagenten ist die Extraktion der Nutzerintention aus diesem Antwortgraphen.
Der letzte Schritt ist die Integration der, aus der Antwort, erhaltenen Resultate in den Graphen.

5.8.2. Aufbau des Dialogmanagers

Da sich alle Indikatoren voneinander unterscheiden und jeder Indikator genau einer Fehlerklasse zuzuordnen ist, vgl. Unterkapitel 5.5, muss für jede Fehlerklasse eine eigene Analyse des Graphen vorgenommen werden. Des Weiteren unterscheiden sich die Frageformulierungen je nach Fehlerklasse, da der Nutzer jeweils eine andere Aufgaben lösen soll. Somit müssen die Frageformulierungen individuell erstellt werden. Dies macht wiederum eine separate Interpretation des Antwortgraphen notwendig. Ein Beispiel hierfür bietet folgende Nutzeraussage: „hänge die Wäsche auf den Wäscheständer“. Sollte der Nutzer eine DANN-Anweisung identifizieren, könnten die Worte als solche interpretiert werden und ein Problem wäre beseitigt. War die Aufgabenstellung eine Koreferenz aufzulösen, so ist dies mit der gegebenen Antwort nicht möglich, da diese mehrere Entitäten enthält und somit muss der Nutzer erneut gefragt werden. Dieses Beispiel zeigt, dass je nach Fehlerklasse der Antwortgraph unterschiedlich zu interpretieren ist. Auch die Integration der Ergebnisse des Dialogmanagers in den Graphen ist bei jeder Fehlerklasse anders. Daher sind alle vier Aufgaben des Dialogmanagers aus Abschnitt 5.8.1 für jede Fehlerklasse individuell zu lösen. Aus diesen Gründen bietet sich für den Dialogmanager ein Aufbau an, bei dem die einzelnen Fehlerklassen unabhängig voneinander bearbeitet werden, vgl. Abbildung 5.3. Dieser Ansatz besitzt darüber hinaus den Vorteil, dass der Dialogmanager effizient um weitere Fehlerklasse erweitert werden kann und somit eine wesentliche Anforderung an den Dialogagenten erfüllt.

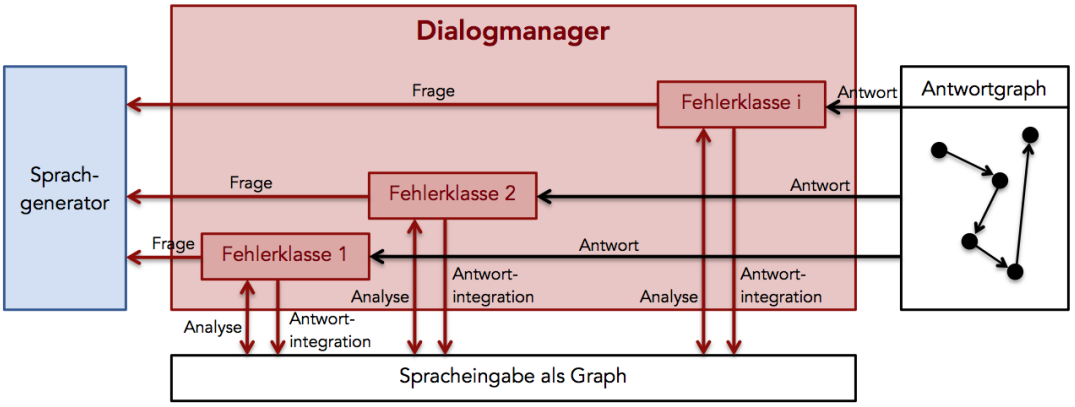


Abbildung 5.3.: Aufbau des Dialogmanagers

In Abbildung 5.3 repräsentieren die einzelnen Fehlerklassen die jeweilige Entscheidungslogik des Dialogmanagers, welche die Probleme der betrachteten Fehlerklasse bearbeitet. Die vier Aufgaben des Dialogmanagers sind in Form von Pfeilen eingetragen. Hierbei steht der mit „Frage“ annotierte Pfeil für die, in einer Fehlerklasse, mit Hilfe des integrierten Textgenerators, erzeugten Fragen. Der Antwortpfeil steht für die Übergabe des Antwortgraphen an die entsprechende Fehlerklasse, welche somit die benötigten Informationen erst aus dem

Antwortgraphen extrahieren muss. Die Suche des Dialogmanagers nach Indikatoren wird durch den Analysepfeil repräsentiert. Das Einfügen der gewonnen Erkenntnisse in den Graphen symbolisiert der mit „Antwortintegration“ annotierte Pfeil. Die Nummerierung der Fehlerklassen von 1, 2, ... i, deutet die einfache Erweiterbarkeit des Dialogmanagers um weitere Fehlerklassen an. Abbildung 5.3 stellt einen detaillierten Ausschnitt aus Abbildung 5.2 dar, somit ist die Bedeutung der Farben identisch. Rot steht für selbst erstellte Komponenten, blau für eingebundene Elemente.
Gemäß den bisherigen Ausführungen besitzt jede vom Dialogmanager behandelte Fehlerklasse eine eigenen Entscheidungslogik bzw. Dialogstrategie, um die jeweiligen Probleme zielführend zu lösen. Für die Entscheidungslogik bietet sich ein graphbasierter Ansatz, wie bei den graphbasierten Dialogmanagern, an, vgl. Unterkapitel 5.4. Daher können die Dialogstrategien, der im Dialogmanager bearbeiteten Fehlerklassen, auch in Form der zugrundeliegenden Graphen dargestellt werden, vgl. Abbildung 5.4, 5.5 und 5.6. Diese Graphen stellen den Ablauf graphbasierter Dialogmanager dar und dürfen nicht mit dem Graphen von PARSE verwechselt werden, der die Nutzeraussage enthält. Die Aufrufe der einzelnen Knoten sind deterministisch vorgegeben, wobei die Kanten den Bedingungen der Zustandsübergänge entsprechen und die Knoten die Dialogaktivitäten beinhalten. Darüber hinaus handelt es sich jeweils um gerichtete Graphen. Die Entscheidungslogik jeder Fehlerklasse setzt hierbei die systemgeführte Dialoginitiative zur Kommunikation ein. Zu beachten ist an dieser Stelle auch, dass der Dialogmanager, nachdem eine Fehlerklasse bearbeitet wurde, die Initiative wieder an PARSE übergibt. Dies geschieht vor dem Hintergrund, dass zunächst die anderen Agenten mit Hilfe der neu integrierten Information weitere Probleme auflösen sollen, bevor der Nutzer erneut gefragt wird. Dies soll die Anzahl der Fragen an den Anwender reduzieren.

5.8.3. Dialogstrategie Wort falsch erkannt

In diesem Abschnitt wird die Dialogstrategie vorgestellt, mit der die Fehlerklasse *Wort falsch erkannt* bearbeitet wird.
Ursprünglich sah der Ansatz zur Beseitigung von Problemen dieser Fehlerklasse wie folgt aus: Der Nutzer wurde gefragt, ob er ein bestimmtes Wort meint. Mit Hilfe der Antwort fand anschließend die Verifizierung oder Korrektur dieses Wortes im Graphen statt. Allerdings hatte diese Vorgehensweise zwei entscheidende Nachteile. Bei längeren Eingaben war für den Anwender nicht mehr nachvollziehbar auf welches Wort sich die Frage bezog. Das zweite Problem offenbart folgendes Beispiel, bei dem ein Ausschnitt der initialen Nutzereingabe vom Spracherkenner wie folgt interpretiert wurde: „get the instant Neil from the fridge“. Die Frage, ob der Anwender „Neil“ meinte, würde dieser in der Regel bejahen, da er das Wort als „meal“ verstanden und damit als korrekt interpretiert hätte. Aus diesen Gründen, wurde der ursprüngliche Ansatz verworfen.
Das erste Problem ließe sich dadurch lösen, dass dem Nutzer statt dem betrachteten Wort, die Textstelle in der Frage genannt wird, die das entsprechende Wort enthält. Somit kann der Anwender nachvollziehen, um welche Textstelle es sich handelt und ob er dieses Wort in dem Kontext sagen wollte. Allerdings würde er die neue Frage mit dem Wort „Neil“ immer noch als korrekt einordnen, da er wieder „meal“ versteht. Um auch dieses Problem zu beseitigen ist es notwendig die Rollen zu tauschen. Das heißt, der Anwender sagt etwas und das System versucht dessen Aussage zu interpretieren. Somit bekommt der Spracherkenner die Gelegenheit das entsprechende Wort in der Antwort des Nutzers neu und korrekt zu klassifizieren. Daher ist es zielführend, dem Anwender in einer Frage die kritische Aussage zu nennen und diesen zu bitten, seine Worte nochmals zu wiederholen.

Die Dialogstrategie für die Fehlerklasse *Wort falsch erkannt* ist in Abbildung 5.4 dargestellt. Der Startknoten ist „Suche nach niedrigen Konfidenzwerten“. Hierbei wird ausschließlich nach Konfidenzwerten des Spracherkenners gesucht, die kleiner als 0,8 sind.

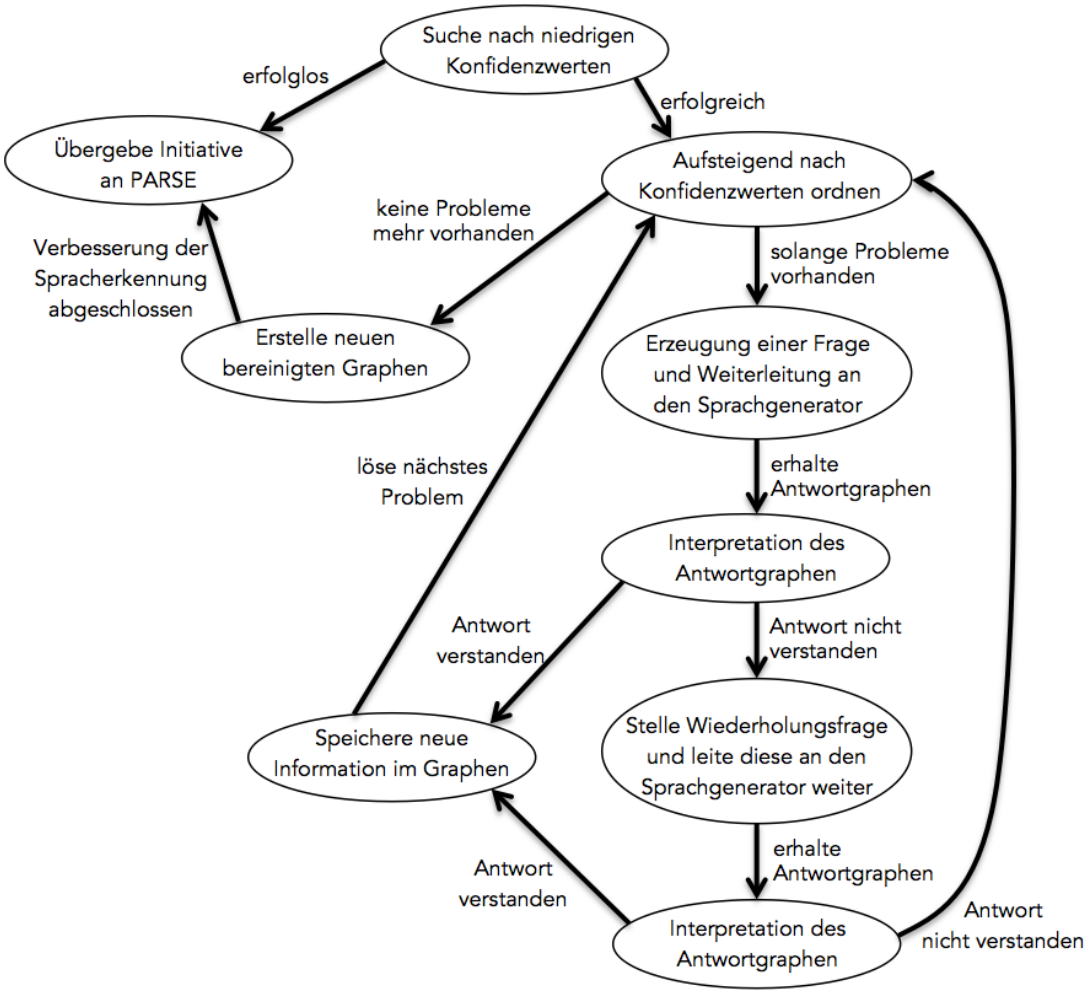


Abbildung 5.4.: Dialogstrategie für *Wort falsch erkannt*

Sofern Wörter, mit niedriger Konfidenz entdeckt werden, findet eine Sortierung in aufsteigender Reihenfolge statt. Anschließend werden diese Wörter, beginnend mit dem Wort, welches die niedrigste Konfidenz besitzt, sukzessive bearbeitet. Hierfür wird zunächst eine Frage generiert, die den Nutzer bittet, eine bestimmte Textstelle mit einem kritischen Wort zu wiederholen. Nach der Weiterleitung dieser Frage an den Sprachgenerator findet die Interpretation des Antwortgraphen statt. Wurde das gesuchte Wort verifiziert oder ersetzt, wird diese Informationen im Graphen gespeichert und anschließend wird mit dem nächsten Knoten fortgefahren. Verifiziert bedeutet in diesem Kontext, dass das im Knoten enthaltene Wort durch die Antwort des Nutzers bestätigt wurde. Ein Wort wird ersetzt, falls die Antwort des Nutzers hierfür ein anderes Wort mit einer Konfidenz von mehr als 0,8 enthält. Für den Fall, dass die Antwort nicht hilfreich war, wird sie einmal in abgewandelter Form wiederholt, um dem Anwender eine zweite Gelegenheit zu bieten, den ursprünglichen Graphen zu korrigieren. Gelingt dies, wird die Information gespeichert und danach das nächste kritische Wort bearbeitet. Schlägt auch dieser Versuch der Problemlösung fehl, wird ebenso mit dem nächsten Fehler fortgefahren, da es unwahrscheinlich ist, dass der Nutzer die Frage beim dritten Mal korrekt beantworten wird. Sobald alle Knoten mit niedriger Konfidenz behandelt wurden, wird ein neuer Graph mit den korrigierten Wörtern erstellt. Dies ist erforderlich, da Knoten eingefügt und entfernt werden können, was die Korrektheit, der von den anderen Agenten und Vorverarbeitungsstufen eingebrachten Informationen, in Form von Knoten, Kanten und Attributen beeinträchtigen

oder verfälschen kann. Durch die Erstellung eines neuen Graphen, beginnen auch die anderen Komponenten mit ihrer Interpretation von Neuem und müssen sich somit nicht um Fehler, die durch Einfüge- oder Löschoperationen entstehen können, kümmern. Nach der Generierung des neuen Graphen ist die Bearbeitung durch den Dialogmanager für diese Fehlerklasse abgeschlossen. Daher wird nun die Initiative vom Dialogagenten an PARSE übergeben. Für den Fall, dass zu Beginn keine Wörter mit niedriger Konfidenz gefunden werden, findet die Übergabe der Initiative an die Rahmenarchitektur unmittelbar statt.

5.8.4. Dialogstrategie Koreferenz falsch aufgelöst

Die Koreferenzauflösung hat das Ziel, dass Wörter die sich auf die gleiche Entität beziehen, sich gegenseitig referenzieren. Entstehen hierbei Fehler, dann referenziert z. B. ein Pronomen auf ein anderes Nomen als vom Nutzer intendiert. Dies ist ein Beispiel für die Fehlerklasse *Koreferenz falsch aufgelöst*. PARSE stellt für diese Probleme Konfidenzen bereit, die angeben mit welcher Wahrscheinlichkeit eine Entität auf eine andere Entität verweist. Es bietet sich daher an, den Anwender zu fragen, ob die Koreferenz richtig aufgelöst wurde, wenn die Konfidenzwerte für zwei oder mehr Zielentitäten größer als 0,8 sind oder es mehrere Zielentitäten gibt, die alle eine niedrige Konfidenz besitzen. Als Zielentität werden die Wörter bezeichnet, auf die sich eine Entität beziehen könnte. Im folgenden Fall entspricht die „Wasserflasche“ einer Zielentität und das Wort „sie“ einer Ursprungsentität, also einer Entität von der die Referenz ausgeht: „Gib mir die Wasserflasche, sie steht auf dem Tisch.“
Die Herausforderung ist an dieser Stelle, eine Frage zu formulieren mit deren Hilfe der Nutzer in der Lage ist, den Fehler zu korrigieren. Ein Problem ist hierbei, dass den meisten Anwendern der Begriff der Koreferenzauflösung unbekannt ist, daher muss die Aufgabe mit anderen Worten umschrieben werden. Eine Möglichkeit wäre die folgende Frage: „Bezieht sich das Wort sie auf die Waschmaschine?“ Allerdings können in längeren Eingaben sowohl die „Waschmaschine“, wie auch das Pronomen „sie“ mehrfach vorkommen. In diesem Fall ist es dem Anwender nicht möglich zu erkennen, auf welche Textstelle sich die Frage bezieht. Zudem merken sich die Menschen in der Regel nicht den genauen Wortlaut ihrer Instruktion. Hilfreich wäre es in diesem Zusammenhang, eine Frage zu formulieren, welche eine Textpassage enthält, die alle vorkommenden Entitäten umfasst. Ein Beispiel dafür wäre folgende Formulierung: „Worauf bezieht sich sie in der folgenden Textstelle? Wäsche in die Waschmaschine und hänge sie“. Diese Frage offenbart ein weiteres Problem. Nur die Textstelle zu erwähnen in der sich die Entitäten befinden reicht nicht aus, es ist darüber hinaus erforderlich den Kontext zu den entsprechenden Entitäten bereitzustellen. Für die erste Entität „Wäsche“ wäre es wünschenswert, über die Handlung die an dieser Entität vorgenommen wird, informiert zu werden und für die letzte Entität wäre es hilfreich zu erfahren, was anschließend mit ihr geschehen soll. Aus diesen Gründen wird die Textpassage zu Beginn erweitert, bis die vorangegangene Verbphrase enthalten ist und am Ende die nächste Nominalphrase eingebunden wurde. Somit ergibt sich folgende Frage an den Anwender: „Worauf bezieht sich sie in der folgenden Textstelle? Tue die Wäsche in die Waschmaschine und hänge sie anschließend auf den Wäscheständer.“
Allerdings existiert noch ein weiteres Problem und zwar könnte in der betrachteten Textpassage auch mehrmals z. B. das Pronomen „sie“ auftauchen. Dies erhöht die Schwierigkeit, eine adäquate Frage zu generieren bzw. die entsprechende Antwort zu verstehen enorm, da man gleiche Ausdrucksformen durchnummerieren oder anders in der Frage hervorheben müsste. Eine Ausdrucksform steht in diesem Zusammenhang für die Wörter, die eine Entität im Text repräsentieren, z. B. sind „die grüne Tasse“, „Tasse“ oder „sie“ Ausdrucksformen für die Entität der grünen Tasse. Aufgrund der Problematik der Identifikation von Entitäten mit gleichen Ausdrucksformen beschränkt sich der hier eingesetzte Dialogmanager auf Fälle, in denen die Ausdrucksform maximal einmal als Zielentität und einmal als Ursprungsentität vorkommt. Hierfür wird die Ursprungsentität in der Frage um das Wort

„letzte“ ergänzt, z. B. „Worauf bezieht sich das letzte sie in folgender Textstelle? Hole eine Tasse, stelle sie auf den Tisch, fülle sie mit Tee.“ Da der Koreferenzauflöser von PARSE derzeit als Zielentität auch nur einmal die gleiche Ausdrucksform erlaubt, ist der hier vorgestellte Ansatz absolut praktikabel. Dennoch sollte in Zukunft die Vorgehensweise dieses Dialogmanagers angepasst werden, da der Koreferenzauflöser der Rahmenarchitektur jederzeit ausgetauscht werden kann.

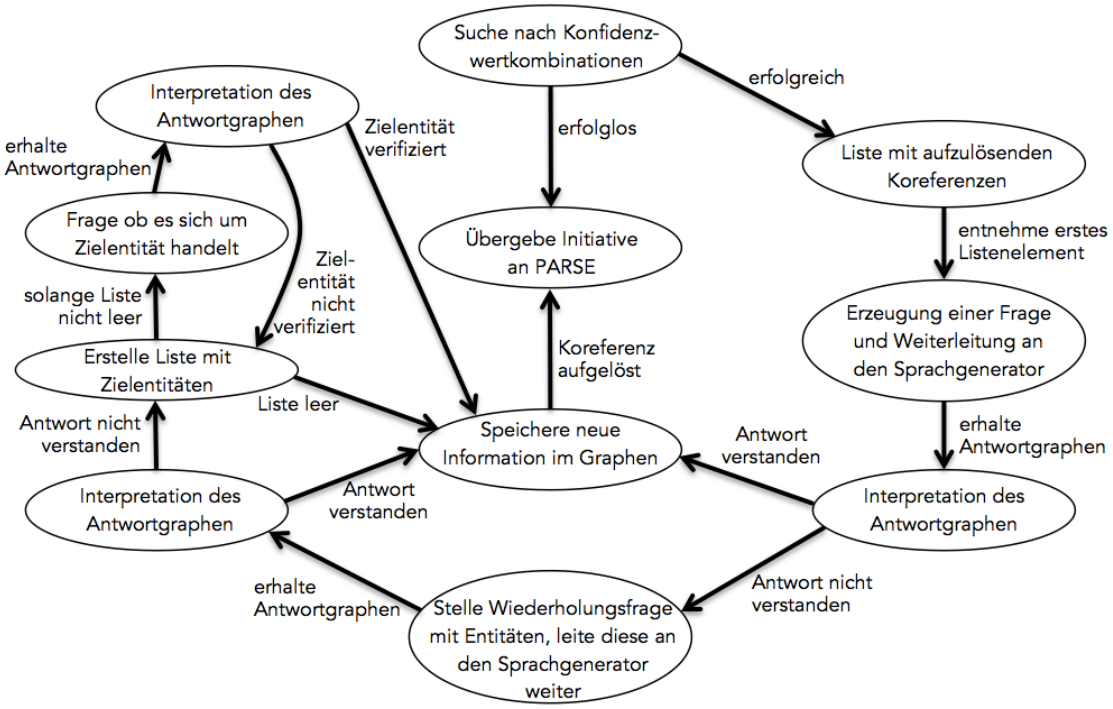


Abbildung 5.5.: Dialogstrategie für *Koreferenz falsch aufgelöst*

Die Dialogstrategie für die Fehlerklasse *Koreferenz falsch aufgelöst* ist in Abbildung 5.5 als gerichteter Graph dargestellt. Für die folgenden Ausführungen steht der Begriff Graph allerdings für die gespeicherte Nutzeraussage der Rahmenarchitektur. Der Knoten „Suche nach Konfidenzwertkombinationen“ repräsentiert den Startknoten. Diese Suche identifiziert in drei verschiedenen Fällen Koreferenzen, die durch den Dialogagenten bearbeitet werden sollen. Verweist eine Entität auf eine Zielentität mit einem Konfidenzwert von eins, so wird diese als referenziertes Objekt angesehen, sofern keine weitere Zielentität mit einem Konfidenzwert von eins existiert. Liegen mehrere Zielentitäten mit einer Konfidenz von eins vor, so charakterisiert dies den ersten Fall, in dem der Dialogagent zur Auflösung der Koreferenz eingesetzt wird. Besitzt eine Ursprungsentität mehrere Zielentitäten, deren Konfidenzwerte über einem Schwellenwert liegen und existiert gleichzeitig keine Zielentität mit einem Konfidenzwert von eins, so definiert dies den zweiten Fall, in dem der Nutzer befragt werden soll. Der vertrauenswürdige Wert wird in dieser Abschlussarbeit auf 0,8 bzw. 80 Prozent Wahrscheinlichkeit, dass eine Entität auf eine andere Entität verweist, festgelegt. Dieser Wert resultiert aus Erfahrungswerten, die während der Erstellung dieses Dialogagenten gewonnen wurden und somit auf einer sehr kleinen Stichprobe. In Zukunft sollte dieser Wahrscheinlichkeitswert allerdings durch Verfahren des maschinellen Lernens und somit der Auswertung einer großen Menge an Trainingsdaten bestimmt werden, um eine optimale Auswahl der zu untersuchenden Koreferenzen zu erhalten. Existieren mehrere Zielentitäten und besitzt hiervon keine einen Konfidenzwert von mehr als 0,8, soll ebenfalls der Dialogagent zur Koreferenzauflösung eingesetzt werden. Dieses Kriterium definiert somit den dritten Fall.

Hat die eben beschriebene Suche eine fragwürdige Koreferenzauflösung entdeckt, so wird diese in eine Liste mit kritischen Koreferenzen eingefügt. Der Abbildung 5.5 ist zu entnehmen, dass aus dieser Liste nur das erste Element bearbeitet wird. Dies hat den Hintergrund, dass der Dialogagent bei PARSE nur dann zum Einsatz kommen soll, wenn alle anderen Versuche den Nutzer zu verstehen gescheitert sind. Daher sollen nach der Auflösung einer Koreferenz die anderen Agenten mit Hilfe dieser neu gewonnen Information die Gelegenheit bekommen, weitere Probleme im Graphen zu beseitigen und somit möglicherweise auch weitere Koreferenzen aufzulösen. Gelingt dies nicht in allen Fällen, wird der Dialogagent erneut aktiv und kann dann die nächste kritische Koreferenzauflösung bearbeiten.
Um nun ein Problem der Fehlerklasse *Koreferenz falsch aufgelöst* zu bearbeiten, wird der Anwender gefragt, worauf sich eine bestimmte Entität bezieht. Falls die Antwort dem Antwortgraphen entnommen werden kann, findet eine Speicherung der neuen Information im Graphen statt und PARSE erhält wieder die Initiative. Gelingt dies nicht, wird eine zweite Frage generiert, die zusätzlich noch die gesuchten Entitäten aufzählt und dadurch versucht, den Nutzer bei der Problemlösung zu unterstützen. Schlägt auch dieser Schritt fehl, wird einzeln zu jeder Zielentität gefragt, ob sich die Ursprungsentität in der genannten Textpassage darauf bezieht. Sobald der Anwender dies für eine Zielentität bestätigt, werden alle gewonnen Informationen im Graphen gespeichert und die Rahmenarchitektur aufgerufen. Hierbei kann die Situation auftreten, dass der Nutzer die Koreferenzbeziehung für alle Zielentitäten verneint. In diesem Fall wird der menschlichen Expertise vertraut und davon ausgegangen, dass die gesuchte Entität tatsächlich nicht in der Liste der Zielentitäten enthalten ist. Daher wird auch diese Information im Graphen gespeichert und anschließend die Initiative an PARSE übergeben. Aufgrund dieser Vorgehensweise findet in allen Fällen eine Auflösung der Koreferenz statt.

5.8.5. Dialogstrategie DANN-Anweisung nicht erkannt

Die Fehlerklasse *DANN-Anweisung nicht erkannt* tritt auf, falls eine WENN-Bedingung ohne DANN-Anweisung vorkommt. Das Problem, derartige Indikatoren im Graphen zu entdecken, kann wie folgt gelöst werden. Falls zwischen zwei WENN-Bedingungen oder einer WENN-Bedingung und dem Ende des Graphen keine DANN-Anweisung steht, so ist dies ein Indikator dafür, dass diese nicht erkannt wurde. Somit können Probleme dieser Fehlerklasse relativ einfach identifiziert werden.
Die Herausforderung liegt daher darin, dem Anwender das zu lösende Problem verständlich zu beschreiben, denn WENN-DANN-SONST-Bedingungen sind nicht allen Menschen bekannt. Zudem kann keine Frage gestellt werden, in der die Bedingung genannt wird und anschließend nach der DANN-Anweisung gefragt wird, da die als WENN-Bedingung identifizierten Knoten des Graphen auch Teile der oder die ganze DANN-Anweisung enthalten könnten. Aufgrund dieser Problematik werden verschiedene Frageformulierungen eingesetzt, um dem Kenntnisstand der verschiedenen Nutzer gerecht zu werden.

Die Dialogstrategie dieser Fehlerklasse sieht daher wie in Abbildung 5.6 dargestellt aus: Zunächst wird nach fehlenden DANN-Anweisungen gesucht. Das Verfahren für diesen Suchprozess wurde bereits im ersten Absatz dieses Abschnitts erläutert. Sobald ein entsprechender Indikator im Graphen entdeckt wurde, beginnt die Erzeugung der ersten Fragen. Hierbei wird eine vom Anwender erwähnte Textstelle wiederholt und dieser gebeten die DANN-Anweisung erneut einzusprechen. Dadurch sollen bereits alle Nutzer, die mit dem Konzept der WENN-DANN-SONST-Bedingung vertraut sind, die Frage beantworten können. Die wiederholte Textpassage enthält die betrachtete WENN-Bedingung und die darauf folgenden Wörter des Graphen bis zur nächsten WENN-Bedingung. Diese Textstelle ist in allen zu dieser Fehlerklasse generierten Fragen enthalten, um dem Anwender seine eigene Aussage in Erinnerung zu rufen. Falls eine DANN-Anweisung mit Hilfe der Antwort identifiziert werden kann, erfolgt das Einfügen der neu gewonnenen Information im

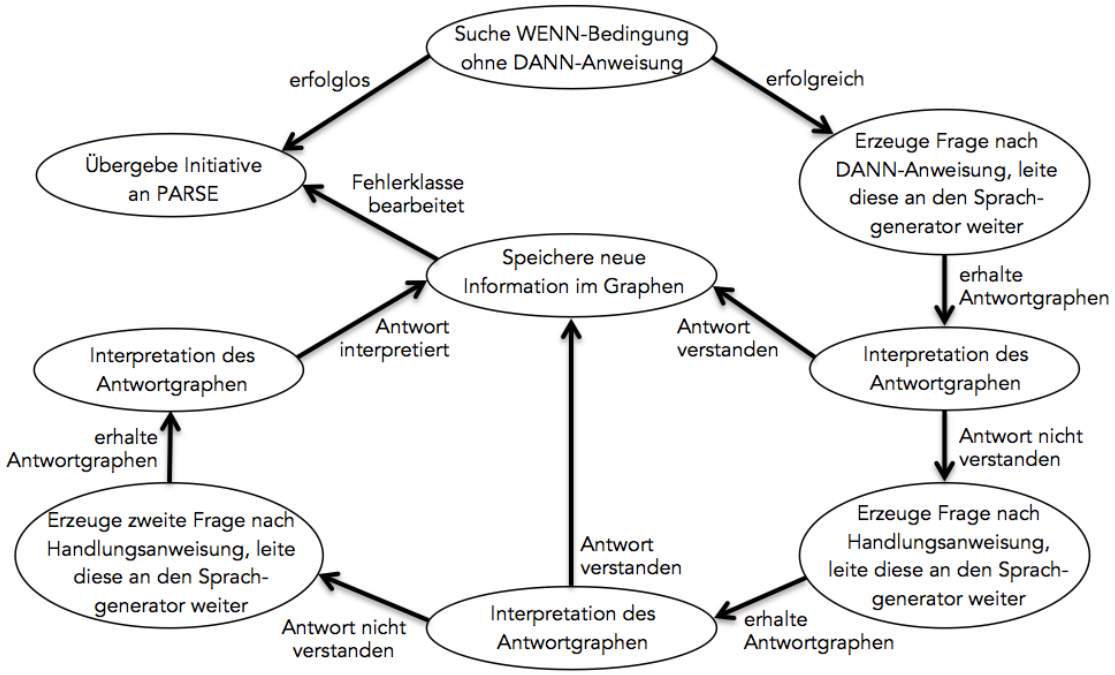


Abbildung 5.6.: Dialogstrategie für *DANN-Anweisung nicht erkannt*

Graphen. War die Antwort nicht hilfreich, wird eine zweite Frage generiert. Dieser wird der Hinweis beigefügt, dass die Textpassage eine Bedingung enthält und der Nutzer die Handlungsanweisung wiedergeben soll, die auszuführen ist, falls die Bedingung eingetreten ist. Löst die Antwort auf diese Frage das Problem, wird die neue Information in den Graphen eingefügt. Gelingt keine Beseitigung des Fehlers, wird eine dritte, letzte Frage formuliert. Diese enthält den gleichen Inhalt, wie die zweite Frage, allerdings in anderen Worten ausgedrückt. Das Ziel ist hierbei, dem Nutzer die Aufgabe auf verschiedenen Wegen zu erklären, damit dieser zumindest bei einer Variante die Intention der Frage versteht. Falls die DANN-Anweisung aus dem Antwortgraphen extrahiert werden kann, wird diese anschließend im Graphen vermerkt. Schlägt dies erneut fehl, wird dieses Problem für nicht lösbar erklärt, damit es der Dialogagent beim nächsten Aufruf nicht erneut aufgreift und den Anwender somit in eine Endlosschleife verwickelt. In beiden Fällen werden dem Graphen neue Informationen hinzugefügt, was in Abbildung 5.6 mit dem Pfeil „Antwort interpretiert“ verdeutlicht wird. Nachdem neue Informationen gespeichert wurden, findet stets eine Rückgabe der Initiative an die Rahmenarchitektur statt, damit die anderen Agenten mit Hilfe der neuen Annotationen weitere Fehler beseitigen können. Dadurch wird bei jedem Aufruf des Dialogagenten maximal ein Problem dieser Fehlerklasse beseitigt. Wird in einer kompletten Nutzereingabe keine WENN-Bedingung ohne DANN-Anweisung entdeckt, findet eine unmittelbare Rückgabe der Initiative an PARSE statt.
Bei den Frageformulierungen zu dieser Fehlerklasse ist festzuhalten, dass wenn der Nutzer nicht versteht, was eine Bedingung ist, keine sinnvolle Auflösung des Problems erreicht werden kann.

6. Implementierung

Dieses Kapitel stellt die Implementierung des Dialogagenten und die hierbei verwendeten Konzepte und eingesetzten Werkzeuge vor. Die zugrundeliegende Programmiersprache ist Java. Zunächst werden die selbst erstellten Komponenten des Dialogagenten sowie die Hilfsklassen beschrieben, da diese im Dialogmanager eingesetzt werden. Anschließend wird das Entwurfsmuster des Dialogmanagers erläutert und dessen Implementierung vorgestellt. Danach findet eine Beschreibung der Klassen statt, welche die Fehlerklassen bearbeiten.

6.1. Komponenten des Dialogagenten

Gemäß Abbildung 5.2 bindet der Dialogagent die Vorverarbeitungsstufen von PARSE ein. Deren Quellcode wird an dieser Stelle allerdings nicht beschrieben, da diese nicht verändert wurden und lediglich das Ergebnis dieser Vorverarbeitungsstufen, der Antwortgraph, für den Dialogagenten relevant ist. Die vorgestellten Hilfsklassen besitzen alle statische Methoden, damit keine Objektinstantiierung für deren Aufruf notwendig ist. Somit ist die Struktur dieser Klassen vergleichbar mit der der Hilfsklassen (Package java.util) von Java. Im Folgenden werden die Klassen des Sprachgenerators und des Rekorders vorgestellt sowie auf verschiedene Hilfsklassen eingegangen. Abschließend findet noch eine kurze Beschreibung der vom Dialogagenten erstellten Audiodateien statt.

6.1.1. Sprachgenerator

Der Sprachgenerator des Dialogagenten wird mit Hilfe von zwei Klassen realisiert. Die erste Klasse `Synthesizer` enthält nur die statische Methode `enunciateQuestion`, welcher eine Zeichenkette, i. d. R. die Frage, die dem Nutzer gestellt werden soll, übergeben wird. In dieser Methode wird nun die zweite Klasse des Spracherkenners `WatsonTTS` instantiiert und anschließend die erhaltene Zeichenkette an die Methode `synthesizeQuestion` dieser Klasse übergeben. Im Konstruktor von `WatsonTTS` wird eine Verbindung zu Watson[1] aufgebaut. Watson ist ein Werkzeug von IBM, welches unter anderem für die Sprachausgabe von Text oder die Transformation einer Audiodatei in Text eingesetzt werden kann. Für Ersteres wird das Werkzeug in der Methode `synthesizeQuestion` eingesetzt. Die Klasse `Synthesizer` fungiert in diesem Zusammenhang als Hilfsklasse, die nicht instantiiert werden muss.

[1] Watson Developer Cloud https://www.ibm.com/watson/developercloud/text-to-speech.html abgerufen am 05.04.2017

Das Einbinden von Watson zur Sprachausgabe hat den Nachteil, dass der Dialogagent auf eine stabile Internetverbindung angewiesen ist, um mit dem Nutzer zu interagieren. Allerdings müssen diverse andere Komponenten von PARSE für die Bearbeitung ihrer Aufgabenstellungen ebenso Online-Ressourcen einbinden, daher stellt dies keine zusätzliche Anforderung an die Rahmenarchitektur dar.

6.1.2. Rekorder

Um die Antwort des Nutzers als Audiodatei zu erhalten, muss diese aufgenommen werden. Hierfür ist neben einem Mikrofon auch Software nötig, welche die vom Mikrofon erhaltenen Signale in einer Datei ablegt. Der Dialogagent kombiniert hierfür eine angepasste Quellcodevorlage[2] von Sun Microsystems zur Aufzeichnung von Tonsignalen mit einem Kodierer[3] für FLAC[4]-Dateien. FLAC (Free Lossless Audio Codec) ist ein Audioformat für Audiodateien.
Die Funktionalität des Rekorders wird von den vier Klassen `GainUserAnswer`, `VoiceRecorder`, `VoiceListeningThread` und `FlacFileEncoder` realisiert. In der Hilfsklasse `GainUserAnswer` wird der `VoiceRecorder` in der statischen Methode `getUserAnswer` instantiiert. Der `VoiceRecorder` bietet dem Anwender eine graphische Benutzerschnittstelle, mit deren Hilfe dieser eine Tonaufnahme starten und beenden kann. Das Starten einer Tonaufnahme erzeugt den `VoiceListeningThread`, welcher mit Hilfe des `FlacFileEncoder` eine Audiodatei erstellt und diese unter dem in einer Konfigurationsdatei hinterlegten Pfad speichert.

6.1.3. Hilfsklassen

Die Hilfsklasse `GainUserAnswer` erfüllt in der Methode `getUserAnswer` neben dem Starten des `VoiceRecorder`, vgl. Abschnitt 6.1.2 noch weitere Aufgaben. Eine ist die Abfrage des Pfads unter dem die Antwort liegt. Danach wird die Klasse `BuildGraph` mit diesem Pfad instantiiert. Diese Klasse gibt über die Methode `getGraph` den Antwortgraphen des Nutzers aus. Dieser entspricht dem Rückgabewert der Methode `getUserAnswer`.
Neben dieser Hilfsklasse verfügt der Dialogagent noch über die Klasse `GraphOperations`, welche Methoden für allgemeine Operationen auf dem Graphen enthält. Die beiden meistbenutzten Methoden sind `getSubsequentNounNode` und `getPreviousVerbNode`. Letztere hat die Aufgabe das erste Verb vor dem übergebenen Wort zurückzuliefern. Hierzu wird der Graph ab dem übergebenen Knoten rückwärts durchsucht, bis die annotierten Phrase einer Verbalphrase entspricht. Anschließend wird dieser Knoten zurückgegeben. Die Methode `getSubsequentNounNode` soll die nächste Nominalphrase nach dem übergebenen Knoten zurückliefern. In diesem Kontext ist zu beachten, dass Nominalphrasen häufig aus mehreren Wörtern bestehen. Daher muss zunächst geprüft werden, ob das übergebene Wort bereits in einer Nominalphrase enthalten ist. Trifft dies zu, wird über die nächsten Knoten des Graphen iteriert, bis das aktuelle Wort keiner Nominalphrase mehr entspricht. Danach wird weiter iteriert, bis zum letzten Wort der folgenden Nominalphrase und dieses anschließend zurückgegeben. Falls das übergebene Wort in keiner Nominalphrase enthalten ist, wird nur der letzte Schritt ausgeführt.

6.1.4. Audiodateien des Dialogagenten

Der Dialogagent speichert die Fragen des Systems und Antworten des Nutzers mit Hilfe des Sprachgenerator bzw. des Rekorders in Audiodateien ab. Diese besitzen alle einen

[2] SimpleSoundCapture Example http://www.java2s.com/Code/Java/Development-Class/Thisisasimpleprogramtorecordsoundsandplaythemback.htm abgerufen am 06.04.2017
[3] javaFlacEncoder https://sourceforge.net/projects/javaflacencoder/files/ abgerufen am 06.04.2017
[4] flac https://xiph.org/flac/ abgerufen am 06.04.2017

Zeitstempel im Dateinamen, damit die Aussagen anschließend ohne Suchaufwand in der korrekten Reihenfolge wiedergeben werden können. Durch die Aufzeichnung der Dialoge kann der in Unterkapitel 4.6 vorgestellte Korpus um weitere Audiodateien und erstmals um Dialoge ergänzt werden.

6.2. Entwurfsmuster des Dialogmanagers

Ein Ziel des Dialogmanagers ist die effektive Bearbeitung von Fehlern im Graphen. Diese Aufgabe wird jeweils für einzelne Fehlerklassen realisiert, vgl. Unterkapitel 5.8. Daher sind weitere Ziele die unabhängige Bearbeitung der einzelnen Problemstellungen und die effiziente Erweiterbarkeit des Dialogmanagers um weitere Fehlerklassen. Darüber hinaus soll der Rechenaufwand möglichst gering sein, da das System in Echtzeit Ergebnisse, sprich Fragen an den Nutzer, liefern soll. Zu beachten ist an dieser Stelle auch, dass der Dialogagent regelmäßig von PARSE neu aufgerufen wird und zwar immer dann, wenn die anderen Agenten keine weiteren Probleme mehr lösen können. Zur Speicherung von Informationen zwischen zwei Aufrufen steht dem Dialogagenten lediglich der Graph der Rahmenarchitektur, welcher die Nutzeraussage enthält, zur Verfügung.
Um diese Ziele zu erreichen und die erwähnten Einschränkungen zu berücksichtigen, bietet sich der Einsatz eines geeigneten Entwurfsmusters, zur Strukturierung des Quellcodes, an. Ein Entwurfsmuster, welches die vorgestellten Kriterien erfüllt ist die Zuständigkeitskette (engl. Chain of Responsibility)[Gei15, S. 211]. Im Folgenden findet eine Beschreibung dieses Entwurfsmusters statt. Danach wird das vorgestellte Konzept auf den Dialogmanager übertragen. Abschließend wird die Implementierung der Zuständigkeitskette im Dialogmanager beschrieben.

6.2.1. Entwurfsmuster Zuständigkeitskette

Bei der Zuständigkeitskette wird eine Anfrage an eine Kette von Objekten gesendet. Dabei findet solange eine Weiterleitung der Anfrage statt, bis ein Objekt diese bearbeitet. Bei diesem Entwurfsmuster wird somit das Objekt, welches eine Anfrage erzeugt, von dem Objekt, welches diese beantwortet, entkoppelt. Das bedeutet, dass das anfragende Objekt nicht weiß, ob und von wem eine Anfrage bearbeitet wird. Ebenso ist dem bearbeitenden Objekt das anfragende Objekt unbekannt. Darüber hinaus kennen beide nicht die Struktur beziehungsweise die Länge der Kette. Den Bearbeitern sind lediglich ihre direkten Nachfolger bekannt. Diese lose Kopplung spart Objektreferenzen und verringert Abhängigkeiten zwischen den einzelnen Objekten. Mit einem entsprechenden Konfigurationsmechanismus kann die Kette sogar zur Laufzeit erweitert oder verkürzt werden und lässt sich somit dynamisch an neue Anforderungen anpassen. Die Zuständigkeitskette kann auch aus lediglich einer Bearbeiterklasse bestehen.[Gei15, S. 211ff.]

6.2.2. Übertragung der Zuständigkeitskette auf den Dialogmanager

Wird der Dialogmanager mit Hilfe einer Zuständigkeitskette realisiert, so hat dieser folgenden Aufbau. Die bearbeitenden Objekte entsprechen den zu lösenden Fehlerklassen. PARSE repräsentiert das anfragende Objekt und der Graph mit seinen Fehlern kann als Anfrage angesehen werden, die es zu bearbeiten gilt. Der Graph wird somit von Fehlerklasse zu Fehlerklasse weitergeleitet, bis eine einen Indikator entdeckt und die Bearbeitung des Fehlers startet. Hierbei ist PARSE gemäß den Eigenschaften der Zuständigkeitskette nicht bekannt, von welcher Fehlerklasse ein Problem bearbeitet wird. Nach einer Fehlerbehandlung findet die Übergabe der Initiative an die Rahmenarchitektur, wie in den Dialogstrategien in Unterkapitel 5.8 beschrieben, statt, d. h. es werden keine weiteren Fehlerklassen in der Kette aufgerufen. Die anderen Agenten von PARSE versuchen nun anhand der vom Dialogagenten eingefügten Informationen, weitere Probleme im Graphen

zu beseitigen. Sobald diese keine weiteren Veränderungen mehr am Graphen vornehmen, wird der Dialogagent erneut aufgerufen und der Graph wieder durch die Kette geschickt. Falls die Fehlerklassen des Dialogmanagers keine Indikatoren finden, wird der Graph nicht verändert und die Initiative geht automatisch wieder zurück an die Rahmenarchitektur. Ob dies geschieht, kann vom Dialogmanager aufgrund des gewählten Entwurfsmusters allerdings nicht festgestellt werden. Diese Tatsache stellt jedoch kein Problem dar, da dies bedeutet, dass in den eingebunden Fehlerklassen kein Fehler auftrat und der Graph somit aus Sicht des Dialogmanagers korrekt ist. Durch den Einsatz der Zuständigkeitskette können die Bearbeitungen der einzelnen Fehlerklassen unabhängig voneinander implementiert werden, wodurch die Erweiterbarkeit des Dialogmanagers extrem erleichtert wird.

6.2.3. Implementierung der Zuständigkeitskette im Dialogmanager

In diesem Abschnitt wird die Implementierung des Entwurfsmusters Zuständigkeitskette anhand eines UML-Klassendiagramms erläutert, vgl. Abbildung 6.1. Dieses Diagramm stellt einen Ausschnitt der Implementierung dar. Es wurden nur die Klassenattribute eingezeichnet, die für die folgenden Ausführungen Relevanz besitzen. Ferner sind einige Variablennamen für die bessere Darstellbarkeit anders bezeichnet als dies im Quellcode der Fall ist. Die weiteren Beschreibungen erläutern die Realisierung der Zuständigkeitskette im Kontext dieser Abschlussarbeit. Die Implementierung orientiert sich an Kapitel 4.1 aus dem Buch „Entwurfsmuster: Das umfassende Handbuch“[DB01].

Der erste Schritt bei der Implementierung des Entwurfsmusters ist die Erstellung einer Schnittstelle, welche die notwendigen Methoden für die Realisierung einer Zuständigkeitskette definiert. In Abbildung 6.1 repräsentiert die Klasse `IDefectCategory` diese Schnittstelle. Die abstrakte Klasse `AbstractDefectCategory` implementiert `IDefectCategory` und füllt die beiden Schnittstellenmethoden mit Anweisungen. Darüber hinaus besitzt `AbstractDefectCategory` noch zwei weitere abstrakte Methoden. In `processDefectCategory` wird geprüft, ob die Methode `analyseGraph` wahr zurückliefert. Falls dem so ist, wird `solveDefectCategory` aufgerufen und die Fehlerklasse bearbeitet. Dadurch wird die Zuständigkeitskette nicht weiter durchlaufen und die Initiative des Systems wird an PARSE zurückgegeben, nachdem die Bearbeitung der Fehlerklasse abgeschlossen ist. Da `analyseGraph` nur dann wahr zurückgibt, falls für die betrachtete Fehlerklasse Indikatoren gefunden wurden, findet somit ein Aufruf von `solveDefectCategory` nur in diesem Fall statt. Aufgrund dieser Vorgehensweise wird für jede implementierte Fehlerklasse die Methode `analyseGraph` ausgeführt, bis eine dieser Klassen Indikatoren entdeckt hat. Somit sollte diese Methode nur die Funktionalität für das Entdecken der Indikatoren enthalten, um die Laufzeit beim Durchlaufen der Zuständigkeitskette nicht unnötig zu verlängern. Liefert `analyseGraph` falsch zurück, wird geprüft, ob der Klassenvariablen `nextCategory` ein Wert zugeordnet ist. Trifft dies zu, wird die nächste Fehlerklasse aufgerufen und dieser hierbei der Graph übergeben. In diesem Kontext repräsentiert `nextCategory` die nächste Fehlerklasse. Falls diese Klassenvariable keinen Wert besitzt, heißt das, dass die Zuständigkeitskette vollständig durchlaufen wurde und keine Fehlerklasse Indikatoren gefunden hat. Somit ist der Graph gemäß der Implementierung des Dialogagenten korrekt und die Rahmenarchitektur bekommt wieder die Initiative.
Für die Initialisierung der Zuständigkeitskette ist noch eine Klasse notwendig, die die Kette erstellt. Diese Aufgabe wird von `DialogAgent` übernommen. Für die Erzeugung der Kette werden in `init` alle zu bearbeitenden Fehlerklasse instantiiert. Dabei wird der ersten Fehlerklasse mit der Methode `nextDefectCategory` die zweite Fehlerklasse übergeben und dort in Form der geerbten Klassenvariable `nextCategory` gespeichert. Auf die gleiche Art und Weise wird der zweiten Fehlerklasse die dritte Fehlerklasse übergeben, der Dritten die Vierte, usw. Dadurch entsteht die zu durchlaufende Zuständigkeitskette. Die Klassen-

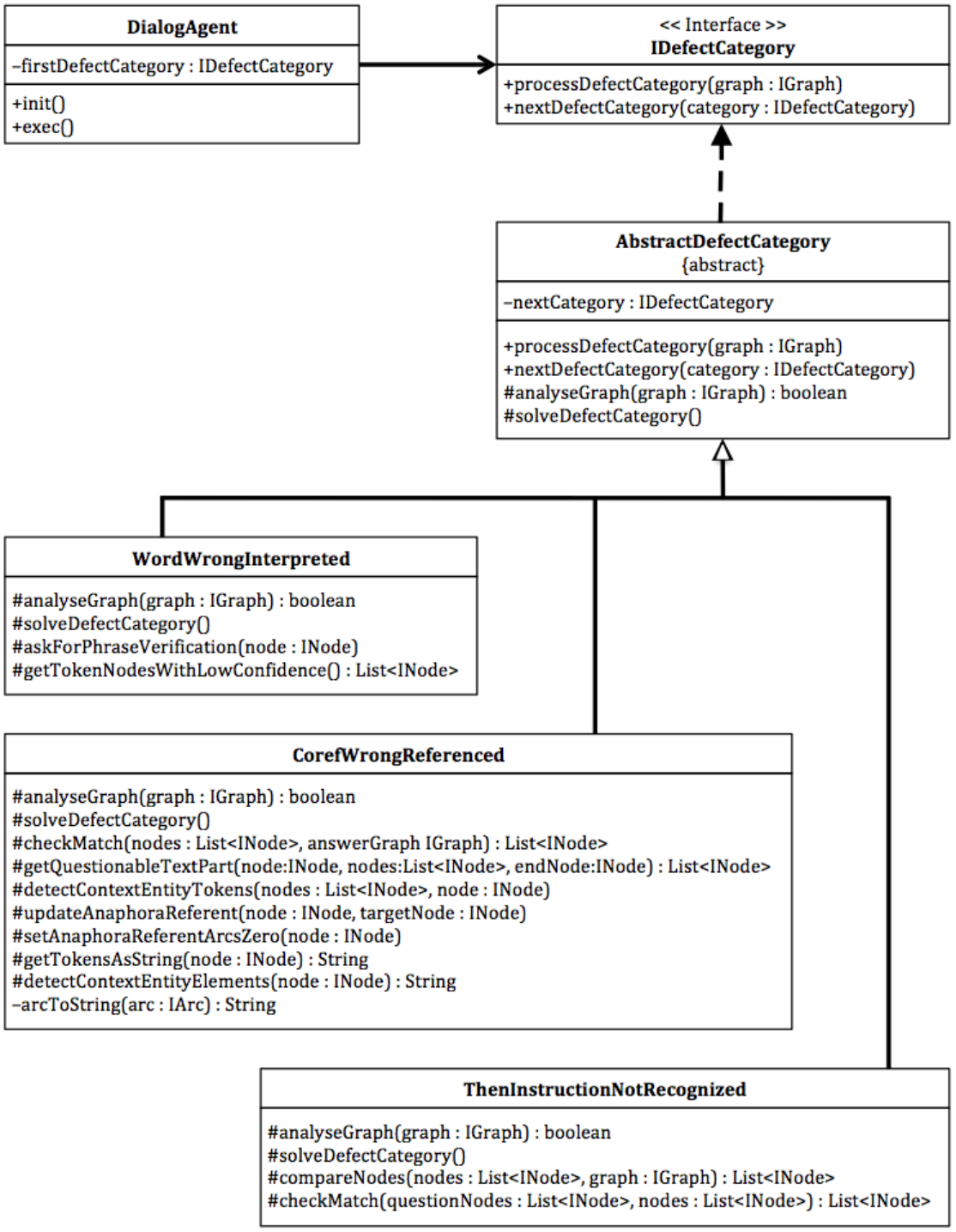


Abbildung 6.1.: UML Diagramm des Dialogmanagers

variable `firstDefectCategory` im `DialogAgent` repräsentiert in diesem Zusammenhang die erste Fehlerklasse der Kette. In der Methode `exec` wird die Methode `processDefectCategory` von `firstDefectCategory` aufgerufen und somit der ersten Fehlerklasse der Zuständigkeitskette der Graph zur Bearbeitung übergeben.

6.3. Implementierte Fehlerklassen des Dialogmanagers

Dieses Unterkapitel beschreibt die Implementierungen der drei Fehlerklassen *Wort falsch erkannt*, *Koreferenz falsch aufgelöst* und *DANN-Anweisung nicht erkannt*. Hierbei wer-

den die Methoden und ihre Funktionalität für die Beseitigung von Fehlern der einzelnen Fehlerklassen vorgestellt. Das UML-Diagramm in Abbildung 6.1 enthält alle Methoden, welche die Klassen zur Bearbeitung der Fehlerklassen besitzen.

6.3.1. Wort falsch erkannt

Die Fehlerklasse *Wort falsch erkannt* wird in `WordWrongInterpreted` bearbeitet. Da diese Klasse von `AbstractDefectCategory` erbt, muss sie die Methoden `analyseGraph` und `solveDefectCategory` implementieren. Die Suche nach Indikatoren dieser Fehlerklasse erfolgt in `analyseGraph`. Hierfür wird eine Liste mit Wörtern erstellt, denen der Spracherkenner einen niedrigen Konfidenzwert zugewiesen hat. In diesem Zusammenhang bedeutet niedrig ein Wert von kleiner 0,8. Dieser Schwellenwert wurde aufgrund von Beobachtungen beim Erstellen des Dialogagenten festgelegt. Aus wissenschaftlicher Sicht wäre die Auswertung des Korpuses von PARSE geeigneter, um diesen Wert zu bestimmen. Hierfür sind alle vom Spracherkenner falsch erkannten Wörter manuell zu annotieren. Anschließend müsste der Konfidenzwert ermittelt werden, welcher die meisten Wörter korrekt klassifiziert. Dieser Wert könnte dann als geeigneter Schwellenwert herangezogen werden. Allerdings ist diese Vorgehensweise sehr aufwändig und selbst bei der Berücksichtigung des kompletten Korpuses wäre die Stichprobe aus wissenschaftlicher Sicht noch sehr klein. Der entscheide Grund, der gegen dieses Verfahren spricht, ist jedoch, dass der Spracherkenner von Watson keine deterministischen Konfidenzwerte liefert. Daher müssten die Audiodateien mehrfach an Watson geschickt werden, um einen Mittelwert für die jeweiligen Konfidenzen der einzelnen Wörter zu erhalten. Anschließend würde dieser Mittelwert zur Bestimmung des Schwellenwertes herangezogen. In diesem Zusammenhang wären die Werte, die Watson selbst für geeignet hält ein Wort als korrekt einzustufen, der beste Ansatz um einen validen Schwellenwert zu erhalten. Da Watson allerdings proprietär ist, steht dieser Wert nicht zur Verfügung. Aus diesen Gründen wird der Schwellenwert, gemäß den gemachten Beobachtungen, dass ein Konfidenzwert von größer gleich 0,8 ein relativ guter Indikator für die Korrektheit eines Wortes ist, auf diesen Wert festgelegt. Um diesen Schwellenwert zukünftig dennoch ohne Eingriffe in den Quellcode verändern zu können, wird dieser, genau wie alle anderen Schwellenwerte, aus einer Konfigurationsdatei geladen und kann daher sogar zur Laufzeit angepasst werden.
Enthält die erstellte Liste nach der Suche im Graphen mindestens ein Wort mit niedriger Konfidenz, d. h. kleiner als 0,8, liefert die Methode `analyseGraph` wahr zurück und dadurch wird Methode `solveDefectCategory` aufgerufen. In dieser wird nun jedes in der Liste enthaltene Wort an die Methode `askForPhraseVerificiation` übergeben, welche anschließend den Nutzer bittet einen bestimmten Teil seiner Aussage zu wiederholen. Hierfür ist zunächst die Generierung einer Frage erforderlich. Diese besteht aus einem vorgegebenen Frageteil und dem dynamisch generierten Textabschnitt, in dem das zu verifizierende Wort enthalten ist. Mit den beiden Methoden `getPreviousVerbNode` und `getSubsequentNounNode` aus der Hilfsklasse `GraphOperations`, vgl. Abschnitt 6.1.3, wird das erste und letzte Wort der benötigten Textstelle aus dem Graphen extrahiert und anschließend diese beiden und alle dazwischenliegenden Wörter zur Textpassage zusammengefasst. Dieser Prozess der Frageformulierung in Textform entspricht dem Textgenerator, der, gemäß Unterkapitel 5.7, in den Dialogmanager integriert ist. Die Ausgabe der Frage erfolgt durch den in Abschnitt 6.1.1 beschriebenen Sprachgenerator, indem der Methode `enunciateQuestion` des `Sysnthesizer` die Frage als Zeichenkette übergeben wird. Nach der Aufzeichnung der Antwort mit Hilfe der Methode `getUserAnswer` der Klasse `GainUserAnswer`, vgl. Abschnitt 6.1.2, wird die Antwort des Nutzers durch den Spracherkenner und die anderen im Dialogagenten eingebunden Vorverarbeitungsstufen weitergereicht, bis die Methode `askForPhraseVerificiation` den Antwortgraphen erhält.
Um den Antwortgraphen zu interpretieren, implementiert die Methode `askForPhraseVerificiation` elf verschiedene Fälle. Eine grundlegende Annahme für die Extraktion einer

sinnvollen Antwort aus dem Antwortgraphen ist, dass der Antwortgraph ähnlich viele Wörter enthält, wie die in der Frage enthaltene Textpassage die der Anwender wiederholen sollte. Weicht die Anzahl der enthaltenen Knoten um mehr als einen Knoten ab, ist dies ein Indiz dafür, dass der Nutzer die Frage falsch verstanden und die Antwort entweder verkürzt oder um eigene Worte verlängert hat. Dennoch wird diese Situation als einer der elf Fälle bearbeitet, damit auch mit diesen Aussagen Probleme des Spracherkenners gelöst werden können. Hierfür wird geprüft, ob ein Knoten i im Antwortgraphen das gleiche Wort wie der Knoten j vor dem betrachteten Knoten im Graphen enthält. Trifft dies zu und entsprechen sich die Wörter im Knoten i + 2 des Antwortgraphen und j + 2 im Graphen auch, kann dies bedeuten, dass der Nutzer in diesem Teil des Antwortgraphen versucht hat die ursprüngliche Textstelle zu wiederholen. Liegt die Konfidenz des Knoten i + 1 im Antwortgraphen daher über 0,8, so wird der Knoten j + 1 im Graphen entweder verifiziert, falls dieser das gleiche Wort enthält oder falls nicht durch i + 1 ersetzt.
Ein weiterer Sonderfall sind Fragen mit sehr kurzen Textpassagen, d. h. mit drei oder weniger Wörtern. Da hierbei nicht genug Knoten für die Fallunterscheidungen aus Tabelle 6.1 zur Verfügung stehen, werden die Wörter mit niedriger Konfidenz in der Textstelle mit denen an der gleichen Position im Antwortgraphen verglichen. Dafür ist es notwendig, dass die Textpassage die gleiche Länge wie der Antwortgraph besitzt. Andernfalls wird ein Fehler verursacht, wenn z B. Knoten drei der Textpassage mit dem dritten Wort der Antwort verglichen werden soll, diese aber nur aus zwei Wörtern besteht. Daher kann die Nutzeraussage nur bei übereinstimmender Länge verwendet werden. Aufgrund der fehlenden Fallunterscheidungen ist dieser Sonderfall die einzige Situation, in der mehr als ein Knoten mit einer Frage verifiziert werden kann. Zum Beispiel könnten bei einer Textpassage, bestehend aus drei Knoten, die ersten beiden Wörter eine niedrige Konfidenz aufweisen. Besitzen das erste und zweite Wort der Antwort dagegen einen hohen Konfidenzwert, verifizieren bzw. ersetzen diese jeweils den entsprechenden Knoten der Textstelle im Graphen.

Die übrigen neun Fälle sind in Tabelle 6.1 dargestellt. Hierbei steht jeder Buchstabe (a, b, c, d, w, x, y, z) für ein Wort. In der Problem-Spalte steht auf der linken Seite die in der Frage enthaltene Textpassage und auf der rechten Seite die Antwort des Nutzers. Die eingeklammerten Buchstaben stellen links die Wörter dar, die eine niedrige Konfidenz aufweisen und rechts die Wörter die die Knoten auf der linken Seite verifizieren oder ersetzen sollen. Der Pfeil bedeutet in diesem Kontext „wird verifiziert durch“ bzw. „wird ersetzt durch“. Verifiziert wird ein Knoten dann, wenn der Antwortknoten das gleiche Wort enthält und somit der ursprüngliche Knoten korrekt war, also lediglich die Konfidenz zu niedrig war. Für die Verifizierung reicht ein Konfidenzwert von mindestens 0,5 im Antwortknoten aus, da der Spracherkenner zweimal das gleiche Wort vorgeschlagen hat. Falls der Antwortknoten ein anderes Wort mit einer Konfidenz von über 0,8 besitzt, wird der ursprüngliche Knoten ersetzt. Um das Einfügen von Fehlern im Graphen zu vermeiden, liegt der Schwellenwert in diesem Fall höher, da der Spracherkenner unterschiedliche Wörter vorgeschlagen hat. Zusätzlich zu den Anforderungen an den Schwellenwert müssen für die Ersetzung bzw. Verifizierung noch die Vergleiche in der Spalte Bedingungen erfüllt sein. Das Ziel dieser Vergleiche ist, sicherzustellen, dass der Nutzer auch tatsächlich die entsprechende Textpassage wiederholt hat und nicht irgendwelche Wörter mit hoher Konfidenz eingesprochen hat. In diesem Zusammenhang prüft „b == y“, ob das Wort aus Knoten b identisch mit dem Wort aus Knoten y ist. Trifft dies zu, ist diese Bedingung erfüllt. Die Fälle eins und drei entsprechen Spezialfällen von Fall zwei am Anfang bzw. Ende des Graphen. Ebenso sind die Fälle vier und sechs Spezialfälle von Fall fünf sowie sieben und neun von Fall acht. Zu beachten ist in diesem Kontext auch, dass die Textpassagen mindestens vier Wörter umfassen müssen, um bei den Vergleichen Fehler der Klasse `ArrayIndexOutOfBoundsException` zu vermeiden. Daher stellen die Probleme der Fälle in Tabelle 6.1 in der Regel nur die Ausschnitte der Textpassage dar, die für Verifizierung oder Ersetzung relevant sind.

Tabelle 6.1.: Neun der elf bearbeiteten Fälle der Fehlerklasse *Wort nicht erkannt*

Fall	**Problem**	**Bedingungen**
1	(a) → (x) b y c z	 b == y c == z
2	a x (b) → (y) c z	a == x c == z
3	a x b y (c) → (z)	a == x b == y
4	(a) → (w) ↘ (x) b y c z	 b == y c == z
5	a w (b) → (x) ↘ (y) c z	a == w c == z
6	a w b x (c) → (y) ↘ (z)	a == w b == x
7	(a) → (x) (b) ↗ c y d z	 c == y d == z
8	a x (b) → (y) (c) ↗ d z	a == x d == z
9	a x b y (c) → (z) (d) ↗	a == x b == y

Liefern diese elf Fälle keine Lösung, wird dem Nutzer die Frage erneut in leicht abgewandelter Form gestellt. Der Unterschied besteht darin, dass der Anwender explizit darauf hingewiesen wird nur seine eigenen Wörter aus der Textpassage zu wiederholen. Gelingt auch mit der Antwort auf die zweite Frage keine Beseitigung des Fehlers, wird an dieser Stelle abgebrochen, da es unwahrscheinlich ist, dass der Nutzer in einem dritten Versuch eine adäquate Antwort liefert, nachdem er bereits zweimal scheiterte. In diesem Kontext ist anzumerken, dass der Anwender in einigen Fällen das Problem gar nicht lösen kann. Besitzen zwei aufeinanderfolgende Knoten im Graphen eine niedrige Konfidenz und sind beide Wörter falsch, ist es dem Nutzer nicht mehr möglich diesen Fehler zu korrigieren, außer die beiden Knoten werden zu einem Knoten verschmolzen, vgl. Fälle 7, 8, 9 oder es handelt sich um eine kurze Textpassage. Das Problem zweier falscher aufeinanderfolgender Wörter lässt sich an folgendem Beispiel verdeutlichen: „Gib mir bitte den Regenschirm.“, war die ursprüngliche Aussage des Nutzers und „Gib mit biete den Regenschirm“ das Er-

gebnis des Spracherkenners. Die Korrektur des Wortes „mit“ zu „mir“ entspricht Fall zwei. Wiederholt der Anwender nun seine Aussage korrekt ist allerdings die Bedingung „c == z“ nicht mehr erfüllt. Das gleiche Problem stellt sich für das Wort „biete“. Somit können die Fehler nicht beseitigt werden. Hierbei stellt sich die Frage, aus welchem Grund zwei Knoten mit niedriger Konfidenz nicht gleichzeitig bearbeitet werden. Dies liegt daran, dass sich dadurch die Anzahl der Fälle vervielfachen würde. Allein das Pendant zu Fall acht würde aus mindestens zwei Fällen bestehen, wenn drei Knoten auf zwei Knoten reduziert werden können, vgl. Fall 1 und 2 in Tabelle 6.2. Zudem würde Fall 3 neu entstehen und müsste auch bearbeitet werden. Des Weiteren stellt sich dann die Frage, ob die gleichzeitige Bearbeitung von zwei aufeinander folgenden Knoten mit niedriger Konfidenz ausreichend ist. Dann könnten auch die Fälle mit drei, vier oder mehr Knoten mit niedriger Konfidenz hintereinander berücksichtigt werden. Hierbei ist jedoch zu beachten, dass die Fehleranfälligkeit mit jedem zusätzlich austauschbaren Knoten steigt, da es immer schwieriger wird sicherzustellen, dass es sich noch um die gefragte Textstelle handelt. Da allerdings ein wesentliches Ziel des Dialogagenten die Beseitigung von Problemen ist, ohne dabei neue Fehler in den Graphen zu integrieren, werden nur die beschriebenen elf Fälle für die Fehlerklasse *Wort falsch erkannt* implementiert. Diese setzen dem Anwender einen engen Rahmen für die Fehlerkorrektur und sollen somit sicherstellen, dass eine Antwort keine neuen Fehler generiert. Falls sich im praktischen Einsatz des Dialogagenten herausstellen sollte, dass dies nicht ausreicht, können in Zukunft weitere Fälle hinzugefügt werden.

Tabelle 6.2.: Beispielfälle bei zwei gleichzeitig zu bearbeitenden Wörtern

Fall	**Problem**	**Bedingungen**
1	a w (b) → (x) (c) → (y) (d) ↗ e z	a == w e == z
2	a w (b) → (x) (c) ↗ (y) (d) ↗ e z	a == w e == z
3	a x (b) → (y) (c) ↗ (d) ↑ e z	a == x e == z

Nachdem die Bearbeitung aller Knoten mit niedriger Konfidenz abgeschlossen ist, wird der Graph neu erstellt und an die Rahmenarchitektur übergeben. Anschließend können die Vorverarbeitungsstufen und Agenten von PARSE diesen korrigierten Graphen bearbeiten. Damit in `WordWrongInterpreted` aufgrund unlösbarer Probleme des Spracherkenners nicht ständig die Methode `solveDefectCategory` aufrufen wird, werden die Konfidenzwerte für alle Wörter auf 1,0 gesetzt, bevor der Graph neu gebaut wird.

6.3.2. Koreferenz falsch aufgelöst

In der Klasse `CorefWrongReferenced` wird die Fehlerklasse *Koreferenz falsch aufgelöst* bearbeitet. Die Methode `analyseGraph` sucht nach Koreferenzenkanten im Graphen. Diese Kanten existieren zwischen Entitäten, die sich möglicherweise referenzieren. Zu den Koreferenzkanten werden die Ursprungs- und Zielentitäten, vgl. Abschnitt 5.8.4, ermittelt. Zeigt

eine Ursprungsentität auf mehrere Zielentitäten, wird geprüft, ob einer der folgenden Indikatoren vorliegt. Die Ursprungsentität besitzt mehrere Zielentitäten mit einer Konfidenz von eins oder mehrere Zielentitäten mit einem Konfidenzwert im Bereich [0,8;1). Der dritte Indikator liegt vor, falls mehrere Zielentität vorhanden sind aber alle einen Konfidenzwert von kleiner 0,8 aufweisen. Während dieser Analyse werden die Ursprungsentitäten, die einen Indikator aufweisen, in eine Liste eingetragen, welche als Klassenvariable implementiert ist. Da die Ursprungsentitäten ohnehin identifiziert werden, bietet das Ablegen in der Liste den Vorteil, dass in der Methode `solveDefectCategory` nicht erneut nach ihnen gesucht werden muss, wodurch Laufzeit eingespart wird. Enthält der Graph einen Indikator wird `solveDefectCategory` aufgerufen, ansonsten wird der Graph an die nächste Fehlerklasse in der Zuständigkeitskette übergeben.
Der erste Schritt in `solveDefectCategory` ist die Bestimmung der Zielentitäten für die betrachtete Ursprungsentität. Anschließend wird die Textpassage aus dem Graphen extrahiert, welche alle Zielentitäten sowie die Ursprungsentität enthält. Zudem soll die Textstelle mit einer Verbphrase beginnen und einer Nominalphrase enden, damit der Nutzer die Entitäten einordnen kann. Aufgrund dieser speziellen Anforderungen kann in diesem Fall nicht auf Methoden der Hilfsklasse `GraphOperations` zurückgegriffen werden. Deswegen wurde für diesen Zweck die Methode `getQuestionableTextPart` erstellt, welche die Textpassage als Liste mit Knoten als Inhalt zurückgibt.
Als nächstes wird in `solveDefectCategory` geprüft, ob das Wort bzw. die Wörter der Ursprungsentität mit den Wörtern einer Zielentität übereinstimmen. Trifft dies zu, wird in der Frage die Ursprungsentität mit „letzte“ gekennzeichnet, siehe Abschnitt 5.8.4. Nach der Frageformulierung, dieser Vorgang entspricht dem Textgenerator, wird die Frage über den Sprachgenerator, vgl. Abschnitt 6.1.1, ausgegeben.
Nachdem der Anwender geantwortet hat und die Antwort mit Hilfe des Rekorders, vgl. Abschnitt 6.1.2, aufgezeichnet wurde, findet in der Methode `checkMatch` eine Prüfung statt, welche Entitäten im Antwortgraphen enthalten sind. Falls die Zielentität aus einem Wort besteht, wird lediglich der Antwortgraph durchsucht und falls das entsprechende Wort enthalten ist, wird es in die Liste der in der Antwort enthaltenen Entitäten aufgenommen. Besteht die Zielentität aus mehreren Wörtern, z. B. „die grüne Tasse“, wird zunächst versucht die komplette Phrase im Antwortgraphen zu identifizieren. Gelingt dies nicht, wird in einem zweiten Schritt nur nach dem Nomen der Zielentität gesucht. Falls dieses in der Antwort enthalten ist, wird auch diese Entität der Liste hinzugefügt. Nach Prüfung aller übergebenen Zielentitäten gibt `checkMatch` die erstellte Liste an die aufrufende Methode `solveDefectCategory` zurück.
Enthält die Liste aus checkMatch keine Entität hat der Nutzer keine valide Antwort gegeben. Sind mehrere Zielentitäten in der Liste enthalten, hat der Anwender entweder mehrere genannt oder der Spracherkenner hat fälschlich weitere Entitäten erkannt. In allen Fällen kann die Antwort nicht sinnvoll verarbeitet werden, daher wird der Nutzer erneut gefragt. Bei der zweiten Frage werden dem Anwender zusätzlich noch die enthaltenen Zielentitäten genannt, um die Antwort zu erleichtern. Für die Extraktion aller Wörter einer Zielentität wird die Methode `getTokensAsString` eingesetzt. Schlägt auch dieser Versuch fehl, wird für jede einzelne Zielentität in Form einer Ja/Nein-Frage nachgefragt, ob diese gemeint ist. Antwortet der Anwender nicht mit Ja oder Nein wird er darauf aufmerksam gemacht, dass es sich um eine Ja/Nein-Frage handelt und die Frage erneut gestellt. Sobald der Nutzer eine Zielentität bestätigt oder im Fall der ersten beiden offenen Fragen der Antwortgraph genau eine Zielentität enthält, wird deren Konfidenzwert in der dazugehörigen Koreferenzkante auf eins gesetzt und der für alle anderen relevanten Zielentitäten in den entsprechenden Koreferenzkanten auf null. Zudem werden alle veränderten Kanten als vom Dialogagent verifiziert markiert, damit die anderen Agenten die Konfidenzwerte nicht mehr überschreiben können. Das Setzen der neuen Konfidenzwerte sowie die Verifizierung der Kanten wird in der Methode `updateAnaphoraReferent` durchgeführt. Für den Fall,

dass die Ursprungsentität zu keiner Zielentität gehört und der Anwender daher in den ersten beiden Fragen keine Entität nennt und anschließend alle individuellen Nachfragen bezüglich einer Zielentität verneint, werden in `setAnaphoraReferentArcsZero` die Konfidenzwerte alle Koreferenzkanten von der Ursprungsentität zu den Zielentitäten auf null gesetzt und die Kanten als verifiziert gekennzeichnet.
Mit Hilfe der Methode `detectContextEntityTokens` können die zu einer Entität gehörenden Knoten ermittelt werden, welche die Wörter der Nutzeraussage enthalten. Diese Methode wird von `checkMatch` und `solveDefectCategory` aufgerufen. Die beiden Methoden `detectContextEntityElements` und `arcToString` sind Hilfsmethoden für den Protokollierer (engl. Logger).

6.3.3. DANN-Anweisung nicht erkannt

Für die Bearbeitung der Fehlerklasse *DANN-Anweisung nicht erkannt* wurde die Klasse `ThenInstructionNotRecognized` erstellt, vgl. Abbildung 6.1. Die Methode `analyseGraph` sucht nach WENN-Bedingungen auf die keine DANN-Anweisung folgt. Hierfür wird der Graph durchlaufen und sobald eine WENN-Bedingung beginnt, werden die betrachteten Knoten in einer Liste gespeichert. Falls einer WENN-Bedingung eine DANN-Anweisung folgt, werden alle Listenelemente entfernt und die Suche wird nach dem gleichen Schema fortgeführt. Folgt einer WENN-Bedingung eine weitere WENN-Bedingung oder das Ende des Graphen, ist dies ein Indikator für eine fehlende DANN-Anweisung. Daher wird nun die Methode `solveDefectCategory` aufgerufen. Die erstellte Liste entspricht in diesem Fall der Textpassage, welche die DANN-Anweisung enthalten sollte und wird daher bei der Fragegenerierung berücksichtigt.
Die Formulierung der kompletten Frage findet in `solveDefectCategory` statt. In einer ersten Frage wird der Anwender gebeten, die DANN-Anweisung in der ausgegebenen Textpassage zu wiederholen. Gelingt dies nicht, wird eine zweite Frage mit der Textstelle generiert, in welcher dem Nutzer erklärt wird, dass seine Aussage eine Bedingung enthält und der Anwender die Handlungsanweisung wiederholen soll, falls diese Bedingung eingetreten ist. Schlägt auch dieser Versuch fehl, wird die zweite Frage erneut leicht abgewandelt ausgegeben. Falls die entsprechende Antwort auch nicht weiterhilft, wird an dieser Stelle abgebrochen und die entsprechende Stelle im Graphen als nicht auflösbar deklariert, damit sie bei zukünftigen Aufrufen des Dialogagenten nicht erneut bearbeitet wird. Andernfalls würden dem Anwender regelmäßig die gleichen Fragen gestellt, was nicht zielführend für die Problemlösung sowie die Nutzerfreundlichkeit des Systems ist.
Um herauszufinden, ob die vom Nutzer eingesprochene Aussage und durch die Vorverarbeitungsstufen in einen Antwortgraph überführte Antwort eine DANN-Anweisung enthält, wird der Antwortgraph in der Methode `compareNodes` mit der Textstelle verglichen. Diese Methode liefert alle übereinstimmenden Knoten zurück. Anschließend wird in `checkMatch` beginnend mit dem ersten Knoten der Textpassage geprüft, ob zwei übereinstimmende Knoten hintereinander liegen. Trifft dies zu, werden diese als Beginn der neuen DANN-Anweisung angesehen und die weiteren übereinstimmenden Wörter dieser DANN-Anweisung hinzugefügt. Die Prüfung auf zwei hintereinander liegende Knoten dient dazu, die WENN-Bedingung nicht aufgrund von Fehlern des Spracherkenners zu zerstückeln. Das vermiedene Problem wird an folgendem Beispiel verdeutlicht: „Falls noch Bowle im Kühlschrank ist, hole mir ein Glas mit Bowle.“ Wiederholt der Anwender nun die DANN-Anweisung „hole mir ein Glas mit Bowle“ und versteht der Spracherkenner fälschlich „Bowle mit ein Glas mit Bowle“, führt dies zu folgender Struktur, falls die DANN-Anweisung schon mit dem ersten übereinstimmenden Knoten beginnt:

Falls noch *Bowle* im Kühlschrank ist *mir ein Glas mit Bowle.*

Die normal geschriebenen Wörter entsprechen einer WENN-Bedingung, die kursiv geschriebenen Wörter einer DANN-Anweisung. In diesem Beispielsatz ist ersichtlich, dass

„Bowle“ die erste WENN-Bedingung in zwei WENN-Bedingungen aufgespalten hat. Somit besitzt der Satz nun die folgende Struktur: 1. WENN-Bedingung „Falls noch“, 1. DANN-Anweisung „Bowle“, 2. WENN-Bedingung „im Kühlschrank ist“, 2. DANN-Anweisung „mir ein Glas mit Bowle“. Daher ist der Vergleich, ob zwei übereinstimmende Knoten hintereinander liegen, sinnvoll. Denn dadurch kann das eine falsch verstandene Wort „Bowle“ die WENN-Bedingung nicht aufspalten.
Nach der Bestimmung der neuen DANN-Anweisung wird diese in `solveDefectCategory` in den Graphen integriert und die Knoten der Textpassage im Graphen als verifiziert gezeichnet, damit die anderen Agenten die Änderungen des Dialogagenten nicht überschreiben.

7. Evaluation

In diesem Kapitel wird die Evaluation des Dialogagenten vorgestellt. Hierfür werden zunächst im Unterkapitel, Aufbau und Durchführung, die einzelnen Szenarien beschrieben, die während der Laborstudie bearbeitet wurden. Danach findet eine Einführung der Metriken statt, die für die Bewertung der gewonnenen Daten eingesetzt werden. Dieses Unterkapitel wird anhand der zu lösenden Fehlerklassen gegliedert, da in einzelnen Szenarien mehrere Fehlerklassen bearbeitet werden können. Abschließend findet eine Bewertung der in der Laborstudie gesammelten Daten statt.

7.1. Aufbau und Durchführung

Für die Evaluation des Dialogagenten wurde eine Laborstudie mit zehn Teilnehmern durchgeführt. Ein Vorteil einer Laborstudie gegenüber einer Feldstudie ist in diesem Zusammenhang, dass die Evaluation nicht durch Hintergrundgeräusche bei der Spracheingabe verfälscht wird. Daher kommunizierten die Teilnehmer einzeln in einem Raum mit geschlossenen Fenstern und Türen mit dem Dialogagenten. Zur Spracheingabe wurde stets dasselbe Mikrofon des Typs „RODE NT 1-A“ verwendet. Der Grund für den Einsatz eines speziellen Mikrofons war, die Nutzeraussagen in einer möglichst guten Qualität zu erhalten, um somit ein Problem beim Verständnis natürlicher Sprache, vgl. 5.1.1, zu beseitigen und die Aussagekraft der Evaluation zu erhöhen. Für die Sprachausgabe wurde ein MacBook Air mit dem Betriebssystem „macOS Sierra“ eingesetzt. Auf diesem Rechner lief auch die Rahmenarchitektur inklusive des Dialogagenten.
Während der Evaluation sollten die Teilnehmer drei verschiedene Szenarien, vgl. Anhang A, lösen. In den folgenden Abschnitten findet die Beschreibung der Szenarien in der gleichen Reihenfolge statt, in der sie bearbeitet wurden. Da die Teilnehmer in zwei Szenarien Vorlagen mit dem Inhalt früherer Nutzereinsprachen erhielten, war es notwendig zuerst das Szenario durchzuführen, in dem die Teilnehmer selbst eine Anleitung einsprechen müssen, damit die Teilnehmer nicht von den Vorlagen beeinflusst werden. Die beiden anderen Szenarien können zwar unabhängig voneinander durchgeführt werden, aber um einen einheitlichen Ablauf der Evaluation zu gewährleisten, wurde die Reihenfolge gemäß Anhang A eingehalten. Alle Szenarien sind in englischer Sprache verfasst, um den Teilnehmer mental auf die Evaluation vorzubereiten, die dieser auch in Englisch durchführen musste, da PARSE nur für Eingaben in englischer Sprache konzipiert ist. Aufgrund der Tatsache, dass die Nummerierung der Szenarien im Korpus von PARSE fortlaufend ist und dieser bereits sieben Szenarien enthält, bekommt das für diese Evaluation erstellte Szenario die

Bezeichnung „Szenario 8“. Die beiden anderen in der Evaluation verwendeten Audiodateien stammen aus dem Szenario 4 und dem Szenario 6 des Korpuses von PARSE.

7.1.1. Szenario 8

In diesem Szenario bekamen die Teilnehmer die Aufgabe, einem Haushaltsroboter in eigenen Worten die Einzelschritte zu erklären, die der Roboter durchführen muss, um die Wäsche aus der Waschmaschine in den Wäschetrockner zu legen, vgl. Anhang A.1. Nachdem der Teilnehmer seine Instruktionen auf einmal eingesprochen hat, soll dieser die Fragen des Dialogagenten beantworten und somit Fehler im Graphen beseitigen. Da der Teilnehmer in seiner Wortwahl frei war, konnten Probleme zu allen implementierten Fehlerklassen des Dialogagenten, vgl. Unterkapitel 6.3, auftreten. Das Primärziel war allerdings die Evaluation der Fehlerklasse *Wort falsch erkannt*. In diesem Kontext findet in Abschnitt 7.3.2 auch eine Bewertung des eingesetzten Schwellenwertes statt, mit dessen Hilfe Worte als korrekt oder fragwürdig eingestuft werden.

7.1.2. Szenario 6

Bei diesem Szenario erhielten die Teilnehmer eine bereits eingesprochene Handlungsanweisung für den Haushaltsroboter in Textform und sollten nun die Rückfragen des Dialogagenten beantworten. Da in diesem Fall die Implementierung der Fehlerklasse *Koreferenz falsch aufgelöst*, vgl. Abschnitt 6.3.2, bewertet werden soll, ist es notwendig auf eine Spracheingabe zurückzugreifen die Fehler für diese Fehlerklasse besitzt. Deswegen durften die Teilnehmer die initiale Handlungsanweisung nicht selbst einsprechen, sondern es wurde auf eine geeignete Nutzeraussage aus dem Korpus von PARSE zurückgegriffen. Ein weiterer Vorteil dieser Vorgehensweise besteht darin, dass die Ergebnisse der Studienteilnehmer vergleichbar sind, da alle die selbe Audiodatei bzw. deren Graphen bearbeitet haben. Damit die Teilnehmer der Studie die Fragen des Dialogagenten beantworten konnten, war es notwendig, ihnen die initiale Spracheingabe mitzuteilen. Dies geschah in Form eines Textes, vgl. Anhang A.2.1. Aufgrund der Tatsache, dass eines der Koreferenzprobleme in diesem Szenario durch ein falsch verstandenes Wort hervorgerufen wurde, fand darüber hinaus eine weitere Evaluation der Fehlerklasse *Wort falsch erkannt* statt, da die Nutzer zunächst Fehler des Spracherkenners korrigieren sollten.

7.1.3. Szenario 4

In diesem Szenario soll der Teilnehmer die Fehlerklasse *DANN-Anweisung nicht erkannt* lösen. Damit der Graph einen solchen Fehler auch enthält, darf der Teilnehmer die Handlungsanweisung, genauso wie in dem Szenario in Abschnitt 7.1.2, nicht selbst einsprechen. Daher wird ihm in Textform die bereits eingesprochene Aussage vorgelegt, vgl. Anhang A.2.2. Zu dieser soll der Teilnehmer anschließend die Fragen des Dialogagenten beantworten. Da das hier vorliegende Problem der Fehlerklasse *DANN-Anweisung nicht erkannt* nicht durch die Korrektur von Fehlern des Spracherkenners zu beseitigen ist, wird auf eine Bearbeitung der Fehlerklasse *Wort falsch erkannt* verzichtet. Somit muss der Nutzer in diesem Szenario keine Fehler des Spracherkenners lösen.

7.2. Metriken

Für die Evaluation der in der Laborstudie gesammelten Daten ist die Definition geeigneter Metriken notwendig. Dieses Unterkapitel führt daher für alle Fehlerklassen Kennzahlen ein, anhand derer die Daten interpretiert werden können. Zudem werden noch Metriken für die Bewertung des gewählten Schwellenwertes der Fehlerklasse *Wort falsch erkannt* vorgestellt.

7.2.1. Kennzahlen für die Evaluation des Schwellenwertes

Bevor die Metriken für die Evaluation des Schwellenwertes für die Konfidenz berechnet werden können, ist es notwendig, die Anzahl der richtig positiven (rp), falsch positiven (fp), falsch negativen (fn) und richtig negativen (rn) Werte zu bestimmen. Richtig positiv bedeutet, dass ein Objekt eine bestimmte tatsächliche Eigenschaft besitzt und diese auch erkannt wurde. Das Pendant dazu ist richtig negativ, hier besitzt das Objekt diese Eigenschaft nicht und auch dies wurde korrekt erkannt. Bei den falsch negativen Werten verfügt das Objekt tatsächlich über eine Eigenschaft, wird jedoch deklariert, als hätte es diese Eigenschaft nicht. Ein Beispiel dafür ist eine grüne Tasse die als blaue Tasse klassifiziert wurde. Die gesuchte Eigenschaft war in diesem Zusammenhang, ob eine Tasse grün ist. Im Falle einer Einordnung als falsch positiv besitzt ein Objekt die gesuchte Eigenschaft nicht, wird aber deklariert als hätte es diese. Ein Beispiel hierfür ist eine blaue Tasse, die als grüne Tasse eingeordnet wird. Auch in diesem Fall wurde nach grünen Tassen gesucht. Aus diesen Werten lassen sich nun die Metriken zur Evaluation des Schwellenwertes wie folgt berechnen:

$$Präzision = \frac{rp}{rp + fp} \tag{7.1}$$

Die Präzision (engl. precision) gibt an, wie viele der mit der gesuchten Eigenschaft klassifizierten Objekte diese auch tatsächlich besitzen.

$$Ausbeute = \frac{rp}{rp + fn} \tag{7.2}$$

Das Verhältnis der Objekte, die eine gesuchte Eigenschaft besitzen und auch so deklariert wurden, zu der Anzahl aller Objekte die diese Eigenschaft besitzen, wird durch die Metrik Ausbeute (engl. recall) ausgedrückt.

$$Genauigkeit = \frac{rp + rn}{rp + fp + fn + rn} \tag{7.3}$$

Die Genauigkeit (engl. accuracy) gibt die Anzahl der korrekt klassifizierten Objekte im Verhältnis zu allen Objekten an.

$$F1 = 2 * \frac{Präzision * Ausbeute}{Präzision + Ausbeute} \tag{7.4}$$

Da sich die beiden Kennzahlen Präzision und Ausbeute in der Regel entgegengesetzt entwickeln, bildet das F1-Maß eine Kombination aus den beiden Metriken, um die Güte einer Auswertung anhand einer einzigen Kennzahl beschreiben zu können. Entgegengesetzt entwickeln bedeutet in diesem Kontext: Steigt die Präzision, dann sinkt die Ausbeute und umgekehrt.

7.2.2. Kennzahlen für Wort falsch erkannt

Die Kennzahlen für die Bewertung der Implementierung der Fehlerklasse *Wort falsch erkannt* werden in diesem Abschnitt definiert. Der Name einer Metrik wird bei deren Einführung in Klammern notiert. Beurteilt wird bei dieser Fehlerklasse die Auflösung der in der Nutzeraussage entdeckten Indikatoren (Anzahl Indikatoren), d. h. Wörter mit einem Konfidenzwert von kleiner als 0,8. In diesem Zusammenhang ist die Anzahl der Fragen (Anzahl Fragen), die der Dialogagent für die Korrektur beziehungsweise Verifizierung eines Wortes

stellt, eine wichtige Kennzahl, denn diese gibt zum einen Aufschluss über die Effektivität der Fragestellung und zum anderen, ob die Antworten zielführend ausgewertet werden. An dieser Stelle ist anzumerken, dass der Dialogagent dem Anwender maximal zwei Fragen für die Auflösung eines fragwürdigen Wortes stellt, vgl. Abschnitt 5.8.3. Bei der Auswertung dieser Fehlerklasse wird zudem unterschieden, ob ein Wort verifiziert (Anzahl verifizierte Aussagen) oder ob ein Fehler korrigiert (Anzahl gelöste Fehler) werden soll. Im ersten Fall kann durch den Nutzer ein neuer Fehler (Anzahl neu erzeugte Fehler) in den Graphen integriert werden, falls der Spracherkenner in der Antwort mit hoher Wahrscheinlichkeit ein anderes Wort, als das zu verifizierende Wort, erkannt hat. Dieses Problem tritt im zweiten Fall nicht auf, denn aus einem falsch identifizierten Wort kann mit einer nicht korrekt verstandenen Antwort kein zusätzliches falsch erkanntes Wort entstehen. Die Summe aus den Kennzahlen (Anzahl verifizierte Aussagen), (Anzahl gelöste Fehler) und (Anzahl neu erzeugte Fehler) gibt an, wie viele Antworten der Dialogagent verstanden hat (Anzahl verstandene Antworten).
Neben diesen Metriken die Aussagen über die Häufigkeit treffen, besitzt diese Fehlerklasse noch drei weitere relative Kennzahlen.

$$Fehlerrate = \frac{Anzahl\ neu\ erzeugte\ Fehler}{Anzahl\ Indikatoren} \tag{7.5}$$

Die Fehlerrate gibt an, wie viele Fehler durch die Antworten des Nutzers im Verhältnis zur Anzahl der betrachteten Indikatoren erzeugt wurden.

$$Lösungsrate = \frac{Anzahl\ verifizierte\ Aussagen + Anzahl\ gelöste\ Fehler}{Anzahl\ Indikatoren} \tag{7.6}$$

Mit Hilfe der Lösungsrate wird das Verhältnis zwischen den Antworten die einen Fehler beseitigt oder ein Wort verifiziert haben und der Anzahl der Indikatoren beschrieben.

$$Verständnisrate = \frac{Anz.\ verif.\ Aussagen + Anz.\ gel.\ Fehler + Anz.\ neu\ erz.\ Fehler}{Anzahl\ Indikatoren} \tag{7.7}$$

Die Verständnisrate gibt an, wie viele Antworten des Anwenders im Verhältnis zur Anzahl der Indikatoren verstanden wurden. Das Verstehen einer Antwort bedeutet in diesem Kontext, dass der Dialogagent den Graphen verändert. Daher wird neben der Anzahl der verifizierten Wörter bzw. gelösten Fehler auch die Anzahl der neu generierten Fehler in dieser Metrik berücksichtigt. Die Verständnisrate kann somit auch durch die Addition der Fehlerrate mit der Lösungsrate berechnet werden.

7.2.3. Kennzahlen Koreferenz falsch aufgelöst

Für die Evaluation der Fehlerklasse *Koreferenz falsch aufgelöst* sind die folgenden Kennzahlen relevant. Da die Probleme dieser Fehlerklasse immer aufgelöst werden, vgl. Abschnitt 5.8.4, ist die Erstellung einer Metrik, welche die Erfolgsrate bei der Auflösung dieser Fehlerklasse widerspiegelt, nicht zielführend. Entscheidend ist somit, ob die einzelnen Entitäten korrekt koreferiert werden.

$$Lösungsrate = \frac{Anzahl\ der\ korrekt\ bearbeiteten\ Indikatoren}{Anzahl\ aller\ Indikatoren} \tag{7.8}$$

Daher wird die Lösungsrate eingeführt, die die Anzahl der richtig bearbeiteten Indikatoren im Verhältnis zu allen aufgetretenen Problemen dieser Fehlerklasse beschreibt.

$$\text{ø } Anzahl\ Fragen = \frac{Anzahl\ Fragen\ für\ korrekt\ bearbeitete\ Indikatoren}{Anzahl\ der\ korrekt\ bearbeiteten\ Indikatoren} \tag{7.9}$$

Die zweite Metrik im Kontext dieser Fehlerklasse ist die durchschnittliche Anzahl an Fragen, die für eine korrekte Auflösung einer Koreferenz nötig war, vgl. Formel 7.9. Diese Kennzahl ist vor allem für die Bewertung der Frageformulierung des Dialogagenten zu berücksichtigen.

7.2.4. Kennzahlen DANN-Anweisung nicht erkannt

Eine Einführung der Metriken für die Fehlerklasse *DANN-Anweisung nicht erkannt* findet in diesem Abschnitt statt. Antworten auf Fragen zu dieser Fehlerklasse können zu drei verschiedenen Resultaten führen. Ein mögliches Ergebnis ist die korrekte Identifikation einer DANN-Anweisung. Alternativ könnte dem Graphen eine falsche DANN-Anweisung hinzugefügt werden. Die dritte Möglichkeit ist, dass aufgrund einer unpräzisen Nutzerantwort überhaupt keine DANN-Anweisung identifiziert wird. Aus diesen Fällen ergeben sich für diese Fehlerklasse die folgenden Metriken.

$$Lösungsrate = \frac{Anzahl\ der\ korrekt\ hinzugefügten\ DANN\text{-}Anweisungen}{Anzahl\ der\ gesuchten\ DANN\text{-}Anweisungen} \tag{7.10}$$

Die Lösungsrate beschreibt den Anteil der korrekt hinzugefügten DANN-Anweisungen, bezogen auf alle gesuchten DANN-Anweisungen.

$$Hinzufügungsrate = \frac{Anzahl\ der\ hinzugefügten\ DANN\text{-}Anweisungen}{Anzahl\ der\ gesuchten\ DANN\text{-}Anweisungen} \tag{7.11}$$

Mit der Hinzufügungsrate wird das Verhältnis zwischen der Anzahl der hinzugefügten DANN-Anweisungen und der Anzahl aller gesuchten DANN-Anweisungen ausgedrückt. Diese Metrik beschreibt somit den Anteil der Antworten, aus denen eine Textstelle extrahiert werden konnte, die gemäß der Nutzeraussage einer DANN-Anweisung entspricht, unabhängig davon, ob dies tatsächlich zutrifft.
Für die Bewertung der Frageformulierung des Dialogagenten für diese Fehlerklasse sind die beiden folgenden Kennzahlen relevant.

$$\text{ø } Anzahl\ Fragen\ gelöst = \frac{Anzahl\ Fragen\ f.\ korrekt\ hinzugefügte\ DANN\text{-}Anw.}{Anzahl\ der\ korrekt\ hinzugefügten\ DANN\text{-}Anw.} \tag{7.12}$$

Die durchschnittliche Anzahl der Fragen für korrekt identifizierte DANN-Anweisungen setzt die Anzahl der Fragen für richtig hinzugefügte DANN-Anweisungen ins Verhältnis zur Anzahl der korrekt hinzugefügten DANN-Anweisungen.

$$\text{ø } Anzahl\ Fragen\ hinzugefügt = \frac{Anzahl\ Fragen\ für\ hinzugefügte\ DANN\text{-}Anw.}{Anzahl\ der\ hinzugefügten\ DANN\text{-}Anw.} \tag{7.13}$$

Mit der durchschnittlichen Anzahl der hinzufügten Fragen wird das Verhältnis zwischen der Anzahl der Fragen für hinzugefügte DANN-Anweisungen und der Anzahl der hinzugefügten DANN-Anweisungen ausgedrückt.

7.3. Auswertung der Laborstudie

In diesem Unterkapitel werden die in der Laborstudie gesammelten Daten ausgewertet. Hierfür wird zunächst ein Überblick über die Studienteilnehmer und deren Merkmale gegeben. Anschließend findet eine Bewertung des in der Fehlerklasse *Wort falsch erkannt* eingesetzten Schwellenwertes statt. Danach werden die weiteren Ergebnisse der Evaluation zu dieser Fehlerklasse vorgestellt. In den beiden folgenden Abschnitten werden die Resultate der Auswertungen zu den Fehlerklassen *Koreferenz falsch aufgelöst* und *DANN-Anweisung nicht erkannt* präsentiert. Abschließend wird ein Fazit über die Evaluationsergebnisse der einzelnen Fehlerklassen im Kontext des Dialogagenten gezogen.

7.3.1. Studienteilnehmer

An der Studie haben insgesamt zehn Personen teilgenommen. Alle besaßen die deutsche Staatsangehörigkeit und ihre Muttersprache war Deutsch. Eine Teilnehmerin wurde zweisprachig erzogen und hatte daher noch Vietnamesisch als Muttersprache. Die eine Hälfte der Befragten war männlich und die andere Hälfte weiblich. Vier Personen gaben an, noch nie in einem englischsprachigen Land gelebt zu haben. Ein Teilnehmer gab hier einen Zeitraum von null bis sechs Monaten an und die übrigen sechs Monate bis ein Jahr. Zwei Personen gaben in der Selbsteinschätzung ihre Englischkenntnisse als elementar, drei als fortgeschritten, eine als erfahren und vier als professionell an. Die Hälfte der Teilnehmer besaß keine Programmierkenntnisse. Je eine Person gab hier professionell und erfahren an. Die restlichen drei Personen verfügten nach ihrer eigenen Einschätzung über elementare Programmierkenntnisse. Das Alter der Teilnehmer lag zwischen 22 und 32 Jahren.

7.3.2. Bewertung des Schwellenwertes für Wort falsch erkannt

Mit Hilfe der Metriken aus Abschnitt 7.2.1 lässt sich der gewählte Schwellenwert für die Bearbeitung der Fehlerklasse *Wort falsch erkannt* bewerten. Da die Wörter, deren Konfidenzwert unter dem Schwellenwert liegt, als fragwürdig eingestuft werden und somit Rückfragen des Dialogagenten erzeugen, ist eine geeignete Wahl des Schwellenwertes wichtig, um den Nutzer nicht mit unnötigen Fragen zu beschäftigen. Der Konfidenzwert drückt in diesem Zusammenhang die Wahrscheinlichkeit aus, mit der ein Wort vom Spracherkenner als korrekt erkannt klassifiziert wurde.
Zur Einordnung des gewählten Schwellenwertes von 0,8, vgl. Abschnitt 6.3.1 wurden die zehn Nutzereingaben aus Szenario 8 annotiert und die vorgestellten Metriken für sechs verschiedene Schwellenwerte berechnet. Hierfür sind zunächst in jeder Instruktion die falschen Wörter markiert worden. Danach wurde überprüft, ob diese mit dem jeweiligen Schwellenwert als falsch eingestuft werden. Auf diese Weise konnten die Werte rp, fp, rn und fn generiert werden, aus welchen anschließend die Präzision, Ausbeute, Genauigkeit und das F1-Maß berechnet wurden. In diesem Zusammenhang ist zu berücksichtigen, dass die Konfidenzen und somit indirekt die Werte für diese vier Kennzahlen von dem, von der Rahmenarchitektur extern als Spracherkenner, eingebundenen Werkzeug Watson abhängen. Der Fokus liegt daher in diesem Abschnitt darauf, den bestmöglichen Schwellenwert, in Abhängigkeit von den von Watson gelieferten Werten, zu identifizieren.

Die Ergebnisse der Auswertung der initialen Spracheingaben der zehn Teilnehmer sind in Tabelle 7.1 dargestellt. Insgesamt haben die Anwender 393 Wörter eingesprochen, von denen 84 tatsächlich falsch waren. Der im Kontext des Dialogagenten aussagekräftigste Wert ist die Präzision. Denn diese repräsentiert den Anteil der Fragen an den Nutzer, welche sich tatsächlich nach einem falsch erkannten Wort erkundigen. Die anderen Fragen verifizieren lediglich ein Wort mit einem niedrigen Konfidenzwert. Eine hohe Präzision vermeidet somit unnötige Fragen und verringert dadurch das Risiko, dass der Anwender

Tabelle 7.1.: Bewertung des Schwellenwertes für die Konfidenzwerte des Spracherkenners

Schwellenwert	**< 0,7**	**< 0,75**	**< 0,8**	**< 0,85**	**< 0,9**	**< 0,95**
Präzision	0,7143	**0,7302**	0,7222	0,6552	0,6238	0,5656
Ausbeute	0,4762	0,5476	0,6190	0,6786	0,7500	**0,8214**
F1-Maß	0,5714	0,6259	0,6667	0,6667	**0,6811**	0,6699
Genauigkeit	0,8473	0,8601	**0,8677**	0,8550	0,8499	0,8270

bei Verifizierungsfragen neue Fehler in den Graphen integriert. Der Schwellenwert für die höchste Präzision wäre bei den zehn Nutzereingaben 0,75 gewesen. Der im Dialogagenten eingesetzte Wert von 0,8 entspricht dem zweitbesten Wert für diese Kennzahl. Die Ausbeute beschreibt den Anteil der gefundenen falschen Wörter im Verhältnis zu allen falschen Wörter. Dies erklärt auch den kontinuierlichen Anstieg dieser Kennzahl bei steigenden Konfidenzwerten, da hierdurch immer mehr tatsächliche Fehler berücksichtigt werden. Bemerkenswert ist in diesem Zusammenhang, dass selbst bei einem Schwellenwert von 0,95 lediglich 82,14 Prozent aller Fehler als solcher markiert werden. Das bedeutet, dass die Konfidenz als einziger Indikator nicht ausreicht, um den Nutzer nach allen Fehlern des Spracherkenners zu Fragen. Daher müssen in Zukunft weitere semantische Indikatoren gefunden werden, die darauf hinweisen, dass ein Wort vom Spracherkenner falsch erkannt wurde. Von diesem Fakt mal abgesehen besitzt die Ausbeute keine hohe Aussagekraft für diesen Anwendungsfall, da es entscheidend ist, mit möglichst wenig Fragen viele Fehler zu beseitigen und die Ausbeute versucht mit vielen Fragen viele Fehler zu lösen. Das F1-Maß fasst die Präzision und die Ausbeute gemäß Formel 7.4 zu einer Kennzahl zusammen. Dadurch wird das F1-Maß von den Werten der Ausbeute beeinflusst und ist daher aus den gleichen Gründen wie diese nicht sehr ausdrucksstark. Die Genauigkeit repräsentiert den Anteil der korrekt klassifizierten Wörter im Verhältnis zu allen Wörtern. Da viele richtig eingeordnete Wörter ein hohes Verständnis der Rahmenarchitektur nach sich ziehen, ist dies eine wichtige Kennzahl für den Dialogagenten. Das beste Resultat für die Genauigkeit wurde bei 0,8, dem im Dialogagenten eingesetzten Wert, erzielt.
Zusammenfassend lässt sich sagen, dass der gewählte Schwellenwert von 0,8 bei den zwei wichtigsten Kennzahlen, der Präzision und der Genauigkeit, einmal den besten und einmal den zweitbesten Wert erreicht hat und dadurch eine gute Wahl war. Ein Schwellenwert von 0,75 liefert ein ähnlich gutes Ergebnis und wäre somit eine mögliche Alternative. Trotz allem ist aus wissenschaftlicher Sicht die Stichprobe für eine quantifizierbare Aussage zu klein. Um belastbare Zahlen zu erhalten, müsste die Stichprobengröße um ein Vielfaches erweitert werden.

Tabelle 7.2.: Bewertung des Schwellenwertes für die Konfidenzwerte des Spracherkenners ohne den schlechtesten Teilnehmer

Schwellenwert	**< 0,7**	**< 0,75**	**< 0,8**	**< 0,85**	**< 0,9**	**< 0,95**
Präzision	0,6000	**0,6222**	0,6200	0,5424	0,5143	0,4607
Ausbeute	0,4706	0,5490	0,6078	0,6275	0,7059	**0,8039**
F1-Maß	0,5275	0,5833	**0,6139**	0,5818	0,5950	0,5857
Genauigkeit	0,8693	0,8784	**0,8815**	0,8602	0,8511	0,8237

Neben den Ergebnissen für alle Nutzereingaben wurden noch zwei weitere Auswertungen vorgenommen. Da ein Teilnehmer der Studie sehr große Probleme mit der englischen Aussprache hatte und dadurch in keinem Szenario auch nur ein Problem lösen konnte und darüber hinaus in dessen Graphen bestehend aus 63 Wörtern 33 Fehler auftraten, wurde

eine Auswertung für den Schwellenwert ohne diesen Teilnehmer vorgenommen. Diese alternative Evaluation bestätigt die Aussage, dass die Stichprobe für belastbare Zahlen zu klein ist, da die neuen Werte deutlich vom Gesamtergebnis abweichen, vgl. Tabelle 7.2. Besonders die Werte für die Präzision liegen deutlich niedriger. Dennoch ist die Tendenz bei allen Kennzahlen außer dem F1-Maß die Gleiche. Somit behalten die Aussagen zum Gesamtergebnis ihre Gültigkeit. Die Abweichung beim F1-Maß lässt sich mit den deutlich geringeren absoluten Werten der Präzision erklären, da diese in das F1-Maß einfließt.

Tabelle 7.3.: Bewertung des Schwellenwertes für die Konfidenzwerte des Spracherkenners für Teilnehmer die im englischsprachigen Ausland gelebt haben

Schwellenwert	**< 0,7**	**< 0,75**	**< 0,8**	**< 0,85**	**< 0,9**	**< 0,95**
Präzision	0,5625	0,6000	**0,6250**	0,5333	0,5135	0,4490
Ausbeute	0,3103	0,4138	0,5172	0,5517	0,6552	**0,7586**
F1-Maß	0,4000	0,4898	0,5660	0,5424	**0,5758**	0,5641
Genauigkeit	0,8816	0,8904	**0,8991**	0,8816	0,8772	0,8509

Die zweite zusätzliche Auswertung umfasst nur die Teilnehmer die bereits im englischsprachigen Ausland gelebt haben. Das Ziel hierbei ist herauszufinden, ob bessere Englischkenntnisse einen Einfluss auf die Wahl des Schwellenwertes haben. In diesem Kontext wurde nicht die eigene Einschätzung der Teilnehmer über ihre Englischfähigkeiten herangezogen, da diese subjektiv ist, sondern auf das objektive Kriterium, eine Weile im englischsprachigen Ausland gelebt zu haben, zurückgegriffen. Tabelle 7.3 zeigt, dass in diesem Fall der beste Schwellenwert, dem im Dialogagenten gewählten Schwellenwert, von 0,8, entspricht. Sowohl die Präzision als auch die Genauigkeit erreichen hierbei ihren besten Wert. Dies lässt darauf schließen, dass der verwendete Schwellenwert für Anwender mit guten Englischkenntnissen die beste Wahl ist.

7.3.3. Bewertung Wort falsch erkannt

Die Auswertung der gesammelten Daten für Szenario 8 hat für die Fehlerklasse *Wort falsch erkannt* folgende Resultate geliefert. Von den 393 eingesprochenen Wörtern waren 84 tatsächlich falsch. Von diesen wurden 52 mit einem Schwellenwert von 0,8 als falsch identifiziert, vgl. 7.3.2. Daneben lagen noch 20 vom Spracherkenner korrekt erkannte Wörter unter diesem Schwellenwert und mussten daher auch in Rückfragen verifiziert werden. Somit hat der Indikator, Konfidenzwert kleiner als 0,8, 72 Fragen aufgeworfen. Allerdings wurde die Anzahl der zu bearbeitenden Indikatoren pro Teilnehmer auf fünf begrenzt, um den Anwender nicht zu demotivieren. Aufgrund der Tatsache, dass der Dialogagent für jedes Wort bis zu zwei Fragen stellen konnte, bedeutete dies maximal zehn Fragen für einen Teilnehmer, bezüglich dieser Fehlerklasse. Bei einigen Spracheingaben war die Erkennungsrate der Wörter jedoch so gut, dass die Zahl der Indikatoren unter fünf lag. Da zudem manche Teilnehmer einen Teil der Rückfragen im ersten Versuch beantworten konnten, lag die Anzahl der gestellten Fragen in Summe bei 80, vgl. Tabelle 7.4. Diese 80 Fragen bezogen sich auf 44 Indikatoren. Bei den Fragen zu diesen Indikatoren wurden sechs Wörter korrekt identifiziert, vier Fehler gelöst und ein neuer Fehler erzeugt. Anzumerken ist an dieser Stelle, dass eine korrekte Lösung aufgrund eines Programmierfehlers nicht in den Graphen integriert wurde. Diese Lösung wurde selbstverständlich nicht als erfolgreich eingestuft. Nachdem alle zehn Teilnehmer den Dialogagenten evaluiert hatten, wurde der Fehler im Quellcode beseitigt. Somit wurden insgesamt elf Antworten verstanden und es ergibt sich eine Verständnisrate von 0,25. Die Fehlerrate von 0,0227 resultiert aus einer falsch Verstanden Antwort, die ein zu verifizierendes Wort verändert hat. Die Lösungsrate

liegt bei 0,2273 d. h. 22,73 Prozent aller Indikatoren wurden durch Dialogagenten korrekt aufgelöst. Dieser niedrige Wert für Lösungsrate besitzt verschiedene Ursachen. Aufgrund der Implementierung der Fehlerklasse *Wort falsch erkannt*, vgl. Abschnitt 6.3.1, konnten nur Wörter korrigiert werden, die nicht gleichzeitig von einem weiteren falschen Wort umgeben waren. Ein anderes Problem war die unpräzise Aussprache einiger Teilnehmer, die großen Einfluss auf das Verständnis ihrer Aussagen hatte, vgl. Tabelle 7.5. Zudem haben einige Teilnehmer die Fragen falsch interpretiert und die vorgesagten Textstellen in anderen Worten wiedergegeben, wodurch aufgrund der Implementierung keine Fehlerbeseitigung mehr möglich war. Denn der Ansatz des Dialogagenten ist die Fehler des Spracherkenners durch die Wiederholung fragwürdiger Wörter zu beseitigen. Dies kann allerdings bei einer Neuformulierung der Aussage nicht gelingen. Eine wesentliche Erkenntnis der Evaluation ist somit, dass die Fragen eindeutiger formuliert werden sollten, um den Nutzer zukünftig davon abzubringen Textstellen in anderen Worten wiederzugeben.

Tabelle 7.4.: Kennzahlen für Wort falsch erkannt

	Szenario 8	**Szenario 6**
Anzahl Indikatoren	44	33
Anzahl Fragen	80	58
Anzahl verifizierte Aussagen	6	11
Anzahl gelöste Fehler	4	1
Anzahl neu erzeugte Fehler	1	0
Anzahl verstandene Antworten	11	12
Fehlerrate	0,0227	0,0000
Lösungsrate	0,2273	0,3636
Verständnisrate	0,2500	0,3636

Bei Szenario 6 haben die Teilnehmer die Handlungsanweisung nicht selbst eingesprochen, sondern es wurde eine Audiodatei aus dem Korpus von PARSE verwendet. Aufgrund der Tatsache, dass das für die Spracherkennung eingesetzte Werkzeug Watson nicht deterministisch ist, vgl. Abschnitt 6.3.1, traten in derselben Audiodatei unterschiedlich viele Indikatoren auf. Bei sieben Teilnehmern traten drei Indikatoren auf und bei den restlichen drei Teilnehmer traten vier Indikatoren auf. Somit waren für Szenario 6 in Summe 33 Indikatoren zu bearbeiten, vgl. Tabelle 7.4. Hieraus resultierten 58 Fragen, acht weniger als maximal möglich. Denn für jeden Indikator werden bis zu zwei Fragen gestellt. Daraus lässt sich schließen, dass acht Indikatoren mit der ersten Antwort aufgelöst werden konnten. Insgesamt verifizierten die Teilnehmer elf Aussagen, haben einen Fehler gelöst und keine neuen Fehler generiert. Von den zehn Teilnehmern gelang somit nur einer Person die Beseitigung des, in der Aussage enthaltenen, Fehlers. Insgesamt wurden somit zwölf Antworten verstanden was zu einer Verständnisrate von 0,3636 führt. Da die Teilnehmer keine neuen Fehler generiert haben, entspricht die Verständnisrate in diesem Fall der Lösungsrate. Beide liegen deutlich höher als in Szenario 8, was sich darauf zurückführen lässt, dass alle Indikatoren mit Hilfe der Implementierung des Dialogagenten auflösbar waren, da kein fragwürdiges Wort von einem anderen kritischen Wort umgeben war. Ein weiterer möglicher Umstand für den Anstieg der Raten könnte sein, dass Szenario 6 nach Szenario 8 durchgeführt wurde und die Teilnehmer daher bereits mit der Fragestellung vertraut waren. Die Fehlerrate von null in diesem Szenario und 0,0227 in Szenario 8 zeigt, dass die Implementierung des Dialogagenten fast keine neuen Fehler durch den Nutzer zulässt. Somit wurde ein wesentliches Ziel des Dialogagenten erreicht. Allerdings kam es auch in Szenario 6 vor, dass Teilnehmer die zu wiederholende Textstelle umformuliert haben. Dies bestätigt die Erkenntnis aus der Evaluation von Szenario 8, dass eine eindeutigere Frage-

formulierung notwendig ist.

Tabelle 7.5.: Kennzahlen für Wort falsch erkannt unter Berücksichtigung der Spracherfahrung der Teilnehmer für die englische Sprache

	Szenario 8		**Szenario 6**	
	kein Ausl.	Ausland	kein Ausl.	Ausland
Anzahl Indikatoren	20	24	13	20
Anzahl Fragen	38	42	26	32
Anz. verifizierte Aussagen	2	4	1	10
Anzahl gelöste Fehler	1	3	0	1
Anz. neu erzeugte Fehler	0	1	0	0
Anz. verst. Antworten	3	8	1	11
Fehlerrate	0,0000	0,0417	0,0000	0,0000
Lösungsrate	0,1500	0,2917	0,0769	0,5500
Verständnisrate	0,1500	0,3333	0,0769	0,5500

Tabelle 7.5 enthält die Resultate der Laborstudie für die Fehlerklasse *Wort falsch erkannt*, aufgeteilt nach Teilnehmern, die im englischsprachigen Ausland gelebt haben und solchen, die dies nicht getan haben. Sechs der zehn Teilnehmer besaßen Spracherfahrung im Englischen durch längerfristige Auslandsaufenthalte, vgl Abschnitt 7.3.1. Die Ergebnisse zeigen, dass die Lösungsrate für Teilnehmer mit derartiger Spracherfahrung in Szenario 8 nahezu doppelt so hoch liegt, wie für Teilnehmer ohne Auslandserfahrung in englischsprachigen Ländern. In Szenario 6 hat sich die Lösungsrate sogar mehr als versiebenfacht, d. h. hier konnte mehr als jeder zweite Indikator von den Teilnehmern mit Spracherfahrung aufgelöst werden. Diese Resultate zeigen eindeutig, dass eine präzisere Aussprache und besseres Sprachverständnis enorm dazu beitragen die Fragen des Dialogagenten effektiv zu beantworten. Ein weiteres Indiz hierfür ist, dass keiner der Teilnehmer mit Spracherfahrung mehr als fünf Indikatoren durch die initiale Handlungsanweisung generiert hat. Bei den anderen Teilnehmern lag dieser Wert stets über fünf. Somit hat sich die Begrenzung der Anzahl der Fragen des Dialogagenten für die Evaluation nur bei den Teilnehmern ausgewirkt, die nicht im englischsprachigen Ausland gelebt haben. Aufgrund des guten Abschneidens der Teilnehmer mit Spracherfahrung im Englischen, sollte in Zukunft eine weitere Laborstudie für den Dialogagenten und somit für PARSE durchgeführt werden, die ausschließlich Muttersprachler der englischen Sprache als Teilnehmer zulässt. Erst durch eine derartige Studie kann die tatsächliche Güte des Systems und insbesondere des Dialogagenten aussagekräftig bewertet werden. Vor diesem Hintergrund kann erst nach dieser zusätzlichen Evaluation abschließend bewertet werden, ob die Art der Frageformulierung für englische Muttersprachler einer Überarbeitung bedarf oder bereits in der aktuellen Variante für eine effektive Fehlerbeseitigung ausreicht. Falls PARSE für nicht englische Muttersprachler eingesetzt werden soll, ist eine eindeutigere Frageformulierung anzuraten.

7.3.4. Bewertung Koreferenz falsch aufgelöst

Dieser Abschnitt beschreibt die Ergebnisse der Auswertung zu Szenario 6 im Kontext der Fehlerklasse *Koreferenz falsch aufgelöst*. Die korrekte Transkription der Audiodatei für dieses Szenario ist in Anhang A.2.1 dargestellt. Ein Koreferenzfehler resultiert hierbei aus einer fehlerhaften Interpretation des Spracherkenners. Dieser hat die Wörter „fill it“ als „Philip“ erkannt. Daher hat der Koreferenzauflöser für das Wort „you“ in der dritten Zeile zwei mögliche Koreferenzen mit niedriger Konfidenz, wodurch dieses Wort zu einem Indikator für diese Fehlerklasse wurde, vgl. Abschnitt 5.8.4. Diese Koreferenzen sind „Philip“

und „you“ in Zeile eins. Die korrekte Koreferenz ist „you“ in der ersten Zeile. Diese Auflösung kann der Anwender auf zwei Arten erreichen. Zum einen indem er „Philip“ bei den Fragen zur Fehlerklasse *Wort falsch erkannt* zu „fill it“ korrigiert. Zum anderen mit Hilfe einer korrekten Antwort auf die Frage des Dialogagenten zu dieser Fehlerklasse. In beiden Fällen wird das Problem beseitigt, weshalb auch beide Lösungen im Kontext der Evaluation dieser Fehlerklasse als Erfolg gewertet werden. Falls die Koreferenz bereits mit Hilfe der Antwort auf die Frage der Fehlerklasse *Wort falsch erkannt* aufgelöst ist, wird dies so gewertet, als wäre das Problem durch eine Frage zu dieser Fehlerklasse gelöst worden. Dies ist für eine plausible Berechnung der Kennzahl Ø Anzahl Fragen notwendig, da eine Problemlösung ohne Frage diese Metrik verzerren würde. Die Koreferenz für „you“ wurde von einem Teilnehmer mit Hilfe der Fehlerklasse *Wort falsch erkannt* aufgelöst. Alle anderen Korrekturen dieses Fehlers wurden durch diese Fehlerklasse, *Koreferenz falsch aufgelöst*, vorgenommen. Die zweite aufzulösende Koreferenz war das „it“ in der ersten Zeile, welches sich auf „the green cup“ bezieht. Als Alternative Koreferenzen bot der Koreferenzauflöser den Teilnehmern „the table“ und „hey Armar“ an.

Tabelle 7.6.: Kennzahlen für Koreferenz falsch aufgelöst

	you → you	**it → the green cup**	**gesamt**
Lösungsrate (insgesamt)	0,40	0,60	0,50
Ø Anzahl Fragen (insgesamt)	2,00	1,50	1,75
Lösungsrate (im Ausland)	0,50	0,83	0,67
Ø Anzahl Fragen (im Ausland)	2,33	1,40	1,75
Lösungsrate (k. Ausland)	0,25	0,25	0,25
Ø Anzahl Fragen (k. Ausland)	1,00	2,00	1,50

Die Auswertung zu dieser Fehlerklasse ist in Tabelle 7.6 abgebildet. Die Spalte „gesamt“ fasst die Ergebnisse für beide Koreferenzprobleme in der jeweiligen Zeile zusammen. In den ersten beiden Zeilen sind die Ergebnisse aller zehn Teilnehmer dargestellt. Anschließend findet eine Beschreibung der Resultate für die sechs Teilnehmer statt, welche bereits im englischsprachigen Ausland (im Ausland) gelebt haben. Die letzten beiden Zeilen zeigen die Ergebnisse der Teilnehmer die noch nicht in einem englischsprachigen Land (k. Ausland) gelebt haben.
Die Lösungsrate (insgesamt) von 0,40 für die Koreferenz von „you → you“ bedeutet, dass vier der zehn Teilnehmer diese Koreferenz richtig aufgelöst haben. Bei der Koreferenzauflösung „it → the green cup“ lagen 60 Prozent der Teilnehmer korrekt. Das heißt, dass insgesamt die Hälfte aller Koreferenzen mit Hilfe des Dialogagenten richtig aufgelöst wurden. Dieses Ergebnis lässt sich allerdings so interpretieren, dass in den anderen Fällen die Anwender die Fragen nicht verstanden haben, da diese Fehlerklasse immer aufgelöst wird, vgl. Abschnitt 5.8.4. Studienteilnehmer, die bereits im englischsprachigen Ausland gelebt haben, weisen mit 0,67 eine deutlich höhere Lösungsrate auf als Teilnehmer die weniger Spracherfahrung im Englischen besitzen, welche nur auf eine Lösungsrate von 0,25 kommen. Dies untermauert die eben aufgestellte These, dass Teilnehmer mit Sprachdefiziten im Englischen die Fragen nicht verstanden haben. Auch eine Betrachtung der Sprachaufnahmen zeigt, dass einige Anwender einfach die Textstellen aus den Fragen wiederholt haben, anstatt eine konkrete Antwort zu geben. Eine weitere Ursache hierfür könnte sein, dass die Nutzer immer noch davon ausgingen, Probleme der Fehlerklasse *Wort falsch erkannt* zu bearbeiten und somit gar nicht mehr auf die tatsächliche Fragestellung geachtet haben. Vor diesem Hintergrund ist zu erwähnen, dass die Implementierung während der Evaluation eine Antwort mit mehreren Entitäten nicht abgelehnt hat und somit die Nutzer

nicht auf die falsch verstandene Frage hingewiesen wurden. Im Zuge dessen kann es auch zu Fehlinterpretationen der Antworten durch den Dialogagenten gekommen sein. Da sich diese allerdings sowohl positiv als auch negativ auf die Ergebnisse zu dieser Fehlerklasse ausgewirkt haben könnten, wird an dieser Stelle davon ausgegangen, dass die tatsächlichen Resultate im Wesentlichen mit denen aus Tabelle 7.6 übereinstimmen und diese daher für die Evaluation dieser Fehlerklasse als valide anzusehen sind. Die Implementierung wurde zudem inzwischen angepasst und erlaubt nur noch Antworten, die maximal eine der gesuchten Entitäten enthalten.
Die Kennzahl Ø Anzahl Fragen (insgesamt) zeigt, dass die Mehrheit der Teilnehmer, welche die Koreferenzen aufgelöst haben, sich mit der Frage nach dem „you“ schwerer getan hat als mit der Auflösung des „it“. Das Ergebnis für Ø Anzahl Fragen (k. Ausland) kann an dieser Stelle ignoriert werden, da diese Werte jeweils nur einem Teilnehmer entsprechen und somit eher zufällig sind als wissenschaftlich belastbar. Eine Erklärung für die höheren Werte bei „you“ könnte sein, dass die Menschen das „you“ eher dem Haushaltsroboter zuordnen würden als einem anderen „you“ und somit die Frage zunächst verwirrend finden. Dies deckt sich auch mit den Reaktionen der Teilnehmer auf diese Frage während der Evaluation. Daher wäre es in solchen Fällen sinnvoll, den Anwendern in der Frageformulierung noch weitere Hilfestellungen zu geben, um Missverständnissen vorzubeugen.

7.3.5. Bewertung DANN-Anweisung nicht erkannt

Die Ergebnisse der Auswertung der Daten zu Szenario 4 und somit für die Fehlerklasse *DANN-Anweisung nicht erkannt* werden hier vorgestellt. Die korrekte Lösung dieser Fehlerklasse bezüglich des Textes in A.2.2 ist „put it in the cupboard“. Diese Aussage entspricht der DANN-Anweisung für die zuvor genannte WENN-Bedingung „if there are clean dishes“.

Tabelle 7.7.: Kennzahlen für DANN-Anweisung nicht erkannt

	DANN-Anweisung nicht erkannt
Lösungsrate (insgesamt)	0,30
Hinzufügungsrate (insgesamt)	1,00
Ø Anzahl Fragen gelöst (insgesamt)	1,33
Ø Anzahl Fragen hinzugefügt (insgesamt)	1,10
Lösungsrate (im Ausland)	0,33
Hinzufügungsrate (im Ausland)	1,00
Ø Anzahl Fragen gelöst (im Ausland)	1,00
Ø Anzahl Fragen hinzugefügt (im Ausland)	1,00
Lösungsrate (kein Ausland)	0,25
Hinzufügungsrate (kein Ausland)	1,00
Ø Anzahl Fragen gelöst (kein Ausland)	2,00
Ø Anzahl Fragen hinzugefügt (kein Ausland)	1,25

Tabelle 7.7 zeigt die Resultate der Auswertung für die Fehlerklasse *DANN-Anweisung nicht erkannt*. Die Hinzufügungsrate (insgesamt) mit einem Wert von 1,00 besagt, dass alle Teilnehmer eine DANN-Anweisung in den Graphen integrieren konnten. Jedoch zeigt die Lösungsrate (insgesamt) von 0,30, dass nur drei der zehn Teilnehmer die DANN-Anweisung auch korrekt hinzugefügt haben. Das bedeutet, dass entweder die Implementierung, vgl. Abschnitt 6.3.3, den Antwortgraphen falsch interpretiert oder die Nutzer dem System eine

falsche Antwort liefern. Eine Analyse der Implementierung hat gezeigt, dass diese korrekt arbeitet. Somit kann das Problem nur noch auf der Seite des Anwenders liegen. Eine Auswertung der Audiodateien mit den Nutzerantworten hat dies bestätigt. Das Problem ist, dass die Teilnehmer der Studie die Frage in 60 Prozent falsch verstanden haben und einfach nur die Textstelle wiederholt haben. Dadurch wurde die komplette Textstelle als DANN-Anweisung deklariert. Dies hat anschließend zu einem Fehler im Bedingungsanalysierer von PARSE geführt, da dessen Heuristiken keine DANN-Anweisungen ohne eine WENN-Bedingung erlauben. Aufgrund dieses Fehlers ist das komplette System abgestürzt. Ein weiterer Teilnehmer hat eine falsche DANN-Anweisung erzeugt, ohne das System zum Absturz zu bringen. Aus diesen Ergebnissen lässt sich schließen, dass die Fragen des Dialogagenten für diese Fehlerklasse neu formuliert werden müssen. Zudem sollte bereits im Dialogagenten verhindert werden, dass der Nutzer die komplette Textstelle in eine DANN-Anweisung transformieren kann. Diese Funktionalität ist mittlerweile in den Quellcode eingefügt worden. Daneben sollte auch der Bedingungsanalysierer dahingehend verändert werden, dass er in solchen Fällen eine Warnung ausgibt, aber das System nicht zum Absturz bringt.
Eine Betrachtung der Lösungsrate (im Ausland), welche nur Teilnehmer umfasst, die bereits im englischsprachigen Ausland gelebt haben, von 0,33 zeigt, dass die Ursache für die niedrige Lösungsrate sehr wahrscheinlich nicht vom Sprachverständnis der Teilnehmer abhängt, sondern die Frageformulierung des Dialogagenten das Problem darstellt.
Für die folgende Schlussfolgerung wird die Kennzahl durchschnittliche Anzahl Fragen gelöst (kein Ausland) ignoriert, da diese lediglich für einen Teilnehmer steht und somit nicht aussagekräftig ist. Die niedrigen Werte von durchschnittliche Anzahl Fragen gelöst und durchschnittliche Anzahl Fragen hinzugefügt in allen anderen Kategorien mit 1,00 bis 1,33 zeigen, dass die Antwort in der Regel ohne Rückfragen als DANN-Anweisung in den Graphen übertragen werden konnten. Dies belegt, dass die Extraktion von DANN-Anweisungen aus dem Antwortgraphen in der Implementierung effektiv umgesetzt wurde.

7.3.6. Fazit der Evaluation des Dialogagenten

In den beiden Fehlerklassen *Wort falsch erkannt* und *Koreferenz falsch aufgelöst* waren die Ergebnisse der Teilnehmer mit Spracherfahrung im Englischen deutlich besser als bei Personen mit geringeren Englischkenntnissen. Für die dritte Fehlerklasse *DANN-Anweisung nicht erkannt* kann diese Schlussfolgerung nicht gezogen werden. Dies liegt allerdings an der schlecht gewählten Frageformulierung für diese Fehlerklasse und somit nicht zwingend daran, dass kein Zusammenhang der Resultate mit den Englischkenntnissen besteht. Aus diesen Gründen sollte für die Bewertung des Dialogagenten eine zweite Laborstudie durchgeführt werden, an der nur englische Muttersprachler teilnehmen dürfen. Dies würde belastbarere Aussagen über die Fähigkeiten des Dialogagenten liefern als diese Evaluation.
Die Implementierung des Dialogagenten hat sich während der Evaluation als kaum fehleranfällig gezeigt. Dennoch wurden aufgrund unerwarteter Nutzereingaben drei logische Fehler in der Implementierung bzw. beim Konzept entdeckt. Dadurch kann die Evaluation diesbezüglich als Erfolg betrachtet werden, da sie zur Verbesserung des Dialogagenten beigetragen hat. Denn alle entdeckten logischen Fehler wurden bereits korrigiert.
Bei der Bewertung des Schwellenwertes für die Fehlerklasse *Wort falsch erkannt* hat sich gezeigt, dass der gewählte Wert von 0,8 besonders bei Nutzern mit guten Englischkenntnissen geeignet ist. Bei der Berücksichtigung aller Studienteilnehmer wäre ein Schwellenwert von 0,75 eine bessere Alternative gewesen.

8. Zusammenfassung und Ausblick

Abschließend wird in diesem Kapitel ein Fazit über den in dieser Abschlussarbeit entwickelten Dialogagenten gegeben. Hierbei wird zunächst beschrieben, ob und inwieweit die gesteckten Ziele erreicht wurden. Danach wird auf die zukünftigen Herausforderungen bei der Weiterentwicklung des Dialogagenten eingegangen.
Das grundlegende Ziel, einen Dialogagenten für PARSE zu erstellen, der einzelne bisher nicht auflösbare Probleme der Rahmenarchitektur bearbeiten kann, wurde erreicht. Diese Anforderung ist für drei verschiedene Fehlerklassen realisiert worden. Dadurch werden Probleme des Spracherkenners, des Koreferenzauflösers und des Bedingungsanalysierers behandelt. Für den Spracherkenner wurde die Fehlerklasse *Wort falsch erkannt* bearbeitet. Die Ergebnisse zeigen, dass der Dialogagent hierbei eine Lösungsrate zwischen 22 und 55 Prozent besitzt. Diese Rate ist stark von den Englischkenntnissen der einsprechenden Personen sowie von den vorhandenen Fehlern abhängig. Ein Grund hierfür ist, dass die Implementierung des Dialogagenten für diese Fehlerklasse nicht alle auftretenden Probleme beseitigen kann. Dies ist allerdings beabsichtigt, um das Ziel des Dialogagenten zu erreichen, möglichst keine neuen Fehler zu generieren. In einem Szenario lag die Fehlerrate bei null Prozent und im anderen Szenario bei 2,27 Prozent. Aufgrund dieser niedrigen Werte kann davon gesprochen werden, dass dieses Ziel erreicht wurde. Die hierfür notwendige restriktive Implementierung hatte sicherlich Auswirkungen auf die Lösungsrate und ist somit ein Grund für die geringen Werte dieser. Die zweite Ursache sind die mangelnden Englischkenntnisse einiger Studienteilnehmer. Diese haben sich sowohl bei der Aussprache als auch beim Verständnis der Fragen bemerkbar gemacht und somit die Lösungsrate negativ beeinflusst.
Die Bewertung des Schwellenwertes für die Konfidenzwerte der Fehlerklasse *Wort falsch erkannt* hat gezeigt, dass dieser mit 0,8 gut gewählt war. Für Teilnehmer mit guten Englischkenntnissen, war dies der optimale Schwellenwert. Bei der Betrachtung aller Teilnehmer hat ein Schwellenwert von 0,75 bessere Resultate erzielt.
Beim Koreferenzauflöser bearbeitet der Dialogagent die Fehlerklasse *Koreferenz falsch aufgelöst*. Die Lösungsrate liegt für diese Fehlerklasse bei insgesamt 50 Prozent. Auch in diesem Fall zeigt sich, dass diese Rate deutlich von den Englischkenntnissen der Nutzer abhängt. Denn Studienteilnehmer, die bereits eine Weile im englischsprachigen Ausland gelebt haben, wiesen eine Lösungsrate zwischen 50 und 83 Prozent auf.
Die Fehlerklasse *DANN-Anweisung nicht erkannt* wurde beim Bedingungsanalysierer bearbeitet. Hier hat die Evaluation gezeigt, dass die gewählte Frageformulierung für diese Fehlerart ungeeignet war, daher sind die Ergebnisse nicht wirklich repräsentativ, werden aber der Vollständigkeit halber an dieser Stelle dennoch erwähnt. Alle Nutzer waren in der

Lage, dem Dialogagenten eine DANN-Anweisung zu nennen, allerdings taten dies aufgrund der oft falsch verstandenen Frage nur 30 Prozent korrekt.
Eine erweiterbare Architektur war eine zusätzliche Anforderung an den Dialogagenten. Dieses Ziel wurde durch den Einsatz des Entwurfsmusters Zuständigkeitskette im Dialogmanager erfolgreich umgesetzt.
Daneben sollte der Dialogagent domänenunabhängig funktionieren. Auch dieses Ziel ist realisiert worden, indem bei der Implementierung komplett auf domänenspezifische Elemente verzichtet wurde.

Eine Herausforderung bei der Weiterentwicklung des Dialogagenten ist das Finden einer besseren Balance zwischen einer niedrigen Fehlerrate und einer hohen Lösungsrate bei der Fehlerklasse *Wort falsch erkannt*. Bei den anderen beiden Fehlerklassen steht die Verbesserung der Frageformulierung im Vordergrund. Daneben sollen zukünftig weitere Fehlerklassen mit Hilfe des Dialogagenten bearbeitet werden können. In diesem Zusammenhang besteht die Herausforderung in der Erstellung neuer Konzepte zur Lösung dieser Fehlerklassen und in der anschließenden Implementierung dieser Konzepte.
Zukünftig sollten die Schwellenwerte der einzelnen Fehlerklassen mit Hilfe von Verfahren des maschinellen Lernens berechnet und regelmäßig angepasst werden. Dadurch kann die Relevanz der gestellten Frage weiter gesteigert werden.
Darüber hinaus sollte der Dialogagent in Zukunft ausschließlich von englischen Muttersprachlern evaluiert werden, um dessen tatsächliche Fähigkeiten objektiver bewerten zu können.
Neben diesen Weiterentwicklungen des Dialogagenten bietet sich auch eine Erweiterung von PARSE um zusätzliche Funktionalitäten an, welche dem Dialogagenten indirekt helfen. Eine Möglichkeit wäre die Integration einer Komponente zur Stimmerkennung. Mit deren Hilfe soll die individuelle Tonspur eines jeden Menschen dazu verwendet werden, die Hintergrundgeräusche effektiver herauszufiltern. Dadurch steigt die Qualität der Resultate des Spracherkenners und somit das allgemeine Verständnis von Nutzeraussagen. Die Bereitstellung zusätzlicher Indikatoren durch PARSE würde dem Dialogagenten darüber hinaus helfen, korrekte Interpretationen des Nutzers häufiger als solche zu erkennen. Dies würde die Anzahl der Fragen, die lediglich der Verifizierung von Informationen dienen, deutlich senken und somit die Nutzerfreundlichkeit des Dialogagenten erhöhen.

Literaturverzeichnis

[ARA+06] Asfour, T ; Regenstein, K ; Azad, P ; Schröder, J ; Bierbaum, A ; Vahrenkamp, N ; Dillmann, R: ARMAR-III : An Integrated Humanoid Platform for Sensory-Motor Control. In: *2006 6th IEEE-RAS International Conference on Humanoid Robots* (2006), S. 169–175. `http://dx.doi.org/10.1109/ICHR.2006.321380`. – DOI 10.1109/ICHR.2006.321380 (zitiert auf Seite 22).

[BR03] Bohus, Dan ; Rudnicky, Alexander I.: RavenClaw : Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. (2003) (zitiert auf Seite 12).

[DB01] Dybkjær, Laila ; Bernsen, Niels O.: Usability Evaluation in Spoken Language Dialogue Systems. In: *Proceedings of the Workshop on Evaluation for Language and Dialogue Systems - Volume 9* (2001), 3:1—-3:10. `http://dx.doi.org/10.3115/1118053.1118055`. – DOI 10.3115/1118053.1118055 (zitiert auf den Seiten 4, 5 und 48).

[DDN05] Denecke, Matthias ; Dohsaka, Kohji ; Nakano, Mikio: Fast Reinforcement Learning of Dialogue Policies Using Stable Function Approximation. (2005). – ISSN 03029743 (zitiert auf Seite 15).

[Gei15] Geirhos, Matthias: *Entwurfsmuster: Das umfassende Handbuch*. 1. Auflage. Bonn : Rheinwerk Verlag GmbH, 2015. – 643 S. – ISBN 978–3–8362–2762–9 (zitiert auf Seite 47).

[Hey16] Hey, Tobias: *Kontext- und Korreferenzanalyse für gesprochene Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Diss., 2016. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/hey_ma` (zitiert auf den Seiten 3, 21, 22, 23 und 33).

[JM09] Jurafsky, Daniel ; Martin, James H.: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2. ed. Prentice Hall, Pearson Education International, 2009. – 1024 S. – ISBN 978–0–13–504196–3 (zitiert auf den Seiten 3, 4, 5, 6, 7, 20, 21 und 22).

[Koc15] Kocybik, Markus: *Projektion von gesprochener Sprache auf eine Handlungsrepräsentation*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Diss., 2015. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/kocybik_ba` (zitiert auf Seite 21).

[LJE+06] Lee, Cheongjae ; Jung, Sangkeun ; Eun, Jihyun ; Jeong, Minwoo ; Lee, Gary G.: A Situation-based Dialogue Management using Dialogue Examples. (2006) (zitiert auf Seite 11).

[LP97] Levin, Esther ; Pieraccini, Roberto: A Stochastic Model of Computer-human Interaction for Learning Dialogue Strategies. (1997) (zitiert auf Seite 14).

[McT98] McTear, Michael F.: Modelling spoken dialogues with state transition diagrams: experiences with the CSLU toolkit. In: *Icslp* 5 (1998), 4. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.6579&rep=rep1&type=pdf (zitiert auf den Seiten 9 und 10).

[MDS+12] Morbini, Fabrizio ; DeVault, David ; Sagae, Kenji ; Gerten, Jillian ; Nazarian, Angela ; Traum, David: FLoReS: A Forward Looking, Reward Seeking, Dialogue Manager. In: *4th International Workshop on Spoken Dialog Systems* (2012) (zitiert auf Seite 13).

[MRGRD14] Mariani, Joseph ; Rosset, Sophie ; Garnier-Rizet, Martine ; Devillers, Laurence: *Natural Interaction with Robots, Knowbots and Smartphones*. New York, New York, USA, 2014. – 397 S. http://dx.doi.org/10.1007/978-1-4614-8280-2. http://dx.doi.org/10.1007/978-1-4614-8280-2. – ISBN 978–1–4614–8279–6 (zitiert auf Seite 4).

[PCD+01] Pieraccini, R. ; Caskey, S. ; Dayanidhi, K. ; Carpenter, B. ; Phillips, M.: ETUDE, a recursive dialog manager with embedded user interface patterns. In: *2001 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2001 - Conference Proceedings* (2001), S. 244–247. http://dx.doi.org/10.1109/ASRU.2001.1034633. – DOI 10.1109/ASRU.2001.1034633. ISBN 078037343X (zitiert auf Seite 10).

[PLE97] Pieraccini, R ; Levin, E ; Eckert, W: AMICA: The AT&T Mixed Initiative Conversational Architecture. In: *Proc. of European Conference on Speech Communications and Technology (Eurospeech'97)* (1997), S. 1875–1878 (zitiert auf Seite 14).

[RPT00] Roy, N ; Pineau, J ; Thrun, S: Spoken Dialogue Management Using Probabilistic Reasoning. In: *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics* (2000), Nr. Acl, 93–100. http://dx.doi.org/10.3115/1075218.1075231. – DOI 10.3115/1075218.1075231 (zitiert auf Seite 16).

[RX99] Rudnicky, Alexander ; Xu, Wei: An agenda-based dialog management architecture for spoken language systems. In: *IEEE Automatic Speech Recognition and Understanding Workshop* (1999), 1–337. http://www.cs.cmu.edu/~xw/asru99-agenda.pdf (zitiert auf den Seiten 11 und 12).

[SBP97] Sadek, M.D. ; Bretier, P. ; Panaget, F.: ARTIMIS : Natural Dialogue Meets Rational Agency. (1997) (zitiert auf Seite 14).

[Sch04] Schlangen, David: Causes and Strategies for Requesting Clarification in Dialogue. In: *Proceedings of the 5th Workshop of the ACL SIG on Discourse and Dialogue* (2004) (zitiert auf Seite 5).

[SD03] Sang, Erik F. Tjong K. ; De Meulder, Fien: Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In: *CONLL '03 Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003* 4 (2003), 142–147. http://dx.doi.org/10.3115/1119176.1119195. – DOI 10.3115/1119176.1119195 (zitiert auf Seite 21).

[SP00] Seneff, Stephanie ; Polifroni, Joseph: Dialogue Management in the Mercury Flight Reservation System. In: *ANLP/NAACL 2000 Workshop on Conversational systems* - 3 (2000), 11–16. http://dx.doi.org/10.3115/1117562.1117565. – DOI 10.3115/1117562.1117565 (zitiert auf Seite 11).

[Ste16] Steurer, Vanessa: *Strukturerkennung von Bedingungen in gesprochener Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Diss., 2016. https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/steurer_ba (zitiert auf Seite 22).

[TD11] Tur, Gokhan ; De Mori, Renato ; Tur, Gokhan (Hrsg.) ; De Mori, Renato (Hrsg.): *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. Chichester, UK : John Wiley & Sons, Ltd, 2011. http://dx.doi.org/10.1002/9781119992691. http://dx.doi.org/10.1002/9781119992691. – ISBN 9781119992691 (zitiert auf den Seiten 5, 9, 11, 12 und 13).

[Tho09] Thomson, Blaise: *Statistical methods for spoken dialogue management*, Diss., 2009. http://mi.eng.cam.ac.uk/~brmt2/papers/2010-thomson-thesis.pdf. – 178 S. (zitiert auf den Seiten xi, 5, 6, 7, 9, 10, 12, 14 und 16).

[TSY08] Thomson, Blaise ; Schatzmann, Jost ; Young, Steve: Bayesian update of dialogue state for robust dialogue systems. In: *Proceedings of ICASSP* (2008), S. 4937–4940. ISBN 1424414849 (zitiert auf Seite 16).

[WT15] Weigelt, Sebastian ; Tichy, Walter F.: ProNat: An Agent-Based System Design for Programming in Spoken Natural Language. In: *Proceedings - International Conference on Software Engineering* 2 (2015), S. 819–820. http://dx.doi.org/10.1109/ICSE.2015.264. – DOI 10.1109/ICSE.2015.264. – ISBN 9781479919345 (zitiert auf den Seiten 19, 20 und 23).

Anhang

A. Evaluationsszenarien

A.1. Scenario 8

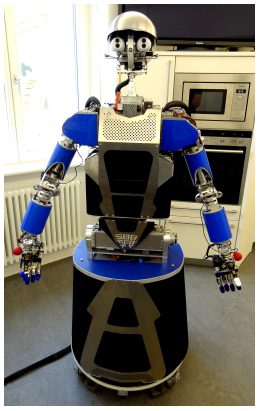
You have bought the new household robot Armar III. At the moment it is just able to fulfill basic tasks. The good news is the robot is able to learn more sophisticated tasks. Therefor you have to explain him a bunch of basic tasks in the correct order. In the case Armar understood everything it has learned a new skill. Otherwise the robot will ask you about the parts of your statement it did not understand. Since Armar is able to understand context, there might come up questions about the context and conditions in the text.

At the moment you are sitting in front of a laptop and have to do very interesting stuff. So you want to teach Armar to take out the laundry of the washing machine and put it into the dryer.

The robot is already aware that you will teach him a new instruction. So it will be fine if your statement just contains the instruction.

Some basic task examples:

- *Go to the cupboard and open it.*
- *Armar cut the orange into halves.*
- *Take the water bottle and bring it to me.*


(a) Armar, the robot


(b) closed washing machine

Advices:

- *Imagine the essential steps you would do putting the laundry into the dryer after the washing machine has finished.*
- *Say or imagine the complete instruction at least once before you start the record.*
- *The robot will ask you a lot of questions. Please be patient and keep calm.*

Thank you for participating.

A.2. Correct instructions

In the following part you should answer the questions of Armar for already expressed instructions.

A.2.1. Scenario 6

In this scenario you told the robot:

Hey Armar, can you please get the green cup from the table, it is located quite in the middle, please fill it afterwards with water from the fridge, the water is in the fridge right next to the orange juice, then you can bring the cup to me. Afterwards empty the dishwasher, there are red cups in there, so you have to open it and then put them into the cupboard.

Please answer the questions of Armar.

A.2.2. Scenario 4

In this scenario you told the robot:

Hi robo, please look at the table, if there are dirty dishes please put it in the dishwasher and if there are clean dishes put it in the cupboard.

Please answer the questions of Armar.