Karlsruhe Institute of Technology

Themenextraktion zur Domänenauswahl für Programmierung in natürlicher Sprache

Masterarbeit
von

Jan Keim

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

Erstgutachter:	Prof. Dr. Walter F. Tichy
Zweitgutachter:	Jun.-Prof. Dr.-Ing. Anne Koziolek
Betreuender Mitarbeiter:	Dipl.-Inform. Sebastian Weigelt
Zweiter betr. Mitarbeiter:	M.Sc. Tobias Hey

Bearbeitungszeit: 24.08.2017 – 23.02.2018

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

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

Karlsruhe, 23.02.2018

. . . . . .

(**Jan Keim**)

Kurzfassung

Für den Menschen sind Kontexte von Anweisungen für die Programmierung in natürlicher Sprache einfach ersichtlich, für den Rechner ist dies nicht der Fall. Eine Art des Kontextwissens ist das Verständnis der Themen. Hierfür wird im Rahmen des PARSE-Projekts zur Programmierung in natürlicher Sprache ein Ansatz zur Themenextraktion vorgestellt, welcher mit Hilfe der Wikipedia als Informationsquelle passende Themen aus den Anweisungen extrahiert. Dafür wird eine Auflösung von mehrdeutigen Nomen benötigt, weshalb in dieser Arbeit ebenfalls ein Werkzeug dafür entwickelt wird. Als einen konkreten Anwendungsfall für die extrahierten Themen wird die Auswahl von passenden Ontologien zu diesen Themen angegangen. Durch diese Auswahl wird ermöglicht, statt einer großen allgemeinen Ontologie mehrere kleine domänenspezifische Ontologien einzusetzen. Auf einem Korpus mit Anweisungen für die Programmierung in natürlicher Sprache konnte für die Auflösung von Mehrdeutigkeiten eine Genauigkeit von 87,80% erreicht werden. Um die Themenextraktion zu evaluieren, wurde eine Umfrage durchgeführt, die ergab, dass das erste extrahierte Thema in bis zu 63,6% der Fälle treffend war. Unter den ersten vier extrahierten Themen konnte sogar in knapp 91% der Fälle mindestens ein passendes Thema gefunden werden. Zuletzt wurden in der Evaluation der Ontologieauswahl gute Ergebnisse erzielt, wobei ein F_1-Maß von 90,67% und ein F_2-Maß von 89,94% erreicht wurden. Dabei wurde festgestellt, dass Schwierigkeiten auftreten, wenn bei der Auswahl mehrere Ontologien benötigt werden. Der Ansatz könnte zukünftig daher mit einer verbesserten Erkennung von unterschiedlichen Themen in der Eingabe weiter ausgebaut werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Im Alltag findet man heutzutage unzählige Computer und digitale Systeme. Diese reichen von Laptops über Smartphones bis hin zu intelligenten Endgeräten wie Fernsehern oder Kühlschränken und sind vielfach in den Haushalten vertreten. Sie sollen den Nutzern den Alltag erleichtern und das Leben komfortabler gestalten. Vor allem digitale Assistenzsysteme wie Amazon Echo, Apples HomePod oder Google Home liegen aktuell im Trend. Viele Systeme, die heutzutage zu finden sind, lassen sich auch erweitern, indem durch Software neue Funktionalitäten hinzugefügt werden. Das bringt die Freiheit, Geräte nach individuellen Bedürfnissen zu konfigurieren und anzupassen. Vielen Menschen fehlt es allerdings an einer grundlegenden Voraussetzung, dem Beherrschen von Programmiersprachen. In Zukunft werden digitale Systeme, Roboter und andere softwaregestützte Geräte immer wichtiger werden. Damit möglichst viele Menschen die Vorteile der Digitalisierung nutzen können, soll die die Hürde zur Erstellung eigener Programme gesenkt werden. Dies soll unter anderem mit Hilfe der Programmierung in gesprochener Sprache ermöglicht werden. Die Menschen werden unter anderem durch die virtuellen Assistenzsysteme immer mehr an das Bedienelement Sprache gewöhnt, wodurch sich die Sprache hervorragend eignet, um Systemen neue Funktionalität beizubringen und dadurch den Nutzern mehr Freiheiten zu ermöglichen.

Allerdings bringt die Aufgabe, ein System für die Programmierung in gesprochener Sprache zu erstellen, viele verschiedene Herausforderungen mit sich. Um Anweisungen korrekt umsetzen zu können, ist es wichtig, dass alle dazu notwendigen Informationen aus der gesprochenen Eingabe extrahiert werden können. Dabei muss auch ein Verständnis für die Art des Befehls, seinen Kontext und seine Domäne aufgebaut werden. Das Wissen über den Kontext und die Domäne soll helfen, Missverständnisse zu minimieren und mögliche Aktivitäten zu erkennen. Kontextwissen ist vielfältig und kann über verschiedene Formen erhalten werden. Eine Art des Kontextwissens ist das Verständnis des Themas von Dialogen beziehungsweise Dialogteilen. Der Mensch kann intuitiv die Thematiken bestimmen. Für die Eingabe „Roboter, gehe zum Tisch und greife das Glas“ können beispielsweise vom Menschen mühelos Themen wie „Haushalt“ oder „Küche“ extrahiert werden. Rechner sind dazu nicht ohne Weiteres in der Lage. Zur Themenextraktion gibt es bereits verschiedene Ansätze, die jedoch nicht auf die Programmierung in gesprochener Sprache ausgelegt sind und deshalb nicht dafür eingesetzt werden können. Ziel dieser Arbeit ist es daher, ein System zu Extraktion von Themen aus Anweisungen für die Programmierung in gesprochener Sprache zu erstellen.

Für die Themenextraktion wird eine Auflösung von mehrdeutigen Nomen benötigt. Ohne

diese ist beim Verarbeiten unklar, ob beispielsweise mit „Bank“ das Finanzinstitut, die Sitzgelegenheit oder eine der anderen Bedeutungen gemeint ist. Daher ist für eine korrekte Themenextraktion wichtig, dass die Bedeutung von mehrdeutigen Wörtern bekannt ist. Aus diesem Grund wird in dieser Arbeit zuerst die Auflösung von Mehrdeutigkeiten angegangen. Das Problem mehrdeutige Wörter aufzulösen existiert in der Rechnerlinguistik seit langem. Bereits 1955 wurde es in [Yng55] als Problem im Bereich der maschinellen Übersetzung erkannt. Bis heute gilt es als eines der schwersten Probleme der Verarbeitung natürlicher Sprache [SW03]. Über Kontextinformationen kann der Mensch die Mehrdeutigkeiten in den meisten Fällen zuverlässig auflösen. Diese Informationen müssen den Rechnern entsprechend vermittelt werden, damit eine Auflösung der Mehrdeutigkeiten stattfinden kann.

Ein möglicher Einsatzbereich der Themenzuordnung ist das Zuordnen der Eingaben zu verschiedenen domänenspezifischen Ontologien. In den Ontologien sind Informationen zu den jeweiligen Domänen enthalten, wie beispielsweise Informationen über Objekte, die auftreten können, oder über Methoden, die ein Zielsystem ausführen kann. Diese Informationen können genutzt werden, um nötige Informationen für die Quelltextgenerierung zu erhalten. Aktuell wird für die Informationsbeschaffung eine große allgemeine Ontologie benutzt. Große Ontologien haben den Nachteil, dass der Aufbau und die Wartung komplex und aufwändig sind. Kleinere domänenspezifische Ontologien sind entsprechend leichter zu erstellen und der Wartungsaufwand bei Änderungen ist geringer, da unter anderem weniger Konsistenzprobleme auftreten. Zusätzlich kann die Abbildung des Textes auf die Ontologie vereinfacht werden, da die Abbildungsmöglichkeiten geringer sind. Wenn jedoch mehrere Ontologien eingesetzt werden sollen, dann benötigt es eine Auswahl der passenden Ontologien. Mit Hilfe der in der Themenextraktion ermittelten Themen kann ermittelt werden, welche der verschiedenen zur Auswahl stehenden Ontologien passend sind und ausgewählt werden sollen. Die Entwicklung des Systems zur Auswahl der passenden Ontologien soll daher in dieser Arbeit angegangen werden.

Die Arbeit ist wie folgt strukturiert: In Kapitel 2 werden die Grundlagen eingeführt, die für das Verständnis dieser Arbeit notwendig sind. Danach wird in Kapitel 3 das Projekt *PARSE* vorgestellt und auf eine kurze Übersicht über Arbeiten zu diesem Projekt gegeben. Die verwandten Arbeiten zur Themenextraktion sowie zum Verschmelzen von Ontologien werden in Kapitel 4 behandelt. In Kapitel 5 werden die Problemstellungen der Auflösung von Mehrdeutigkeiten, der Themenextraktion für die Programmierung in natürlicher Sprache sowie die Ontologieauswahl betrachtet und mögliche Ansätze analysiert, bevor ein Konzept zu Umsetzung entwickelt wird. Die Beschreibung der Implementierung der verschiedenen Werkzeuge zur Auflösung von Mehrdeutigkeiten, zur Themenextraktion sowie zur Auswahl geeigneter Ontologien ist in Kapitel 6 zu finden. Die Evaluation sowie eine Diskussion über die Qualität der Ansätze finden sich in Kapitel 7. Zuletzt wird in Kapitel 8 eine Zusammenfassung der Arbeit gegeben und die Arbeit mit einem Ausblick auf mögliche zukünftige Arbeiten abgeschlossen.

2. Grundlagen

Natürliche Sprache ist die Art und Weiße, wie wir alltäglich kommunizieren. Die maschinelle Verarbeitung von natürlicher Sprache ist jedoch nicht unproblematisch. Dabei bringen vor allem die Freiheiten der natürlichen Sprache Probleme mit sich, die die Disziplin der Verarbeitung von natürlicher Sprache zu lösen versucht. Die Verarbeitung von natürlicher Sprache funktioniert dabei häufig über maschinelle Lernverfahren, dessen Grundlagen in Abschnitt 2.1 vorgestellt werden. Danach wird in Abschnitt 2.2 auf Metriken eingegangen, die für die Evaluation von Klassifikatoren genutzt werden können. In Abschnitt 2.3 werden dann die Grundlagen zur Verarbeitung von natürlicher Sprache vorgestellt. Ein Problem bei maschinellen Lernverfahren, vor allem in Verbindung mit natürlicher Sprache, ist häufig, genug Trainingsdaten zu erhalten. Eine mögliche Datenquelle ist die Wikipedia; eine Einführung zu Wikipedia ist in Abschnitt 2.6 zu finden. Um Abhängigkeiten und Beziehungen zwischen Entitäten kennzeichnen zu können, ist es wichtig, geeignete Datenstrukturen einzusetzen. Graphen sind eine Möglichkeit dafür und werden in Abschnitt 2.4 eingeführt. Des Weiteren sind Ontologien eine Option, gerade in der Verarbeitung von Sprache, um Wissen darzustellen. Eine Einführung dazu ist in Abschnitt 2.5 zu finden.

2.1. Maschinelles Lernen

Gerade im Bereich der Verarbeitung natürlicher Sprache wird häufig maschinelles Lernen eingesetzt. Eine Definition für maschinelles Lernen ist unter anderem bei Mitchell [Mit97] zu finden:

Definition 2.1.1. Ein System lernt aus Erfahrung E in Hinblick auf eine Klasse von Aufgaben T und einem Performanzmaß P, wenn seine Leistungen bei Aufgaben aus T gemessen mit P durch Erfahrung aus E steigt.

Daraus lässt sich ableiten, dass der Begriff *Maschinelles Lernen* sehr weit gefasst ist und unterschiedlichste Lösungen zielbringend sein können. Zum Beispiel lässt das Performanzmaß P viele Freiheiten zu. So kann unter anderem für P die Genauigkeit der Ergebnisse gemeint sein, aber auch die benötigte Zeit, um das Ergebnis zu erhalten. Gerade in Echtzeitsystemen kann ein schnelleres Ergebnis das gewünschte erlernte Verhalten sein, während in anderen Systemen, wie beispielsweise zur Auflösung von Mehrdeutigkeiten, eher die Genauigkeit wichtig ist.

Maschinelle Lernverfahren lassen sich anhand verschiedener Merkmale unterteilen. Zunächst gibt es die Unterscheidung in überwachte und unüberwachte Lernverfahren. Der

Unterschied der beiden Verfahren liegt darin, ob Trainingsdaten mit den erwarteten Ergebnissen annotiert vorliegen oder nicht. Außerdem unterscheidet man zwischen induktiven und deduktiven Lernverfahren. Induktion ist der Prozess des plausiblen Schließens vom Speziellen zum Allgemeinen, wodurch man die „Wahrheit erweitert“. Deduktion ist andererseits wahrheitserhaltend und setzt darauf, über logische Schlüsse aus den Prämissen die Konsequenzen zu ermitteln. Unter anderem kann dadurch die explizite Darstellung implizit vorhandenen Wissens entstehen. Dadurch ist deduktives Lernen auch stets korrekt, wenn das Ausgangswissen korrekt ist. Allerdings ist das Problem von deduktivem Lernen, dass sich das Ausgangswissen oft gar nicht oder nur sehr schwer formulieren lässt. Bei induktiven Lernen existiert dieses Problem nicht, allerdings wird nach [GD95] durch die Trainingsdaten oder durch das Verfahren auch immer eine Voreingenommenheit des Klassifikators erzeugt. Hier steht man vor der Aufgabe diese Voreingenommenheit möglichst gering zu halten, indem nach Möglichkeit ausgeglichene Trainingsdaten verwendet. Des Weiteren können Lernverfahren noch darin unterschieden werden, ob sie inkrementell arbeiten können, man also später durch weitere Trainingsdaten den Klassifikator erweitern kann, ohne ihn komplett neu lernen zu lassen.

Ein Beispiel für ein induktives, überwachtes lernendes System, das im Bereich der natürlichsprachigen Verarbeitung häufig eingesetzt wird, ist der Naive-Bayes Klassifikator [Mur06]. Für diesen empirischen Lerner werden für jede Instanz anhand ihrer Merkmale die Wahrscheinlichkeiten ermittelt, dass die Instanz zu einer der verschiedenen Klassen gehört. Das System erhielt seinen Namen aufgrund des Theorems von Bayes [BP63], das in Gleichung 2.1 zu sehen ist.

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)} \tag{2.1}$$

Das Theorem besagt, dass sich die *a-posteriori*-Wahrscheinlichkeit durch die Kombination von drei Wahrscheinlichkeiten berechnen lässt. Die *a-posteriori*-Wahrscheinlichkeit $P(C_k|x)$ ist die Wahrscheinlichkeit, dass eine Hypothese auf Grund der beobachteten Daten wahr ist. Der erste Ausdruck der Formel ist $P(x|C_k)$, die Auftrittswahrscheinlichkeit der beobachteten Daten, wenn die Hypothese gilt. Der zweite Ausdruck im Zähler stellt die Wahrscheinlichkeit dar, dass eine Hypothese gültig ist. Diese Wahrscheinlichkeit wird *a-priori*-Wahrscheinlichkeit genannt. Im Nenner der *a-posteriori*-Wahrscheinlichkeit steht die allgemeine Auftrittswahrscheinlichkeit der beobachteten Daten.

Wenn von unabhängigen Merkmalen ausgegangen wird, dann wird eine sehr große Menge an Trainingsdaten benötigt, damit die Wahrscheinlichkeiten für jede Kombination der Merkmale mit jeweils ihren möglichen Werten zufriedenstellend festgestellt werden kann. Ebenso ist die Berechnung entsprechend komplex. Der Zähler kann wie in Gleichung 2.2 dargestellt werden.

$$\begin{aligned} p(C_k, x_1, ..., x_n) &= p(x_1, ..., x_n, C_k) \\ &= p(x_1|x_2, ..., x_n, C_k)p(x_2|x_3, ..., x_n, C_k)...p(x_{n-1}|x_n, C_k)p(x_n|C_k)p(C_k) \end{aligned} \tag{2.2}$$

Deshalb wird beim naiven Modell davon ausgegangen, dass die Merkmale unabhängig voneinander sind und nur noch der Bezug zur Klasse entscheidend ist. Dadurch kann nun wie in Gleichung 2.3 vereinfacht werden. Das Modell aus Gleichung 2.2 kann dann durch das Modell aus Gleichung 2.4 ersetzt werden. Für einen Klassifikator muss nun lediglich die Klasse ermittelt werden, für die die Wahrscheinlichkeit am größten ist. Die Formel dazu ist in Gleichung 2.5 zu sehen.

$$p(x_i|x_{i+1}, ..., x_n, C_k) = P(x_i|C_k) \tag{2.3}$$

$$p(C_k|x_1, ..., x_n) = p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{2.4}$$

$$y = \underset{k \in \{1,...,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{2.5}$$

Durch das naive Modell werden weniger Trainingsdaten benötigt, für einen genauen Klassifikator ist eine umfangreiche Beispielmenge dennoch essentiell. Nach Russel und Norvig [RN95] skaliert das Naive-Bayes-Lernen auch gut für große Probleme, da vor allem keine aufwändige Suche benötigt wird. Ebenso kommt dieses Lernverfahren auch mit verrauschten oder fehlenden Daten zurecht, weil es lediglich auf Wahrscheinlichkeiten basiert.

Um die verschiedenen Lernalgorithmen und lernenden Systeme wie den Naive-Bayes-Klassifikator einsetzen zu können, gibt es bereits verschiedene Bibliotheken. Eine solche Bibliothek für die Programmiersprache Java ist Weka [WFHP16], die in dieser Arbeit eingesetzt wird. Weka bietet neben den verschiedenen Lernalgorithmen auch Algorithmen für die Vorverarbeitung der Daten sowie bei Bedarf eine grafische Bedienoberfläche, mit der sich Daten und Ergebnisse visualisieren lassen.

2.2. Beurteilung eines Klassifikators

Bei der Klassifizierung, vor allem beim Einsatz von statistischen Verfahren, kann eine Klassifikation sowohl korrekt als auch inkorrekt sein. Um einen Klassifikator anhand seiner Ergebnisse beurteilen zu können, gibt es verschiedene Metriken, die in [Pow11] gesammelt sind und von denen eine Auswahl hier vorgestellt werden sollen.

Die Ergebnisse eines binären Klassifikators lassen sich in in verschiedene Kategorien einteilen. Zum einen gibt es Elemente, die vom Klassifikator als positiv eingestuft wurden. Dabei kann der Klassifikator richtig klassifiziert haben, wodurch man ein richtig-positives Ergebnis erhält. Wurde fälschlicherweise das Element als positiv kategorisiert, dann wird das Ergebnis falsch-positiv genannt. Das selbe gilt für negative Testergebnisse: Wenn das Element korrekt als negativ eingestuft wurde, dann ist das Ergebnis richtig-negativ. Entsprechend erhält man ein falsch-negatives Ergebnis, wenn es fälschlicherweise negativ kategorisiert wurde. Mit Daten über die Verteilung der Ergebnisse in diesen Kategorien können Metriken zur Beurteilung des Klassifikators berechnet werden. Zur Einordnung der Ergebnisse in diese Kategorien benötigt es einen Goldstandard. Im Goldstandard ist vermerkt, was das gewünschte Ergebnis einer Klassifikation eines Elements ist. Durch den Vergleich der Ergebnisse des Klassifikators mit dem Goldstandard werden die jeweiligen Ergebnisse in die vier Kategorien eingeordnet.

Eine oft genutzte Metrik ist die *Präzision* (*Precision*), mit der bestimmt wird, wie häufig positive Ergebnisse korrekt klassifiziert wurden. Dabei wird der Anteil an richtig-positiven Ergebnissen unter allen als positiv klassifizierten Ergebnissen wie in Gleichung 2.6 bestimmt. Je höher die Präzision ist, desto mehr positive klassifizierte Elemente wurden korrekt klassifiziert. Bei einer Präzision von Null wurde kein positiv klassifiziertes Element korrekt klassifiziert, während bei einer Präzision von Eins alle positiv klassifizierten Elemente korrekt klassifiziert wurden.

$$\text{Präzision} = \frac{\#\text{richtig-positiv}}{\#\text{richtig-positiv} + \#\text{falsch-positiv}}, \text{Präzision} \in [0, 1] \tag{2.6}$$

Eine weitere häufig genutzte Metrik ist die *Ausbeute* (*Recall*), bei der der Anteil der gefundenen Elemente, die korrekt als positiv eingestuft wurden, an der Menge aller Elemente,

die positiv sind, bestimmt wird. Entsprechend ist die Ausbeute aus Gleichung 2.7 hoch, wenn viele der positiv einzuordnenden Elemente auch positiv eingeordnet wurden. Eine niedrige Ausbeute weißt darauf hin, dass viele der eigentlich positiven Elemente nicht als solche erkannt wurden.

$$\text{Ausbeute} = \frac{\#\text{richtig-positiv}}{\#\text{richtig-positiv} + \#\text{falsch-negativ}}, \text{Ausbeute} \in [0, 1] \tag{2.7}$$

Präzision und Ausbeute sind Metriken, die sich in vielen Fällen gegenseitig beeinflussen. Eine hohe Ausbeute kann beispielsweise negative Auswirkungen auf die Präzision haben. Wenn zum Beispiel ein Klassifikator jedes Element als positiv bewertet, dann liegt die Ausbeute bei 100%, während die Präzision entsprechend niedrig ausfällt. Deshalb sollten beide Metriken zusätzlich noch zusammen betrachtet werden. Aus diesem Grund gibt es das *F-Maß* (*F-score*), das ein gewichtetes harmonisches Mittel der Präzision und Ausbeute ist. Dadurch nimmt das *F-Maß* Werte zwischen Null und Eins an. Die allgemeine Formel des F-Maß ist in Gleichung 2.8 zu finden. Das α bestimmt die Gewichtung von Ausbeute und Präzision, wobei beim Wert von Eins, dem F_1-Maß, das in Gleichung 2.9 zu sehen ist, ein Gleichgewicht herrscht. Je größer der Wert von α ist, desto stärker wird die Ausbeute gewichtet, während bei Werten, die sich der Null annähern, die Präzision stärker gewichtet wird. Folglich wird beim F_2-Maß in Gleichung 2.10 die Ausbeute viermal so hoch wie die Präzision gewichtet.

$$\mathrm{F}_\alpha = (1 + \alpha^2) * (\text{Präzision} * \text{Ausbeute}) / (\alpha^2 * \text{Präzision} + \text{Ausbeute}) \tag{2.8}$$

$$\mathrm{F}_1 = 2 * (\text{Präzision} * \text{Ausbeute}) / (\text{Präzision} + \text{Ausbeute}) \tag{2.9}$$

$$\mathrm{F}_2 = 5 * (\text{Präzision} * \text{Ausbeute}) / (4 * \text{Präzision} + \text{Ausbeute}) \tag{2.10}$$

Zuletzt wird hier die *Ausfallrate* (*Fall-out*), auch Falsch-Positiv-Rate genannt, vorgestellt. Bei ihr werden diejenigen Elemente betrachtet, die fälschlich als positiv klassifiziert wurden. Die Ausfallrate in Gleichung 2.11 ist somit der Anteil an fälschlicherweise als positiv bewerteten negativen Elemente. Eine hohe Ausfallrate weißt entsprechend darauf hin, dass negative Elemente oft positiv klassifiziert werden.

$$\text{Ausfallrate} = \frac{\#\text{falsch-positiv}}{\#\text{richtig-negativ} + \#\text{falsch-positiv}}, \text{Ausfallrate} \in [0, 1] \tag{2.11}$$

2.3. Verarbeitung von natürlicher Sprache

Die Verarbeitung von natürlicher Sprache, im Englischen *Natural Language Processing*, kurz *NLP*, genannt, beschäftigt sich damit, Sprache rechnergestützt zu verarbeiten. Die rechnergestützte Sprachverarbeitung beinhaltet grundlegende Tätigkeiten wie die Extraktion von Satzinformationen, zu der man unter anderem die Annotation von Wortarten oder die Ermittlung von Abhängigkeiten in Sätzen zählen kann. Aber auch konkrete Anwendungsfälle werden mit der rechnergestützten Sprachverarbeitung angegangen. Zu diesen gehören unter anderem die Analyse der Stimmungslage in Texten und die Zuordnung von Dokumenten zu Themen, aber auch die Übersetzung in andere Sprachen. Auch die Generierung von Quelltext aus gesprochener Sprache, wie es das Ziel von [Wei] ist, ist eines der konkreten Anwendungsfälle. Im Folgenden sollen Grundlagen zur Annotation von Wortarten in Abschnitt 2.3.1, zur Syntaxanalye in Abschnitt 2.3.2 sowie zur Auflösung von Mehrdeutigkeiten in Abschnitt 2.3.3 und der Themenextraktion in Abschnitt 2.3.4 vermittelt werden.

Tabelle 2.1.: Auszug aus dem Penn Treebank Set an Etiketten auf Wortebene [MMS93]

Etikett	**Bedeutung**
CC	Konjunktion
CD	Kardinalzahl
DT	Determinativ
IN	Präposition oder unterordnende Konjunktion
JJ	Adjektiv
NN	Nomen, singular
NNS	Nomen, plural
NNP	Eigenname, singular
NNPS	Eigenname, plural
PRP	Possessivpronomen
RB	Adverb
VB	Verb, Grundform
VBD	Verb, Vergangenheitsform
VBN	Verb, Partizip Perfekt
VBP	Verb, Gegenwart, nicht dritte Person singular
VBZ	Verb, Gegenwart, dritte Person singular
WDT	Determinativ

2.3.1. Annotation von Wortarten

Ein wichtiger Bestandteil der Verarbeitung natürlicher Sprache ist das Annotieren von Wortarten, wie sie in Abbildung 2.1 unterhalb der Wörter zu sehen sind. So ist „ jumped“ ein Verb, während „fox“ und „dog“ Nomen und „quick“, „brown“ und „lazy“ Adjektive darstellen. Dies erkennt man an den annotierten Etiketten. Diese Etiketten sind mit dem *Penn Treebank* Projekt [MMS93] eingeführt worden. Beispielsweise steht *NN* für Nomen, *DT* für Artikel, *VBD* für Verben in der Vergangenheitsform und *JJ* für Adjektive. Ein Ausschnitt an Etiketten und ihren Bedeutungen ist in Tabelle 2.1 zu sehen.

The	quick	brown	fox	jumped	over	the	lazy	dog
DT	JJ	JJ	NN	VBD	IN	DT	JJ	NN

Abbildung 2.1.: Satz mit annotierten Wortarten, annotiert durch das Werkzeug Stanford CoreNLP [MSB+14]

Viele weiterführende Analysen basieren auf Wortart-Annotationen. Aus diesem Grund ist eine hohe Genauigkeit essentiell, da Fehler in diesem Schritt auch alle daraus folgenden Schritte beeinflussen können. Gerade mehrdeutige Wortformen stellen eine der Herausforderungen dar. Beispielsweise ist das Wort „orange“ im Satz „I feel like eating an orange“ ein Nomen, während es im Satz „He wears orange sneakers“ ein Adjektiv ist. Obwohl das Problem der Mehrdeutigkeiten existiert, bekommt man nach Charniak [Cha97] bereits gute Ergebnisse mit einer Genauigkeit von etwa 90%, wenn lediglich die häufigste Wortform pro Wort annotiert wird. Andere Verfahren zum Annotieren der Wortarten, die die Fehler der naiven Variante minimieren sollen, setzen unter anderem *Hidden Markov Modelle* [Blu04] oder den regelbasierten Etikettierer nach Brill [Bri92] ein. Bei Hidden Markov Modellen wird ein Zustandssystem modelliert, indem von jedem Zustand Übergangswahrscheinlichkeiten in andere Zustände existieren. Dabei hängen die Übergangswahrscheinlichkeiten ausschließlich vom aktuellen Zustand ab. Dieses Modell ist „hidden“, also verborgen, weil die Zustände von außen nicht beobachtet werden können, sondern lediglich die *Emissionen*

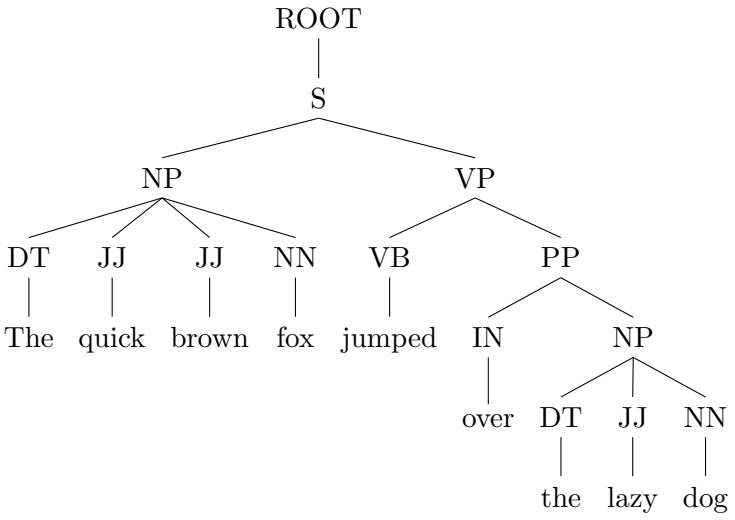

Abbildung 2.2.: Beispiel eines Syntaxbaumes, annotiert durch das Werkzeug Stanford CoreNLP [MSB+14]

genannten Ausgabesymbole. Bei der Wortartmarkierung wird auf diese Weise die Auftrittswahrscheinlichkeit von Wörtern in Verbindung mit ihren Wortarten als Ausgabesymbole modelliert. Beispielsweise kann damit das Wissen, dass nach einem Artikel oft ein Nomen folgt, modelliert werden. Aktuelle Systeme zur Annotation von Wortarten wie etwa *CoreNLP* [MSB+14] kommen auf Genauigkeiten von etwa 97%.

2.3.2. Syntaxanalyse

Ein weiterer grundlegender Bestandteil der Verarbeitung natürlicher Sprache ist die Syntax- bzw. Satzanalyse. Dabei wird aus dem Satz ein Syntaxbaum erstellt, wie er in dem Beispiel in Abbildung 2.2 dargestellt ist. Dies ermöglicht dem Nutzer die Analyse der Syntax des Satzes und der Verbindungen der einzelnen Wörter untereinander. So ist in dem Beispiel unter anderem die Nominalphrase (*NP*) „the quick brown fox“ zu sehen. Der Satzteil „ jumped over the lazy dog“ stellt eine Verbalphrase dar und ist entsprechend mit *VP* gekennzeichnet. Phrasen können auch andere Phrasen enthalten, wie etwa die Präpositionalphrase (*PP*) „over the lazy dog“ innerhalb der Verbalphrase enthalten ist. Solche Syntaxbäume werden mit der Hilfe von kontextfreien Grammatiken erstellt [Jur09]. Die resultierenden Bäume sind nicht immer eindeutig, da es bei semantisch mehrdeutigen Sätzen auch verschiedene Ableitungen nach den Grammatikregeln geben kann. Beispielsweise ist der Satz „I saw the man with a telescope“ mehrdeutig. Abbildung 2.3 zeigt zwei mögliche Syntaxbäume für diesen Satz. Auf der einen Seite gibt es die Interpretation, dass man den Mann gesehen hat, der bei einem Teleskop stand. Eine andere Interpretation lautet, dass man den Mann gesehen hat, indem man ein Teleskop nutzte.

Ein weitere Form, die Verbindung zwischen den Wörtern darzustellen, ist der Abhängigkeitsgraph. Bei Abhängigkeitsgraphen wie dem aus dem Beispiel in Abbildung 2.4 werden die Abhängigkeiten zwischen Wörtern ermittelt. Im Beispiel existiert unter anderem eine Abhängigkeit vom Typ „amod“ zwischen „fox“ und „brown“, somit ist „brown“ in diesem Satz ein Adjektiv, das das Wort „fox“ näher beschreibt. Weiterhin gibt es unter anderem eine Abhängigkeit von „ jumped“ zu „fox“ vom Typ „nsubj“, was bedeutet, dass in diesem Satz „fox“ das Subjekt ist.

Zuletzt gibt es noch Zerleger, die den Satz in einfache, nichtrekursive Phrasen unterteilen, im Englischen *Chunking* genannt. Dadurch entstehen nicht-überlappende Abschnitte des Satzes, nach [Jur09] üblicherweise Nominalphrasen, Verbalphrasen, Adjektivphrasen

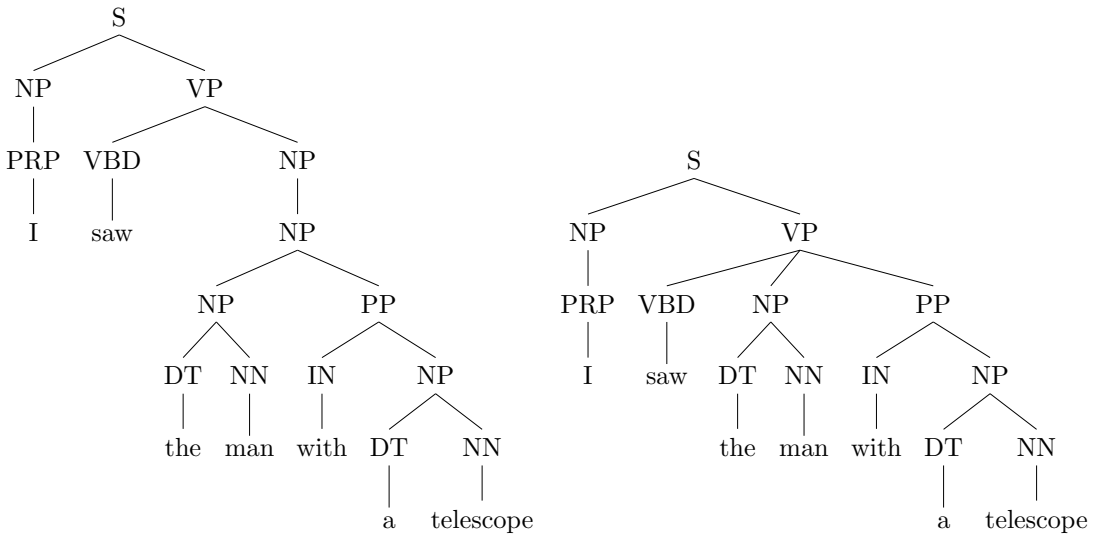

Abbildung 2.3.: Beispiel eines nicht eindeutigen Satzes und der daraus resultierenden Syntaxbäume

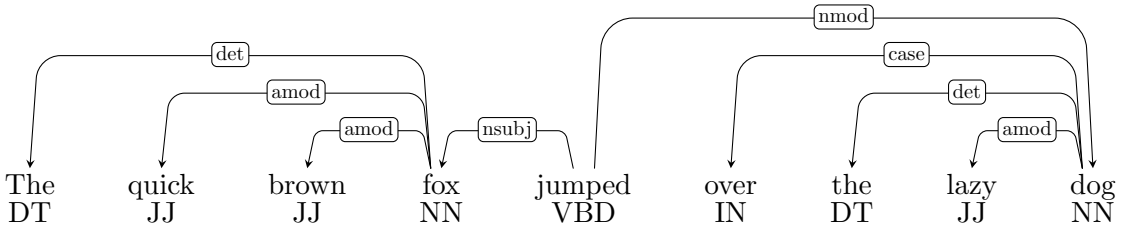

Abbildung 2.4.: Abhängigkeiten im Satz, annotiert durch das Werkzeug Stanford CoreNLP [MSB+14]

und Präpositionalphrasen. Damit können die inhaltstragenden Elemente jeweils ermittelt werden, wobei die Hierarchien vernachlässigt werden. Oftmals wird dieses vereinfachte Verfahren angewandt, wenn beispielsweise nur das Interesse an der Ermittlung der Hauptnominalphrasen existiert.

2.3.3. Auflösung von Mehrdeutigkeiten

Eine Problematik der Verarbeitung natürlicher Sprache, die in der Rechnerlinguistik seit langem existiert, ist das Auflösen von Mehrdeutigkeiten, im Englischen *Word Sense Disambiguation (WSD)* genannt. Bereits 1955 ist es in [Yng55] als ein Problem im Bereich der maschinellen Übersetzung erkannt worden und gilt bis heute als eines der schwersten Probleme im Bereich der Verarbeitung natürlicher Sprache [SW03]. Russel und Norvig bezeichnen in [RN95] dieses Problem als sehr wichtig im Bereich des Verständnisses von natürlicher Sprache und geben an, dass die Disambiguierung abhängig ist vom Wissen über die Welt, aktuelle Situationen und den Gebrauch der Sprache.

Während der Mensch durch Kontextinformationen Mehrdeutigkeiten größtenteils zuverlässig auflösen kann, muss dieses Wissen den Rechnern entsprechend vermittelt werden. Ein Beispiel für ein mehrdeutiges Wort wäre „table“. Je nach Satz und Kontext kann mit „table“ ein Möbelstück, ein Set aus Daten, das in Zeilen und Spalten angeordnet ist, ein Begriff für eine Gesellschaft, die an einem Tisch versammelt ist, oder eine Landschaftsform gemeint sein. Im Satz „Robot, go to the table and grab the glas“ ist zum Beispiel die Bedeutung als Möbelstück gemeint, wobei der Kontext des Satzes entscheidend sein kann. Viele Ansätze zur Auflösung der Mehrdeutigkeiten nutzen probabilistische Systeme wie beispielsweise den Naive-Bayes-Klassifikator aus Abschnitt 2.1. Ein einfacher probabilistischer Ansatz wäre, die häufigste und dadurch wahrscheinlichste Bedeutung jedes Wortes zu ermitteln und diese Bedeutung zu nutzen. Andere Ansätze betrachten zusätzlich den Kontext des Wortes in Form von umgebenden Wörtern, wie etwa der Ansatz von Lesk [Les86]. In der Evaluation in [Mih07] kommt die Annotation des wahrscheinlichsten Wortes auf eine Genauigkeit von 72,58%, ein Ansatz nach Lesk auf 78.02%.

2.3.4. Extraktion von Themen

Die Extraktion von Themen aus Texten ist ebenfalls eine bekannte Problematik der Verarbeitung natürlicher Sprache, zu der bereits verschiedene Herangehensweisen existieren. Hierbei wird bei den bisherigen Ansätzen versucht, eine – oftmals homogene – Menge an Themen für ein Dokument zu finden. Hilfreich ist die Themenextraktion, um dadurch weiteres Kontextwissen nutzen zu können, aber auch die automatisierte Kategorisierung von Texten an sich kann ein praktischer Anwendungsfall sein. Die Aufgabe ist komplex, da das Dokument mit seiner Menge an Wörtern auf kurze, aber möglichst passende Beschreibungen als Thema abgebildet werden muss. Dabei kann das Thema im Dokument enthalten sein, es kann allerdings auch eine Beschreibungen passender sein, die sich nicht im Text befindet. Gerade dieser Fall drängt zum Einsatz von Informationsquellen außerhalb des Textes. Näheres zu Arbeiten im Bereich der Themenextraktion ist in Abschnitt 4.1 zu finden.

2.4. Graphen

Ein Graph ist eine abstrakte Datenstruktur, in der Punkte und die Relationen zwischen ihnen gespeichert werden. Graphen werden unter anderem von Diestel in [Die00] wie in Definition 2.4.1 definiert.

Definition 2.4.1. Ein Graph G besteht aus dem dem Paar G=(V,E), wobei $E \subseteq [V]^2$. Die Elemente aus E sind daher eine zweielementige Teilmenge von V und stellen die Kanten des Graphen dar. Die Elemente von V sind die Knoten des Graphen.

Es gibt gerichtete und ungerichtete Graphen. Bei einem gerichteten Graphen ist die Reihenfolge der Knoten in einer Kantendefinition entscheidend. Diese gibt die Richtung der Kante an. Zum Beispiel werden Zustandsautomaten häufig als gerichtete Graphen dargestellt; die Kanten geben dabei den möglichen Zustandsübergang an. Bei ungerichteten Graphen werden Kanten keine Richtung vorgegeben, sie sind bidirektional. Außerdem können sowohl bei ungerichteten als auch bei gerichteten Graphen die Kanten gewichtet sein. Dies wird häufig dazu genutzt, um der Verbindung zwischen zwei Knoten eine Länge oder einen Aufwand zur Benutzung festzulegen. Wird das Gewicht nicht explizit notiert, dann ist die Kante nicht gewichtet beziehungsweise wird vom Kantengewicht Eins ausgegangen.

Ein Graph wird üblicherweise mit Punkten für die Knoten und Linien zwischen den Punkten für die Kanten visualisiert. Gerichtete Kanten werden in Darstellungen häufig als Pfeile dargestellt. Ein beispielhafter ungerichteter Graph ist in Abbildung 2.5 abgebildet.

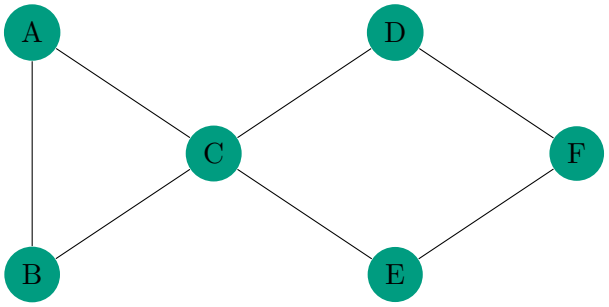

Abbildung 2.5.: Ungerichteter Beispielgraph mit 6 Knoten und 7 Kanten

In Graphen gibt es Pfade, manchmal auch Wege genannt. Ein Pfad ist eine Menge von paarweisen Knoten, die miteinander durch Kanten verbunden sind. Ein Beispiel für einen Pfad ist in der Abbildung 2.5 der Pfad A→B→C→D→F→E. Ein Zyklus liegt vor, wenn ein Knoten mehrfach an unterschiedlicher Stelle im Pfad vorkommt, dadurch also einen Kreis bildet. Zu sehen ist dies beispielsweise in Abbildung 2.5 bei dem Pfad C→D→F→E→C.

Einen Graph nennt man zusammenhängend, wenn es für jedes Knotenpaar zwischen diesen Knoten einen Weg gibt. Dies bedeutet, dass man eine einzige zusammenhängende Komponente im Graphen hat, nicht mehrere. Der Graph in Abbildung 2.5 ist zusammenhängend. Nicht zusammenhängend wäre er, wenn Knoten C entfernen werden würde. Dadurch würden die Knoten A und B nicht mehr mit dem restlichen Graphen zusammenhängen. Um einen nicht zusammenhängenden Graphen zu erhalten, könnte alternativ Knoten C behalten werden, die Kanten {A,C} und {B,C} jedoch entfernt werden. Dann würden A und B ebenfalls nicht mehr mit dem restlichen Graphen zusammenhängen.

In der Graphentheorie spricht man von benachbarten beziehungsweise adjazenten Knoten, wenn diese Knoten über eine Kante miteinander verbunden sind. In Abbildung 2.5 wären das zum Beispiel die Knoten A und B. Ebenso nennt man zwei Kanten benachbart, wenn diese einen gemeinsamen Knoten besitzen, so wie etwa die Kanten {A,C} und {D,C} aus dem Beispielgraphen. Einen Knoten und eine Kante werden außerdem inzident genannt, wenn die Kante den Knoten mit einem anderen Knoten verbindet.

Es gibt verschiedene Arten, wie diese abstrakte Datenstruktur gespeichert werden kann. Nach Cormen et al. in [CLRS01] stellt die sogenannte Adjazenzmatrix eine Möglichkeit dafür dar. Eine Adjazenzmatrix ist eine zweidimensionale Matrix, in der gespeichert wird, welche Knoten des Graphen miteinander verbunden sind. Für jeden Knoten existiert in der Matrix eine Zeile sowie eine Spalte. Ist in der Matrix der Eintrag an Stelle ij eine Zahl größer oder gleich Eins, dann existiert zwischen den Knoten i und j eine Kante mit

dem Kantengewicht gleich der eingetragenen Zahl. Adjazenzmatrizen werden neben der Speicherung des Graphen auch häufig in Algorithmen genutzt, da sie den Einsatz von Methoden der linearen Algebra erlaubt.

2.5. Ontologien

Ontologie ist ein Begriff, der seinen Ursprung in der Philosophie hat. Die philosophische Ontologie gehört zur allgemeinen Metaphysik und ist die Lehre vom Sein. Die Lehre handelt unter anderem von Erkenntnissen über Attribute und Wissen über die Dinge an sich [SS09][Hes02].

Die Auslegung des Begriffs Ontologie ist in der Informatik nicht unähnlich zur Philosophie. In der Informatik werden Ontologien als eine Art Datenbank genutzt, um Wissen zu speichern. Dies wird vor allem genutzt, um einen gewissen Gegenstandsbereich zu modellieren.

Eine Ontologie besitzt Konzepte (*concepts*), Klassen und Attribute, zwischen denen es Beziehungen (Relationen) und Regeln geben kann. Diese können auch vererbt werden, ähnlich wie in verschiedenen Programmiersprachen, wodurch eine Vererbungshierarchie entsteht.

Durch die erzeugte Struktur der Ontologien kann die Ontologie auch auf Konsistenz geprüft und sogar fehlendes Wissen durch maschinelle Lernverfahren ergänzen werden, indem durch logisches Folgern (*Inferenz*) Informationen erzeugt werden.

Eine der bekanntesten Ontologien ist die Weltwissenontologie *Cyc* mit ihrem wissenschaftlichen Ableger *ResearchCyc* [Cyc]. Das Projekt wurde 1984 mit dem Ziel gestartet, eine umfassende Wissensdatenbank zum Allgemein- und Weltwissen zu erstellen. Bis heute wird Cyc weiterentwickelt und verbessert.

Einsatzgebiete für Ontologien findet man beispielsweise in der Bioinformatik, in der Ontologien unter anderem dazu genutzt werden, genetische Daten zu speichern. Bei der Verarbeitung von natürlicher Sprache werden Ontologien ebenfalls eingesetzt. Dort werden sie zum Beispiel genutzt, damit Informationen über Semantik und Beziehungen zwischen Wörtern zur Verfügung gestellt werden können. Körner in [Kö14] und Landhäußer et al. in [LKT+15] haben beispielsweise *ResearchCyc* eingesetzt, um zu ermitteln, welche Informationen beziehungsweise Argumente eine Nominalisierung oder ein Verb benötigen, um vollständig spezifiziert zu sein.

In verschiedenen Fällen kann es notwendig sein, Ontologien zu verschmelzen. Das Verschmelzen von Ontologien stellt dabei eine besondere Herausforderung dar, denn Ontologien lassen sich selten trivial verschmelzen. Der Aufbau von Ontologien kann sich unterscheiden, weshalb erst ein Abgleich der Ontologien stattfinden muss. Dabei wird versucht, gleiche oder sehr ähnliche Konzepte über beispielsweise linguistische und strukturelle Ähnlichkeiten zu finden. Zum Problem der Ontologieverschmelzung (*Ontology Merging*) ähneln sich der Ontologieabgleich (*Ontology Matching*) und die Ontologieangleichung (*Ontology Alignment*) beziehungsweise die Ontologieabbildung(*Ontology Mapping*). Der Unterschied der Begriffe liegt nach Choi et al. in [CSH06] darin, dass bei der Ontologieverschmelzung aus zwei (oder mehr) Ontologien eine einzige Ontologie generiert werden soll. Beim Ontologieabgleich werden Abbildungen zwischen den Ontologien gesucht, über die man das Wissen der beiden Ontologien verbinden kann; die Ontologien werden allerdings weiterhin separat behandelt. Man kann dies daher als nötigen Schritt vor einer Verschmelzung sehen. Der Ontologieabgleich ist ein Prozess, dessen Ergebnis eine Ontologieangleichung ist. Zuletzt ist eine Ontologieabbildung eine gerichtete Abbildung einer Ontologie auf eine andere. Dies entspricht nach Euzenat und Shvaiko in [ES+07] einer mathematischen Abbildung.

2.6. Wikipedia

Die Wikipedia [Wik18a] ist eine freie Online-Enzyklopädie, die 2001 veröffentlicht wurde. Sie zählt zu den am meisten genutzten Enzyklopädien [LIJ+15]. Verfügbar ist sie in einer Vielzahl an Sprachen wie beispielsweise Englisch und Deutsch, teilweise sogar in Dialekten wie dem Alemannischen. Im Englischen besitzt die Wikipedia dabei aktuell knapp 5,5 Millionen Artikel. Die Urheber der Artikel sind angemeldete und nicht angemeldete Autoren. Dies hat den Vorteil, dass sich im Grunde genommen jeder an der Wikipedia beteiligen und somit Wissen hinzufügen oder korrigieren kann. Dadurch kann die Wikipedia immer weiter wachsen und einzelne Artikel werden immer exakter, da Fehler von einer Vielzahl an Menschen korrigiert werden können.

Jimmy Wales, Mitbegründer der Wikipedia, sprach in einem Interview [Wik10] vom Ziel „eine frei lizenzierte und qualitativ hochstehende Enzyklopädie zu schaffen“ und damit lexikalisches Wissen zu verbreiten. Nach Giles in [Gil05] war Wikipedia 2005 bereits ähnlich gut in Genauigkeit und Abdeckung wie die bekannte *Encyclopedia Britannica*.

Die Wikipedia ist über Artikel aufgebaut. Artikel behandeln jeweils ein bestimmtes Thema, wie etwa der Artikel „Archimedes“ Wissen über den griechischen Mathematiker, Physiker und Ingenieur Archimedes bereitstellt. Im Text der Artikel wurden zu bestimmten Wörtern Querverweise erstellt, die zu anderen Artikeln führen, wodurch der Nutzer auf diese Weise Unbekanntes schnell nachschlagen und sich weiteres Wissen aneignen kann.

Damit das Wissen von Wikipedia strukturiert eingesetzt werden kann, wurde das DBPedia-Projekt [LIJ+15] gestartet. Dafür wird die Wissensbasis aus Wikipedia extrahiert und in Wissensdatenbanken gespeichert. Eine bestimmte Form davon ist eine Ontologie. Dabei wird nicht nur das Wissen aus den Wikipedia-Artikeln aus einer bestimmten Sprache extrahiert, sondern versucht, das Wissen aus möglichst allen Sprachen zu kombinieren.

3. Programmieren in natürlicher Sprache mit PARSE

Das Projekt *PARSE* (*Programming ARchitecture for Spoken Explanations*) [Wei] befasst sich mit dem Programmieren in natürlicher Sprache. Das Ziel des Projekts ist es, Nutzern zu ermöglichen, Zielsystemen neue Handlungen basierend auf bereits bekannten Handlungen beizubringen. Dazu sollen Anweisungen eines Nutzers an das System in ausführbaren Quelltext umgewandelt werden. So kann beispielsweise dem Haushaltsroboter ARMAR-III [ARA+06], im folgenden ARMAR genannt, beigebracht werden, wie der Befehl „Bring mir den Orangensaft aus dem Kühlschrank“ umgesetzt werden soll. Der Nutzer erklärt dazu dem Roboter die einzelnen Zwischenschritte: Zuerst muss ARMAR zum Kühlschrank gehen, diesen öffnen, den Orangensaft identifizieren und greifen, den Kühlschrank schließen und schließlich zur Person gehen, um den Saft zu übergeben.

Verschiedene Verarbeitungsschritte müssen ausgeführt werden, um die Transformation der Eingabe in Quelltext zu ermöglichen. PARSE setzt auf einen agentenbasierten Ansatz, der in Abbildung 3.1 zu sehen ist. Die Eingabe wird anfangs durch Module vorverarbeitet, die die gesprochene Sprache erkennen und in Text umwandeln sowie eine erste Sprachverarbeitung durchführen, bei der unter anderem die Wortarten annotiert werden. Während dieses Prozesses wird die Eingabe umgewandelt und in einem Graphen gespeichert. Auf dem Gra-

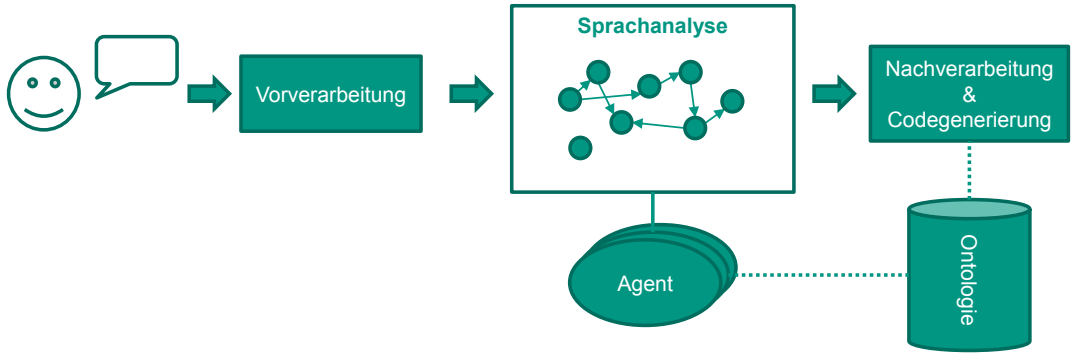

Abbildung 3.1.: Schematische Darstellung des agentenbasierten Ansatzes von PARSE [WT15]

phen können verschiedenen Agenten arbeiten, die für die Sprachanalyse zuständig sind und die Semantik analysieren sollen. Dafür lesen sie benötigte Informationen aus und überführen ihre Ergebnisse geeignet in den Graph. Nachdem die Eingabe von den verschiedenen Agenten verarbeitet und der Graph mit zusätzlichem Wissen angereichert wurde, werden Ontologien genutzt, um weitere benötigte Informationen zur Generierung des Quelltextes für das ausführende System zu erhalten. Abschließend sollen die gesammelten Informationen genutzt werden, um den Quelltext zu generieren, mit dem das Zielsystem, wie etwa der Haushaltsroboter ARMAR, die gewünschten Aktionen umsetzen kann.

Viele bisherige Ansätze im Bereich der natürlichsprachigen Verarbeitung setzen auf eine Fließbandarchitektur, bei der Eingaben von den einzelnen Verarbeitungsschritten in einer festgelegten Reihenfolge nacheinander verarbeitet werden. Dies ist der Hauptunterschied des agentenbasierten Ansatzes von PARSE. Der Vorteil des agentenbasierten Ansatzes mit einem Graphen als geteilten Datenspeicher ist die Möglichkeit der parallelen Ausführung der unterschiedlichen Verarbeitungsschritte. Beim agentenbasierten Ansatz müssen die einzelnen Agenten nicht auf den Abschluss anderer Verarbeitungsschritte warten, zu denen sie keine Abhängigkeiten haben. Gleichzeitig wird die erneute Ausführung von Agenten erlaubt, sollten andere Verarbeitungsschrittes neue Erkenntnisse liefern, die zu besseren Ergebnissen führen können.

Für PARSE gab es bereits verschiedene Arbeiten, die sich mit den unterschiedlichen Thematiken und Problemen der Programmierung in natürlicher Sprache beschäftigt haben und damit Funktionalitäten und Eigenschaften von PARSE mitgestalteten. Anfangs wurde im Rahmen einer Bachelorarbeit [Gü15] untersucht, wie Nutzer dem Haushaltsroboter ARMAR Anweisungen geben. Dafür wurde ein Sprachkorpus aus den Anweisungen der Probanden aufgebaut. Mit dem daraus resultierenden Wissen wurde in einer Bachelorarbeit [Sch15] versucht, eine geeignete Repräsentation der Eingabe mitsamt der zu annotierenden Informationen zu ermitteln. Das Ergebnis war die Speicherung der eingegebenen natürlichen Sprache sowie weiterer Informationen in einem Graph. Dabei können Knoten und Kanten mit verschiedenen Attributen angereichert werden, um die Informationen zu speichern.

Damit die gesprochenen Anweisungen in den Graph überführt werden können, wurden in einer Bachelorarbeit [Pas15] zuerst verschiedene automatische Spracherkennungssysteme evaluiert. Dabei wurde ein Analyse-Werkzeug erstellt, mit dem die gesprochene Eingabe mithilfe der automatischen Spracherkennung transkribiert und zu einem Korpus hinzugefügt werden kann. Danach wurde in einer weiteren Bachelorarbeit [Koc15] ein Werkzeug entwickelt, das die von der automatischen Spracherkennung erstellten Transkription auf den Graphen abbildet. Der Graph wurde außerdem mit ersten Informationen durch grundlegende Sprachverarbeitungsanalysen, wie der Annotation von Wortarten, ergänzt. So wird die textuelle Eingabe wortweise in speziellen Knoten gespeichert, indem über die Attribute der Knoten unter anderem das Wort der Eingabe, die Grundform des Wortes sowie die Wortart innerhalb des Satzes gespeichert sind. Gleichzeitig zeigen bestimmte gerichtete Kanten die Reihenfolge der Knoten an. Ein Ergebnis der Arbeit war jedoch, dass übliche Sprachverarbeitungswerkzeuge in diesem Kontext eher schlecht abschneiden. Deshalb wurde daraufhin die Verbesserung der Ergebnisse der automatischen Spracherkennung [Sch16] als auch der Wortart-Markierung [Kie16] angegangen.

In den darauf folgenden Arbeiten konnten erste Agenten entwickelt werden. So wurde in einer Bachelorarbeit [Ste16] ein Agent entwickelt, um Kontrollstrukturen in den Anweisungen zu erkennen und angemessen verarbeiten zu können. Desweiteren wurden Agenten zur Kontext- und Korreferenzanalyse erstellt [Hey16] sowie die Erkennung von Aktionen [Ou16] angegangen. Außerdem wurde in einer Masterarbeit [Sch17] ein Dialogagent entwickelt, der bei fehlerhafter Verarbeitung der Eingaben beim Nutzer nachfragen soll, um

Fehler korrigieren zu können.

Bei PARSE werden gesprochene Anweisungen verarbeitet, was zu verschiedenen Herausforderungen und Problemen führen kann. Kennzeichnend für gesprochene Sprache sind die strukturellen Unterschiede zur geschriebenen Sprache. Die Grammatik in gesprochenen Sätzen ist häufig fehlerhaft und die Syntax unvollständig. Gesprochene Sprache ist in der Regel spontan, wodurch diese und weitere Fehler begünstigt werden. Wiederholungen von Wörtern oder ganzen Satzteilen sind bezeichnend dafür. Beispielsweise entstehen Stotterer oder Wiederholungen, um Fehler zu korrigieren oder um den Gedankengang neu zu ordnen. Ein Merkmal der gesprochenen Sprache sind auch Füllwörter wie etwa „ähm“ beziehungsweise „ehm“, die ein Zögern oder eine Gedankenpause überdecken sollen. Diese können bei der maschinellen Verarbeitung zu Problemen führen, wenn Wiederholungen oder Füllwörter auftreten, diese aber vom Verarbeitungssystem nicht erwartet werden. Beispielsweise bei der Annotation von Wortarten wird in der Regel auch die Umgebung des Wortes betrachtet. Stehen dort durch Füllwörter oder Wiederholungen Wörter, die vom System nicht erwartet wurden, können fehlerhafte Klassifikationen der Wortarten und verfälschte Ergebnisse entstehen.

In PARSE werden die gesprochenen Anweisungen mit der Hilfe von Systemen zur automatischen Spracherkennung in Textform transformiert. Dies ermöglicht den Einsatz von Methoden aus der Verarbeitung von geschriebenen Texten, dennoch existieren neben den bereits genannten Unzulänglichkeiten wie grammatischen Fehlern oder Füllwörtern dadurch weitere Unterschiede. Ein Eingabetext in der Form, wie er verarbeitet werden muss, ist in folgendem Beispiel zu sehen:

Beispiel

okay go to the kitchen table ehm get the popcorn and bring it to me

In gesprochener Sprache existieren keine Satzzeichen, wodurch nach der Transformation in die Textform keine Satzzeichen vorkommen. Ohne diese Gliederung der Texte sind die Ergebnisse von Sprachverarbeitungswerkzeugen in der Regel ungenauer. Dies liegt vor allem daran, dass Sprachverarbeitungswerkzeuge lediglich auf geschriebene Texte mit einer Satzgliederung ausgelegt und trainiert sind. Die verschlechterte Güte der Werkzeuge beim Einsatz auf gesprochener Sprache wurde auch unter anderem in [Koc15] festgestellt. Dort wurden Werkzeuge zur Wortartenerkennung auf Texten mit und ohne Satzzeichen getestet, wobei die Genauigkeit bei den Texten ohne Satzzeichen um bis zu 5% sank. Deshalb sind klassische Ansätze aus dem Bereich der textuellen natürlichsprachigen Verarbeitung nicht ohne weiteres einsetzbar. Vor allem Ansätze, die auf Ergebnisse aus Verarbeitungsschritten wie der Annotation von Wortarten aufbauen, können durch falsche Ergebnisse stark negativ beeinflusst werden. Daher muss darauf geachtet werden, dass neben den Unzulänglichkeiten der natürlichen Sprache auch Fehler in Vorverarbeitungsschritten vorliegen können. Deshalb sollten Ansätze möglichst robust gegenüber diesen Problemen sein.

4. Verwandte Arbeiten

In dieser Arbeit sollen aus Eingaben, die aus Anweisungen für die Programmierung in natürlicher Sprache bestehen, Themen extrahiert werden. Unter anderem Magatti et al. schreiben in ihrer Arbeit [MCCS09], dass die Extraktion von Themen durch den Menschen langsam ist und es regelmäßig vorkommen kann, dass sich nicht auf ein bestimmtes Thema geeinigt werden kann. Zudem sind die Themen auch häufig inkonsistent, da nach der Meinung von Mei et al. [MSZ] diese Themen subjektiv gewählt werden. Dadurch fließt eine starke Voreingenommenheit ein. Automatische Verfahren sind objektiver und in der Regel schneller als der Mensch. Daher sind automatische Verfahren gerade für größere Textsammlungen essentiell. Außerdem sind automatische Verfahren notwendig, wenn die Themenextraktion als Zwischenschritt in einem System eingesetzt werden soll. In Abschnitt 4.1 wird daher auf solche Verfahren eingegangen.

Es gibt verschiedene Anwendungsfälle, in denen es nötig ist, das Wissen aus Ontologien zusammenzunehmen. Dafür müssen die Ontologien verschmolzen werden. Ein beispielhafter Anwendungsfall ist die Kombination von Daten verschiedener Anbieter für Suchanfragen. Ein anderer Anwendungsfall ist die Kombination der Ontologie einer Bibliothek mit der eines Buchversandhauses, um die Informationen aus beiden Ontologien zu kombinieren. Beide haben dabei grundsätzlich ähnliche Konzepte, unterschiedliche Strukturen sind jedoch nicht unwahrscheinlich. Einfache Ansätze versuchen dabei lediglich gleichlautende Konzepte aufeinander abzubilden, während komplexere Ansätze auch Semantik und andere Strukturen der Ontologien für die Abbildung berücksichtigen. In dieser Arbeit sollen Themen extrahiert und danach passende Ontologien zu den Themen zugeordnet werden. Um das Wissen aus diesen Ontologien in Kombination nutzen zu können, sollen diese themenspezifischen Ontologien verschmolzen werden. Einige der komplexeren Ansätze zur Verschmelzung von Ontologien werden daher in Abschnitt 4.2 vorgestellt.

4.1. Themenextraktion

Es gibt verschiedene Ansätze zur Themenextraktion, wovon jedoch viele die *Latent Dirichlet Allocation (LDA)* von Blei et al. aus [BNJ03] nutzen. Die Autoren entwickelten ein generatives probabilistisches Modell für Sammlungen von diskreten Daten, wie etwa Textsammlungen. Damit können Beobachtungen gemacht werden, warum sich bestimmte Daten ähneln. Mit dem LDA-Verfahren lassen sich Modelle für Themen erstellen, in denen die Wahrscheinlichkeiten enthalten sind, dass die verschiedenen Themen ein Dokument

beschreiben. Das LDA-Verfahren kann somit automatisch Themen in Dokumenten entdecken. Viele Verfahren nutzen die Themenmodelle aus der LDA als Grundlage, um die resultierenden Themen oder Themenwahrscheinlichkeiten dann weiterzuverarbeiten.

Hingmire und Palshikar nutzen für ihre Arbeit [HCPC13] das LDA-Verfahren als Grundlage, um einen Klassifikator trainieren zu können. So werden einige Dokumente mit einen LDA-basierten Algorithmus verarbeitet und die Themenmodelle ermittelt. Diese werden daraufhin von Experten zu einem Themenbezeichner zugeordnet. Danach wird ein sogenannter *Expectation-Maximization*-Algorithmus angewandt und ein Naive-Bayes Klassifikator trainiert. Dabei wird der Klassifikator zuerst mit den Anfangsdokumenten und den Themenbezeichnern der Experten trainiert. Dieser Klassifikator wird danach auf Dokumente angewandt, die noch keinen Themenbezeichner zugewiesen bekommen haben. Daraus können die Wahrscheinlichkeiten für das Auftreten jedes Themenbezeichners ermittelt werden. Mit diesen Informationen wird daraufhin der Naive-Bayes Klassifikator aktualisiert. Dies wird solange wiederholt, bis sich der Klassifikator nicht mehr wesentlich ändert. Mit dem so trainierten Klassifikator können daraufhin weitere Dokumente mit Themenbezeichnern versehen werden. Vorteil dieses Ansatzes ist, dass man so nur noch wenige vorher annotierte Dokumente braucht. In der Evaluation werden in verschiedenen Durchläufen Dokumente aus zwei bis vier verschiedenen Datensets – und dadurch Bezeichner – gewählt und klassifiziert. Aus den Ergebnissen wird jeweils das F_1-Maß ermittelt. Bei den Durchläufen mit lediglich zwei bis drei verschiedenen Datensets werden Werte von 0.92 bis 0.986 erreicht, bei vier Datensets fällt dieser Wert auf 0.73 bis 0.78.

Mei et al. haben in [MSZ] bereits einige Jahre zuvor angemerkt, dass manuelle Bezeichner zu subjektiv sind. Außerdem missfällt ihnen, dass man Experten über die jeweiligen Themen benötigt, um korrekt klassifizieren zu können. Deshalb schlagen sie einen automatischen Ansatz vor, um Themenmodelle zu Themenbezeichnern zuzuweisen. Für die Generierung der Themenmodelle können sowohl das LDA-Verfahren als auch andere Ansätze gewählt werden. Die Themenmodelle werden jeweils weiterverarbeitet, wobei die Bedeutung des Themenmodells interpretiert werden soll. Dafür nutzen sie mehrere Maße, um die semantische Relevanz der Themen an sich und in Kombination zu ermitteln. Mit diesem Maß kann dann der beste Bezeichner ermittelt werden. In der Evaluation sollten Personen die Themen aus diesem Ansatz mit anderen Systemen, wie die manuelle Annotation, vergleichen. Die Evaluation zeigt, dass die so ermittelten Themen oft gut sind und ähnlich zu den Bezeichnern, die von Experten annotiert wurden. Im Vergleich zu einem einfachen Ansatz, bei dem lediglich die wahrscheinlichsten Bezeichner aus dem Themenmodell genommen werden, liefert dieses Verfahren auf Grundlage der Bewertung der Testpersonen bessere Ergebnisse.

Ansätze wie dieser zählen zu den textbasierten Ansätzen, die lediglich Informationen aus dem Text nutzen. Reine textbasierte Ansätze weisen jedoch einige Probleme auf. So merken etwa Medelyan in [OID] und Hulpus et al. in [HHKG13] jeweils in ihren Arbeiten an, dass textbasierte Ansätze zur Extraktion von Themen zu limitiert sind, da sie nur Themen vorschlagen können, die auch tatsächlich im Text erwähnt werden. Ebenso sieht Medelyan in textbasierten Arbeiten das Problem von Rechtschreibfehlern oder Synonymen, die das Endergebnis sehr stark beeinflussen können. Daher gibt es im Bereich der Themenextraktion immer mehr Ansätze, die auf das Wissen aus externen Quellen wie Wissensdatenbanken setzen.

Magatti et al. nutzen in ihrer Arbeit [MCCS09] die Themenextraktion über das *Latent Dirichlet Allocation* Modell in Verbindung mit Baumhierarchien. An bisherigen Ansätzen kritisieren die Autoren, dass Themen nicht richtig interpretiert werden und ihre Bedeutung nicht erfasst wird. Die Hauptaufgabe sehen die Autoren darin, Themen so präzise zu interpretieren, dass die Bedeutung jedes Themas erfasst werden kann. Dies versuchen

sie mit Hilfe der Baumhierarchien zu lösen. Aus Quellen wie etwa dem *Google Directory Service* und dem *OpenOffice English Thesaurus* werden alle Konzepte entnommen und als Knoten genutzt. Die Konzepte werden danach in Ist-Ein-Beziehungen zueinander gesetzt, um die Baumhierarchie herzustellen. So ist beispielsweise das Konzept „Katze“ eine Unterkonzept des Konzepts „Katzenartige“. Zu diesen Konzepten, die später als gewählte Themenbezeichner genutzt werden sollen, werden weitere Informationen und eine Liste an Wörtern, die im Thema vorkommen, gespeichert. Auf die zu annotierenden Dokumente wird nun die *LDA*-Methode angewandt. Die daraus resultierende Wortliste an Themen wird über ein Ähnlichkeitsmaß mit den Knoten im Baum verglichen und jedes Thema wird dem ähnlichsten Knoten zugewiesen. Im Anschluss ermittelt ein Algorithmus den Konsens der gefunden Knoten im Baum. Dabei wird geprüft, ob die Knoten beispielsweise im selben Unterbaum oder im selben Pfad sind oder ob es einen gemeinsamen Knoten gibt, der die Konzepte verbindet. Die Evaluation aus [MCCS09] sieht den Test von 50 Themen vor. Nach Aussage der Autoren sind die Ergebnisse vielversprechend. Dabei werden jedoch keine exakten Daten darüber genannt, ob und wie die Themen passend sind. Hier fehlt ein aussagekräftiger Gütewert, der die Qualität dieses Ansatzes bewertet und vergleichbar mit anderen Ansätzen macht.

Eine externe Wissensquelle nutzen Medelyan et al. in ihrer Arbeit [OID]. Sie nutzen Wikipedia als Datenquelle beziehungsweise als sogenanntes *kontrolliertes Vokabular*. Ein kontrolliertes Vokabular bringt Vorteile gegenüber reinen textbasierten Ansätzen, die lediglich auf Themen aus dem Text beschränkt sind, da Themenbegriffe so nur durch das Vokabular beschränkt sind. Die Autoren setzen auf einen Ansatz in zwei Schritten, dem Ermitteln von möglichen Kandidaten und das Herausfiltern der vielversprechendsten unter diesen Kandidaten. Zuerst werden daher mögliche Kandidaten für Themen aus dem Text ermittelt, indem N-Gramme aus dem Dokument extrahiert werden. Ein N-Gramm ist eine Zerlegung des Textes in Fragmente, hier Wörter, wobei immer n viele Fragmente zusammengenommen werden. Die extrahierten N-Gramme werden über einen errechneten *Keyphraseness*-Wert daraufhin untersucht, mit welcher Wahrscheinlichkeit sie jeweils ein Kandidat für ein Thema sind. Übertrifft der *Keyphraseness*-Wert einen festgelegten Schwellwert, dann werden die passenden N-Gramme zu Begriffen aus dem kontrollierten Vokabular zugewiesen. Da das kontrollierte Vokabular hier Wikipedia als Grundlage nutzt, werden somit die N-Gramme zu Wikipedia-Artikeln zugewiesen. Sollte mehr als ein Artikel in Frage kommen, muss die Mehrdeutigkeit des N-Gramms aufgelöst werden. Die eindeutigen N-Gramme werden dabei genutzt, um die mehrdeutigen N-Gramme aufzulösen. Dazu wird jeweils ein Ähnlichkeitswert zwischen den verschiedenen Bedeutungen des mehrdeutigen N-Gramms und den eindeutigen N-Grammen ermittelt. Dieser Ähnlichkeitswert wird dann mit der Auftrittswahrscheinlichkeit der Bedeutung gewichtet. Der höchste ermittelte Wert bestimmt, zu welcher Bedeutung ein N-Gramm zugeordnet wird. Nach der Zuordnung werden verschiedene statistische und semantische Eigenschaften der Kandidaten ermittelt. Diese werden mit Hilfe eines Naive-Bayes Klassifikators interpretiert und zu den endgültigen Themen zugewiesen. Der Naive-Bayes Klassifikator wurde zuvor mit Hilfe von manuell annotierten Dokumenten trainiert. Die Evaluation des Ansatzes ergab, dass die Ergebnisse vergleichbar mit menschlichen Ergebnissen sind.

In [HHKG13] schreiben Hulpus et al. neben den bereits genannten Nachteilen textbasierter Ansätze, dass diese nicht auf die Strukturen und ähnliche Informationen hinter den Konzepten achten. Deshalb setzen Hulpus et al. auf einen graphbasierten Ansatz, dessen Graphen auf allgemeinen Wissensdatenbanken wie Wikipedia beziehungsweise DBpedia basieren. Der graphbasierte Ansatz hat den Vorteil, dass hier anerkannte Algorithmen zur Ermittlung einer Zentralität der Knoten im Graph eingesetzt werden können. Diese sind nach Ansicht der Autoren besser geeignet, als die einfache Ermittlung eines Konsens, wie es bei Magatti et al. [MCCS09] vorkommt, da dieses Verfahren häufig zu generel-

le Themen liefert. Der Ansatz bedarf allerdings einer Vorverarbeitung, die entsprechend gute Ergebnisse liefern muss. Zuerst findet eine Extraktion von Schlüsselbegriffe mit Hilfe des LDA-Verfahrens statt. Daraufhin werden die Mehrdeutigkeiten der Schlüsselbegriffe aufgelöst, wobei ein Ansatz gewählt wird, der Konzepte aus DBpedia liefert. Mit diesen ermittelten Konzepten, die die jeweiligen Bedeutungen der Schlüsselbegriffe darstellen, wird dann in zwei Schritten ein Graph erstellt. Im ersten Schritt zu Erstellung des Graphen wird für jedes Konzept jeweils ein eigener Graph erstellt, der von den Autoren *Bedeutungsgraph* (*Sense Graph*) genannt wird. Der *Bedeutungsgraph* besteht dabei aus dem Ausgangskonzept und allen weiteren Konzepten, die über zwei Verbindungen in DBpedia mit dem Ausgangskonzept verbunden sind. Im nächsten Schritt werden alle diese Graphen zu einem von den Autoren *Themengraph* (*Topic Graph*) genannten Graphen verschmolzen. Ausgangskonzepte, die nicht mit der Hauptkomponente des Graphen verbunden sind, werden als Rauschen in den Daten interpretiert und entsprechend aus dem Graphen entfernt. Auf den Themengraphen können dann verschiedene Algorithmen zur Ermittlung der Zentralität angewandt werden, womit die wahrscheinlichsten Themen ermittelt werden können. Im Ansatz beschreiben Hulpus et al. die Algorithmen *Closeness Centrality*, *Betweenness Centrality*, *Information Centrality* und *Random Walk Betweenness Centrality*. Bei der *Closeness Centrality* ist ein Knoten wichtiger, je näher er zu allen anderen Knoten im Graphen liegt. Die *Betweenness Centrality* bewertet einen Knoten höher, wenn er den Informationsfluss zwischen Knoten fördert. Dies geschieht, wenn viele der kürzesten Pfade zwischen anderen Knoten des Graphen durch diesen Knoten verlaufen. Die *Information Centrality* ist ähnlich definiert und bewertet die Information aller Pfade durch den Knoten. Die Information eines Pfades ist dabei invers proportional zu seiner Pfadlänge. Zuletzt misst die *Random Walk Betweenness Centrality* wie oft ein Knoten durch einen Wanderer besucht wird, wenn dieser zufällige Pfade im Graphen wandern würde. Hierbei entschieden sich Hulpus et al. dafür, die Algorithmen ein wenig anzupassen, damit die Ausgangskonzepte eine entsprechend gewichtigere Rolle spielen. Ihre Arbeit stützen Hulpus et al. auf eine Nutzerstudie, in der die Themen, die mit obiger Variante erstellt wurden, von verschiedenen Personen bewertet wurden. Die Ergebnisse wurden daraufhin mit Ergebnissen des textbasierten Ansatzes aus [MSZ] verglichen, wobei der Ansatz von Hulpus et al. besser als diese abschnitt. Die Autoren stellen fest, dass graphbasierte Ansätze eine bessere Auswahl an Themen ergeben, gleichzeitig aber auch weniger sensitiv gegenüber der Art des Textes sind. Jedoch ergab die Evaluation, dass häufig etwas zu allgemeine Themen vorgeschlagen werden. Offen bleibt, wie dieser Ansatz im Vergleich zu weiteren Ansätzen abschneidet. Der Ansatz liefert laut Evaluation oft zu allgemeine Themen, was die Autoren bei anderen Ansätzen kritisiert haben. Großer Vorteil dieses Ansatzes ist jedoch, dass er unüberwacht abläuft und daher keine annotierten Trainingsdaten braucht.

Einen ähnlichen Ansatz wie Hulpus et al. nutzen Coursey et al. in ihrer Arbeit [KRW]. In dieser Arbeit wird ebenfalls ein unüberwachter, graphbasierter Ansatz mit einem Zentralitätsmaß genutzt. Allerdings unterscheiden sich die Arbeiten darin, wie die Graphen erstellt werden und welcher Algorithmus zum Berechnen des Zentralitätsmaßes genutzt wird. In der Arbeit von Coursey et al. wird ein enzyklopädischer Graph aus der gesamten Wikipedia aufgebaut. Knoten stellen dabei die Artikel und Kategorien aus der Wikipedia dar, Kanten die Relation zwischen ihnen. Für den Aufbau des Graphen wird von einem Ausgangsartikel ausgegangen, Querverweise im Artikel sowie die Kategorien des Artikels stellen Knoten dar, die mit Kanten zum Ausgangsknoten verbunden werden. Auf diese Weise kann rekursiv der enzyklopädische Graph aufgebaut werden. Dieser Graph muss nur einmal erstellt werden und kann anschließend gespeichert werden. Auf den Text wird dann das *Wikify!*-Werkzeug eingesetzt, das in [RA07] näher beschrieben ist. Das Werkzeug sucht Konzepte im Text, bei denen angenommen wird, dass sie relevant sind, und verlinkt diese zur Wikipedia. Die so gefundenen Konzepte werden im Wikipedia-Graphen markiert und eine angepasste Version des PageRank-Algorithmus von Page et al. [PBMW99] so-

wie des personalisierten PageRank-Algorithmus von Haveliwala [Hav02] angewandt. Dabei wird für jeden Knoten eine Wahrscheinlichkeit errechnet. Diese sagt aus, wie wahrscheinlich es ist, dass dieser Knoten besucht wird, wobei die gefundenen Konzepte aus dem Text stärker gewichtet werden. In Gleichung 4.1 sind die Berechnung für den PageRank-Wert nach Brin und Page aus [PBMW99].

$$S(V_i) = (1-d) + d * \sum_{j \in \text{ In}(V_i)} \frac{1}{|\text{Out}(V_i)|} * S(V_j) \tag{4.1}$$

$\text{In}(V_i)$ steht dabei für die Menge an Knoten, die auf den Knoten i mit einer gerichteten Kante zeigen. Entsprechend steht $\text{Out}(V_i)$ für die Menge an Knoten, auf die der Knoten i zeigt. Bei ungerichteten Graphen gilt, dass $\text{In}(V_i)$ und $\text{Out}(V_i)$ gleich sind. Der angepasste PageRank-Algorithmus ändert sich insofern, dass sich der erste Summand ändert und eine Beeinflussung durch die Ausgangsbedeutungen der Bedeutungsgraphen erhält. Die Formel dafür ist in Gleichung 4.2 zu sehen.

$$S(V_i) = (1-d) * \text{Bias}(V_i) + d * \sum_{j \in \text{ In}(V_i)} \frac{1}{|\text{Out}(V_i)|} * S(V_j) \tag{4.2}$$

Die Formel für die Beeinflussung lautet

$$\text{Bias}(V_i) = \frac{f(V_i)}{\sum_{j \in \text{ InitialNodeSet}} f(V_j)} \tag{4.3}$$

$f(V_i)$ kann hier nach Aussage der Autoren in [KRW] verschiedene Formen annehmen. Ein Beispiel hierfür ist, dass $f(V_i)$ für Knoten aus der Menge der Ausgangsbedeutungen 1 zurückgibt, ansonsten 0. Dementsprechend sollen hier die Ausgangsbedeutungen etwas stärker gewichtet werden. Da, wie in Gleichung 4.2 ersichtlich, der Wert für einen Knoten auch von den Werten der benachbarten Knoten abhängt, werden Knoten, die mit einem Ausgangsknoten verbunden sind, größere Werte erhalten. In [KRW] entschieden sich die Autoren, bei der Berechnung von $f(V_i)$ unter anderem Werte aus dem LDA-Verfahren zu nutzen. Die Knoten mit den größten Werten entsprechen schließlich den möglichen Themen. In drei Experimenten zeigten Coursey et al., dass ihr Ansatz vielversprechend ist. Zuerst testeten sie ihren Ansatz ohne den Part mit *Wikify!*, um keine von *Wikify!* abhängigen Fehler zu messen. Im zweiten Test untersuchten sie den Ansatz auf 150 Artikeln und nutzen den Ansatz zusammen mit *Wikify!*. Im letzten Test wendeten sie den Ansatz auf Artikel speziell aus dem Themenfeld der Informatik an. Die Ergebnisse sind ähnlich gut, teilweise sogar besser, als die der menschlichen Teams, da diese manchmal ein oder mehrere geeignete Themen gar nicht in Betracht gezogen hatten.

4.2. Verschmelzen von Ontologien

Das Verschmelzen von Ontologien ist wichtig, wenn man die Informationen aus verschiedenen Ontologien kombinieren möchte. In dieser Arbeit soll als Anwendungsfall die Zuordnung von Eingaben zu Themen stattfinden, wonach die Themen zu passenden themenspezifischen Ontologien zugeordnet werden sollen. Um das Wissen aus den themenspezifischen Ontologien verbinden zu können, ist eine Verschmelzung der Ontologien entsprechend notwendig.

Um eine Abbildung von Konzepten zweier Ontologien aufeinander zu finden, gibt es verschiedene Strategien, grundlegende Ansätze sind jedoch oftmals ähnlich. Stets wird versucht, sowohl terminologische als auch semantische und strukturelle Ähnlichkeiten zu finden, um eine Abbildung erzeugen zu können. Unterschiedlich sind aber die spezifischen Methoden zum Ermitteln der Ähnlichkeiten sowie die Skalierbarkeit der Ansätze.

Hu und Qu setzen dabei in ihrem System *Falcon* [HQ08] auf einen Teile-und-Herrsche-Ansatz. In der ersten Phase wird dafür die Ontologie strukturbasiert in Blöcke aufgeteilt, indem strukturelle Ähnlichkeiten zwischen Domänenentitäten rekursiv ermittelt werden. Danach werden in der zweiten Phase die Blöcke der verschiedenen Ontologien anhand von sogenannten *Ankern* gruppiert. Ein Anker ist dabei ein Paar von übereinstimmenden Entitäten aus jeweils einer der beiden Ontologien, die zuvor über linguistische und andere Analysen ermittelt wurden. Je mehr Anker zwischen zwei Blöcken existieren, desto ähnlicher sind sich diese Blöcke. Die Blockpaare werden anhand eines Schwellwertes für diese Ähnlichkeit bestimmt. Im letzten Schritt werden zusätzliche Verbindungen zwischen den Blöcken über weitere linguistische und strukturelle Techniken ermittelt. Über eine Greedy-Auswahl wird anhand der besten Blockpaaren die Ontologieausrichtung erstellt. Durch den lokalen Ansatz durch Unterteilung in Blöcke und die gierige Auswahl können auch große Ontologien effizient miteinander verglichen werden.

Einen sehr ähnlichen Ansatz nutzen Seddiqui und Aono in ihrer Arbeit [SA09]. Auch sie nutzen Anker, um Gemeinsamkeiten aus den Ontologien zu ermitteln. Ein Anker ist in ihrer Arbeit ebenfalls ein Paar aus ähnlichen Konzepten aus jeweils einer Ontologie. Nach dem Ermitteln der Anker werden nach und nach benachbarte Konzepte analysiert und es wird versucht, diese zu verbinden. Deshalb nennen sie ihren Ansatz entsprechend *Anchorflood*. Die Autoren vertreten hier die These, dass ähnliche Konzepte in einer Ontologie adjazent oder zumindest nahe beieinander sind. Dadurch werden ähnliche Konzepte anhand einer lokalen Suche gefunden, anstatt jeweils die ganze Ontologie zu durchsuchen. Durch die lokale Suche ist der Ansatz zudem gut skalierbar.

Das Werkzeug *RiMOM*, das von Li et al. in [LTLL09] vorgestellt wird, verfolgt einen etwas anderen Ansatz. Das Werkzeug führt verschiedene Messungen für die paarweise Ähnlichkeit der jeweiligen Ontologieentitäten durch. Beispielsweise werden verschiedene linguistische Ähnlichkeitsmaße eingesetzt. Darunter zählen die Wortdistanz der Bezeichner der Entitäten ebenso wie eine Vektordistanz. Für die Vektordistanz wird für die Bezeichner jeweils ein Vektor aufgebaut. In dem Vektor befinden sich Kontext- und Metainformationen. Die Informationen werden dafür anhand der Häufigkeit des Auftretens sowohl im Kontext als auch allgemein in Zahlenwerte transformiert. Die Vektoren werden dann mit der Kosinus-Ähnlichkeit verglichen, bei dem der Kosinus des Winkels zwischen den Vektoren bestimmt wird. Die Ähnlichkeitsmaße werden gewichtet kombiniert und strukturell propagiert, unter anderem an die Nachbarn. Letztendlich können auf diese Weise ähnliche Konzepte ermittelt und zur Ergebnisabbildung hinzugefügt werden. Die Autoren gehen jedoch nicht darauf ein, wie gut der Ansatz skaliert. Gerade bei großen Ontologien müssen viele Konzepte miteinander verglichen werden, was die Laufzeit entsprechend erhöhen könnte.

Der Ansatz von Cruz et al. in [CAS09] ähnelt stark dem *RiMOM*-Ansatz von Li et al. Das *Agreementmaker* genannte Werkzeug besitzt ebenfalls verschiedene Vergleichsalgorithmen, um Konzepte und ihre Merkmale miteinander zu vergleichen. Dabei ist die Verarbeitung in drei Schichten unterteilt. Die erste Schicht vergleicht die Konzepte syntaktisch und lexikalisch über verschiedene Metriken, wie etwa der Übereinstimmung der Zeichenketten oder der Länge der Bezeichner. In der zweiten Schicht werden die strukturellen Eigenschaften der Ontologien, wie beispielsweise Vererbungshierarchien, verglichen. Zuletzt werden in der dritten Schicht die Ergebnisse aus den anderen beiden Schichten kombiniert. Dafür werden die Ergebnisse zuerst pro Konzeptpaar in einer Ähnlichkeitsmatrix gesammelt. Die Einträge werden dann gewichtet. Die Gewichtung kann entweder manuell oder automatisch durch verschiedene Evaluationsmethoden bestimmt werden. Aus den gewichteten Einträgen wird letztendlich ein Ähnlichkeitswert ermittelt. Ist dieser über einem Schwellwert, dann wird das Konzeptpaar zur Ergebnisabbildung hinzugefügt. Die Unterschiede zum *RiMOM*-Ansatz von Li et al. in [LTLL09] liegen vor allem in den eingesetzten Methoden zur Ermittlung der Ähnlichkeitsmaße.

Die *Ontology Alignment Evaluation Initiative (OAEI)* [OAE] hat sich als Ziel gesetzt, Stärken und Schwächen der verschiedenen Systeme mit ihren Techniken herauszuarbeiten. Dafür werden jährliche Tests für eine vergleichbare Evaluation der Ansätze veröffentlicht. Unter anderem wurden die vier zuvor beschriebenen Werkzeuge Falcon, Anchorflood, RiMOM und Agreementmaker mit den Tests der OAEI evaluiert. Die Evaluation des OAEI ist in mehrere Kategorien aufgeteilt. Die erste Kategorie besteht aus allgemeinen *Richtwert*-Testfällen, die durchschnittliche Tests darstellen. Sie wurden generiert und sollen einen stabilen Vergleich ermöglichen. Die Testfälle der *Verzeichnis*-Kategorie sollen besondere Herausforderungen darstellen. Die Verzeichnisse sind unter anderem aus Google und Yahoo erstellt und besitzen oft nur eine vage Terminologie und auch terminologische Fehler. Zuletzt gibt es noch die *Anatomie*-Kategorie, in der real existierende Ontologien aus dem biomedizinischen Bereich verarbeitet werden sollen. Shvaiko und Euzenat haben in [PJ13] die Ergebnisse zwischen 2004 und 2010 aufgearbeitet und miteinander verglichen. Die Ergebnisse ergeben, dass alle Ansätze vergleichbar gut abschneiden. Ebenso zeigt sich, dass die Ergebnisse über die Jahre besser werden. Dabei beeinflussen bessere Ergebnisse in einer Kategorie kaum bis gar nicht die Ergebnisse in einer anderen Kategorie, was für die Stabilität der Ansätze spricht. In den *Richtwert*-Testfällen sind Präzision, Trefferquote und das F-Maß jeweils sehr gut mit Werten von 90% und höher. Die Ergebnisse aus der *Anatomie*-Kategorie sind ähnlich und häufig nur wenige Prozentpunkte schlechter als in den *Richtwert*-Testfällen. So ist im Jahr 2010 das F-Maß für den *Agreementmaker* mit 88% lediglich um einen Prozentpunkt schlechter. *RiMOM* sticht im Jahr 2009 heraus, da die Ergebniswerte in der *Anatomie*-Kategorie mit 79% im Vergleich zu 91% in den *Richtwert*-Testfällen auffallend schlechter sind. In der *Verzeichnis*-Kategorie sind die Ergebnisse aller Systeme schlechter und variieren stark. Das F-Maß liegt in dieser Kategorie für die verschiedenen Systeme im Bereich von 37% bis 63%.

Abseits des OAEI unterteilen Choi et al. in der Studie [CSH06] die Systeme für Ontologieabbildungen in drei Kategorien. In die ersten Kategorie fallen Ansätze für Ontologieabbildungen zwischen integrierten globalen und lokalen Ontologien. Dabei wird die Abbildung genutzt, um ein Konzept aus der einen Ontologie zu transformieren und anschließend damit eine Anfrage an die andere Ontologie zu stellen. Die Vorteile liegen darin, dass es laut Choi et al. leichter ist, Abbildungen und Abbildungsregeln zu finden, da die integrierte globale Ontologie ein gemeinsames Vokabular bereitstellt und die lokalen Ontologien ähnlich dazu sind. Als Beispiel nennen sie *MOMIS* [BBGV03] von Beneventano et al., das nach der Vorverarbeitung eine globale virtuelle Sicht auf die verschiedenen Informationsquellen anbietet. In die zweiten Kategorie fallen Systeme für Ontologieabbildungen zwischen lokalen Ontologien. Der Vorteil von Systemen dieser Kategorie liegt darin, dass die Abbildungen kontextualisiert werden können, da sie nur lokal gespeichert sind. Die Abbildungen ermöglichen Kompatibilität zwischen lokalen Ontologien, die nicht verschmolzen oder integriert werden können, da sie zueinander eigentlich inkonsistent und inkompatibel sind. Choi et al. argumentieren außerdem, dass diese Abbildungen besser wartbar und skalierbarer sind, da die Änderungen lokal ohne Rücksicht auf andere Abbildungen gemacht werden können. Ein Beispiel für ein System aus dieser Kategorie ist *GLUE* [DMDH02] von Doan et al. Bei *GLUE* wird über verschiedene Lerner wie einem Inhaltslerner und einem Namenslerner versucht, die ähnlichsten Konzepte zwischen zwei Ontologien zu finden, um eine Abbildung zu erstellen. In die dritten Kategorie fallen nach Choi et al. Werkzeuge für das Verschmelzen von Ontologien. Hier liegt die Herausforderung besonders darin, nicht nur die Gemeinsamkeiten, sondern auch die Konflikte zwischen Ontologien zu finden. Ein Beispiel für diese Kategorie ist das halbautomatische Werkzeug *SMART* [NM99] von Noy und Musen. Das System sucht über verschiedene Metriken Ähnlichkeiten und macht anschließend dem Benutzer Vorschläge. Bei Konflikten wird der Benutzer gefragt, ob die geplante Auflösung in Ordnung ist.

5. Analyse und Entwurf

In dieser Arbeit soll zunächst ein Ansatz für die Themenextraktion entwickelt werden. Dabei sollen die Anweisungen für die Programmierung in gesprochener Sprache analysiert und passende Themen extrahiert werden. Die Eingabe der Themenextraktion besteht bei der Programmierung in gesprochener Sprache aus Transkriptionen von Anweisungen des Nutzers an das Zielsystem. Da es Spracheingaben sind, existieren in den Eingabetexten keine Satzzeichen zur Gliederung des Textes und die Satzstruktur kann verändert sein. Ebenso finden sich weitere Eigenheiten und Herausforderungen, die bereits in Kapitel 3 beschrieben wurden. Ziel ist es, zu jedem Eingabetext Themen zu finden, mit denen die Thematik des Eingabetextes beschrieben werden kann. Dabei wird die Extraktion von mehreren Themen angestrebt. Damit soll ermöglicht werden, unterschiedliche Themen in der Eingabe erkennen zu können. Dies ist bei der die Spracheingaben für die Programmierung wichtig, da dort, im Gegensatz zu Dokumenten, oftmals verschiedene Themen vorkommen. So kann beispielsweise in einer Eingabe ein Thema das System sein, das angesprochen wird, wie etwa ARMAR. Andere Themen der Eingabe beziehen sich dann wiederum auf die Aufgabe und die Umgebung, in der das System arbeiten soll, wie zum Beispiel „Küche“. Im Forschungsgebiet der Themenextraktion ist ein hauptsächliches Anliegen die Klassifikation von Texten, damit dadurch die Thematik zusammengefasst werden kann. Dieses Anliegen ist auch für diese Arbeit vorgeschehen, jedoch ist vor allem wichtig, dass die extrahierten Themen dazu genutzt werden können, zusätzliche Informationen zu den Eingabetexten gewinnen zu können.

Die Ziele der bisherigen Ansätze zur Themenextraktion unterschieden sich daher geringfügig von den Zielen in dieser Arbeit. Ebenso sind bisherige Ansätze nicht für die Programmierung in natürlicher Sprache erstellt worden. Deshalb wird in Abschnitt 5.1 die Themenextraktion gerade im Bezug auf die Herausforderungen der Programmierung in natürlicher Sprache analysiert und es werden Ansätze dazu diskutiert. Anschließend wird ein Entwurf vorgestellt.

Ein Anwendungsfall für die Themenextraktion soll die Auswahl von domänenspezifischen Ontologien sein. Passend zu den Themen der Eingabe sollen eine oder mehrere Ontologien ausgewählt werden, die Hintergrundwissen über die Eingabe bereitstellen können. Dies ist vor allem wichtig, um Informationen über beispielsweise das Zielsystem, für das die Programmierung bestimmt ist, sowie die Fähigkeiten und Möglichkeiten des Systems zu erhalten. Sollten bei der Auswahl mehrere passende Ontologien gefunden werden, ist eine Verschmelzung der Ontologien zur Kombination des Wissens notwendig. In Abschnitt 5.2

befindet sich daher die Analyse und der Entwurf zur Auswahl und Verschmelzung von Ontologien.

5.1. Themenextraktion

Wie bereits in der Einleitung zu diesem Kapitel erwähnt, soll in dieser Arbeit eine Extraktion von Themen aus einer Eingabe stattfinden, wobei sich die Anforderungen und Ziele in dieser Arbeit von denen üblicher Ansätze im Forschungsgebiet der Themenextraktion unterscheiden. Ein Unterschied zu den üblichen Ansätzen ist das abgewandelte Ziel der Themenextraktion. Anstatt für eine Eingabe eine relativ homogene Menge an Themen zu ermitteln, die die gesamte Eingabe zusammenfasst und beschreibt, soll es auch möglich sein, eine inhomogene Menge an Themen zu erhalten. Dies ist wichtig, da bei der Programmierung mit natürlicher Sprache unterschiedliche Themen in der Eingabe vorkommen können, die bei der Themenextraktion entsprechend beachtet werden sollen.

Ein anderer Unterschied ist, dass die Eingaben für die Themenextraktion keine wohlformulierten Dokumente sind, sondern gesprochene Anweisungen. Zusätzlich zu den Herausforderungen von Spracheingaben, die in Kapitel 3 aufgeführt sind, stellt bei diesen Eingaben vor allem die Länge der Eingabe die Themenextraktion vor Herausforderungen. Kurze Eingaben bringen vor allem Schwierigkeiten für die Ermittlung von Schlüsselbegriffen der Eingabe. Um für ein Dokument Schlüsselbegriffe zu ermitteln, setzen viele Ansätze zur Themenextraktion auf Verfahren wie die *Latent Dirichlet Allocation*, auf das in Abschnitt 4.1 eingegangen wurde. Dafür wird anhand eines statistischen Verfahrens die Wahrscheinlichkeiten für das Auftreten von Themen sowie die Auftrittswahrscheinlichkeit von Wörtern in bestimmten Themen berechnet. Bei der Programmierung in natürlicher Sprache werden jedoch Anweisungen verarbeitet, die in der Regel kurz gehalten sind und nur aus wenigen Sätzen bestehen. In den kurzen Anweisungsfolgen, die verarbeitet werden sollen, lassen sich mit einem statistischen Modell nur wenige hilfreiche Informationen extrahieren, da insgesamt nur wenige Informationen enthalten sind. Deshalb sind Verfahren wie das LDA-Verfahren nicht einsetzbar. Dennoch lassen sich einige Ansätze zur Themenextraktion nutzen, wenn eine angemessene Alternative zur Extraktion von Schlüsselbegriffen gefunden wird. Bei diesen Ansätzen wird das LDA-Verfahren eingesetzt, um eine erste Menge von Schlüsselbegriffen in den Dokumenten zu extrahieren. Danach werden die Schlüsselbegriffe weiterverarbeitet, um die eigentlichen Themen zu ermitteln. Diese Weiterverarbeitung kann auch für diese Arbeit genutzt werden. Deshalb sollen dafür gute Ansätze der Themenextraktion gefunden werden. Für die Extraktion von Schlüsselbegriffen muss jedoch eine Alternative gefunden werden.

Einer der Ansätze, die das LDA-Verfahren zur Ermittlung von Schlüsselbegriffen in Dokumenten anwenden, ist [HCPC13] von Hingmire und Palshikar. Die Autoren trainierten dabei mit den Ergebnissen aus dem LDA-Verfahren in Verbindung mit manuellen Zuordnungen von Themen zu Trainingsdokumenten einen Klassifikator. Neben dem Problem, dass ein adäquater Ersatz für das LDA-Verfahren gefunden werden muss, gibt es jedoch bei diesem Ansatz weitere Kritikpunkte. So könnte der resultierende Klassifikator zu stark von der Auswahl der initialen Trainingsdokumente abhängen. Die von Experten annotierten Dokumente und Themenbezeichner haben einen erheblichen Einfluss darauf, wie die weiteren Dokumente annotiert werden. Während des Trainingsprozesses werden weitere Dokumente und ihr unüberwacht annotierter Bezeichner jedoch nicht mehr überprüft. Dennoch wird der Naive-Bayes-Klassifikator mit der Annahme trainiert, dass diese Bezeichner korrekt sind. Es wird versucht, den Erwartungswert zu maximieren, ohne den Erwartungswert außerhalb der anfänglichen Expertenbezeichner zu kontrollieren. Daraus resultieren auch die Evaluationsergebnisse, die zeigen, dass der Wert für das F_1-Maß mit der Anzahl der verschiedenen Datensets erheblich sinkt. Das selbe Problem der Voreingenommenheit bei Klassifikatoren existiert auch im Ansatz [OID] von Medelyan et al. Dort

wird der Klassifikator mit Hilfe von manuell annotierten Dokumenten trainiert. Bereits in Kapitel 4 wurde erwähnt, dass manuelle Annotationen oft subjektiv und inkonsistent und durch starke Voreingenommenheit geprägt sind. Bei dem Training von Klassifikatoren für die Themenextraktion muss dieser Sachverhalt entsprechend beachtet werden.

Ein Aspekt, der bei den Ansätzen ebenfalls beachtet werden sollte, ist die Laufzeit. Einige Ansätzen sind allein durch schlechte Laufzeiten weniger attraktiv. Manche Ansätze lassen sich durch den Wachstum der genutzten Informationsquellen unter Umständen gar nicht mehr effizient einsetzen. Beispielsweise wird im Ansatz [KRW] von Coursey et al. aus der Wikipedia ein großer Graph erstellt. Wikipedia wächst immer weiter, womit auch der dazugehörige Graph entsprechend wächst, auf dem für jeden Knoten ein Wert berechnet wird. Dieser Schritt dauert entsprechend länger, je größer der Graph ist. Zwar wächst der Aufwand bei dem eingesetzen Algorithmus linear mit der Anzahl an Kanten und Knoten im Graph, da die Wikipedia aber ständig wächst, könnte ab einer gewissen Größe dieser Verarbeitungsschritt zu lange dauern. Andere Ansätze nutzen Algorithmen, die zwar bessere Ergebnisse versprechen, aber wesentlich schlechter skalieren, weshalb diese mit dem Wachstum der Informationsquellen immer unattraktiver werden.

Der Ansatz aus [HHKG13] stellt eine Möglichkeit vor, kleinere themenspezifische Graphen aus der Wikipedia zu erstellen. Dabei werden Daten aus der Wissensdatenbank von DBpedia genutzt, um für einzelne Nomen jeweils einen *Bedeutungsgraphen* zu erstellen. Dafür werden die Namen von Wikipedia-Konzepten, also der Titel eines Wikipedia-Artikels, benötigt. Für die Bedeutungsgraphen werden aus dem Ausgangskonzept verwandte Konzepte ermittelt, die bis zu zwei Verbindungen vom Ausgangskonzept entfernt sind. Hulpus et al. nutzen dafür die Verbindungen *dcterms:subject*, *skos:broader*, *skos:broaderOf*, *rdfs:subClassOf* und *rdfs:type* aus der Wissensdatenbank von DBpedia. Die Verbindungen des Typs *dcterms:subject* verbinden dabei Konzepte mit ihren Kategorien. Die hierarchieähnliche Struktur zwischen den Kategorien wird mit den Verbindungsarten *skos:broader* und *skos:broaderOf* ausgedrückt. Mit *rdfs:type*-Verbindungen werden Konzepte aus Wikipedia mit Entitäten aus der Wissensdatenbank von DBpedia verbunden. Die Hierarchie innerhalb der Ontologie wird vom Verbindungstyp *rdfs:subClassOf* beschrieben. Die Auswahl der Verbindungen ist jedoch ausbaufähig. Zum einen sind nicht alle Verbindungen vertreten, die zum Aufbau eines Bedeutungsgraphen nützlich und sinnvoll sein können. So gibt es beispielsweise die Verbindung *rdfs:seeAlso*, die auf ähnliche weiterführende Artikel verweist, aber nicht genutzt wird. Zum anderen bilden die in [HHKG13] genutzten Verbindungen vor allem hierarchische Strukturen ab, was sich in der Evaluation wiederspiegelt. Dort wurde unter anderem festgestellt, dass meist etwas zu generelle Themen vorgeschlagen werden. Ein weiteres Problem im Ansatz von Hulpus et al. ist der Umgang mit nicht zusammenhängenden Themengraphen. Wie bereits in Abschnitt 4.1 beschrieben, schreiben die Autoren, dass Teile des Graphen, die nicht mit der Hauptkomponente verbunden sind, als Rauschen interpretiert und deshalb ignoriert beziehungsweise entfernt werden. Dabei bleibt aber offen, wie eine Hauptkomponente definiert ist und wie vorgegangen werden soll, wenn der Themengraph zum Beispiel aus zwei nahezu gleich großen, nicht zusammenhängenden Teilgraphen besteht. Abseits der ungenauen Definition gehen durch die Entfernung eines Teilgraphen Informationen verloren, die für die Themenextraktion wichtig sein könnten. Wenn homogene Themen ausgewählt werden sollen, kann dies eine akzeptable Methode sein, da diese Teilgraphen möglicherweise tatsächlich ein Rauschen sind. Wenn jedoch auch inhomogene Themen aus mehreren Gebieten extrahiert werden sollen, wie es in dieser Arbeit geschehen soll, ist diese Handhabung problematisch. Nicht zusammenhängende Graphen entstehen vor allem dann, wenn verschiedene mögliche Themengebiete in der Eingabe vorkommen. Durch die Entfernung eines Teilgraphen, entfernt man den Bezug zu dem Themengebiet, der von diesem Teilgraphen abgedeckt wird. Dadurch kann ein wichtiges Themengebiet komplett verloren gehen und die Themenex-

traktion ist entsprechend ungenauer.

Auf Graphen, deren Knoten mögliche Themen repräsentieren, können mittels Graphzentralitätsalgorithmen mögliche Themenkandidaten ermittelt werden. Diese Herangehensweise verfolgen unter anderem die Ansätze [HHKG13] und [KRW]. Ein Graphzentralitätsalgorithmus weist einzelnen Knoten einen Wert zu, der die Wichtigkeit des Knotens im Graphen darstellen soll. Wenn die Knoten mögliche Themen darstellen, dann kann dadurch die Wichtigkeit eines Themas ermittelt werden. In [HHKG13] werden deshalb verschiedene Zentralitätsalgorithmen vorgestellt und getestet. Gerade für größere Graphen tendieren diese aber zu schlechter Performanz. Im Algorithmus *Random Walk Betweenness Centrality* werden beispielsweise Adjazenzmatrizen erstellt, invertiert und anderweitig manipuliert. Die Größe einer Adjazenzmatrizen hängt von der Größe des Graphen ab, wodurch die Operationen auf den Matrizen entsprechend länger dauern. Ein anderes Beispiel ist der Algorithmus *Closeness Centrality*, bei dem für jeden Knoten die durchschnittliche Länge der kürzesten Pfade zu allen anderen Knoten berechnet werden. Selbst moderne Algorithmen zur Ermittlung der kürzesten Pfade wie [Tho99] haben als untere Grenze eine lineare Laufzeit im Bezug auf die Größe des Graphen. Da von einem Knoten die kürzesten Pfade zu allen anderen Knoten ermittelt werden müssen, entsteht für diesen Rechenschritt ein quadratischer Aufwand. Der Rechenschritt ist außerdem für jeden Knoten nötig, wodurch die Laufzeit entsprechend mehr als quadratisch mit der Größe des Graphen wächst. Der angepasste PageRank-Algorithmus aus dem Ansatz von Coursey et al. [KRW] skaliert in dieser Hinsicht besser. Der Algorithmus hängt nicht von allen Knoten ab, sondern lediglich von benachbarten und skaliert deshalb linear. Dies verspricht auch bei größeren Graphen bessere Laufzeiten.

Ein Problem für viele Graphzentralitätsalgorithmen sind neben der Laufzeit auch unzusammenhängende Graphen. Viele Algorithmen setzen auf Pfade innerhalb des Knotens und auf die Länge dieser Pfade. Wenn kein Pfad existiert, haben diese Algorithmen entsprechend Probleme, da nicht definiert ist, wie verfahren werden soll. Dadurch ist der angegebene Algorithmus in der dargestellten Form oft nicht ausführbar. Zum Beispiel setzt die *Random Walk Betweenness Centrality* auf zufällige Pfade zwischen den Knoten. Für die Berechnung, wie sie in [HHKG13] beschrieben wird, wird dafür eine Adjazenzmatrix erstellt, die invertiert werden muss. Wenn ein Graph nicht zusammenhängend ist, dann ist die dazugehörige Adjazenzmatrix oftmals nicht invertierbar, wodurch der Ansatz nicht durchführbar ist. Daher ist ein großer Vorteil des PageRank-Algorithmus, dass er auch ohne Probleme auf unzusammenhängende Graphen angewandt werden kann.

5.1.1. Entwurf der Auflösung von Mehrdeutigkeiten

Eine Extraktion von Themen sollte nach [HHKG13] auch auf die Strukturen hinter den Konzepten achten. Dies soll die Qualität der extrahierten Themen steigern und gleichzeitig Themen vorschlagen können, die nicht direkt im Text erwähnt werden. Um die Informationen für das richtige Konzept hinter einem Wort zu erhalten, wird eine Auflösung möglicher Mehrdeutigkeiten benötigt. Daher soll in dieser Arbeit auch ein Ansatz für die Auflösung von Mehrdeutigkeiten, die als Problem der Verarbeitung von natürlicher Sprache in Abschnitt 2.3.3 beschrieben ist, umgesetzt werden.

Für die Auflösung von Mehrdeutigkeiten gibt es bereits einen vielversprechenden Ansatz von Mihalcea [Mih07][RA07]. Der Ansatz nutzt die Wikipedia als Korpus für die Vorkommen von mehrdeutigen Wörtern inklusiver ihrer Bedeutung. Wikipedia-Artikel stellen dabei die Bedeutungen von mehrdeutigen Wörtern dar. Innerhalb der Wikipedia-Artikel werden von menschlichen Autoren Querverweise zu anderen Wikipedia-Artikeln in den Fließtext eingearbeitet. Die Querverweise stellen die Auflösung der Mehrdeutigkeiten bereit, da Querverweise auf jeweils denjenigen Artikel verwiesen, der die Bedeutung des

Wortes darstellt. Die Wikipedia ist somit ein Korpus für die Auflösung von Mehrdeutigkeiten, da in ihr Sätze mit mehrdeutigen Wörtern, für die korrekte Bedeutung annotiert ist, enthalten sind. Im folgenden Beispiel befindet sich der Quelltext eines Satzes in einem Wikipedia-Artikel:

Beispiel

[[Cicero]] describes visiting the tomb of Archimedes, which was surmounted by a [[sphere]] and a [[cylinder (geometry)|cylinder]], which Archimedes had requested to be placed on his tomb, representing his mathematical discoveries.

In der *Wikitext*-Syntax[1], in der die Artikel geschrieben sind, stehen die doppelten eckigen Klammern für die Querverweise. Bei den Querverweisen steht der Identifikator des Artikels, auf den verwiesen werden soll, an erster Stelle. Der Identifikator eines Artikels ist gleichzeitig der Titel des Artikels. Sollte der Identifikator nicht in den Textfluss passen, kann eine andere Darstellung angegeben werden. Diese Darstellung findet man nach einem durchgehenden senkrechten Strich: |. Im Beispiel ist dies bei „[[cylinder (geometry)|cylinder]]“ zu sehen. Dort steht entsprechend der Teil vor dem Trennstrich für den Identifikator des Artikels, während der hintere Teil der Anzeigetext im Satz ist. Die Querverweise können für die Auflösung der Mehrdeutigkeit genutzt werden, da durch sie explizit annotiert ist, welche Bedeutung das Wort besitzt. Im Beispiel ist die Bedeutung „cylinder (geometry)“ gemeint, nicht eine der anderen Bedeutungen für „cylinder“ wie etwa aus den Themenbereichen der Feuerwaffen oder der Motoren. Existiert kein senkrechter Strich, so ist das angegebene Wort gleichzeitig Identifikator und Anzeigetext.

Mit den Querverweisen und weiteren Informationen aus der Umgebung des Querverweises kann ein Klassifikator trainiert werden. Mihalcea setzt in ihrer Arbeit den Naive-Bayes-Klassifikator ein, mit dem bereits andere Ansätze gute Ergebnisse erzielt haben. Lee und Ng haben in einer empirischen Studie [LN02] über verschiedene Lernalgorithmen zur Auflösung von Mehrdeutigkeiten gezeigt, dass mit dem Naive-Bayes Klassifikator ein System nach dem Stand der Forschung erstellt werden kann.

Für das Training werden Instanzen benötigt, mit denen der Klassifikator trainiert werden kann. Eine Instanz besteht aus mehreren Attributen, die die Informationen über das mehrdeutige Wort darstellen, welche aus den Querverweisen zur Auflösung der Mehrdeutigkeit sowie aus dem Umfeld des mehrdeutigen Wortes entnommen werden. Nach dem Ansatz von Mihalcea besteht eine Instanz aus dem Wort, für das die Mehrdeutigkeit aufgelöst wird, und seiner Wortart. Dazu kommen die benachbarten Wörtern, die sich drei Positionen links sowie rechts des mehrdeutigen Wortes befinden, inklusive ihrer Wortarten im Satz. Außerdem werden noch die Informationen über das nächste Verb und Nomen, die sich links und rechts der Ausgangswortes befinden, sowie bis zu fünf Kontextwörter hinzugefügt. Nach Mihalcea sind diese Kontextwörter die Wörter, die am häufigsten im Abschnitt des mehrdeutigen Wortes vorkommen, wobei jedes dieser Wörter mindestens drei mal auftreten muss. In Abbildung 5.1 sind diejenigen Merkmale markiert, die sich in einer Instanz zur Auflösung der Mehrdeutigkeiten für das Wort „cylinder“ zur Bedeutung „cylinder (geometry)“ befinden. Dabei wurden die Kontextwörter ausgelassen, um das Beispiel kurzzuhalten. Unterhalb der Wörter befinden sich die jeweiligen Wortarten. In diesem Beispiel sind die Wörter „sphere“, „and“ und „a“ die benachbarten Wörter links des Ausgangswortes und die Wörter „which“, „Archimedes“ und „had“ die benachbarten Wörter rechts des Ausgangswortes. Das Nomen und das Verb links des mehrdeutigen Wortes werden durch „sphere“ und „surmounted“ repräsentiert, das Nomen und das Verb rechts des mehrdeutigen Wortes durch „Archimedes“ und „had“.

[1] https://en.wikipedia.org/wiki/Help:Wikitext

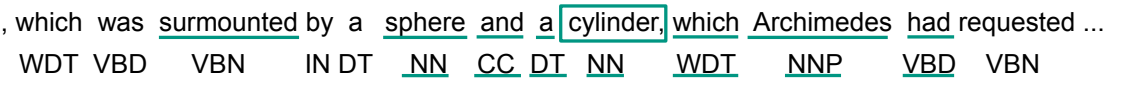

Abbildung 5.1.: Auszug aus einem Satz, aus dem für das Wort „cylinder“ zur Bedeutung „cylinder (geometry)“ eine Trainingsinstanz kreiert wird. Markierte Elemente entsprechen Merkmalen, die für die Erstellung der Instanz genutzt werden. Unterhalb der Wörter stehen abgekürzt die dazugehörigen Wortarten (vgl. Tabelle 2.1).

In [Mih07] schreibt Mihalcea außerdem, dass für Eigennamen keine Auflösung der Mehrdeutigkeit geschehen soll. Eigennamen sollten nicht über die Auflösung von Mehrdeutigkeiten verarbeitet werden, sondern über einen eigenständigen Prozess zur Erkennung von Eigennamen. Deshalb soll in dieser Arbeit ebenso verfahren werden und keine Auflösung von mehrdeutigen Wörtern, die Eigennamen darstellen, stattfinden.

Der größte Vorteil dieses Ansatzes ist, dass mit Wikipedia ein großer annotierter Korpus bereits vorliegt. Zusätzlich wird der Korpus stetig erweitert und verbessert, wodurch vorstellbar ist, dass sich die Ergebnisse des Ansatzes zu späteren Zeitpunkten ebenfalls verbessern. Die Texte aus den Artikeln haben ebenso eine andere Verteilung als bisherige, in der Regel künstlich kreierte Korpora. Laut Mihalcea wird bei der Erstellung der Wikipedia-Texte nicht auf eine gleichmäßige Verteilung geachtet, sondern es entsteht eine natürlichere Verteilung, zumindest im Kontext einer Enzyklopädie. In der Evaluation in [Mih07] erreicht der Klassifikator eine Genauigkeit von 84,65% auf einem Teil der Testkorpora der SENSEVAL-2 [PY01] und SENSEVAL-3 [MCK04], während ein Vergleichsansatz [KR00] auf eine Genauigkeit von 78,02% kommt.

Laut den Richtlinien zur Erstellung von Wikipedia-Artikeln [Wik18b] sollen Querverweise im Text nur dorthin führen, wo es Erläuterungen für Fachbegriffe oder weiterführende Informationen zum aktuellen Artikel gibt. Dies kann für diesen Ansatz hilfreich sein, da dadurch implizit sichergestellt wird, dass der Kontext der Bedeutung des mehrdeutigen Wortes auch der Kontext ist, in dem man jene Bedeutung erwarten kann. Dies kann allerdings auch Nachteile mit sich bringen, da dadurch manche Vorkommen nicht einbezogen werden. Die Informationen über andere Kontexte, in denen eine Bedeutung eines mehrdeutigen Wortes vorkommen kann, bleiben verborgen und die wahre Verteilung über das Vorkommen einer Bedeutung kann verfälscht werden.

Ein weiterer Nachteil, der sich aus diesem Ansatz ergibt, sind fehlende Auflösungsmöglichkeiten für manche Begriffe. Beispielsweise existiert in Wikipedia kein konkret passender Begriff für das Wort „front“, wie er in dem Ausdruck „in front of you“ vorkommt. WordNet [Fel98] liefert dafür eine Bedeutung mit der Erklärung „the immediate proximity of someone or something“, die hier passen würde. Desweiteren spricht Mihalcea ebenfalls von dem Problem, dass für manche Wortbedeutungen eine starke voreingenommene Verteilung existiert. Ein Beispiel hierfür ist das Wort „Bank“, das in fast jedem Vorkommen in Wikipedia mit der Bedeutung als finanzielles Institut annotiert ist, während die Bedeutung als Sitzgelegenheit entsprechend vernachlässigt wird und kaum jemals annotiert wird. Dadurch fehlen unter Umständen ausreichend Beispiele, um diese Bedeutung zuverlässig erkennen zu können.

Inkonsistente Querverweise stellen ebenfalls ein Problem dar. Die Benutzer verweisen an einigen Stellen nicht auf den Hauptartikel, sondern auf sogenannte Weiterleitungsseiten. Ein Beispiel dafür ist der Verweis auf die Weiterleitungsseiten „Fridge“, die lediglich auf den eigentlichen Artikel „Refrigerator“ weiterleitet. Dadurch wird zum einen die korrekte

Tabelle 5.1.: Die verschiedenen Attribute der Instanzen für die Auflösung der Mehrdeutigkeiten. POS steht für *Part Of Speech* und steht für die Wortart. Negative Zahlen beschreiben die Positionen links des Wortes, positive Zahlen die Positionen rechts.

#	Attribut	#	Attribut
0	Bedeutung	11	Wort +2
1	Wort	12	Wort +2 POS
2	Wort POS	13	Wort +3
3	Wort -3	14	Wort +3 POS
4	Wort -3 POS	15	Nomen links des Wortes
5	Wort -2	16	Verb links des Wortes
6	Wort -2 POS	17	Nomen rechts des Wortes
7	Wort -1	18	Verb rechts des Wortes
8	Wort -1 POS		
9	Wort +1		
10	Wort +1 POS		

Verteilung der Vorkommen einer Bedeutung verfälscht, zum anderen können so mehrdeutige Wörter je nach Umgebung zu verschiedenen, aber synonymen, Begriffen aufgelöst werden. Dies muss entsprechend in der Weiterverarbeitung bedacht werden. Alternativ könnte versucht werden, diese synonymen Verweise vor dem Training aufzulösen.

Zuletzt ist ein Nachteil, dass ausschließlich Nomen disambiguiert werden können. Andere Mehrdeutigkeiten wie beispielsweise für Adjektive oder Verben können mit diesem Verfahren nicht angegangen werden. So kann zum Beispiel nicht bestimmt werden, welche Bedeutung des Verbs „umfahren“ im Satz „Ich werde das Hindernis umfahren“ gemeint ist. Für den Einsatz in Verbindung mit der Themenextraktion genügt allerdings die Auflösung von Nomen.

Trotz der Nachteile überwiegen die Vorteile mitsamt des größten Vorteils: Die Auflösung der Mehrdeutigkeiten geschieht zu Konzepten aus der Wikipedia. Dadurch können nach der Auflösung der Mehrdeutigkeit in anderen Verarbeitungsschritten, wie beispielsweise der Themenextraktion, auch Informationen aus der Wikipedia zu den einzelnen Konzepten genutzt werden, da der Titel des Wikipedia-Artikels nun bekannt ist.

Damit dieser Ansatz im Kontext der natürlichsprachigen Programmierung mit PARSE genutzt werden kann, muss er jedoch angepasst werden. So sind die einzelnen Befehle in der Regel eher kurz und der Kontext ist sehr beschränkt. Daher ist die Einbindung von Kontextwörtern wenig hilfreich für die Klassifikation. Deshalb soll auf die Nutzung der Kontextwörter im Merkmalsvektor verzichtet werden. Die verschiedenen Attribute, die für die Auflösung der Mehrdeutigkeiten genutzt werden sollen, sind in Tabelle 5.1 zu sehen. Die Bedeutung ist dabei lediglich in den Trainingsinstanzen vorhanden. Aufgelöst werden sollen alle Nomen im Satz, wobei Eigennamen ausgeschlossen werden.

Bei diesem Ansatz wurde außerdem der Entschluss gefasst, keine zusammengesetzten Nomen aufzulösen. Kritisch beim Auflösen von zusammengesetzten Nomen ist vor allem, dass für jedes aufzulösende zusammengesetzte Wort entsprechende Trainingsdaten existieren müssen, was jedoch in der Wikipedia nicht unbedingt gegeben ist. Desweiteren gibt es noch das Problem bei der Anwendung in PARSE, da dort, nach aktuellem Stand, keine zusammengesetzten Nomen erkannt werden können. Dies bringt das Problem auf, dass zuerst eine zuverlässige Erkennung von zusammengesetzten Nomen entwickelt werden müsste. Ein einfacher Ansatz ist zum Beispiel, aufeinanderfolgende Nomen stets als zusammengesetzte

Nomen zu betrachten, was jedoch fehlerbehaftet sein kann. Da vor allem auch keine Satzzeichen existieren, wäre eine hohe Rate an Fehlklassifikationen bei Aufzählungen und über Satzgrenzen hinaus die Folge. Hier müsste zunächst eine zuverlässige Methode entwickelt werden, zusammengesetzte Nomen zu erkennen.

Für die Klassifikation soll außerdem noch eine alternative Methode eingebaut werden: Anstatt lediglich die wahrscheinlichste Klasse auszuwählen, sollen mit dieser Option die drei wahrscheinlichsten Klassen in Erwägung gezogen werden. Befindet sich das Ausgangswort als Teil in der Bezeichnung einer der drei wahrscheinlichsten Bedeutungen, wird diese Bedeutung bevorzugt. Kommt das Wort in mehreren Bedeutungen vor, wird unter diesen die wahrscheinlichste Bedeutung ausgewählt. Beispielsweise kann es durch die Wörter in der Umgebung des aufzulösenden Wortes *orange* dazu kommen, dass die wahrscheinlichste Bedeutung *Grapefruit* ist und an zweiter und dritter Stelle die Bedeutungen *Orange (fruit)* und *Lemon* stehen. Da das Ausgangswort *orange* ist, wird in diesem Auswahlvorgang die Bedeutung *Orange (fruit)* bevorzugt. Dabei soll jedoch überprüft werden, ob die Wahrscheinlichkeit der so ausgewählten Bedeutung im Vergleich zur wahrscheinlichsten Bedeutung ähnlich ist. Das bedeutet, dass der Unterschied der Wahrscheinlichkeitswerte nicht zu groß ist. Damit soll verhindert werden, dass eine sehr wahrscheinliche Klassifikation im Austausch für eine unwahrscheinliche verworfen wird. Ob diese Methode eine Verbesserung der Klassifikation bringt, soll in der Evaluation untersucht werden.

5.1.2. Entwurf der Themenextraktion

Nachdem die Nomen im Satz disambiguiert und damit zu Wikipedia-Artikeln zugewiesen werden können, wird die Themenextraktion angegangen. In dieser Arbeit sollen Teile aus verschiedenen Ansätzen kombiniert werden, um Nachteile ausgleichen zu können. Dabei sind vor allem die Ansätze [HHKG13] und [KRW] vielversprechend, vor allem, weil diese Verfahren unüberwacht ablaufen und somit kein Trainingskorpus erstellt werden muss. In dieser Arbeit soll die Themenextraktion in den folgenden Schritten ablaufen: Wie in [HHKG13] sollen zuerst die Schlüsselbegriffe der Eingabetexte extrahiert werden, um für jeden Schlüsselbegriff jeweils einen Bedeutungsgraphen zu erstellen. Dabei werden für die Schlüsselbegriffe die aufgelösten Mehrdeutigkeiten, die mit dem Ansatz aus Abschnitt 5.1.1 ermittelt werden, gebraucht, damit die Bedeutungsgraphen aus den Bedeutungen erstellt werden können. Im Anschluss werden die Bedeutungsgraphen zu einem Themengraphen verschmolzen. Auf diesem werden die Werte für die Graphzentralität bestimmt, wodurch wichtige Knoten und dadurch wichtige Themen gekennzeichnet werden sollen. Im Anschluss findet das Auswahlverfahren der Knoten anhand verschiedener Kriterien statt, durch die die gemeinsamen Themen der Bedeutungen ermittelt werden sollen.

Für die Extraktion der Schlüsselbegriffe aus der Eingabe wird in [HHKG13] das LDA-Verfahren eingesetzt. Dieses lässt sich aber, wie bereits in Abschnitt 5.1 erläutert, für Eingaben aus der Programmierung mit natürlicher Sprache nicht einsetzen. Statt das LDA-Verfahren zu nutzen, um Schlüsselbegriffe aus dem Text zu extrahieren, soll in dieser Arbeit eine Extraktion aller Nomen stattfinden. Nomen stellen Schlüsselwörter einer Eingabe dar und bestimmen die Themen, die behandelt werden [XNXT04]. Auf lange Dokumente wendet man das LDA-Verfahren an, um aus der Sammlung von Wörtern aus dem Dokument die wichtigen Schlüsselbegriffe herauszufiltern. Anweisungen für die Programmierung sind jedoch in der Regel zielführender und präziser. Daher werden selten Nomen genutzt, die von der Thematik der Anweisungen abweichen. Somit kann jedes Nomen als wichtig für die Themenextraktion betrachtet werden. Eigennamen sollen hier jedoch nicht verwendet werden, da diese nicht auf die selbe Weise weiterverarbeitet werden können. Dies liegt daran, dass zu Eigennamen keine Auflösung zu Wikipedia-Artikeln beispielsweise durch die Auflösung von Mehrdeutigkeiten stattfindet. Dies ist unter anderem auch dadurch geschuldet, dass zu einigen Eigennamen wie für den Haushaltsroboter ARMAR kein entsprechender

Tabelle 5.2.: Genutzte Arten von Verbindungen zwischen DBpedia-Einträgen

Bezeichner	**Bedeutung**
dcterms:subject	Verbindung eines Konzepts zur dazugehörigen Wikipedia-Kategorie
skos:broader	Hierarchie zwischen Kategorien, allgemeiner
skos:broaderOf	Umgekehrte Verbindung von skos:broader
skos:narrower	Hierarchie zwischen Kategorien, spezifischer
purlg:hypernym	Hyperonymie zwischen Konzepten
purlg:meronym	Teil-Ganzes-Beziehungen zwischen Konzepten
purlg:synonym	Synonymie zwischen Konzepten
rdfs:type	Verbindung zu DBpedia-Ontologie-Entitäten
rdfs:subClassOf	Unterklasse innerhalb der DBpedia-Ontologie
rdfs:seeAlso	Ähnliche, weiterführende Konzepte

Wikipedia-Artikel existiert. Im folgenden Beispiel kann man einen typischen Befehl sehen, wie er von einer Versuchsperson in einem Szenario zur Steuerung des ARMAR genutzt wurde.

Beispiel

okay Armar go to the table grab popcorn come to me give me the popcorn which is in your hand

In dieser Eingabe sind neben dem Eigennamen *Armar* die Nomen *table*, *popcorn* und *hand* vorhanden. Die Nomen sollen nach der Auflösung der Mehrdeutigkeiten als Eingabe für die Themenextraktion verwendet werden.

Nachdem die Schlüsselbegriffe extrahiert wurden, können aus diesen die Bedeutungsgraphen erstellt werden. Dafür wird der Ansatz aus [HHKG13] genutzt, wobei Anpassungen gemacht werden sollen. Wie bereits in der Analyse in Abschnitt 5.1 beschrieben, sind bei der Extraktion der verbundenen Konzepte nicht alle erwünschten Verbindungen vertreten. Deshalb sollen in dieser Arbeit die Verbindungen genutzt werden, die in Tabelle 5.2 zu sehen sind. Darin sind die Verbindungsarten *skos:broader*, *skos:broaderOf*, *rdfs:type*, *rdfs:subClassOf* und *dcterms:subject* vertreten, die bereits in [HHKG13] genutzt wurden, um die hierarchischen Verbindungen abzubilden. Diese werden durch die Verbindungsarten *skos:narrower* und *purlg:hypernym* ergänzt, die die hierarchischen Strukturen weiter verfeinern sollen. Weiterhin wird mit den Verbindungsarten *purlg:meronym*, *purlg:synonym* und *rdfs:seeAlso* versucht, auch Relationen außerhalb der einfachen hierarchischen Strukturen einzubeziehen, die dennoch eine Verwandtschaft und Nähe abbilden. Gleichzeitig sollen nicht nur gerichtete Verbindungen vom aktuellen Konzept zu anderen Konzepten gefunden werden, sondern auch die entgegengesetzten Verbindungen. Dies bringt in der Theorie wenige zusätzliche Verbindungen, da beispielsweise *skos:broader* und *skos:broaderOf* diese Verbindungen schon abbilden. In der Analyse von Konzepten der DBpedia konnte jedoch festgestellt werden, dass in einigen Fällen die entgegengesetzten Verbindungen fehlen. Beispielsweise kann dadurch ein Konzept eine Verbindung des Typs *skos:broader* zu einem anderen Konzept haben, das andere Konzept aber keine *skos:broaderOf*-Verbindung zurück. Außerdem wurde bereits in [HHKG13] darauf hingewiesen, dass einige Konzepte gefunden werden können, die unerwünscht sind. Dazu gehören unter anderem administrative Seiten der Wikipedia, etymologische oder sehr generelle Konzepte. Diese führen dazu, dass unerwünschte Verbindungen zwischen Konzepten entstehen oder völlig zusammenhanglose Konzepte in den Graphen aufgenommen werden. Aus diesem Grund soll eine Liste an Konzepten, die herausgefiltert werden sollen, erstellt werden.

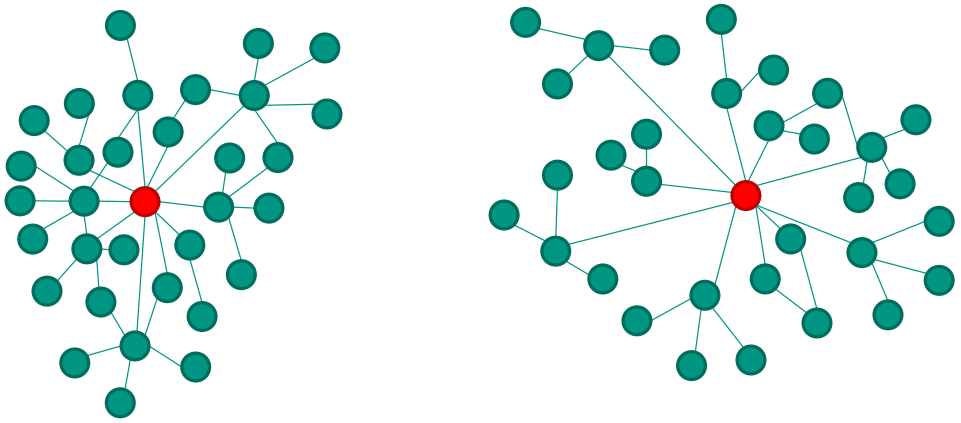

Abbildung 5.2.: Zwei beispielhafte Bedeutungsgraphen. Die rot markierten Knoten repräsentieren die Bedeutung, mit der der Graph erstellt wurde.

In Abbildung 5.2 sind zwei beispielhafte Bedeutungsgraphen zu sehen, in dem die Ausgangsknoten, also diejenigen Knoten, die die Bedeutungen darstellen, mit der der Graph erstellt wurde, rot markiert wurden. Aus den einzelnen Bedeutungsgraphen für die Nomen wird dann der Themengraph erstellt, indem die Bedeutungsgraphen verschmolzen werden. Als Resultat aus den beiden Bedeutungsgraphen könnte der Themengraph aus Abbildung 5.3 entstehen. Auch hier sind zur Verdeutlichung die ursprünglichen Ausgangsknoten rot eingefärbt. Zusätzlich wurden Knoten, die in beiden Bedeutungsgraphen vorkamen und somit die Bedeutungsgraphen im Themengraphen verbinden, hellgrün eingefärbt.

Hulpus et al. entfernen in ihrer Arbeit außerdem unzusammenhängende Teilgraphen. Unzusammenhängende Graphen können in diesem Ansatz öfter auftreten, da Konzepte aus verschiedenen Themengebieten in einer Anweisung vorkommen können. Beispielsweise können in einer Anweisung sowohl das Themen „Robot“ als auch das Thema „Home“ vorhanden sein. Dabei ist denkbar, dass der resultierende Themengraph aus mehreren Teilen besteht, die nicht verbunden sind. Um keine Themen beziehungsweise Themengebiete zu verlieren, sollen nicht zusammenhängenden Teilgraphen nicht entfernt werden. Wenn eine Bedeutung mehrfach in der Eingabe vorkommt, werden zusätzlich die Gewichte aller Kanten in demjenigen Teilgraphen erhöht, der dem Bedeutungsgraphen entspricht. So sollen häufiger vorkommende Begriffe stärker gewichtet werden.

Auf dem Themengraphen können daraufhin die Werte für die Graphzentralität bestimmt werden. Hierbei muss darauf geachtet werden, dass einige Algorithmen zur Berechnung der Graphzentralität Probleme bei nicht zusammenhängenden Graphen haben. Der angepasste PageRank-Algorithmus von Coursey et al. in [KRW] soll daher als Grundlage dienen, da er robust genug für unzusammenhängende Graphen ist. Dabei wird, wie bereits in Abschnitt 4.1 beschrieben, in den PageRank-Algorithmus eine Beeinflussung eingeführt, durch die ein stärkerer Fokus auf die Ausgangsknoten und die Knoten in deren Umgebung gelegt werden. Die Autoren nutzen für ihre Formel für die Beeinflussung jedoch unter anderem Werte, die aus dem LDA-Verfahren stammen. Diese Werte existieren hier allerdings nicht, da das LDA-Verfahren nicht angewandt wurde. Deshalb soll eine andere Lösung gewählt werden. Als Alternative wurde in [KRW] vorgeschlagen, stattdessen für $f(V_i)$ in Gleichung 4.3 für einen Ausgangsknoten eine Eins und ansonsten eine Null zurückzugeben. Dadurch erhält man als Ausgabe der Beeinflussungsfunktion $Bias(V_i)$ die Werte, die in Gleichung 5.1 formuliert sind.

$$\text{Bias}(V_i) = \begin{cases} 0 & , V_i \notin \text{InitialNodeSet} \\ \frac{1}{|\text{InitialNodeSet}|} & , V_i \in \text{InitialNodeSet} \end{cases} \tag{5.1}$$

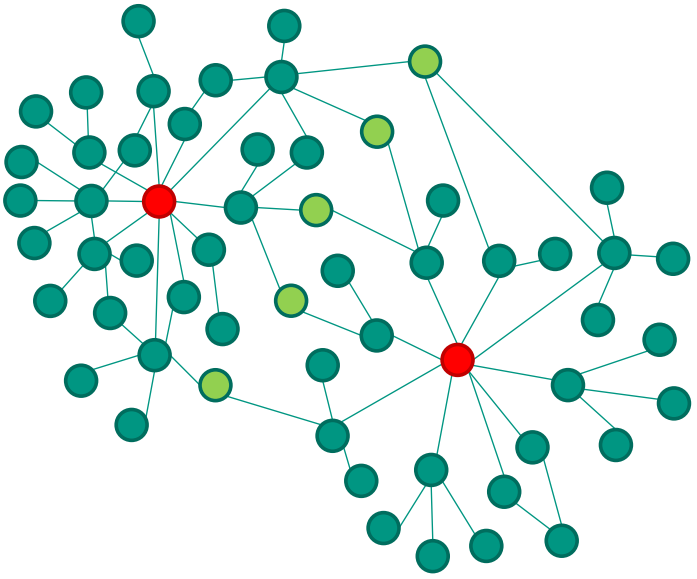

Abbildung 5.3.: Themengraph, der aus zwei Bedeutungsgraphen entstand. Rote Knoten sind die ursprünglichen Bedeutungen, aus denen die Bedeutungsgraphen entstanden. Hellgrüne Knoten repräsentieren gemeinsame Knoten der beiden Bedeutungsgraphen.

Im letzten Schritt können mit den Informationen aus dem Graphen mitsamt den ermittelten Werten der Graphzentralität die Themen ermittelt werden. In Ansätzen wie [HHKG13] oder [KRW] können direkt über die Werte für die Graphzentralität die Themen ermittelt werden. Dabei werden die Knoten, die die höchste Graphzentralität besitzen, als Themen genutzt. Dies beruht vor allem darauf, dass die Ausgangsbegriffe bereits wichtige Begriffe innerhalb eines Dokuments waren und dadurch eher kongruent sind. Für homogene Themen für Dokumente ist dies ausreichend. Allerdings kann es wie bereits erwähnt, in Anweisungen für die Programmierung in natürlicher Sprache notwendig sein, verschiedene Themengebiete zu wählen. Beispielsweise sollen verschiedene Themengebiete wie „Robot“ und „Home“ erkannt werden, wenn diese im Satz vorkommen. Problematisch dabei ist, dass in der Regel bei Zentralitätsalgorithmen höhere Werte innerhalb eines Teilgraphen auftreten und somit andere Teilgraphen nicht beachtet werden. Deshalb soll bei den Knoten zusätzlich betrachtet werden, aus welchen Bedeutungsgraphen sie entstammen. Dadurch kann für jeden Knoten eine *Konnektivität* zu den Ausgangsbedeutungen bestimmt werden. Die Konnektivität besagt, in wie vielen Bedeutungsgraphen des Themengraphen der Knoten ursprünglich vorkam. Knoten mit hoher Konnektivität stellen somit gemeinsame Elemente zwischen den Bedeutungsgraphen, zu denen sie gehören, und damit zwischen den dazugehörigen Ausgangsbedeutungen dar. Dies spricht dafür, dass diese Knoten mögliche Kandidaten für gemeinsame Themen sind, weshalb die Konnektivität der Knoten für die Themenauswahl priorisiert werden soll. Bei Knoten mit gleicher Konnektivität sollen dann diejenigen Knoten ausgewählt werden, die einen höheren Wert für die Zentralität besitzen.

Bei der endgültigen Auswahl von Themen sollen zwei Varianten getestet werden. In der ersten Variante werden diejenigen Themen ausgewählt, für die die Knoten die höchste Konnektivität und innerhalb der selben Konnektivität den höchsten Zentralitätswert besitzen. Diese Knoten stellen somit die wahrscheinlichsten gemeinsamen Themen dar. In der zweiten Variante soll die Auswahl der Themen in Abhängigkeit voneinander stattfinden. Dabei wird im ersten Schritt ein Thema wie in der ersten Variante ausgewählt. Wenn der Knoten zu dem Thema nicht zu jedem Ausgangsknoten verbunden ist, dann muss im nächsten

Schritt noch mindestens ein Thema für die nicht verbundenen Ausgangsbedeutungen gewählt werden. Das soll dafür sorgen, dass zuerst das wahrscheinlichste gemeinsame Thema für möglichst viele Ausgangskonzepte gefunden wird und dann im zweiten Schritt ein oder mehrere möglichst gemeinsame Themen für die restlichen Ausgangskonzepte. Dafür wird versucht, einen oder mehrere Knoten zu finden, die mit möglichst vielen der fehlenden Bedeutungen verbunden sind. Werden mehrere Kandidaten gefunden, wird erneut derjenige gewählt, der die höchste Zentralität besitzt. Zum Beispiel wird im ersten Schritt bei einem Themengraphen mit den fünf Ausgangsbedeutungen „Gate“, „Greenhouse“, „Pond“, „Lawn“ und „Table“ das Thema „Gardening“ gefunden, dessen Knoten eine Konnektivität von Drei besitzt und die Bedeutungen „Gate“, „Greenhouse“ und „Lawn“ verbindet. Für die verbleibenden zwei Ausgangsbedeutungen „Pond“ und „Table“ wird dann im zweiten Schritt versucht, ein gemeinsames Thema über einen gemeinsamen Knoten zu finden. Da kein gemeinsames Thema für die beiden Ausgangsbedeutungen ermittelt werden kann, werden daraufhin Themen für die beiden Bedeutungen einzeln ermittelt, die also nur im Bedeutungsgraphen zu den jeweiligen Bedeutungen zu finden sind und dort jeweils hohe Zentralitätswerte besitzen. Für „Table“ wäre das gefundene Thema „Furniture“.

Wichtig bei der Extraktion von Themen ist außerdem die Anzahl an Themen, die extrahiert werden sollen. Werden zu viele Themen extrahiert, läuft der Ansatz schnell in Gefahr, dass ein Teil der extrahierten Themen qualitativ schlecht ist, da sie beispielsweise zu ungenau oder sogar falsch sind. Werden andererseits zu wenige Themen extrahiert, könnte ein gutes Thema fehlen. Aus diesem Grund ist es sinnvoll, keine feste Anzahl vorzugeben, sondern diese flexibel nach der Eingabegröße zu skalieren. Dabei soll die Anzahl an Themen mit der Anzahl an Bedeutungen im Themengraphen wachsen. Gleichzeitig soll eine obere Grenze an Themen festgelegt werden, um bei großen Eingaben nicht zu viele Themen auszuwählen.

5.2. Zuordnung von Themen zu Domänen

Als Anwendungsfall für die Themenextraktion soll eine Zuordnung der in der Eingabe gefundenen Themen zu einer oder mehreren passenden themenspezifischen Ontologien stattfinden. Dadurch kann themenspezifisches Wissen über die Domäne der Anweisung genutzt werden. Das Wissen umfasst unter anderem die Aktionen, die ausgeführt werden können, sowie die Umgebung und wie das System mit ihr interagieren kann. PARSE nutzt dieses Wissen, wie in Kapitel 3 beschrieben, um die benötigten Informationen zur Erzeugung des Quelltextes zu erhalten. Bisher wurde stets eine einzige große Ontologie genutzt, um das gesamte benötigte Wissen bereitstellen zu können. Große Ontologien haben jedoch den Nachteil, dass sie schwer aufzubauen und zu warten sind. Bei kleineren Ontologien ist dies einfacher, da in der Regel weniger Konsistenzprobleme auftreten können. Außerdem existieren weniger Abbildungsmöglichkeiten, was die Abbildung der Eingabetexte auf die Ontologie vereinfacht. Daher ist es sinnvoll, statt einer großen Gesamtontologie, eine kleinere, spezifische Ontologie pro Themengebiet zu erstellen. Wenn aber nun mehrere Ontologien existieren, dann muss entschieden werden, welche Ontologien jeweils genutzt werden sollen. Deshalb soll eine Zuordnung der Eingabetexte zu passenden Ontologien stattfinden. Für diese Zuordnung existieren mehrere Teilprobleme, die jeweils gelöst werden müssen. Zuerst muss ein Ansatz gefunden werden, mit dem die Ähnlichkeit der Themen aus der Eingabe mit den Themen einer Domänenontologie bestimmt werden kann. Hier muss vor allem auch definiert werden, wie die Thematik einer Ontologie ermittelt wird. Danach muss festgelegt werden, wie und welche Ontologien ausgewählt werden. Zuletzt müssen Ontologien verschmolzen werden können, wenn mehrere Ontologien ausgewählt wurden, damit das Wissen der ausgewählten Ontologien zusammengenommen werden kann.

Zunächst wurde die bereits bestehende Ontologie analysiert, um eine Übersicht zu erhalten, wie eine Domäne für die Programmierung in natürlicher Sprache beschaffen ist. Aktuell

Tabelle 5.3.: Struktur der Domänenontologie von PARSE

Konzept	**Beschreibung**	**Beispielinstanzen**
Thing	Oberkonzept der Ontologie	
DataType	Datentypen	boolean, double
Method	ausführbare Aktionen	bring, grab
Object	auftretende Objekte	Dishwasher, Cupboard
Parameter	Parameter-Arten d. Methoden	grab.what, move.howFar
State	auftretende Zustände	closed, dirty
System	ansprechbares System	Armar, Robot
Value	Werte der Parameter	Dishwasher, Cupboard

gibt es im Projekt PARSE eine manuell erstellte Domänenontologie für den Haushaltsroboter ARMAR, deren Struktur in Tabelle 5.3 abgebildet ist. In der Ontologie befinden sich Informationen über den eingesetzten Roboter, die Funktionen des Roboters, die Parameter der Funktionen sowie die auftretenden Objekte und Zustände. Zusätzlich sind Beziehungen zwischen den Individuen enthalten, die unter anderem beschreiben, wie Objekte miteinander in Beziehung stehen, welche Zustände ein Objekt annehmen kann oder durch welche Methode bestimmte Zustandsübergange ausgelöst werden. Die Ontologie lässt sich dabei in zwei logische Hauptkomponenten unterteilen. Der erste Teil beschreibt das System, der zweite Teil beschreibt die Umgebung, in der das System eingesetzt werden soll. In der Domänenontologie von PARSE gibt es entsprechend den Teil, der den Roboter ARMAR inklusive seiner Funktionen beschreibt. Gleichzeitig existiert die Umgebung vor allem durch die Objekte, die man einer Haushalts- beziehungsweise Küchenumgebung zuordnen kann, wie beispielsweise Einrichtungsgegenstände, Geschirr und elektronische Küchengeräte. Ein Entwurf zur Erstellung von Ontologien für die Umgebung sowie die Systeme ist in Abschnitt 5.2.1 zu finden.

Um später für Eingaben jeweils passende Ontologien finden zu können, müssen den Ontologien Themen zugewiesen werden, ähnlich wie bereits für die Anweisungen der Nutzer, die verarbeitet werden sollen. Diese Themen sollen dann miteinander abgeglichen werden können. Ein einfacher Ansatz wäre die manuelle Annotation von Themen an die Ontologien. Beim Betrachten der existierenden Ontologie erkennt ein menschlicher Betrachter schnell in welchem Kontext die Ontologie eingesetzt werden soll. Allerdings sind manuelle Annotationen subjektiv und oft nicht konsistent, weshalb hier Probleme entstehen können. Deshalb sollte es einen automatischen Ansatz geben, mit dem aus Ontologien Themen extrahiert werden können. Diese Problematik erinnert stark an die Problematik der Themenextraktion aus Texten. Der Unterschied liegt hier lediglich bei der Art der Eingabe. Statt Themen für Texte sollen nun für Ontologien Themen gefunden werden. Dementsprechend soll versucht werden, die Themenextraktion aus Abschnitt 5.1.2 auch hier anwenden zu können. Nachdem Themen für die Ontologien ermittelt werden können, muss ein Ansatz gefunden werden, mit dem diese Themen mit den Themen aus den Eingabetexten verglichen werden können. Dadurch sollen eine oder mehrere Ontologien gefunden werden können, die zu den Eingabetexten thematisch am besten passen. Die Entwürfe zur Ermittlung der Themen einer Ontologie und der Auswahl der passenden Ontologien sind in Abschnitt 5.2.2 zu finden.

Die so ermittelten thematisch passenden Ontologien müssen dann verschmolzen werden. In Abschnitt 4.2 wurden einige Ansätze zur Verschmelzung beziehungsweise Abbildung von Ontologien vorgestellt. Die Ansätze liefern alle zufriedenstellende Ergebnisse, weshalb alle potentiell eingesetzt werden können. Große Unterschiede zwischen den Ansätzen liegen vor allem in der Handhabung größerer Ontologien durch unter anderem Teile-Und-Herrsche-Ansätze oder ähnlichem. Das Ziel in dieser Arbeit soll jedoch sein, kleinere themenspe-

zifische Ontologien zu kombinieren. Dadurch ist die Performanz bei großen Ontologien weniger entscheidend. Wichtig ist vor allem, dass ein Ansatz mit dem Verarbeiten von Ontologien aus verschiedenen Thematiken zurechtkommt. Entsprechend sind vor allem die Ansätze interessant, die in den *Verzeichnis*- und *Anatomie*-Testfällen der OAEI gut abschneiden. Dadurch kann besser sichergestellt werden, dass die Verschmelzung von realen Domänenontologien auch für verschiedene Themenfelder gut funktioniert. Für einfache Ontologien, reicht jedoch auch ein einfaches Verfahren. In Abschnitt 5.2.3 wird der Entwurf zur Verschmelzung von Ontologien beschrieben.

5.2.1. Entwurf der Erstellung von themenspezifischen Ontologien

Das Ziel der Ontologieauswahl ist, statt einer großen allgemeinen Ontologie mehrere kleine domänenspezifischen Ontologien einsetzen zu können. Die domänenspezifischen Ontologien sollen passend anhand der Themen der Anweisungen für die Programmierung in natürlicher Sprache ausgewählt und gegebenfalls miteinander verschmolzen werden. Da zuvor nur eine große Ontologie existierte, benötigt es der Erstellung dieser kleineren Ontologien. Dafür ist es hilfreich, die Struktur für diese Ontologien festzulegen, um zum einen die Erstellung zu vereinheitlichen und zum anderen die späteren Verschmelzungen zu erleichtern. Deshalb wird im folgenden beschrieben, wie die domänenspezifischen Ontologien erstellt werden sollen.

In der Analyse in Abschnitt 5.2 wurde festgestellt, dass sich die aktuell existierende Ontologie in zwei Teile unterteilen lässt. Ein Teil repräsentiert den Akteur, der andere Teil repräsentiert die Umgebung beziehungsweise Umwelt, in der das System agieren soll. Daraus kann abgeleitet werden, wie die themenspezifischen Ontologien gestaltet werden sollen: Die themenspezifischen Ontologien sollen entweder einen Akteur beziehungsweise ein System darstellen oder die Umgebung, in der ein System agieren soll. So ist ein Beispiel für einen Akteur der Roboter ARMAR und ein Beispiel für eine Umgebung ist der Haushalt beziehungsweise die Küche. Die Trennung von Akteuren und Umgebungen hat auch den Vorteil, dass diese einfacher in anderen Kontexten wiederverwendet werden können. Beispielsweise kann eine Ontologie, die nur für den Einsatz von ARMAR im Haushalt entwickelt wurde, auch nur dort eingesetzt werden. Wenn Akteur und Umgebung getrennt werden, dann kann die ARMAR-Ontologie auch beispielsweise mit einer Umgebungsontologie für den Garten kombiniert werden, um so den Einsatz von ARMAR im Garten zu ermöglichen. Gleichzeitig könnte ein andere Akteur auch die Haushalts-Ontologie nutzen, um im Haushalt eingesetzt werden zu können.

Für die Erstellung der Ontologien sollen verschiedenen Regeln befolgt werden, um den Ontologien eine einheitliche Struktur zu geben. Grundsätzlich sollen zwischen allen erstellten Ontologien gemeinsame Grundkonzepte und Verbindungen abgesprochen werden, damit diese auf die gleiche Weise definiert werden können. So ist eine Grundregel, dass sich Entitäten, die sich entsprechen, in den verschiedenen Ontologien jeweils eine gleiche Benennung erhalten sollen. Damit kann das Verschmelzen von Ontologien vereinfacht werden, da dies die Erkennung von sich entsprechenden Entitäten vereinfacht.

Bei der Erstellung von Akteuren sollen zusätzlich einige Grundeigenschaften gemeinsam definiert werden. Dadurch kann sichergestellt werden, dass Akteure ausgetauscht werden können. Die möglichen Eigenschaften der Objekte und Daten, wie beispielsweise *hasMethod* oder *hasState*, sollen daher in allen Akteur-Ontologien vorkommen. Ebenso müssen auch die grundlegenden Klassen wie *Method*, *State* oder *DataType* geteilt werden. Entsprechend sollte ein Konsens zwischen den Akteuren gebildet werden, welche Eigenschaften und Klassen grundlegend existieren müssen. Dies soll vereinfacht und vereinheitlicht werden, indem eine Basisontologie für Akteur-Ontologien erstellt wird. In dieser sind alle bereits existierenden Eigenschaften sowie grundlegenden Klassen vorhanden, sodass für neue Akteure

Tabelle 5.4.: Struktur der Basisontologie für Akteure

Konzept	Beschreibung
Thing	Oberkonzept der Ontologie
DataType	Datentypen
Method	ausführbare Aktionen
Parameter	Parameter-Arten d. Methoden
State	auftretende Zustände
System	ansprechbares System

Tabelle 5.5.: Struktur der Basisontologie für Umgebungen

Konzept	Beschreibung
Thing	Oberkonzept der Ontologie
Object	auftretende Objekte
State	auftretende Zustände

bestenfalls nur noch die Instanzen hinzugefügt werden müssen. Die Struktur der Basisontologie ist in Tabelle 5.4 zu sehen. Darin sind die bereits erwähnten Klassen *Method*, *State* oder *DataType* enthalten. Zusätzlich existieren die Klasse *Parameter*, in der die Parameter der Methoden definiert werden sollen, und die Klasse *System*. In *System* soll der Akteur beziehungsweise das System an sich definiert werden.

Für die Erstellung der Ontologien für die Umgebungen, in denen die Systeme agieren sollen, kann ähnlich verfahren werden. Für die Umgebungsontologien muss sich ebenfalls auf Grundeigenschaften und gemeinsame Grundklassen sowie Strukturen geeinigt werden. Dabei kann es zu Überschneidungen von konzeptuellen Eigenschaften und Strukturen der Akteur-Ontologien kommen. Beispielsweise ist die Modellierung von Zuständen sowohl bei den Akteur-Ontologien als auch den Umgebungsontologien sinnvoll. Damit diese Überschneidungen bei der späteren Verschmelzung leichter zu handhaben sind, soll in solchen Fällen eine gleiche Struktur und Benennung der Klassen oder Eigenschaften geschehen. In dem Beispielfall der Zustände würde man daher die Klasse *State* bei den Akteur- als auch den Umgebungsontologien gleichermaßen erstellen. Ebenso wie für die Akteur-Ontologien ist es auch für die Umgebungsontologien sinnvoll, eine Basisontologie zu erstellen, welche in Tabelle 5.5 zu sehen ist. Die Klasse *Object* stellt darin das Oberkonzept für auftretende Objekte dar. In konkreten Umgebungsontologien sollen Unterklassen der *Object*-Klasse erstellt werden, die Schnittstellen entsprechen. Dadurch sollen Eigenschaften von Objekten abgebildet werden, vor allem im Bezug auf mögliche Methodenparameter beziehungsweise Datentypen aus den Akteur-Ontologien. Ein Beispiel dafür wäre die Schnittstelle *Grabbable* für greifbare Objekte. Hier existiert die Möglichkeit, in der Basisontologie der Umgebungen bereits alle Schnittstellen zu modellieren, die existieren sollen. Die könnte die Abstimmung mit bereits existierenden Umgebungsontologien für neue Ontologien vereinfachen. Gleichzeitig wird dadurch aber die Basisontologie aufgebläht und die Übersichtlichkeit verringert, was die Handhabung wiederum erschwert und eventuell die Vorteile von kleineren Ontologien zunichte macht. Aus diesem Grund fiel die Entscheidung gegen diese Option.

Vorteil des Vorgehens mit Basisontologien ist, dass auf einfache Art und Weise einheitliche und übersichtliche Ontologien erstellt werden können. Da das Grundgerüst für die Ontologien bereits existiert, kann dadurch die Erstellung neuer Ontologien beschleunigt werden. Ein Nachteil ist, dass bei Änderungen wie dem Hinzufügen neuer Eigenschaften alle Ontologien entsprechend geändert werden sollten. Die Änderungen sind nicht zwangsweise notwendig, wenn das System oder die Umgebung die Eigenschaften nicht nutzen, für den Erhalt der Einheitlichkeit werden die Änderungen jedoch empfohlen.

Zuletzt ist es notwendig, dass die Titel passender Wikipedia-Artikel zu den Instanzen an-

notiert werden, damit die Extraktion der Themen aus der Ontologie, wie sie später in Abschnitt 5.2.2 beschrieben wird, angewandt werden kann. So soll beispielsweise für die Instanz *Table* in der Ontologie der Artikel-Titel *Table (furniture)* annotiert werden. Dafür werden Kommentare für die konkreten Instanzen in der Sprache „wiki“ hinzugefügt, damit die Wikipedia-Titel einfach über die Sprache aus allen Kommentaren gefiltert werden können. Bei der Annotation sollen zu möglichst vielen Instanzen passende Wikipedia-Artikel annotiert werden, wobei voraussichtlich nicht für alle möglichen Instanzen passende Artikel in der Wikipedia existieren. Bei der Annotation ist wichtig, dass die Artikel möglichst präzise zu den Bedeutungen der Instanzen passt, damit später durch eine ungenaue Annotation keine Fehlinterpretationen entstehen. So gibt es zum Beispiel einen thematischen Unterschied zwischen *Water* und *Drinking water*, der beachtet werden sollte.

5.2.2. Entwurf der Ontologieauswahl

Mehrere kleine domänenspezifische Ontologien bringen Vorteile gegenüber einer großen allgemeinen Ontologie, wie bereits die Analyse in Abschnitt 5.2 ergab. Allerdings muss beim Einsatz von mehreren Ontologien eine Auswahl von zur Eingabe passenden Ontologien geschehen. Mit Hilfe der mit der Themenextraktion gefundenen Themen kann die Auswahl von einer oder mehreren passenden Ontologien umgesetzt werden. Domänenspezifischen Ontologien besitzen, genau wie die Eingaben, ebenfalls Themen. Die Themen der Ontologien können genutzt werden, um sie mit den Themen der Eingabetexte zu vergleichen und so die passenden Ontologien zu ermitteln.

In dieser Arbeit soll dafür zunächst eine automatische Extraktion der Themen aus der Ontologie stattfinden. Die Extraktion von Themen für die Ontologien unterscheidet sich von der Extraktion von Themen aus den Anweisungen lediglich darin, wie die Liste an Bedeutungen erstellt wird, für die gemeinsame Themen ermittelt werden sollen. Für eine Ontologie sollen dafür alle annotierten Identifikatoren der Wikipedia-Artikel, die nach den Regeln zur Erstellung von Ontologien in Abschnitt 5.2.1 erstellt werden sollten, extrahiert und als Eingabe für die Themenextraktion verwendet werden. Als Resultat erhält man analog zur Themenextraktion für Anweisungen eine Anzahl von Themen für die Ontologie.

Danach können die Themen aus den Anweisungen und die Themen aus den Ontologien verglichen werden. Für den Vergleich der Themen soll pro Thema der Anweisung ein Wert bestimmt werden, der die Übereinstimmung mit den Themen der Ontologie repräsentiert. Dabei soll das Thema aus der Anweisung mit den Themen der Ontologie verglichen und ein Wert für die Ähnlichkeit des Themas mit den Themen der Ontologie ermittelt werden. Nachdem für alle Themen die Ähnlichkeit berechnet wurde, soll davon das Mittel bestimmt werden. Dieses Mittel stellt dann die Übereinstimmung zwischen den Themen der Eingabe und einer Ontologie dar.

$$Sim(t, O) = \begin{cases} 0 & , t \notin \text{Themengraph} \\ \frac{1}{min_{o \in O}(dist(t,o))+1} & , t \in \text{Themengraph} \end{cases} \tag{5.2}$$

Die Berechnung der Ähnlichkeit zweier Themen soll mit der Formel aus Gleichung 5.2 stattfinden. t steht darin für ein Thema der Anweisung, O für die Menge der Ontologiethemen. Für ein Thema der Eingabe ist die Ähnlichkeit von der Länge des kürzesten Pfades zum nächstliegenden Thema der Ontologie abhängig. Die Distanz eines Knoten zu sich selbst ist hier Null, wodurch eine Ähnlichkeit von Eins entsteht, sollte das Thema der Eingabe einem Thema der Ontologie entsprechen. Je weiter das Thema vom nächsten Thema der Ontologie entfernt ist, desto kleiner wird die Ähnlichkeit. Existiert für das Thema kein Knoten im Themengraphen der Ontologie, dann wird die Ähnlichkeit entsprechend mit Null bewertet. Die Übereinstimmung $C(T_E, O)$ der Themen T_E einer Eingabe mit den

Ontologiethemen O berechnet sich dann über den Mittelwert der Werte für die Ähnlichkeit mit der Formel aus Gleichung 5.3.

$$C(T_E, O) = \frac{\sum_{t \in T_E} Sim(t, O)}{|T_E|} \tag{5.3}$$

Nach der Ermittlung der Übereinstimmungen zu allen Ontologien kann die Auswahl der Ontologien stattfinden. Hier gibt es mehrere Optionen, wie die Ontologien ausgewählt werden sollen. Die erste Option ist eine Auswahl der jeweiligen Ontologie, die den besten Übereinstimmungswert hat. Vorteil dieses Ansatzes ist die Einfachheit, wobei gleichzeitig sichergestellt wird, dass in jedem Fall eine Ontologie ausgewählt wird. Der offensichtliche Nachteil ist jedoch, dass nur eine Ontologie ausgewählt wird. Sollte eine andere Ontologie einen ähnlichen Übereinstimmungswert besitzen, dann könnte man argumentieren, dass diese Ontologie ebenfalls ausgewählt werden sollte. Die wäre mit diesem Ansatz jedoch nicht möglich, lässt sich allerdings mit der zweiten Option lösen. Die zweite Option ist die Auswahl derjenigen Ontologien, deren Übereinstimmungswert einen vorher festgelegten Schwellwert übersteigen. Dadurch kann vorher festgelegt werden, wie gut die Themen zu einer Ontologie mindestens passen müssen und es können mehr als eine Ontologie ausgewählt werden. Nachteil daran ist jedoch, dass unter Umständen keine Ontologie ausgewählt wird. Dies führt jedoch dazu, dass der Zweck der Ontologieauswahl, mindestens eine passende Ontologie auszuwählen, damit die Information aus der Ontologie genutzt werden können, verfehlt wird. Daher ist die dritte Option eine Mischung aus den ersten beiden Optionen. Bei dieser Option soll zuerst der beste Übereinstimmungswert ermittelt werden. Der beste Übereinstimmungswert wird genutzt, um mit ihm in Verbindung mit einem *Schwellwertfaktor* den Schwellwert zu bestimmen. Der Schwellwertfaktor ist die Höhe des Anteils, der vom besten Übereinstimmungswert abgezogen werden soll. Bei einem Schwellwertfaktor von 0,1 und dem besten Übereinstimmungswert von 0,5 erhält man so den Schwellwert von 0,45. Mit dem so festgelegten Schwellwert können dann die Ontologien wie in Option zwei ermittelt werden. Dies bringt den Vorteil, dass so auf jeden Fall mindestens eine Ontologie ausgewählt wird. Gleichzeitig wird ermöglicht, dass weitere, ähnlich gute Ontologien der Auswahl hinzugefügt werden können. Durch den anteilig ermittelten Abzug vom besten Übereinstimmungswert wird der Schwellwert außerdem flexibel festgelegt. Damit kann gesteuert werden, wie übereinstimmend andere Ontologien im Vergleich zur passendsten Ontologie sein müssen, damit diese noch ausgewählt werden. Da diese Option die Vorteile der beiden anderen Optionen vereint und die Nachteile ausgleicht, soll diese Option gewählt werden. Dabei soll untersucht werden, wie sich Änderungen des Schwellwertfaktors auswirken.

Zuletzt muss sichergestellt werden, dass sowohl mindestens eine Umgebungs- als auch eine Akteur-Ontologie ausgewählt wird. Das ist notwendig, damit sowohl ein System als auch das Einsatzgebiet des Systems tatsächlich angesprochen und Informationen abgerufen werden können. Dies lässt sich bewerkstelligen, indem der obige Ansatz auf Akteur- und Umgebungsontologien getrennt angewandt wird.

Der Ansatz könnte jedoch bei der Auswahl von Akteuren nicht zielbringend sein. Ein Problem liegt darin, dass Akteure oftmals nicht direkt angesprochen werden, weshalb auch keine Themen dazu existieren. Dadurch ist es schwer über die Thematik des Satzes alleine den Akteur zu ermitteln. Daher soll der Ansatz für Akteure angepasst werden: Zusätzlich soll ermittelt werden, wie gut die Akteure aus den Akteur-Ontologien jeweils mit den ausgewählten Umgebungen umgehen können. So ist beispielsweise ein virtueller Assistent im Garten wenig sinnvoll, da dieser dort mit keinem Objekt interagieren kann. Dies setzt voraus, dass zuerst die Umgebungsontologien ermittelt werden. Danach können aus den ausgewählten Umgebungsontologien die Objekt-Klassen extrahiert werden. Die Objekt-Klassen sind die in Abschnitt 5.2.1 erwähnten Unterklassen, die die Eigenschaften der

Objekte festlegen. Akteure sollten mit möglichst vielen, am besten mit allen, Objekt-Klassen umgehen können. Ein Akteur kann dabei mit einer Objekt-Klassen umgehen, wenn er mit einem Datentyp umgehen kann, der der Objekt-Klasse entspricht. Deshalb soll der Anteil an Objekt-Klassen der Umgebung ermittelt werden, mit denen der Akteur umgehen kann. Dieser Anteil soll dann mit den Werten für die Themenübereinstimmung verrechnet werden, indem ein gewichtetes Mittel berechnet wird. Da eine direkte Übereinstimmung über die Themen aussagekräftiger ist, soll diese wie in Gleichung 5.4 doppelt so stark gewichtet werden.

$$C_A(T_E, O_A, A, U) = \frac{2 * C(T_E, O_A) + \text{Handling}(A, U)}{3} \tag{5.4}$$

A steht für die Akteur-Ontologie, O_A entsprechend für die Themenmenge der Akteur-Ontologie. Handling(A, U) ist der Anteil an Objekt-Klassen der Umgebung U, mit denen der Akteur umgehen kann. Mit diesem so ermittelten Übereinstimmungswert für Akteure C_A kann die Auswahl der passenden Akteur-Ontologien dann wie zuvor beschrieben fortgesetzt werden.

Bei diesem Ansatz werden jedoch nicht zwangsweise nur die passenden Ontologien ausgewählt. Es können durchaus Ontologien ausgewählt werden, die zwar grob etwas mit der Eingabe zu tun haben, aber nicht optimal passen. Bei der Auswahl der Ontologien ist jedoch vor allem wichtig, dass alle Ontologien ausgewählt werden, die benötigt werden, damit die benötigten Informationen aus der Ontologie vorhanden sind. Die Ausbeute ist dementsprechend wichtiger als die Präzision. Allerdings erhöht man dabei auch die Gefahr, gar nicht benötigte Ontologien auszuwählen, durch die möglicherweise Fehler entstehen. Die Möglichkeit, alles abbilden zu können, ist jedoch höher zu bewerten als die Chance auf Fehler durch die Auswahl falscher Ontologien.

5.2.3. Entwurf der Verschmelzung von Ontologien

Zuletzt müssen die ausgewählten Ontologien verschmolzen werden. Wenn die Ontologien nach den in Abschnitt 5.2.1 genannten Regeln erstellt wurden, dann kann ein einfacher Ansatz zur Verschmelzung eingesetzt werden. Ein einfacher Verschmelzungsansatz ist das Abbilden von Entitäten mit gleichen Namen aufeinander. Dafür werden alle Entitäten aus den Ontologien in eine neue Ontologie übertragen. Gleich benannte Entitäten werden dabei zusammengenommen. Dies funktioniert gut, wenn gleiche Namen verwendet wurden und alle Regeln zur Erstellung von Ontologien eingehalten wurden. Allerdings macht dies den Ansatz weniger allgemein einsetzbar. Dies liegt vor allem daran, dass Entitäten mit gleichen Namen als gleich behandelt werden, obwohl sie unter Umständen gar nicht gleich sind. Hier muss festgelegt werden, ob gleichnamige Entitäten mit unterschiedlichen Eigenschaften wirklich gleich sind und aufeinander abgebildet werden sollten. Hier kann man einerseits argumentieren, dass in anderen Domänen andere Eigenschaften der selben Entität existieren und diese beim Verschmelzen der Domänen entsprechend auch zusammengenommen werden sollen. Andererseits kann es auch sein, dass trotz des gleichen Namens etwas anderes gemeint ist. Beispielsweise können zusammenklappbare Gartentische und ausziehbare Esstische in den jeweiligen Ontologien Garten und Küche jeweils Tisch heißen, obwohl leicht unterschiedliche Objekte gemeint sind. Durch eine Kombination der beiden Tisch-Arten würde man folglicherweise Fehler produzieren. Zusätzlich zu diesem Problem existiert das Problem, dass semantisch gleiche Entitäten in den Ontologien unter Umständen andere Namen besitzen. Dies passiert beispielsweise, wenn Synonyme oder andere Schreibweisen von Wörtern verwendet werden. Dadurch werden diese Entitäten nicht aufeinander abgebildet. Alternativ könnte ein Ansatz umgesetzt werden, bei dem zuerst eine Ontologieangleichung oder eine Ontologieabbildung erstellt wird, wobei diese semantische Informationen und ähnliches nutzt, um gleiche Entitäten zu erkennen.

Tabelle 5.6.: Struktur der verschmolzenen Basisontologien

Konzept	**Beschreibung**
Thing	Oberkonzept der Ontologie
DataType	Datentypen
Method	ausführbare Aktionen
Parameter	Parameter-Arten d. Methoden
State	auftretende Zustände
System	ansprechbares System
Object	auftretende Objekte

Damit könnte dann die verschmolzene Ontologie erstellt werden. Dafür müsste ein geeigneter Ansatz für die Ontologieangleichung oder die Ontologieabbildung gewählt werden. Da kleine Ontologien angeglichen werden sollen, die voraussichtlich relativ ähnlich im Aufbau sind, eignen sich alle in Abschnitt 4.2 vorgestellten Ansätze. Leider steht für keine der genannten Ansätzen eine zugängliche Implementierung zur Verfügung. Das Implementieren dieser Ansätze ist jedoch sehr zeitintensiv und bei vielen Ansätzen fehlen einige Detailinformationen, damit eine Implementierung problemlos nachvollzogen werden kann. Da beim Einhalten der Regeln zu Erstellung von Ontologien auch der zuvor erwähnte einfache Ansatz eingesetzt werden kann, soll dieser zum Verschmelzen der Ontologien umgesetzt werden. Dabei soll jedoch eine Schnittstelle erstellt werden, die einen einfachen Austausch des Ansatzes ermöglicht.

Der einfache Ansatz zum Verschmelzen soll die Entitäten aus den Ontologien in einer neuen Ontologie kombinieren. Dabei sollen Entitäten wie beispielsweise Objekte, die den gleichen Bezeichner haben, miteinander verschmolzen werden. Dies bedeutet, dass die Eigenschaften der beiden gleichnamigen Entitäten kombiniert werden. Dadurch sollen auch Strukturen wie Unterkonzepte übernommen werden. Betrachtet man die Basisontologien aus Tabelle 5.4 und Tabelle 5.5 und verschmelzt diese, so soll als Resultat die Struktur aus Tabelle 5.6 entstehen. Dort wurden beispielsweise die Klasse *State* aus den jeweiligen Ontologien verschmolzen.

Zusätzlich zu dem einfachen Verschmelzen der Ontologien soll noch eine zusätzliche Verbindungsart hinzugefügt werden. Diese neue Verbindung soll Datentypen mit Objekt-Instanzen verbinden, deren Klassen den gleichen Namen wie der Datentyp besitzt. Zum Beispiel existiert in einer konkreten Akteur-Ontologien der Datentyp *Grabbable*. In einer Objekt-Ontologie, mit der die Akteur-Ontologie verschmolzen werden soll, existiert als Unterklasse von *Object* die Klasse *Grabbable*, die greifbare Objekte kennzeichnet. So soll in diesem Beispiel die Verbindungen vom Datentyp *Grabbable* zu allen Instanzen der Objekt-Klasse *Grabbable* erstellt werden. Dies soll eine Verbindung des Systems mit der Umgebung herstellen, damit einfach festgestellt werden kann, mit welchen Objekten der Umgebung das System umgehen kann. Diese Information ist vor allem wichtig, um feststellen zu können, ob ein Objekt als Parameter einer Methode fungieren kann.

Nachdem die Ontologien zusammengenommen und die neuen Verbindungen hinzugefügt wurden, soll diese gespeichert sowie an den PARSE-Graphen annotiert werden können.

6. Implementierung

Im Folgenden wird die Implementierung der in Kapitel 5 beschriebenen Entwürfe vorgestellt. Um diese in PARSE einsetzen zu können, müssen zusätzlich Agenten für die verschiedenen Aufgaben implementiert werden. Da eine Auflösung von mehrdeutigen Wörtern für die Themenextraktion benötigt wird, soll außerdem zunächst ein Ansatz implementiert werden, mit dem die Bedeutung eines Nomens aufgelöst und annotiert werden kann. Die verschiedenen Schritte der Implementierung dazu werden in Abschnitt 6.1 beschrieben. Mit den annotierten Bedeutungen kann die Extraktionen der Themen angegangen werden, dessen Implementierung in Abschnitt 6.2 erläutert wird. Zuletzt wird sich in Abschnitt 6.3 der Implementierung der Auswahl der passenden Ontologien mit Hilfe der extrahierten Themen sowie der Verschmelzung der ausgewählten Ontologien gewidmet.

6.1. Auflösung von Mehrdeutigkeiten

Um einen Klassifikator nach dem Vorbild des Ansatzes von Mihalcea [Mih07] zu erstellen, der in der Lage ist, mehrdeutige Wörter aufzulösen, werden verschiedene Schritte benötigt. Diese Schritte sind die Beschaffung und Vorverarbeitung des Trainingskorpus, die Extraktion von Trainingsinstanzen aus dem Korpus, die Vorverarbeitung der Trainingsinstanzen und letztlich das Training des Klassifikators. Die Schritte bringen verschiedene Herausforderungen mit sich, die gelöst werden müssen. Die verschiedenen Schritte, die Herausforderungen sowie die Lösungen dafür sind in Abschnitt 6.1.1 näher beschrieben.

Mit dem trainierten Klassifikator können daraufhin mehrdeutige Nomen aufgelöst und dabei dem Wikipedia-Artikel zugeordnet werden, der sich mit dem Nomen in der aufgelösten Bedeutung beschäftigt. Der Prozess der Auflösung sowie die Speicherung der Bedeutung von Nomen im Graphen von PARSE ist in Abschnitt 6.1.2 beschrieben.

6.1.1. Training des Klassifikators

Der erste Schritt zur Erstellung der Trainingsinstanzen ist die Beschaffung des Trainingskorpus. Dafür werden die Texte aller Wikipedia-Artikel in ihrer Quelltextform benötigt. DBpedia legt regelmäßig eine Kopie aller Wikipedia-Artikel in Form von XML-Dateien an und bietet die letzten fünf Kopien zum Herunterladen an. Darin sind die Quelltexte der Artikel mitsamt einiger Metainformationen enthalten. Für die Verarbeitung sind jedoch lediglich die Quelltexte der Artikel interessant, weshalb diese zuerst extrahiert werden müssen.

Nach der Extraktion der Artikel werden diese von Elementen bereinigt, die für den Ansatz nicht hilfreich sind und nicht benötigt werden. Zum einen sind die zu entfernenden Elemente nicht hilfreich, wenn sie keine vollständigen Sätze beinhalten, in denen Querverweise eingebetten sind. Diese werden jedoch benötigt, um das mehrdeutige Wort und seine Bedeutung aus dem Querverweis und den Kontext des mehrdeutigen Wortes aus dem Satz extrahieren zu können. Zum anderen sind Elemente entfernt werden, die nicht in einer Form vorliegen, in der sie mit üblichen Sprachverarbeitungswerkzeugen verarbeitet werden können. Ein Beispiel für solche Elemente sind die Informationsboxen am Anfang eines Artikels. Durch die Struktur und den Aufbau der Informationsboxen können Sprachverarbeitungswerkzeuge nicht eingesetzt werden. Gleichzeitig befinden in ihnen sich üblicherweise keine in Sätze eingebundenen Querverweise. Da der Ansatz jedoch auf diese setzt, um damit die Bedeutung auch anhand der Umgebung des mehrdeutigen Wortes zu bestimmen, bringen die Informationsboxen für die Auflösung von Mehrdeutigkeiten keinen Mehrwert. Das gleiche Problem existiert bei den meisten Aufzählungslisten, bei Zitaten und bei Überschriften. Bei Aufzählungslisten kommen oftmals nur Stichworte oder unvollständige Sätze vor und bei Zitaten und Überschriften sind normalerweise keine Querverweise eingebaut. Daher werden diese in diesem Schritt ebenso entfernt.

Nach der ersten Filterung wird der Text in einem zweiten Filterschritt von weiteren Besonderheiten befreit, die die Formatierung des Textes beeinflussen. Dabei werden Markierungen im Quelltext entfernt, die beispielsweise kursiven Text bewirken. Zuletzt wird der Text von phonetischen Informationen zur Aussprache von Wörtern sowie Querverweisen, die zu Webseiten außerhalb der Wikipedia führen, befreit. Danach besteht der ursprüngliche Quelltext nur noch aus dem reinen Text der Artikel, in dem ausschließlich die Markierungen für Querverweise auf andere Wikipedia-Artikel eingebettet sind.

Die gefilterten Artikel werden im Anschluss gespeichert, damit dieser Schritt nur bei Bedarf ausgeführt werden muss. Dabei wird jeder Artikel in einer eigenen Datei gespeichert. Hierbei muss jedoch beachtet werden, dass Dateisysteme Probleme haben können, wenn sich eine große Anzahl an Dateien in einem einzigen Verzeichnis befinden. Dies führt dazu, dass die Speicherung selbst kleiner Dateien sehr viel Zeit in Anspruch nimmt, wodurch das Programm erheblich ausbremst wird. Bei ersten Durchläufen benötigte das Speichern der Dateien fast die gesamte Rechenzeit, da die Dateizugriffe nach einer gewissen Anzahl an Dateien im Ausgabeverzeichnis länger brauchten und damit das Speichern erheblich beeinflussten. Deshalb wurde eine Option eingebaut, mit der die Artikel nicht direkt in einem einzigen Ausgabeverzeichnis gespeichert werden, sondern das Ausgabeverzeichnis in weitere Verzeichnisse unterteilt wird, in denen die Artikel aufgeteilt gespeichert werden. Dadurch befindet sich pro Verzeichnis eine eher geringe Anzahl an Dateien, wodurch das Dateisystem keine langen Zugriffszeiten mehr benötigt.

Nach der Vorverarbeitung können die Texte weiter verarbeitet werden, um aus ihnen die Informationen zu extrahieren, mit denen die Trainingsinstanzen kreiert werden können. Ein eigenes Modul ist für die Erzeugung der Trainingsinstanzen aus den gefilterten Texten zuständig. Die vorverarbeiteten Texte werden nacheinander geladen und der Verarbeitung übergeben. In der Verarbeitung wird die Bibliothek *Stanford CoreNLP* [MSB+14] eingesetzt, um die Wortarten der einzelnen Wörter sowie ihre Grundformen zu erhalten. Ebenso können damit Satzgrenzen erkannt werden, was die Trennung der Sätze voneinander erlaubt. Dies ist hilfreich, da die einzelnen Sätze unabhängig voneinander verarbeitet werden sollen.

Um die Instanzen erstellen zu können, werden die Texte zuerst auf Querverweise untersucht. Werden Querverweise gefunden, dann wird die Position und der Inhalt des Querverweis gespeichert, bevor der Querverweis auf seine Fließtextform reduziert wird, damit lediglich reiner Text ohne Sonderzeichen oder ähnliches übrig bleibt. Die reine Fließtext-

form wird gebraucht, um einen möglichst guten und fehlerfreien Einsatz von CoreNLP zu ermöglichen. Nachdem der Text von CoreNLP mit den benötigten Informationen zu Wortarten und Grundformen der einzelnen Wörter annotiert wurde, wird für jeden ursprünglich gefundenen Querverweis eine Trainingsinstanz erstellt, wobei auch die Umgebung des Querverweises betrachtet wird. Eine Trainingsinstanz besteht aus den 19 Merkmalen, die im vorigen Kapitel in Abschnitt 5.1.1 in Tabelle 5.1 aufgeführt wurden. Zu den Attributen gehören die Wörter mitsamt ihren Wortarten, die sich im Satz drei Positionen links sowie rechts des mehrdeutigen Wortes befinden, wobei in dieser Arbeit Stoppwörter übergangen werden sollen. Stoppwörter bringen keine nützlichen Informationen über den Kontext des mehrdeutigen Wortes, weshalb diese übergangen werden. In [Mih07] findet die Filterung von Stoppwörtern nicht statt, Tests ergaben jedoch, dass sich dadurch die Genauigkeit des Klassifikators verbessert. Die Erkennung von Stoppwörtern findet über einen Vergleich mit einer Stoppwort-Liste statt, die aus dem Programm zur statistischen Sprachmodellierung und Dokumentenklassifikation *Rainbow* [McC96] extrahiert wurde. Die weiteren Attribute sind jeweils das erste Nomen sowie Verb, das sich links beziehungsweise rechts des mehrdeutigen Wortes befindet. Wörter als Attributwerte werden jeweils in ihre Grundform umgewandelt und dann erst als Attributwerte genutzt. Das heißt, dass beispielsweise statt der Pluralform „oranges“ die Grundform „orange“ und statt der Vergangenheitsform „went“ die aktive Infinitivform im Präsens „go“ genutzt wird. Damit soll vermieden werden, dass unterschiedliche Formen eines Wortes als verschiedene Wörter behandelt werden. Die unterschiedliche Formen und Einsatzmöglichkeiten eines Wortes, beispielsweise ob ein Nomen im Singular oder Plural oder ein Verb in einer bestimmten Zeitform genutzt wurde, werden bereits über die Wortarten einbezogen.

In dieser Arbeit zur Auflösung der Mehrdeutigkeiten werden Eigennamen ausgeschlossen, wie es auch in [Mih07] geschieht. Eigennamen sollten nicht über die Auflösung von mehrdeutigen Wörtern sondern über einen eigenständigen Prozess erkannt werden. Um Eigennamen für die Verarbeitung ausschließen zu können, gibt es zwei verschiedene Methoden. Die erste Methode ist das Ausschließen von Wörtern, die groß geschrieben werden. Da am Satzanfang kann nicht bestimmt werden, ob das Wort ein Eigenname ist oder nur durch den Satzanfang groß geschrieben wird, werden deshalb auch alle Nomen am Satzanfang ausgeschlossen. Dies Methode ist sehr simpel, verursacht jedoch Fehler, da so Wörter am Satzanfang fälschlicherweise aussortiert werden können. Die zweite Methode ist die Nutzung des Moduls zur Erkennung von Eigennamen von CoreNLP. Dieses verspricht eine höhere Zuverlässigkeit, es werden jedoch nicht immer alle Eigennamen erkannt. Diese Methode hat außerdem den großen Nachteil, dass diese Erkennung sehr zeitintensiv ist. In [Mih07] wird die erste Option verwendet, daher soll in dieser Arbeit ebenfalls die erste Option verwendet werden.

Die Trainingsinstanzen sollen nach der Erzeugung gespeichert werden, um sie bei Bedarf erneut einsetzen zu können, ohne den vorigen Prozess erneut durchlaufen zu müssen. Wenn die Trainingsinstanzen bis zur Speicherung vom Programm im Arbeitsspeicher gehalten werden, können allerdings bei Rechnern mit wenig Arbeitsspeicher Probleme auftreten. Deshalb wurde eine Option eingebaut, mit der die aktuell im Speicher gehaltenen Trainingsinstanzen nach einer festgelegten Anzahl an bearbeiteten Artikeln auf die Festplatte geschrieben und aus dem Speicher entfernt werden können. Dadurch erhält man anstatt einer großen Datei, in der alle Trainingsinstanzen enthalten sind, mehrere kleinere Dateien.

Die Erstellung der Trainingsinstanzen findet parallel statt. Die Erstellung ist größtenteils trivial parallelisierbar, da keinerlei Abhängigkeiten zwischen den Instanzen existieren. Lediglich das Hinzufügen der Instanzen zur Liste der Instanzen muss synchronisiert werden, damit keine Fehler durch Wettlaufsituationen auftreten. Die Parallelisierung findet dabei auf Abschnittsebene statt: Der Text wird zeilenweise eingelesen, wobei eine neue Zeile des Textes jeweils ein Abschnitt des Textes ist. Die Aufgabe, die eingelesene Zeile zu verar-

beiten, wird einer Warteschlange hinzugefügt. Untätige Arbeiter können diese Aufgaben jeweils aus der Warteschlange entnehmen und mit der Verarbeitung beginnen. Bei der Verarbeitung werden zuerst mit Hilfe von CoreNLP die Satzgrenzen erkannt und die Sätze dann nacheinander verarbeitet, um Trainingsinstanzen zu erstellen. Nachdem alle Zeilen aus einer Datei gelesen wurden, wird mit dem Einlesen des nächsten Textes fortgesetzt. Das Einlesen wird lediglich pausiert, wenn die vorher erwähnte Zwischenspeicherung von Instanzen aktiviert ist und die festgelegte Menge an Artikeln erreicht wurde. In diesem Fall wird pausiert, bis die Warteschlange abgearbeitet wurde und die Instanzen gespeichert wurden. Nach der Speicherung wird der gesamte Prozess fortgesetzt, bis alle Texte verarbeitet sind. Das Resultat der Erstellung der Trainingsinstanzen sind über fünf Millionen Trainingsinstanzen mit 283.173 verschiedenen Bedeutungen.

Mit den erstellten Trainingsinstanzen kann das Training des Klassifikators angegangen werden. Die Trainingsinstanzen müssen vor dem eigentlichen Training jedoch weiter vorbereitet werden. Dazu werden auf ihnen Filter angewandt. Der wichtigste Filter ist der sogenannte *StringToNominal*-Filter. Dieser Filter extrahiert die verschiedenen Zeichenketten aus den Attributwerten und erstellt pro Attribut eine Liste mit auftretenden Attributwerten. Dies wird benötigt, da zu jedem Wert in der Liste in der Lernphase eine Auftrittswahrscheinlichkeit zugeordnet und entsprechend für die Klassifikation abgefragt werden muss. Desweiteren wurde zusätzlich ein Filter implementiert, der Bedeutungen herausfiltern soll, für die es nur eine einzige Trainingsinstanz gibt. Eine einzige Trainingsinstanz ist nicht ausreichend, um tatsächlich Aussagen über die Bedeutung treffen zu können. Die Notwendigkeit dieses Filters ist diskutierbar, allerdings verbessert er die Ergebnisse in der internen Validierung, denn vor allem bei Tests wie der Kreuzvalidierung sorgt der Filter für bessere Ergebnisse. Bei diesen Tests werden Instanzen aus der Menge der Trainingsinstanzen für das Training zurückgehalten, um die Güte des Klassifikators auf unbekannten Instanzen überprüfen zu können. Wenn zu einer Bedeutung nur eine Instanz existiert, dann kann diese ausschließlich in der Trainingsmenge oder in der Testmenge sein. Ist eine solche Instanz in der Testmenge, dann entsteht automatisch eine Fehlklassifikation, da die Bedeutung dem Klassifikator unbekannt ist. Etwa 3% der Instanzen in der Menge der Trainingsinstanzen, die aus den Wikipedia-Texten kreiert wurden, sind für ihre Bedeutung die einzigen Beispiele. Der Filter bringt allerdings einen hohen Aufwand mit sich, wodurch das Training entsprechend mehr Zeit in Anspruch nimmt. Die Auswirkungen für beispielsweise die Kreuzvalidierung sind vorhersehbar, die Filterung muss aber mehrfach durchgeführt werden, wodurch die Evaluation entsprechend länger dauert. Deshalb und da die Notwendigkeit diskutierbar ist, ist der Einsatz des Filters optional.

Nach dem Einsatz der Filter kann das eigentliche Training des Klassifikators beginnen. Für den Klassifikator war geplant, eine Implementierung des Naive-Bayes-Klassifikators aus der Bibliothek Weka zu nutzen. Diese Implementierung weißt jedoch Schwächen auf, die eine Benutzung in diesem Ansatz nahezu unmöglich machen. Die größte Schwäche von Weka liegt in den zweidimensionalen Feldern, die für jede Klasse des Klassifikators, also den Bedeutungen der Mehrdeutigkeiten, angelegt werden. Jeder der Werte in einem solchen zweidimensionalen Feld repräsentiert einen Zählwert, der für ein bestimmtes Attribut die Anzahl an Vorkommen für einen bestimmten Attributwert darstellt. Bei der Erstellung der Felder werden diese mit Initialwerten belegt. Diese Art der Implementierung bringt jedoch einen sehr hohen Speicherbedarf mit sich. Vor allem bei Trainingskorpora mit vielen Klassen, in denen pro Attribut viele Attributwerten vorkommen, wird sehr viel Speicher benötigt. Im Anwendungsfall der Auflösung von Mehrdeutigkeiten tritt dieser Fall auf. Dabei treten gleichzeitig viele Attributwerte für eine große Anzahl an Klassen überhaupt nicht auf. Daher ist das zweidimensionale Feld nur sehr spärlich mit anderen Werten als dem Initialwert belegt. Dennoch wird für diese Werte Speicher belegt, was zu einem extremen Speicherverbrauch führt. Eine grobe Abschätzung anhand der vorhan-

denen Trainingsmenge, zu der sich Informationen im Appendix in Tabelle A.1 befinden, ergab einen benötigten Speicheraufwand im Bereich mehrerer Terabyte, was nicht praktikabel ist und weshalb die Implementierung angepasst werden musste. In der angepassten Implementierung werden nur Zählwerte für diejenigen Attributwerte gespeichert, für die der Zählwert vom Initialwert abweicht, da die Attributwerte mindestens einmal für die Klasse und das Attribut auftreten. Dafür wird eine Abbildung erstellt, die Attributwerte auf die dazugehörigen Zählwerte abbildet. Eine Abbildung eines Attributwertes auf einen Zählwert wird dabei erst erstellt, wenn für einen Attributwert der Zählwert zum ersten Mal erhöht werden muss. Wird der Zählwert für einen Attributwert angefragt, der noch nicht initialisiert wurde, dann wird der allgemeine Initialwert zurückgegeben. Dadurch wird nur Speicher benötigt, wenn der Zählwert vom Initialwert abweicht. Im Gegenzug wird bei diesem Ansatz jedoch zusätzlicher Speicher für die Speicherung der Abbildung benötigt. Eine Einsparung des Speicherverbrauchs existiert bei dieser Methode genau dann, wenn weniger als die Hälfte der möglichen Attributwerte einen Zähler benötigen. Da in diesem Anwendungsfall sehr weit weniger als die Hälfte der Attributwerte einen Zähler benötigen, kann eine enorme Einsparung vorgenommmen werden. Statt mehrere Terabyte an Speicher zu benötigen, reichen für diesen Anwendungsfall nun wenige Gigabyte aus.

Eine wichtige Eigenschaft des Klassifikators ist der Einsatz der sogenannten *LaPlace-Glättung* [Jur09], was für eine bessere Klassifikation sorgen soll. Der Zähler für Attributwerte, also der Initialwert, beginnt dabei nicht bei Null, sondern bei einem geringen Wert wie beispielsweise Eins. Die LaPlace-Glättung sorgt damit dafür, dass die Auftrittswahrscheinlichkeit für im Training nicht vorkommende Attributwerte klein, aber nicht Null ist. Ansonsten könnte bei der Multiplikation der Wahrscheinlichkeiten durch einen einzigen Attributwert, der nicht im Training vorkam, unabhängig von der Auftrittswahrscheinlichkeit der anderen Attribute eine Wahrscheinlichkeit von Null bewirken. Beispielsweise könnte bei der Klassifikation eine Instanz vorkommen, bei der fast alle Attribute für eine bestimmte Klasse eine hohe Auftrittswahrscheinlichkeit besitzen, ein einziges Attribut aber nicht im Training vorkam. Ohne LaPlace-Glättung ist die Wahrscheinlichkeit für diese Klasse nun Null, mit der Glättung ist die Wahrscheinlichkeit für diese Klasse lediglich verringert worden. So wird sichergestellt, dass die Abwesenheit eines Attributwertes bei der Berechnung beachtet und diesem Umstand Rechnung getragen wird, dies das Gesamtergebnis aber nicht zu stark beeinflusst oder verfälscht.

Ein weiteres Problem, das angegangen wurde, ist das Auftreten von arithmetischen Unterläufen bei der Berechnung der Wahrscheinlichkeiten für die einzelnen Klassen. In Gleichung 6.1 ist die Formel für die Berechnung der Klasse mit einem Naive-Bayes-Klassifikator zu sehen, in der Wahrscheinlichkeiten miteinander multipliziert werden.

$$y = \underset{k \in \{1,\ldots,|C|\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{6.1}$$

Da die Wahrscheinlichkeiten kleiner als Eins sind, sind die Ergebnisse entsprechend klein und verkleinern sich mit jeder Multiplikation. Dabei kann es passieren, dass Zwischenergebnisse oder das Endergebnis zwischen Null und der kleinsten darstellbaren Gleitkommazahl liegen, wodurch sie auf Null gerundet werden. Dies verfälscht die Ergebnisse, da dadurch die Wahrscheinlichkeit einer Klasse bei Null liegt, obwohl sie eigentlich größer sein sollte. Im schlimmsten Fall tritt dies bei der eigentlich gesuchten Klasse in einem Zwischenergebnis auf, wodurch die Klasse nicht ausgewählt wird und die Genauigkeit des Klassifikators sinkt. Um dieses Problem zu umgehen, kann anstatt mit konkreten Wahrscheinlichkeiten mit den Logarithmus-Werten der Wahrscheinlichkeiten gerechnet werden, wie es in Gleichung 6.2 demonstriert ist. Dadurch ändert sich das Ergebnis mathematisch nicht, aber das Multiplizieren von kleinen Zahlen miteinander und die dadurch potentiell auftretenden

Fehler werden vermieden.

$$y = \underset{k\in\{1,\ldots,|C|\}}{\operatorname{argmax}} \log\left(p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \right) \tag{6.2}$$

$$= \underset{k\in\{1,\ldots,|C|\}}{\operatorname{argmax}} \left(\log\left(p(C_k)\right) + \sum_{i=1}^{n} \log\left(p(x_i|C_k)\right) \right) \tag{6.3}$$

In den ersten Tests ergab sich eine relativ geringe Genauigkeit des Klassifikators. Dies konnte vor allem darauf zurückgeführt werden, dass die Umgebung des Wortes, das klassifiziert werden sollte, einen sehr starken Einfluss hat. Mit den drei Wörtern links und rechts, sowie jeweils das Nomen und Verb links sowie rechts, erhält man 10 Umgebungswörter, die jeweils genausoviel Einfluss auf die Klassifikation wie das eigentliche Wort. Somit hat das eigentliche Wort einen recht geringen Einfluss auf die Klassifikation im Vergleich zu der Umgebung, in der es steht. Dadurch entstanden häufig Klassifikationen zu Klassen, die in dieser Umgebung ebenfalls vorkommen können, aber nicht die Klasse des gesuchten Wortes selbst sind. Die Lösung dafür ist, die Gewichtung des Wortes, für das die Mehrdeutigkeit aufgelöst werden soll, zu erhöhen. Damit das aufzulösende Wort genausoviel Gewicht wie die zehn Wörter der Umgebung besitzt, wird es in der Berechnung zehnfach gewichtet. Tests mit verschiedenen anderen Gewichten ergaben ebenfalls, dass eine zehnfache Gewichtung die besten Ergebnisse bringt. Nach [Mih07] bringt die wahrscheinlichste Bedeutung alleine bereits gute Ergebnisse, was hier ausgenutzt und durch das Einbeziehen der Umgebung weiter verbessert wird.

Da eine große Menge an Trainingsinstanzen existiert, dauert das Training entsprechend lange, allerdings lässt es sich gut parallelisieren. Beim Training werden jeweils die Informationen aus den Instanzen genutzt, um Zähler für das Vorkommen von Merkmalen für die jeweilige Klasse zu erhöhen. Die verschiedenen Zähler sind dabei unabhängig voneinander. Dadurch lässt sich die Verarbeitung der einzelnen Instanzen parallelisieren, indem die Verarbeitung der Instanzen auf die zur Verfügung stehenden Fäden aufgeteilt werden. Problematisch ist lediglich der Fall, wenn ein Zähler von mehreren Fäden gleichzeitig erhöht werden soll. Da im Anwendungsfall sehr viele Instanzen mit vielen verschiedenen Klassen und Merkmalsmöglichkeiten existieren, tritt der Fall, dass mehrere Fäden auf den selben Zähler zugreifen selten auf. Dennoch wird Zugriff auf die einzelnen Zähler synchronisiert, sodass ein simultaner Zugriff auf einen bestimmten Zähler und dadurch Wettlaufsituationen und Fehler vermieden werden können.

Nachdem das Training des Klassifikators abgeschlossen ist, kann dieser gespeichert werden, um die Ergebnisse des Trainings behalten zu können und es nicht erneut ausführen zu müssen. Dabei wird zum einen der Klassifikator gespeichert, aber auch zusätzlich der eingesetzte *StringToNominal*-Filter, indem die erstellten Listen über die Attributwerte der Attribute gespeichert sind. Dies ist notwendig, damit der Klassifikator und der Filter in Kombination erneut eingesetzt werden können. Um die gespeicherten Dateien zusammenzufassen und zu komprimieren, werden beide in eine ZIP-Datei geschrieben, wobei dieses Verhalten optional deaktiviert werden kann.

6.1.2. Einsatz des Klassifikators

Nachdem der Klassifikator trainiert wurde, muss er zur Auflösung von Mehrdeutigkeiten nur noch korrekt eingesetzt werden. Dafür wurde eine Hilfsbibliothek kreiert, in der eine Fassade zum eigentlichen Naive-Bayes-Klassifikator und anderer benötigter Klassen erstellt wurde. Dies soll die Nutzung vereinfachen und Konsistenz bringen, wenn interne Änderungen notwendig sind. Für eine vereinfachte Erstellung von Instanzen für diesen

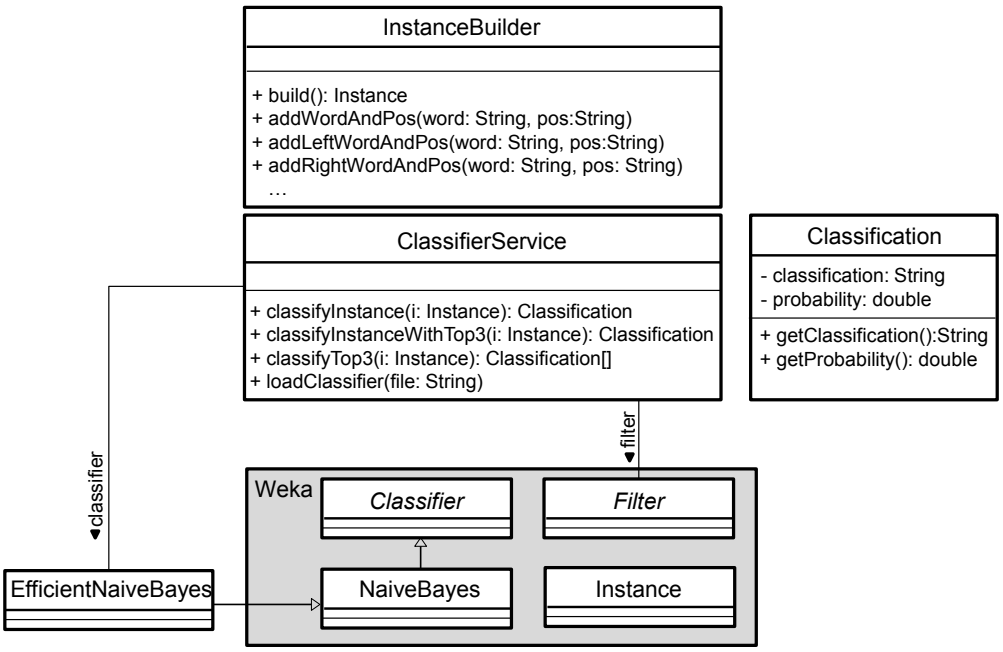

Abbildung 6.1.: Grobes Klassendiagramm für den Klassifikationsdienst

Klassifikator wurde außerdem eine Hilfsklasse angelegt. Das Klassendiagramm zur Hilfsbibliothek und der Hilfsklasse zur Instanzenerstellung ist in Abbildung 6.1 zu sehen. Die Klasse `ClassifierService` stellt die Verbindung zu den eigentlichen Klassen aus Weka her und vereinfacht dabei sowohl die Nutzung des Klassifikators als auch die Nutzung des Ergebnisses einer Klassifizierung. Ergebnisses einer Klassifizierung wird über ein Objekt der Klasse `Classification` repräsentiert. Darin sind die Klasse, also die Bedeutung eines Wortes, als Zeichenkette und die Wahrscheinlichkeit für diese Klassifikation enthalten.

Für die Auflösung von Mehrdeutigkeiten und die Nutzung der erwähnten Schnittstelle muss für das aufzulösende Wort zuerst eine Instanz, ähnlich wie für das Training, erstellt werden. Die Instanz wird erstellt, indem das mehrdeutige Wort, die drei Wörter links sowie rechts sowie die Nomen und Verben links und rechts davon aus der Eingabe extrahiert werden. Die Wörter werden dabei wie auch im Training jeweils in ihrer Grundform verwendet. Für das mehrdeutige Wort und die drei Wörter links sowie rechts werden außerdem die Wortarten ermittelt und als Attribute der Instanz genutzt. Die Wortarten werden aus den entsprechenden Informationen in den jeweiligen Knoten des PARSE-Graphen extrahiert. Stoppwörter werden, wie auch schon im Training, herausgefiltert.

Nachdem eine Instanz erstellt wurde, kann die Mehrdeutigkeit des Wortes mit Hilfe des trainierten Klassifikators aufgelöst werden. In einem Objekt zur Klasse `ClassifierService` ist der trainierte Klassifikator enthalten und das Objekt dient als Schnittstelle für die Klassifikation von Instanzen. Dabei wurde auch die alternative Methode implementiert, die in Abschnitt 5.1.1 vorgestellt wurde. Bei dieser sollen die drei wahrscheinlichsten Bedeutungen in Erwägung gezogen und diejenigen Bedeutungen bevorzugt werden, in denen das Ausgangswort als Teil der Bezeichnung vorkommt.

Damit der Klassifikator beziehungsweise der Klassifikationsdienst in PARSE eingesetzt werden kann, wurde ein Agent erstellt. Der Agent für die Auflösung von Mehrdeutigkeiten nutzt die Eingaben, die er aus dem Graphen von PARSE extrahiert, und kreiert aus den Nomen inklusive der umgebenden Wörter die Instanzen. Diese werden dann mit Hilfe des erstellten Klassifikationswerkzeuges klassifiziert, um die Bedeutung des Wortes zu erhalten.

Anschließend wird die ermittelte Bedeutung als Attribut an den entsprechenden Knoten des Graphen annotiert.

6.2. Themenextraktion

Wie in Abschnitt 5.1.2 beschrieben, soll für die Themenextraktion aus den Nomen innerhalb der Eingabe ein Themengraph erstellt werden. Um einen Themengraphen zu erhalten, muss zu den Nomen jeweils ein Bedeutungsgraph erstellt werden. Für die Erstellung eines Bedeutungsgraphen benötigt es einen Konnektor, der Informationen zu den Bedeutungen und den dazugehörigen Wikipedia-Konzepten sowie die Verbindungen zwischen Wikipedia-Konzepten aus der Wissensdatenbank von DBpedia extrahiert. Dabei sollen diejenigen Verbindungen genutzt werden, die in Tabelle 5.2 aufgeführt sind, um Wikipedia-Konzepte zu ermitteln, die über maximal zwei dieser Verbindungen vom Ausgangskonzept aus erreicht werden können. Dazu wird Bibliothek Apache Jena[1] genutzt, um Anfragen an die Anbindung der Wissensdatenbank von DBpedia zu stellen. Der erstellte Konnektor ist konfigurierbar, sodass die Adressierung der Anbindung zur DBpedia geändert werden kann, damit beispielsweise eine Kopie der DBpedia-Wissensdatenbank, die lokal bereitstellt wird, auch angefragt werden kann. Für die Repräsentation eines Graphen wird die Bibliothek JGraphT[2] genutzt. Die Bibliothek ermöglicht unter anderem eine flexible Nutzung und enthält einen Großteil der benötigten Funktionen, wie beispielsweise die Möglichkeit zur Verschmelzung von Graphen oder die Ermittlung des kürzesten Pfades zwischen zwei Knoten. Desweiteren ist bereits eine allgemeine Implementierung des PageRank-Algorithmus enthalten, mit der eine angepasste Version entwickelt werden konnte, um die Beeinflussung durch die Ausgangskonzepte zu ermöglichen, wie sie in Abschnitt 5.1.2 entworfen wurde. Um den Zugriff auf die Funktionalitäten von JGraphT zu vereinfachen und steuern, wurde eine Klasse erstellt, die einen Themen- oder Bedeutungsgraphen darstellt und benötigte Funktionalitäten bereitstellt. In der Implementierung existiert kein konkreter Unterschied zwischen Themen- und Bedeutungsgraph außer in der Anzahl an Bedeutungen, die dem jeweiligen Graphen zugeordnet sind. Deshalb können beide mit der selben Klasse repräsentiert werden, die im Klassendiagram in Abbildung 6.2 als `TopicGraph` dargestellt ist. Desweiteren wurde eine Klasse erstellt, die einen Knoten im Graphen darstellt und dabei den Bezeichner sowie den benötigten Verweis zur DBpedia-Entität enthält. Diese Klasse heißt im Klassendiagram `WikiVertex`.

Für die Verarbeitung der Werte aus dem PageRank-Algorithmus und die endgültige Auswahl von Themen existieren zwei Klassen, die die beiden Methoden aus Abschnitt 5.1 zur Auswahl der Themen repräsentieren: eine für die Auswahl der Knoten mit der höchsten Konnektivität (`TopConnectivityProcessor`) und eine für die Variante, in der Themen in Abhängigkeit voneinander ermittelt werden sollen (`CombinedProcessor`). Über eine gemeinsame Schnittstelle (`TopicProcessor`) können diese angesprochen werden, um eine Auswahl an Themen zu erhalten. Dabei kann die Anzahl der zu ermittelnden Themen vom Nutzer mit der Methode `setAmountOfTopics()` in der Klasse `TopicExtraction` festgelegt werden oder der Nutzer kann die Anzalh automatisch vom Programm festlegen lassen. Bei der automatischen Festlegung werden doppelt so viele Themen ermittelt wie verschiedene Bedeutungen als Eingabe übergeben werden. Dadurch werden für kleine Eingabegrößen weniger Themen extrahiert, da bei kleinen Eingabegrößen die Qualität der Themen erfahrungsgemäß schnell abnimmt. Die automatisch ermittelte Anzahl an Themen kann über einen Maximalwert, der mit der Methode `setMaxAmountOfTopics()` festgelegt werden kann, nach oben limitiert werden, damit für größeren Eingabegrößen gleichzeitig nicht zu viele Themen extrahiert werden.

[1] https://jena.apache.org/
[2] http://jgrapht.org/

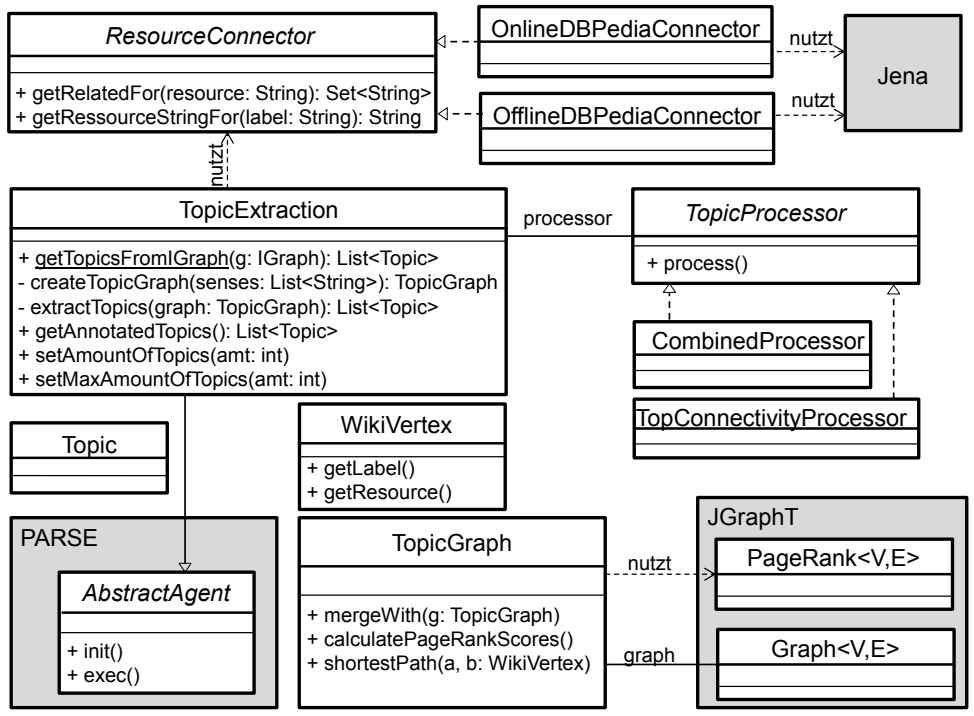

Abbildung 6.2.: Grobes Klassendiagramm der Themenextraktion

Das Erstellen der Bedeutungsgraphen ist zeitintensiv. Der Grund dafür liegt vor allem an der großen Menge an Anfragen, die an die DBpedia-Wissensdatenbank gestellt werden müssen. Die genutzte Bibliothek Apache Jena verfügt über die Möglichkeit, Anfragen in einem Puffer-Speicher zu halten, um wiederholte gleiche Anfragen schneller beantworten zu können. Da aber nicht nur gleiche, sondern viele verschiedene Anfragen gestellt werden müssen, ist auch mit diesem Puffer-Speicher die Datenabfrage der Flaschenhals der Anwendung. Aus diesem Grund wurde ein einfacher Puffer-Speicher für Bedeutungsgraphen eingeführt, in dem die bereits erstellten Bedeutungsgraphen gespeichert werden. Wenn für eine Bedeutung ein Bedeutungsgraph erstellt werden soll, dann wird zuerst überprüft, sich im Puffer-Speicher der dazugehörige Bedeutungsgraph befindet. Befindet sich im Puffer-Speicher der dazugehörige Bedeutungsgraph, dann wird dieser Bedeutungsgraph zurückgegeben. Andernfalls wird ein neuer Bedeutungsgraph erstellt, dieser dann in den Puffer-Speicher übertragen und anschließend zurückgegeben.

Für den Einsatz in PARSE wurde ein Agent erstellt. Dieser liest zunächst die Nomen mitsamt den Bedeutungen aus der Eingabe aus. Nachdem damit die Themen ermittelt wurden, werden diese vom Agenten an den entsprechenden PARSE-Graphen der Eingabe annotiert. Dafür wurde ein eigener Knoten-Typ erstellt, in den an ein Attribut die Liste mit den Themen gespeichert wird. Um das Auslesen der annotierten Themen aus dem PARSE-Graphen zu vereinfachen, wurde außerdem die statische Methode `TopicExtraction.getTopicsFromIGraph(IGraph)` hinzugefügt, die annotierten Themen aus dem Graphen ausliest und als Liste zurückgibt.

6.3. Zuordnung von Themen zu Domänen

Als Anwendungsfall sollen für eine Eingabe und ihre Themen eine passende Domänenontologie zugeordnet werden. Dabei sollen aus den zur Verfügung stehenden Ontologien die passenden Akteur- sowie Umgebungsontologien ausgewählt und im Anschluss verschmolzen werden.

Tabelle 6.1.: Domänenontologien, zu denen die Eingaben zugeordnet werden können. In der Spalte *Typ* kennzeichnet ein „A“ die Akteur-Ontologien, während ein „U“ die Umgebungsontologien markiert.

Typ	**Name**	**Beschreibung**
A	Household Robot	Haushaltsroboter, wie ARMAR
A	Virtual Assistant	virt. Assistenten, z.B. Amazons Echo (Alexa)
A	Drone	Drohnen, wie bspw. ein Quadrocopter
A	Lego Mindstorm	Lego Mindstorm Roboter
U	Kitchen	Küche, mit Küchengeräten, -möbeln und Lebensmitteln
U	Bar	Bar, mit Cocktails, Barmöblierung etc.
U	Garden	Garten, mit Gartenmöblierung, Pflanzen u.ä.
U	Bedroom	Schlafzimmer
U	Children's room	Kinderzimmer
U	Music	Thematik Musik mit Musikrichtungen, Instrumenten etc.
U	Heating	Thematik Heizen (Klimaanlagen, Heizungen u.ä.)
U	Laundry	Thematik Wäsche (Waschmaschine, Trockner etc.)

Die zur Verfügung stehenden Ontologien müssen dafür zuvor erstellt worden sein und folgen im besten Fall den Regeln aus Abschnitt 5.2.1. Um die Zuordnung testen zu können, wurden verschiedene Ontologien erstellt, die in Tabelle 6.1 zu sehen sind. Vier Ontologien sind Ontologien über Akteure, die restlichen acht Ontologien repräsentieren jeweils eine Umgebung. Die Akteur-Ontologien wurden deshalb so gewählt, um verschiedenen Typen von Akteuren zu repräsentieren. Zum einen können Roboter angesprochen werden, zum anderen aber auch virtuelle Assistenzsysteme oder Spielgeräte. Für die Auswahl der erstellten Umgebungsontologien wurde ebenfalls versucht, verschiedene Einsatzgebiete wie die Küche oder den Garten abzudecken. Gleichzeitig ist unter anderem die *Music*-Ontologie erstellt worden. Damit soll gezeigt werden, dass Ontologien auch abstraktere Umgebungen darstellen können, in und mit denen Akteure arbeiten können.

Als Beispiel für die Struktur einer Akteur-Ontologie ist die Struktur der Ontologie über den Haushaltsroboter ARMAR in Tabelle 6.2 dargestellt, bei deren Erstellung die Regeln aus Abschnitt 5.2.1 beachtet wurden. Allgemein ist die hauptsächliche Struktur zur ursprünglichen Ontologie, die im vorigen Kapitel analysiert und in Tabelle 5.3 dargestellt wurde, gleich geblieben. Der größte Unterschied liegt darin, dass das *Object*-Konzept inklusive seiner Instanzen aus der Akteur-Ontologie entfernt wurde, da Objekte im Allgemeinen zur Umgebung gehören. Die entfernten Instanzen befinden sich in der Umgebungsontologie über die Küche, deren Struktur als Beispiel für eine Umgebungsontologie in Tabelle 6.3 zu sehen ist. Die Objekte wurden dabei im Gegensatz zur ursprünglichen Variante weiter unterteilt und spezifiziert. Die Unterteilung ist anhand der Eigenschaften der Objekte geschehen und entspricht dem programmatischen Konzept von Schnittstellen, wie es in den Regeln gefordert wird. So sind zum Beispiel alle Objekte, die greifbar sind, der Klasse *Grabbable* angehörig. Da Instanzen in Ontologien zu mehreren Klassen gehören können, können diese auch mehrere Objekteigenschaften erfüllen. Deshalb können Instanzen beispielsweise gleichzeitig greifbar (*Grabbable*) und essbar (*Consumable*) sein, wie es etwa für die Instanz *Popcorn* der Fall ist.

Damit eine Zuordnung der Themen einer Eingabe zu passenden Ontologien stattfinden kann, müssen zuerst die Ontologien geladen und analysiert werden. Für die Analyse werden die annotierten Identifikatoren der Wikipedia-Artikel extrahiert, deren Erstellung in Abschnitt 5.2.1 beschrieben wurde. Diese Identifikatoren werden als Eingabe für die Themenextraktion genutzt und dem in Abschnitt 6.2 beschriebenen Themenextrahierer übergeben. Dabei wird die unabhängigen Variante der Themenextraktion genutzt, bei der die

Tabelle 6.2.: Struktur der Domänenontologie über Haushaltsroboter

Konzept	Beschreibung	Beispielinstanzen
Thing	Oberkonzept der Ontologie	
DataType	Datentypen	int, Grabbable
Method	ausführbare Aktionen	bring, grab
Parameter	Parameter-Arten d. Methoden	grab.what, move.howFar
State	auftretende Zustände	on, off
System	ansprechbare Systeme	ARMAR, Robot

Tabelle 6.3.: Struktur der Domänenontologie über die Küche

Konzept	Beschreibung	Beispielinstanzen
Thing	Oberkonzept der Ontologie	
Object	auftretende Objekte (Oberkonzept)	-
Closeable	Objekte, die schließbar sind	Dishwasher, Fridge
Openable	Objekte, die öffenbar sind	Dishwasher, Fridge
Consumable	Objekte, die verzehrbar sind	Popcorn, Milk
Controllable	Objekte, die bedienbar sind	Microwave, TV
Grabbable	Objekte, die greifbar sind	Popcorn, Plate
State	auftretende Zustände	closed, dirty

Themen mit der höchsten Konnektivität ausgewählt werden. Diese Option ist für diesen Zweck sinnvoller, da die Ontologien themenspezifisch sind und der zusätzliche Aufwand des anderen Ansatzes hier unnötig ist. Die resultierenden Themen werden zusammen mit dem erstellten Themengraphen und der Ontologie in einem Objekt zwischengespeichert. Die Zuordnung der Themen muss nur einmalig bei der Initialisierung der Ontologieauswahl erstellt werden. Um diesen Schritt bei einem Neustart des Programms nicht erneut durchlaufen zu müssen, kann das Ergebnis der Themenzuordnung auf die Festplatte gespeichert werden. Dabei werden das Objekt, das die Themen, den Themengraphen und die Ontologie hält, exportiert. Hierbei muss jedoch darauf geachtet werden, dass bei Änderungen der Ontologien und ähnlichem diese Objekte nicht mehr aktuell sind und daher erneuert werden müssen.

Nachdem die Ontologien Weise geladen und ihnen Themen zugeordnet wurden, kann der Vergleich stattfinden. Pro Eingabetext werden die ermittelten Themen des Eingabetextes mit denen der Ontologien verglichen, wie es in Abschnitt 5.2.2 beschrieben wurde. Nacheinander wird für jedes Thema des Eingabetextes der Ähnlichkeitswert zu jeder Ontologie errechnet. Anschließend werden die Ähnlichkeitswerte der Themen zu den jeweiligen Ontologien gemittelt, um die Übereinstimmung zu den jeweiligen Ontologien zu erhalten. Danach wird der maximale Wert der Übereinstimmung ermittelt. Von diesem Wert werden 10% abgezogen, um die untere Schwelle für den Übereinstimmungswert zu ermitteln, mit dem eine Ontologie ausgewählt wird. So wird beispielsweise bei einer maximalen Ähnlichkeit von 0,5 von diesem Wert 0,05 abgezogen und der Schwellwert auf 0,45 festgelegt.

Auf diese Weise werden zunächst die Umgebungsontologien ermittelt. Für die Auswahl der Akteur-Ontologien wird der Übereinstimmungswert angepasst, wie es im Entwurf in Abschnitt 5.2.2 vorgesehen ist. Dabei wird der Wert, der bestimmt wie gut ein Akteur mit der Umgebung umgehen kann, einberechnet. Um diesen Wert zu bestimmen, werden zuerst alle Objektklassen wie beispielsweise *Grabbable* oder *Consumable* aus den ausgewählten Umgebungsontologien extrahiert. Diese werden dann jeweils mit den Datentypen aus den jeweiligen Akteur-Ontologien verglichen, da die Datentypen bestimmen, mit was der Akteur umgehen kann. Der Vergleich findet über einen Vergleich der Zeichenketten statt. Als Alternative zum Vergleich der Zeichenketten könnte man einen semantischen Ansatz

wählen. Dabei könnten die Äquivalenz von Datentypen und Objekt-Klassen auch dann erkannt werden, wenn beispielsweise synonymen Bezeichner verwendet werden. Dies könnte das Ergebnis verbessern. Da die Regel, das äquivalente Entitäten auch gleiche Bezeichner verwenden sollen, umgesetzt wurde, genügt der Vergleich der Zeichenketten. Das Ergebnis des Vergleiches ist der Anteil an Objekt-Klassen, für die der Vergleich positiv ausfiel. Der endgültige Übereinstimmungswert für Akteuer bestimmt sich nach Gleichung 5.4. Dabei wird der ursprüngliche Übereinstimmungswert mit dem Vergleichswert verrechnet, wobei die ursprüngliche Ähnlichkeit doppelt so stark gewichtet wird. Danach wird der Akteur mit dem besten Übereinstimmungswert ausgewählt.

Die so ausgewählten Ontologien werden im Anschluss verschmolzen. Dafür wurde der einfacher Ansatz zum Verschmelzen aus dem Entwurf aus Abschnitt 5.2.3 implementiert, der Entitäten mit dem gleichen Namen aufeinander abbildet. Zunächst werden dafür alle Entitäten der Ontologien gesammelt und nacheinander in eine neue Ontologie übertragen. Wenn eine Entität mit gleichem Namen bereits in der neuen Ontologie existiert, werden diese miteinander verschmolzen und ihre Eigenschaften kombiniert. Nachdem so die neue Ontologie kreiert und befüllt wurde, müssen noch die Objekteigenschaften erstellt werden, die Datentyp-Instanzen mit den entsprechenden Instanzen der gleichnamigen Objekt-Klasse verbindet. Dafür wird für jeden Datentypen versucht, eine gleichnamige Objekt-Klasse zu finden. Wenn eine Objekt-Klasse gefunden wird, dann wird die Objekteigenschaft vom Datentypen zu jeder Instanz, die der Objekt-Klasse zugeordnet ist, erstellt. Existiert beispielsweise der Datentyp *Grabbable*, so werden Verbindungen von diesem Datentyp zu den Instanzen der Objekt-Klasse *Grabbable* erstellt. Auch hier sind wieder andere Vergleichsmöglichkeiten denkbar, die beispielsweise semantische Ähnlichkeiten zwischen dem Datentyp und der Objekt-Klasse einbeziehen. Aus den selben Grund wie zuvor, dem Einhalten der Regeln zur Erstellung von Ontologien, genügt ebenfalls der einfache Ansatz.

Die aus der Verschmelzung der ausgewählten Ontologien resultierende Ontologie wird dann gespeichert und an den PARSE-Graphen der Eingabe annotiert. Dafür wird wie für die Themenextraktion in Abschnitt 5.1.2 ein eigener Knoten erstellt und in dem die Ontologie in einem Attribute gespeichert wird. Ebenso wurde für die einfache Nutzung die statische Methode `OntologySelector.getOntologyFromIGraph(IGraph)` erstellt, mit der die annotierte Ontologie aus dem Graphen ausgelesen werden kann.

7. Evaluation

In diesem Kapitel wird die Qualität der der erstellten Werkzeuge zur Auflösung von Mehrdeutigkeiten, Themenextraktion sowie Ontologieauswahl evaluiert. Für die Evaluationen wurden Texte aus dem PARSE-Korpus genutzt. Der PARSE-Korpus entstand in den verschiedenen Arbeiten, die sich mit PARSE auseinandersetzen und in Kapitel 3 beschrieben wurden. Im Korpus sind Anweisungen an den Haushaltsroboter ARMAR in verschiedenen Szenarien enthalten. Die Szenarien sind in Tabelle 7.1 mit einer kurzen Beschreibung gelistet. Die Texte zu den Szenarien sind jeweils Transkripte der Anweisungen an ARMAR, die in verschiedenen Studien erstellt wurden. Dadurch kann die Qualität der Werkzeuge auf realistischen Eingaben getestet werden.

Zuerst wird die Güte der Auflösung von Mehrdeutigkeiten evaluiert. Die Auflösung von Mehrdeutigkeiten ist eine Grundlage für die folgenden Ansätze, weshalb hier eine hohe Genauigkeit wichtig ist. Der Aufbau und die Ergebnisse der Evaluation des Ansatzes dazu sind in Abschnitt 7.1 zu finden.

Danach wird die Qualität der extrahierten Themen ermittelt. Dazu wurde eine Befragung durchgeführt, bei der die Befragten bestimmen sollten, ob die extrahierten Themen passend zu den jeweiligen Eingabetexten sind. Bei der Befragung wurde sich an der Umfrage aus [HHKG13] orientiert, um die Ergebnisse des Ansatzes dieser Arbeit mit der Güte des Ansatzes von Hulpus et al. vergleichen zu können. Details zu der Evaluation der Themenextraktion sowie die Ergebnisse sind in Abschnitt 7.2 zusammengefasst.

Zuletzt findet die Evaluation der Ontologieauswahl statt. Dabei soll geprüft werden, ob die Auswahl von passenden Ontologien zu den Eingabetexten zuverlässig ist. Ebenso sollen

Tabelle 7.1.: Szenarien des PARSE-Korpus

Szenario	**Kurzbeschreibung**
1	Popcorn soll vom Tisch geholt werden
2	Becher vom Tisch soll in Spülmaschine geräumt werden
3	Orangensaft soll aus dem Kühlschrank geholt werden
4	Geschirr soll in Spülmaschine geräumt werden, falls es schmutzig ist
5	Cocktail *Screwdriver* machen, falls möglich mit frischen Orangen
6	Wasser in Tasse füllen und rote Tassen aus Spülmaschine ausräumen
7	Teller säubern, dann Fertiggericht auf dem Teller in Mikrowelle zubereiten
8	Wäsche aus der Waschmaschine nehmen und in den Trockner packen

Tabelle 7.2.: Ergebnisse der Evaluation der Auflösung der Mehrdeutigkeiten mit Instanzen aus Wikipedia

Durchlauf	**Genauigkeit**
1	80,17%
2	80,17%
3	79,69%
4	80,70%
5	79,37%
6	79,67%
7	80,69%
8	79,98%
9	79,54%
10	78,93%
Durchschnitt	79,89%

die Grenzen des Ansatzes ausgemacht werden. Dabei soll eruiert werden, welche Konfiguration die besten Ergebnisse liefert. In Abschnitt 7.3 werden die Resultate aufgelistet und diskutiert.

7.1. Auflösung von Mehrdeutigkeiten

Um die Güte der Auflösung von Mehrdeutigkeiten zu bestimmen, wurden zwei verschiedene Evaluationen durchgeführt. Die erste misst die Genauigkeit des Klassifikators anhand des Trainingskorpus, die zweite misst die Genauigkeit auf dem PARSE-Korpus mit Anweisungen für die Programmierung in natürlicher Sprache.

Für die erste Evaluation war eine zehnfache Kreuzvalidierung vorgesehen. Bei über fünf Millionen Trainingsinstanzen, die in Abschnitt 6.1.1 erstellt wurden, müssen für die Kreuzvalidierung pro Durchlauf 500.000 Instanzen überprüft werden. Aufgrund der großen Menge an Trainingsdaten und der Dauer für die Klassifikation einer Instanz von etwa zwei bis vier Sekunden war dies zeitlich nicht realisierbar, selbst nachdem der Test durch Parallelisierung entsprechend beschleunigt werden konnte. Deshalb wurde der Test angepasst, um ein möglichst vergleichbares Ergebnis zur Kreuzvalidierung zu erhalten, der zeitliche Aufwand allerdings geringer ist. Dafür wurde die Evaluation in mehrere Durchläufe unterteilt. Pro Durchlauf werden aus dem Trainingskorpus 10.000 zufällige Instanzen entfernt und zwischengespeichert, um sie nach dem Training als Testmenge einsetzen zu können. Mit den übrigen Instanzen wird der Klassifikator trainiert. Nach dem Training werden die vor dem Training entfernten Instanzen klassifiziert, wobei ermittelt wird, wie viele der Instanzen korrekt klassifiziert wurden. Insgesamt wurden auf diese Weise zehn Durchläufe durchgeführt. Die Genauigkeit pro Durchlauf ebenso wie der Schnitt aller Durchläufe sind in Tabelle 7.2 zu sehen. Die durchschnittliche Genauigkeit von 79,89% ist durchaus zufriedenstellend und mit dem Ergebnis von 84,65% aus der Evaluation in [Mih07] vergleichbar.

Wichtiger als der Test anhand des Trainingskorpus ist jedoch die Genauigkeit des Ansatzes bei Anweisungen für die Programmierung in natürlicher Sprache, wie sie in PARSE eingesetzt werden. Daher ist der zweite Test eine Evaluation des Klassifikator anhand des PARSE-Korpus. Der Korpus wurde dafür manuell mit dem Goldstandard für die mehrdeutigen Nomen annotiert. Dadurch kann das Ergebnis des Klassifikators mit dem Goldstandard verglichen und die Genauigkeit des Klassifikators ermittelt werden. In Abschnitt 5.1.1 wurden zwei Varianten für die Klassifikation vorgestellt: Die erste Variante, die im folgenden *Top-3* genannt wird, zieht die drei wahrscheinlichsten Klassen in Erwägung und bevorzugt diejenigen Klassen, die das Ausgangswort beinhalten. Die zweite

Tabelle 7.3.: Ergebnisse der Evaluation der Auflösung der Mehrdeutigkeiten mit den zwei verschiedenen Methoden auf dem PARSE-Korpus mit 961 Nomen.

Methode	**korrekt**	**Genauigkeit**
Top-3	846	88,03%
Top-1	845	87,93%

Variante, im folgenden *Top-1* bezeichnet, ist die normale Variante eines Klassifikators, bei der die wahrscheinlichste Klasse ausgewählt wird. Um die beiden Varianten anhand ihrer Genauigkeit vergleichen zu können, wurden beide Ansätze getestet. Die Ergebnisse sind in Tabelle 7.3 aufgeführt. Auf dem PARSE-Korpus erzielt der Klassifikator mit der *Top-3*-Methode eine Genauigkeit von 88,03%. Bei der *Top-1*-Methode wurden 845 von 961 Instanzen korrekt klassifiziert, was einer Genauigkeit von 87,93% entspricht. Damit ist nur eine einzigen korrekten Klassifikation als Unterschied zwischen den beiden Varianten vorhanden. Mit dieser Evaluation konnte also keine echte Verbesserung durch die alternative Variante *Top-3* gegenüber der normalen Klassifikation festgestellt werden. Da das Wort, dessen Mehrdeutigkeit aufgelöst werden soll, bei der Klassifikation bereits stärker gewichtet ist, ist diese zusätzliche Fokussierung auf das zu disambiguierende Wort wohl überflüssig.

Bei der Auflösung der Mehrdeutigkeiten treten verschiedene Probleme auf, die in der Evaluation deutlich wurden. Ein Problem des Ansatzes liegt in den Abhängigkeiten zu den Ergebnissen der vorher stattfindenden Annotation der Wortarten. 18 der 961 Nomen (1,87%) wurde fälschlicherweise nicht als Nomen markiert, weshalb sie auch nicht disambiguiert wurden. Ein anderes Problem des Ansatzes sind Nomen, für die es keinen passenden Begriff in der Wikipedia gibt. Diese Problematik wurde bereits in Abschnitt 5.1.1 angesprochen und ist für einige Nomen in der Evaluation aufgetreten. Ein konkretes Beispiel für dieses Problem ist das Wort „front“, das neun Mal im Korpus in Ausdrücken wie „in front of you“ vorkommt. Eine korrekte Bedeutung für dieses Wort in dem Kontext findet man beispielsweise in WordNet[Fel98]: „the immediate proximity of someone or something“. In Wikipedia gibt es verschiedene Bedeutungen dieses Wortes, unter anderem „Front (military)“ und „Weather front“, für die eigentliche Bedeutung in diesem Fall existiert jedoch kein Äquivalent zur korrekten Bedeutung, wie sie in WordNet existiert. Dadurch kann die Bedeutung des Wortes, ebenso wie Bedeutungen anderer Wörter ohne Präsenz in der Wikipedia, nicht korrekt klassifiziert werden. Zuletzt repräsentieren von den achtzehn Attributen sieben Attribute die Wortarten des aufzulösenden Wortes und der umgebenden Wörter, Die genutzten Wortarten werden durch ein anderes Werkzeug annotiert und bei der Erstellung der Instanzen lediglich ausgelesen. Da die annotierten Wortarten nicht fehlerfrei annotiert werden, kann es dadurch zu einem negativen Einfluss auf die Klassifizierung kommen.

In Abschnitt 6.1 wurde ein optionaler Filter angesprochen, der Bedeutungen, für die nur eine Trainingsinstanz existierte, filtern soll. Für die Evaluationen in diesem Kapitel wurde diese Filterung nicht angewandt. Einzelne Tests ergaben durch den Einsatz des Filters eine Verbesserung von etwa 3% der Genauigkeit des Klassifikators auf den Instanzen der Wikipedia, was auch dem Anteil der Bedeutungen mit lediglich einer Trainingsinstanz entspricht. Da diese Filterung aber sehr zeitaufwändig ist und bei der ersten Evaluation mehrfach durchgeführt werden muss, wurde sie nicht vollständig evaluiert.

Bei näherer Betrachtung des PARSE-Korpus fällt allerdings auf, dass der Umfang an verschiedenen Nomen eher gering ist. Von den 961 Nomen, die aufgelöst werden sollen, entsprechen beispielsweise 120 Nomen (12,5%) dem Wort „fridge“ in der jeweils selben Bedeutung. „table“ (111 Mal, 11,6%), „cup“ (73 Mal, 7,6%) und „orange“ (67 Mal, 7,0%) sind ebenfalls

Tabelle 7.4.: zusätzliche erstellte Szenarien für die Evaluation

#	**Kurzbeschreibung**
1	Drohne, die im Garten eine bestimmte Strecke abfliegen soll
2	Lego Mindstorm Roboter, der zu einer Rassel fahren und diese aufheben soll
3	virtueller Assistent soll Heizung aufdrehen und Musik abspielen

häufig vertreten. Der PARSE-Korpus ist thematisch auf den Haushalt bezogen, weshalb ein gehäuftes Auftreten der selben Begriffe nicht ungewöhnlich ist. Allerdings bleibt dadurch die Frage offen, ob in anderen Szenarien außerhalb des Haushalts die Genauigkeit des Klassifikators ebenfalls so hoch ist. Da die Evaluation anhand der Trainingsinstanzen aber ebenfalls gute Ergebnisse lieferte, kann dies vorerst angenommen werden.

Für die Evaluation der Auflösung von Mehrdeutigkeiten kann zusammengefasst werden, dass der Klassifikator zuverlässig Mehrdeutigkeiten auflöst. Gerade bei der Evaluation auf den Anwendungsfällen von PARSE wurden gute Ergebnisse erzielt. Dies ist vor allem bedeutend, da die Auflösung der Mehrdeutigkeiten für die Themenextraktion benötigt wird. Schlechte Ergebnisse in diesem Schritt können sich negativ auf die Themenextraktion auswirken und dort die Ergebnisse verfälschen.

7.2. Themenextraktion

Die Evaluation der Güte der extrahierten Themen ist eine Herausforderung. Das Erstellen eines allgemeingültigen Goldstandards ist problematisch, da sich dafür auf eine abschließende Menge akzeptierter Themen für jeden Eingabetext geeinigt werden müsste. Daher wurde für die Evaluation der Themenextraktion eine Nutzerstudie durchgeführt. Die Studie wurde nach dem Vorbild der Evaluation aus [HHKG13] erstellt, um die Ergebnisse vergleichen zu können. Dadurch kann unter anderem festgestellt werden, welche Auswirkungen die Änderungen, wie beispielsweise der Austausch des LDA-Verfahrens zur Ermittlung der Schlüsselbegriffe der Eingabe, haben.

Für die Nutzerstudie mussten zuerst die Themen für eine Menge an Eingaben ermittelt werden. Bei der Auswahl der Eingabetexte wurde zunächst der PARSE-Korpus genutzt und für jedes der Szenarien aus Tabelle 7.1 zwei zufällige Texte ausgewählt. In dem Korpus existiert jedoch das Problem, dass die Texte zwar verschieden sind und unterschiedliche Tätigkeiten behandeln, jedoch eine ähnliche Thematik aufweisen, da alle Szenarien im Haushalt beziehungsweise der Küche angesiedelt sind. Zusätzlich ist der Akteur in allen Szenarien stets der Haushaltsroboter ARMAR. Aus diesem Grund wurden weitere synthetische Eingaben erstellt, die andere Themen und insbesondere auch andere Akteure abdecken sollen. Dabei wurde darauf geachtet, dass Form und Art der Eingaben ähnlich zu den Eingaben aus dem PARSE-Korpus sind. Insgesamt wurden drei neue Szenarien erstellt, die in Tabelle 7.4 aufgeführt sind. Zu jedem dieser Szenarien wurden zwei unterschiedliche Texte erstellt. Alle so ausgewählten 22 Texte wurden dann auf zwei Umfragen aufgeteilt. Dabei wurde jeder Umfrage ein Text pro Szenario zugewiesen. Die Ausgangstexte mitsamt den resultierenden Themen sind im Appendix in Abschnitt B.2 zu finden. Zu den ausgewählten Texten wurden dann die Themen extrahiert. Es wurden dabei beide Varianten, die in Abschnitt 5.1 beschrieben wurden, evaluiert, um feststellen zu können, welche Methode geeigneter ist: Beim *CombinedProcessor* ist die Variante der Themenauswahl umgesetzt, bei der die Themen in Abhängigkeit voneinander ermittelt werden, beim *TopConnectivityProcessor* werden die Themen mit der höchsten Konnektivität ausgewählt.

In der Umfrage, für die ein Auszug im Anhang in Abschnitt B.1 zu finden ist, wurde zur Bewertung der Themen den Nutzern der Ausgangstext und nacheinander die dazu extrahierten Themen vorgelegt. Für jedes Thema konnte zuerst ausgewählt werden, ob das

Thema *passend*, *verwandt* oder *unverwandt* ist. Die erste Kategorie (*passend*) sollte ausgewählt werden, wenn das Thema als geeignetes Thema für den Text angesehen wurde. In die zweite Kategorie (*verwandt*) gehören alle Themen, die zwar mit der Eingabe verwandt sind, sich jedoch nicht gut genug als Thema eignen. Zur letzten Kategorie (*unverwandt*) sollten alle Themen zugeordnet werden, die mit dem Text keine Verwandtschaft haben und daher komplett ungeeignet sind. Nach dieser ersten Einschätzung wurden die Teilnehmer der Umfrage noch nach einer Präzisierung ihrer Einschätzung gefragt, wenn die erste oder zweite Kategorie ausgewählt wurde. Wenn die Testperson die erste Kategorie ausgewählt hat, konnte sie auswählen, ob das Thema passend, aber *etwas zu allgemein*, *etwas zu spezifisch* oder *in Ordnung* ist. Zur Präzisierung der zweiten Kategorie konnten die Personen angeben, ob das Thema verwandt ist und dabei *zu allgemein* oder *zu spezifisch* ist oder das Thema aus einem anderen Grund nicht passend ist. Jede Umfrage wurde, wie auch in [HHKG13], drei Testpersonen vorgelegt, damit für jedes Thema eine mehrheitlich Entscheidung gefällt werden konnte. Alle Testpersonen waren zwischen 22 und 27 Jahre alt und studieren oder haben das Studium bereits abgeschlossen. Bei jeder Umfrage nahmen jeweils eine weibliche und zwei männliche Testpersonen teil. Die Testpersonen aus der ersten Umfrage setzten sich aus einer Person aus dem beruflichen Umfeld der Informatik und zwei Testpersonen aus anderen Gebieten zusammen. Bei der zweiten Umfrage waren zwei Personen aus dem beruflichen Umfeld der Informatik und eine Testperson aus einem anderen Gebiet. Für die Bestimmung der Übereinstimmung der Beurteiler wurde pro Umfrage der κ-Wert mit der Methode nach [Eye06] berechnet. Die dazu benötigte Formel ist in Gleichung 7.1 zu finden. Darin ist i der Index für eine der N Fragen, z ist die Anzahl der Kategorien und d ist die Anzahl an Testpersonen. p_i ist das Maß für die Übereinstimmung bei Frage i und wird nach der Formel in Gleichung 7.2 berechnet. Dort ist d_{ij} die Anzahl an Testpersonen, die bei Frage i die Kategorie j ausgewählt haben. Für die erste Umfrage in dieser Arbeit erhält man mit diesem Verfahren einen κ-Wert von 0,243. Für die zweite Umfrage ergibt sich für κ der Wert 0,267. Damit erhält man nach Landis und Koch in [LK77] jeweils eine angemessene Übereinstimmung. Gleichzeitig ist der Wert ähnlich zum κ-Wert von 0,27 aus [HHKG13].

$$\kappa_n = \frac{\sum_i^N p_i - \frac{1}{z^{d-1}}}{1 - \frac{1}{z^{d-1}}} \tag{7.1}$$

$$p_i = \frac{1}{d(d-1)} \sum_{j=1}^{z} (d_{ij}^2 - d_{ij}) \tag{7.2}$$

Eine Zusammenfassung der Antworten für den *CombinedProcessor* ist in Tabelle 7.5 zu finden, für den *TopConnectivityProcessor* befindet sich die Zusammenfassung der Antworten in Tabelle 7.6. In diesen Zusammenfassungen erkennt man, dass die Ergebnisse über alle Themen nahezu gleichmäßig über die drei Kategorien verteilt sind. Das bedeutet, dass 36,73% der Themen des *CombinedProcessor* und 34,14% der Themen des *TopConnectivityProcessor* als passend eingestuft wurden. Dies liegt auch an der Menge an Themen, die zugewiesen wurden, wie die weiteren Statistiken in der Zusammenfassung zeigen. Wenn jeweils nur das erste Thema pro Text betrachtet wird, dann sind 53,03% der Themen passend. Gleichzeitig werden nur noch 18,18% der Themen als komplett unverwandt eingeschätzt, während der Anteil für verwandte Themen bei 28,79% liegt. Mit zunehmender Anzahl an Themen ist die Tendenz erkennbar, dass weniger Themen passend und stattdessen mehr Themen unverwandt sind. Schon durch ein zweites Thema sinkt der Anteil der passenden Themen erheblich um etwa 10% auf 44,70%, der Anteil der Unverwandten steigt um knapp 10% auf 28,03%. Der Anteil der verwandten Themen bleibt annähernd gleich und steigt nur langsam mit der Anzahl an Themen an. Die Kategorien *passend* und *verwandt* können auch zusammengenommen werden. Dadurch erhält man das gemeinsame

Tabelle 7.5.: Genauigkeit der Themenauswahl bei der Verwendung des *CombinedProcessor*

	Passend	**Verwandt**	**Unverwandt**
Erstes Thema	53,03%	28,79%	18,18%
Erste 2 Themen	44,70%	27,27%	28,03%
Erste 3 Themen	44,95%	29,80%	25,25%
Erste 4 Themen	43,18%	29,92%	26,89%
Erste 5 Themen	37,74%	31,72%	29,31%
Alle Themen	36,73%	31,85%	31,42%

Tabelle 7.6.: Genauigkeit der Themenauswahl bei der Verwendung des *TopConnectivityProcessor*

	Passend	**Verwandt**	**Unverwandt**
Erstes Thema	53,03%	28,79%	18,18%
Erste 2 Themen	42,42%	28,79%	28,79%
Erste 3 Themen	44,44%	29,80%	25,75%
Erste 4 Themen	42,05%	30,68%	27,27%
Erste 5 Themen	36,54%	31,42%	29,89%
Alle Themen	34,14%	33,26%	32,60%

Ergebnis für Themen, die zumindest akzeptabel sind, da sie noch verwandt sind. Damit erhält man das gute Ergebnis, dass über zwei Drittel der Themen als akzeptabel eingestuft werden.

Die beiden Auswahlverfahren unterscheiden sich in diesen Evaluationsergebnissen nur geringfügig, die Themen des *CombinedProcessor* sind jedoch etwas öfter passend. Dabei sind bei diesem Auswahlverfahren gerade die zuletzt ausgewählten Themen marginal besser. Die ähnlichen Ergebnisse können damit erklärt werden, dass der *CombinedProcessor* im Ansatz so funktioniert, als würde man den *TopConnectivityProcessor* mehrfach anzuwenden: Zunächst wird ein Thema genau wie beim *TopConnectivityProcessor* mit einer hohen Konnektivität ausgewählt. Für die restlichen Themen wird im zweiten Schritt wiederum versucht, für diese ein Thema mit möglichst hoher Konnektivität auszuwählen. Dies entspricht dem Ansatz des *TopConnectivityProcessor*, wenn der Graph auf die fehlenden Bedeutungen beschränkt ist. Die marginal besseren Ergebnisse des *CombinedProcessor* entstehen dann dadurch, dass darüber auch kleinere Themengebiete in den Eingaben erkannt werden können. Die dazugehörigen Knoten haben eigentlich keine hohe Konnektivität, können aber im zweiten Schritt ausgewählt werden.

Die Vergleichwerte aus Tabelle 7.5 und Tabelle 7.6 eignen sich jedoch nicht optimal zur Bestimmung der Qualität der Ansätze. Wenn beispielsweise zwei von drei Personen ein Thema passend fanden und somit das Thema mehrheitlich passend eingestuft wurde, ist der Anteil dennoch nur bei 66,7%. In [HHKG13] wurden daher Begriffe mitsamt Berechnung vorgestellt, mit denen sich die Güte bestimmen lässt, wenn Mehrheitsentscheidungen beachten werden sollen. Dafür wurde zunächst der Begriff *Treffer* eingeführt. Ein Thema ist ein Treffer, wenn mindestens zwei der drei Testpersonen das Thema passend fanden. Damit kann die Präzision und eine Abdeckung der Themen berechnet werden: Die *Präzision@k* in Gleichung 7.3 gibt an, welcher Anteil der ersten k Themen passend sind. Die *Abdeckung@k*, die in Gleichung 7.4 beschrieben ist, zeigt den Anteil an Eingaben an, für

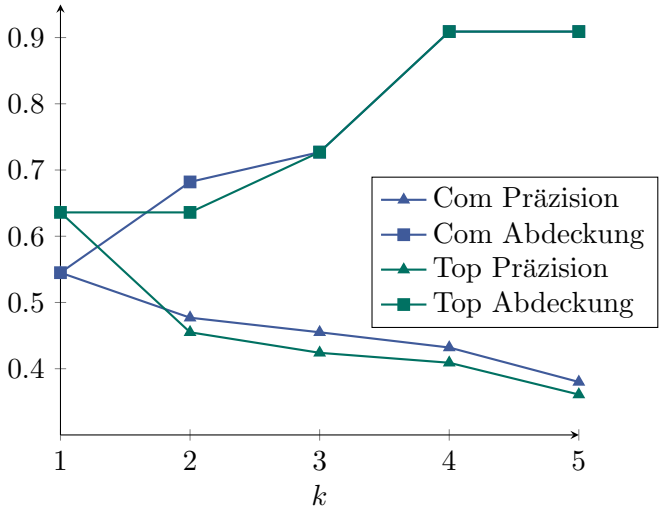

Abbildung 7.1.: Präzision und Abdeckung der ersten k Themen, die als passend eingestuft wurden. Com ist der *CombinedProcessor*, Top der *TopConnectivityProcessor*.

die mindestens ein passendes Thema unter den ersten k Themen enthalten ist.

$$\text{Präzision@}k = \frac{\#\text{Treffer in den ersten } k \text{ Themen}}{k} \tag{7.3}$$

$$\text{Abdeckung@}k = \frac{\#\text{Texte mit min. 1 Treffer in den ersten } k \text{ Themen}}{\#\text{Texte}} \tag{7.4}$$

In Abbildung 7.1 sind die Präzision und die Abdeckung für Werte für k von eins bis fünf für die beiden Varianten *CombinedProcessor* und *TopConnectivityProcessor* abgebildet. Die Linien mit den Dreiecken stellen die Werte für die Präzision dar, die Linien mit den Quadraten zeigen die Abdeckung. Wie zu erwarten sinkt mit steigendem k die Präzision, während die Abdeckung steigt. Beide Varianten verhalten sich dabei ähnlich gut, die Abdeckung ist ab $k = 3$ für beide Varianten gleich. Ab $k = 4$ erreicht die Abdeckung jeweils knapp 91%. Dies zeigt, dass bei mindestens vier Themen bei fast allen Texten mindestens ein passendes Thema ermittelt werden konnte. Bei der Präzision existieren allerdings geringe Unterschiede zwischen den Auswahlverfahren. Die Präzision für das erste Thema liegt für den *TopConnectivityProcessor* mit 63,6% noch über den 54,5% des *CombinedProcessor*. Ab zwei Themen ($k = 2$) liegt die Präzision des *CombinedProcessor* stets etwa zwei bis drei Prozentpunkte über der des *TopConnectivityProcessor*. Bei fünf Themen liegt die Präzision des *CombinedProcessor* bei 37,9%.

Verglichen mit den Werten für Abdeckung und Präzision aus [HHKG13] sind die Werte der Ansätze dieser Arbeit besser. Die Präzision der Ansätze von Hulpus et al. sind für das erste Thema mit maximal 31% erheblich schlechter und sinken bei $k = 5$ auf knapp unter 20%. Die Abdeckung liegt für fünf Themen bei 61%, wodurch die Ergebnisse dieser Arbeit ebenfalls besser sind. Allerdings können die Unterschiede auch von einer unterschiedlichen Interpretation der Begriffe *passend* und *verwandt* kommen. Die Definition der Begrifflichkeiten fand bei Hulpus et al. nicht klar genug statt, um Freiheiten bei der Interpretation auszuschließen. Dennoch zeigen die Werte, dass der Ansatz in dieser Arbeit gute Ergebnisse liefert.

In [HHKG13] wurde zusätzlich eine zweite Auswertung erstellt, für die der Begriff *Treffer* allgemeiner definiert wurde. Ein Thema ist mit dieser neuen Definition dann ein Treffer, wenn mindestens zwei Personen es als passend oder zu allgemein einstufen. Die Ergebnisse

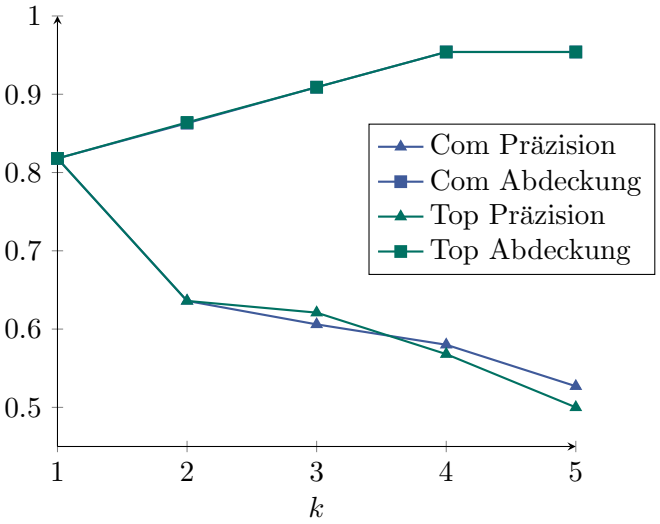

Abbildung 7.2.: Präzision und Abdeckung der ersten k Themen, die als passend oder zu allgemein eingestuft wurden. Com ist der *CombinedProcessor*, Top der *TopConnectivityProcessor*.

der Ansätze dieser Arbeit nach der veränderten Definition sind in Abbildung 7.2 abgebildet. Durch die Inklusion der allgemeinen Themen entstehen entsprechend bessere Werte für Präzision und Abdeckung. In diesem Vergleich sind die Ergebnisse für die beiden Varianten nahezu identisch. Die Abdeckung ist identisch, die Präzision unterschiedet sich für größere k marginal. Auffällig bei der Präzision ist vor allem die hohe Präzision für das erste Thema von knapp 82%, die für $k = 2$ jedoch auf 63,6% abfällt. Für $k = 4$ erreicht die Abdeckung das Maximum von 95,5%. Die Ergebnisse von Hulpus et al. sind bei dieser Auslegung des Begriffs *Treffer* ähnlich. Die Präzision ist für $k = 1$ bei den Ansätzen aus dieser Arbeit etwa 10% besser, die Werte nähern sich jedoch für höhere Werte für k an. Bei den Werten für die Abdeckung sind ab $k = 2$ die Ansätze aus dieser Arbeit und die aus [HHKG13] annähernd gleich. Möchte man möglichst präzise die Themen bestimmen, dann ist diese Messung nicht aussagekräftig. Dennoch zeigt die Messung, dass sich die Ansätze dieser Arbeit mit denen von Hulpus et al. vergleichen lassen. In einigen Fällen kann es allerdings hilfreich sein, auch allgemeinere Themen zu akzeptieren, um vor allem die Themenrichtung zu ermitteln. Dies hilft vor allem, wenn man über die extrahierten Themen die Thematik der Eingabe abgrenzen möchte, wie es unter anderen bei der Ontologieauswahl gebraucht wird. Aber auch in anderen Fällen können allgemeine Themen hilfreich sein, weshalb allgemeine Themen nicht grundsätzlich negativ zu bewerten sind.

In Abschnitt 5.1 wurde bereits festgestellt, dass beim Ansatz [HHKG13], der die Grundlage für diesen Ansatz der Themenextraktion bildet, oftmals zu generelle Themen extrahiert wurden. Deshalb wurde zusätzlich für passende oder verwandte Themen gefragt, ob diese möglicherweise zu allgemein oder zu spezifisch war. Nimmt man die Werte der zu allgemein eingeschätzten passenden und verwandten Themen zusammen, sind beim *CombinedProcessor* 54,69% der Themen zu allgemein und 35,92% zu spezifisch. Beim *TopConnectivityProcessor* sind 41,50% der Themen zu allgemein und 31,04% zu spezifisch. Dies zeigt, dass dieser Ansatz trotz der Änderungen noch immer eher allgemeine Themen extrahiert. Auch beim Vergleich der Graphen aus Abbildung 7.1 und Abbildung 7.2 zeigt sich diese Tendenz, da sich Präzision und Abdeckung wesentlich verbessern, wenn allgemeine Themen akzeptiert werden. Eine Erklärung hierfür lässt sich im Ansatz finden. Es werden Bedeutungs- und Themengraphen erstellt, wobei für die Verbindungen vor allem auch hierarchische Strukturen genutzt werden. Eine Analyse der Themengraphen zeigt, dass neben den Ausgangsbedeutungen vor allem die Knoten der allgemeineren Themen, die in

der Regel eine Anzahl an spezifischeren Themen vereinen, einen höheren Grad besitzen. Ein höherer Grad bedeutet in der Regel auch ein besserer Wert für die Graphzentralität, weshalb diese Knoten auch eher als Themen in Betracht gezogen werden. Weiterhin werden in der Auswahl diejenigen Knoten ausgewählt, die die Bedeutungsgraphen verbinden. Die Verbindung zweier oder mehr Bedeutungen ist oftmals ein allgemeines Thema.

Betrachtet man die Antworten aus der Umfrage zu den Themen detaillierter, so findet man weitere Eigenheiten der Ansätze. So verbessert die Anzahl an verschiedenen Bedeutungen, die als Eingabe der Themenextraktion übergeben werden, im Allgemeinen das Ergebnis. Beispielsweise sind die Texte des ersten Szenarios sehr kurz, wodurch lediglich zwei Bedeutungen für die Themenextraktion genutzt werden können. Die Resultate sind für diese Texte eher unpassend, da die Thematik nur sehr vage abgegrenzt werden kann. Desweiteren verbessern sich meistens die Ergebnisse, wenn der Eingabetext ein homogenes Themenfeld behandelt. Beispielsweise beinhalten die Texte aus Szenario Acht fast ausschließlich Nomen, die thematisch nah am Thema *Wäsche* liegen. Für dieses Szenario wurden überdurchschnittlich viele Themen von den Testpersonen als passend bewertet. Die Präzision liegt beim *TopConnectivityProcessor* beispielsweise für drei Themen bei 100%, für fünf Themen bei 60%. Auf der anderen Seite wird in Szenario Vier über Obst, Säfte, Alkoholika und Küchengeräte gesprochen, was den Ansatz vor eine Herausforderung stellt. Zwar wird mit *Cocktails* ein passendes Thema gefunden, die restlichen Themen sind jedoch weniger passend. Gerade im Bezug auf homogene und inhomogene Eingaben werden die Unterschiede der beiden Varianten ersichtlich. Der *CombinedProcessor* beachtet alle Bedeutungen, was für manche Eingaben vorteilhaft, für andere aber nachteilig sein kann. So kommt in einem Text aus Szenario Acht das Nomen „arms“ vor, da der Roboter mit seinen Armen in die Waschmaschine greifen soll. Bei der Themenfindung werden die Arme mit einbezogen, was für die gesamte Eingabe zu unpassenderen Themen wie *Upper limb anatomy* führt. Im Gegensatz dazu ist der *TopConnectivityProcessor* bei solchen Fällen robuster. Die Variante eignet sich besser, wenn gemeinsame Themen möglichst vieler Bedeutungen gefunden werden sollen und das „Rauschen“ in der Eingabe ignoriert werden soll. Störende Elemente, die unabhängig und im Graphen nicht verbunden sind, werden bei diesem Verfahren oftmals ignoriert. Problematisch wird dies jedoch, wenn die vom *TopConnectivityProcessor* als „Rauschen“ eingestuften Bedeutungen beachtet werden sollen. Beispielsweise behandeln die Texte zu Szenario Elf sowohl das Themengebiet *Heizen* als auch das Themengebiet *Musik*. Hier wird vom *TopConnectivityProcessor* die Thematik *Musik* vernachlässigt, da diese im Vergleich zur Thematik *Heizen* weniger stark in der Eingabe präsent ist. In diesem Fall liefert der *CombinedProcessor* bessere Themen, da von diesem beide Thematiken berücksichtigt werden. Unabhängig vom Auswahlverfahren kann es außerdem passieren, dass ein Themengebiet ein oder mehrere andere Themengebiete bei der Auswahl dominiert. Dadurch können Themengebiete in der Liste der extrahierten Themen weniger stark repräsentiert sein oder sogar gänzlich verdrängt werden.

Zusammenfassend kann man dennoch sagen, dass die Themenextraktion gute Ergebnisse liefert. Die Ergebnisse zeigen, dass der Austausch des LDA-Verfahrens sowie die weiteren Anpassungen keine negativen Auswirkungen haben, sondern das Ergebnis verbessern. Da unter anderem oft zu allgemeine Themen ausgewählt werden, ist jedoch noch Verbesserungspotential vorhanden, worauf in Kapitel 8 kurz eingegangen wird.

7.3. Zuordnung von Themen zu Ontologien

Zuletzt wurde die Zuordnung von Themen zu Ontologien evaluiert. Dafür wurden wie in Abschnitt 7.2 möglichst realistische Eingaben genutzt. Gleichzeitig sollten die Möglichkeiten und Grenzen des Ansatzes ermittelt werden. Mit dem PARSE-Korpus alleine ist dies allerdings nicht möglich, da gerade Fälle, in denen mehrere Ontologien ausgewählt

Tabelle 7.7.: Beschreibung des zur Evaluation genutzten Korpus

Szenario	Synthetisch	#Texte	Domäne(n)
1	Nein	2	Küche
2	Nein	2	Küche
3	Nein	2	Küche
4	Nein	2	Küche
5	Nein	2	Küche
6	Nein	2	Küche
7	Nein	2	Küche
8	Nein	2	Wäsche
E1	Ja	2	Garten
E2	Ja	2	Kinderzimmer
E3	Ja	2	Heizung, Musik
E4	Ja	2	Garten
E5	Ja	2	Bar
E6	Ja	2	Schlafzimmer
E7	Ja	2	Musik
E8	Ja	2	Musik, Bar
E9	Ja	1	Küche, Garten

werden müssen, in diesem nicht existierten. Aus diesem Grund wurden sowohl Transkripte zu den verschiedenen Szenarien aus dem PARSE-Korpus als auch eigene Texte für die Evaluation genutzt, wodurch 33 Eingabetexte für die Evaluation zur Verfügung standen. Eine Übersicht des Korpus findet man in Tabelle 7.7. 16 der Texte sind echte Eingaben aus dem PARSE-Korpus, die in Abschnitt 7.2 bereits zur Evaluation der Themenextraktion genutzt wurden und im Anhang in Abschnitt B.2 zu finden sind. Dazu wurden 17 Texte zusätzlich erstellt, die im Anhang in Abschnitt C aufgeführt sind. Auf den Texten aus diesem Testkorpus wurde der Ansatz zur Themenextraktion angewandt, um mit den resultierenden Themen dann die Ontologieauswahl durchführen zu können. Dabei wurden verschiedene Konfigurationen für die Ontologieauswahl getestet, um den Einfluss der verschiedenen Konfigurationsmöglichkeiten auf die Auswahl feststellen zu können.

In der folgenden Evaluation soll außerdem lediglich die Auswahl der Umgebungsontologien gemessen werden. Bereits in Abschnitt 5.2.2 wurde diskutiert, dass die Zuordnung von Akteuren problematisch ist und es wurde ein Ansatz präsentiert, der die Zuordnung zu den Akteuren verbessern soll. Dabei sollte anhand der Fähigkeit der Akteure zur Interaktion mit den ausgewählten Umweltontologien die Auswahl beeinflusst werden. Bei den getesteten Eingaben war dieser Ansatz jedoch nur begrenzt zielbringend. Zwar wurde mit diesem Ansatz mindestens ein Akteur ausgewählt, der mit der Umwelt agieren kann. In fast jedem Fall wird aber die Akteur-Ontologie über den Haushaltsroboter ARMAR ausgewählt, da dieser in einer Vielzahl an Umgebungen agieren kann. In 94% der Fälle wurde die Ontologie für den Haushaltsroboter ausgewählt, lediglich vier Mal wurde die Ontologie des virtuellen Assistenten ausgewählt. Die anderen beiden Akteur-Ontologien wurden gar nicht ausgewählt. Die Ontologie über ARMAR dominiert dementsprechend die anderen Akteur-Ontologien. Daraus kann geschlossen werden, dass die Auswahl von Akteur-Ontologien über die Themen der Eingabe, wie sie in dieser Arbeit ermittelt werden, nicht funktioniert. Mit dem Ansatz kann lediglich festgestellt werden, welche Akteure für die ausgewählten Ontologien die besten sind, aber nicht, welche Akteur-Ontologie für die Eingabe die beste ist.

Um die Güte der Zuordnung von Umgebungsontologien feststellen zu können, wurde ein

Tabelle 7.8.: Verschiedene Konfigurationen für die Ontologieauswahl und die Themenextraktion (TE) und die jeweils resultierende Präzision und Ausbeute zusammen mit den F_1- und F_2-Werten

TE	Ontologieauswahl		Metriken			
#Themen	Schwellwertfaktor	#Themen	Präzision	Ausbeute	F_1	F_2
2*n	0,10	5	91,89%	89,47%	90,67%	89,94%
2*n	0,15	5	80,95%	89,47%	85,00%	87,63%
2*n	0,20	5	70,83%	89,47%	79,07%	85,00%
5	0,10	5	85,00%	89,47%	87,18%	88,54%
5	0,15	5	79,07%	89,47%	83,95%	87,18%
5	0,20	5	76,09%	92,11%	83,33%	88,38%
2*n	0,10	10	77,78%	92,11%	84,34%	88,32%
5	0,10	10	79,01%	89,47%	83,95%	87,18%

Goldstandard für die Auswahl der passenden Ontologien erstellt. Die für eine Eingabe gefundenen Ontologien können in zwei Gruppen unterteilt werden: benötigte und zusätzliche Ontologien. Zu den benötigten Ontologien gehören alle Ontologien, die für die Eingabe notwendig sind, um alle benötigten Informationen zur Eingabe bereitzustellen. Die benötigten Ontologien entsprechen damit korrekt ausgewählten Ontologien. Zu den zusätzlichen Ontologien gehören alle anderen ausgewählten Ontologien, die entsprechend fälschlicherweise ausgewählt wurden.

Bei der Evaluation wurden verschiedene Konfigurationen getestet, wobei an drei Einstellungsmöglichkeiten Veränderungen stattfanden. Zunächst wurde die Anzahl an Themen, die bei der Themenextraktion für den Eingabetext ermittelt werden, verändert. Die Evaluation der Themenextraktion in Abschnitt 7.2 ergab, dass die Präzision der Themen mit der Anzahl abnimmt, weshalb diese Eigenschaft hier getestet werden sollte. Hier wurde einerseits die Standardmenge der Themen getestet, die der doppelten Menge der Bedeutungen in der Eingabe entspricht (2*n). Andererseits wurde die Festlegung auf fünf Themen getestet, für die sich in der Evaluation eine Abdeckung von über 90% ergab. Außerdem fand eine Anpassung der Konfigurationen beim Schwellwertfaktor statt, wie es in Abschnitt 5.2.2 angesprochen wurde. Der Schwellwertfaktor bestimmt, welcher Anteil des besten Übereinstimmungswertes abgezogen wird, um den Schwellwert zu bestimmen Mit dem Schwellwert wird dann ermittelt, welche Ontologien ausgewählt werden. Bei ersten Tests ergaben sich aus den Beobachtungen der Übereinstimmungswerte ein geeigneter Schwellwertfaktor von 15%, weshalb sowohl dieser als auch der etwas höheren Schwellwertfaktor von 20% und der etwas niedrigere Schwellwertfaktor 10% ausgewählt wurden. Gleichzeitig kann mit diesen Schwellwerten auch überprüft werden, welche Auswirkungen die Verdopplung des Schwellwertfaktors von 10% auf 20% hat. Zuletzt wurde noch getestet, wie die Anzahl an Themen, die den Ontologien zugewiesen werden die Ergebnisse beeinflusst. Da in der Ontologie eine Vielzahl von Bedeutungen enthalten sind, eignet sich die Variante nicht, bei der doppelt so viele Themen extrahiert werden wie Bedeutungen übergeben wurden. Daher wurde zunächst mit fünf Themen evaluiert, um zu überprüfen, ob bei der gleichen Menge an Themen für Ontologie und Eingabetext gute Ergebnisse geliefert werden. Zusätzlich wurde die Anzahl auf zehn Themen verdoppelt, um wiederum die Auswirkungen der Verdopplung an Themen zu evaluieren. Gerade im Bezug auf das in der Evaluation der Themenextraktion in Abschnitt 7.2 gemessene Verhalten der sinkenden Präzision bei steigender Abdeckung wurden dadurch bessere Werte für die Ausbeute erwartet.

Eine Übersicht der Ergebnisse der Evaluation der Ontologieauswahl bei den verschiedenen Konfigurationskombinationen ist in Tabelle 7.8 aufgeführt. Dort findet man in den ersten drei Spalten die verschiedenen Konfigurationsoptionen, in den nächsten vier Spalten

sind die Ergebnisse nach verschiedenen Metriken, die in Abschnitt 2.2 erklärt wurden, zu sehen. Zusätzlich zum F_1-Wert, der zur Kombination der Ergebnisse zur Präzision und Abdeckung berechnet wurde, ist der F_2-Wert berechnet worden, bei dem die Ausbeute stärker gewichtet wird. Für die Ontologieauswahl ist die Ausbeute wichtiger ist als die Präzision, da mit hoher Ausbeute sichergestellt werden kann, dass alle benötigten Ontologien und dadurch alle benötigten Informationen vorhanden sind. Dennoch kann die Präzision nicht vernachlässigt werden, da bei schlechterer Präzision auch die Gefahr steigt, falsche Ontologien auszuwählen und damit falsche Informationen zu erhalten. Daher ist die gewichtete Berechnung mit dem F_2-Maß sinnvoll.

In den Ergebnissen ist zu erkennen, dass bei größeren Schwellwertfaktoren die Präzision sinkt. Wie zu erwarten beeinflusst das Zulassen einer größeren Fehlertoleranz die Präzision negativ. Gleichzeitig steigt die Ausbeute, wobei diese langsamer steigt als die Präzision sinkt, was sich im sinkenden F_1-Wert widerspiegelt. Eine Verdopplung des Schwellwertfaktors verringert die Präzision erheblich, während die Ausbeute gar nicht oder nur wenig steigt. Die Anzahl der Themen für die Eingabetexte auf fünf Themen zu begrenzen anstatt die Standardmenge zu nutzen, hat auf die Ausbeute keinen Einfluss. Für die Berechnung der Übereinstimmungswerte wird jeweils die Nähe der verschiedenen Themen der Eingabetexte zum nächsten Thema der Ontologie genutzt. Wenn durch die Änderung der Anzahl der Themen keine Themen zu einem neuen Themengebiet gefunden werden, ändert sich die Ausbeute entsprechend nicht. Änderungen in der Anzahl der Themen für die Eingabetexte haben jedoch einen Einfluss auf die Präzision der Ontologieauswahl, die bei der Standardmenge höher als bei lediglich fünf Themen ist. Dies lässt sich unter anderem dadurch erklären, dass einzelne Themen, durch die auf falsche Ontologien geschlossen werden könnte, durch die größere Menge weniger ins Gewicht fallen. Bei der Anpassung der Anzahl an Themen, die einer Ontologie zugeordnet werden, ergeben sich auch geringfügige Änderungen. Bei der Verdopplung auf zehn Themen verschlechtert sich die Präzision deutlich, während die Ausbeute nur bei der Standardmenge an extrahierten Themen marginal steigt, was sich auch in den F_1- und F_2-Werten abzeichnet.

Zusätzlich zur Analyse der Präzision und Ausbeute wurde noch die Auswahl der falschen Ontologien untersucht. In Tabelle 7.9 finden sich die Ergebnisse zur Ausfallrate, die in Gleichung 2.11 aus Abschnitt 2.2 eingeführt wurde. Hier ist zu Erkennen, dass die Ausfallraten im Schnitt ziemlich gering ist und im schlimmsten Fall bei 6,19% liegt. Bei der genaueren Betrachtung der fälschlicherweise ausgewählten Ontologien kann man außerdem feststellen, dass einige dieser Ontologien nicht problematisch sind. Sie sind für die Eingabe austauschbar mit der eigentlich korrekten Ontologie und stellen lediglich nicht benötigtes zusätzliches Wissen bereit, weshalb sie einen unproblematischen Fehler darstellen. Durch die Auswahl dieser Ontologien werden keine falschen Informationen oder ähnliches eingeführt, durch die die Übersetzung der Anweisungen aus der Eingabe in Quellcode gefährdet wird. Gleichzeitig behandeln diese Ontologien ähnliche Themen und besitzen gleiche oder äquivalente Konzepte, damit die benötigten Informationen zur Verfügung gestellt werden können. Beispielsweise sind die Konzepte *OrangeJuice* und *Fridge* in den Ontologien über die Küche und die Bar gleich, weshalb für diese Konzepte die Ontologien ausgetauscht werden können. Aus diesem Grund wurde die Anteile der unproblematischen Ontologien unter den fälschlicherweise ausgewählten Ontologien ermittelt, die ebenfalls in Tabelle 7.9 zu finden ist. Darin erkennt man, dass bis zu 40% der fälschlicherweise ausgewählten Ontologien unproblematisch sind. Dieser Wert ist neben Präzision und Ausbeute ein weiterer Faktor, der beim Vergleich der verschiedenen Konfigurationen eine Rolle spielt. Bei ähnlich guten Werten für Präzision und Ausbeute sind diejenigen Konfigurationen zu bevorzugen, die einen höheren Anteil an unproblematischen Ontologien besitzen.

Mit diesen Informationen kann nun die optimale Konfiguration bestimmt werden. Wie bereits diskutiert, ist die Ausbeute wichtiger als die Präzision, weshalb der F_2-Wert für

Tabelle 7.9.: Verschiedene Konfigurationen für die Ontologieauswahl und die Themenextraktion (TE) und die jeweils resultierende Ausfallrate sowie der Anteil unproblematischer Ontologien unter den fälschlicherweise annotierten Ontologien

TE	Ontologieauswahl		Metriken	
#Themen	Schwellwertfaktor	#Themen	Ausfallrate	Anteil unprobl. Ont.
2*n	0,10	5	1,33%	0,00%
2*n	0,15	5	3,54%	0,00%
2*n	0,20	5	6,19%	14,29%
5	0,10	5	2,65%	33,00%
5	0,15	5	3,98%	33,00%
5	0,20	5	4,87%	18,18%
2*n	0,10	10	4,42%	40,00%
5	0,10	10	3,98%	33,33%

die Bestimmung der optimalen Konfiguration genutzt werden soll. Die optimalen Konfiguration anhand der F_2-Werte erhält man bei der Standardmenge an Themen aus der Themenextraktion in Kombination mit einem Schwellwertfaktor von 0,10 und fünf Themen, die den Ontologien zugewiesen werden. Ähnlich gut ist die selbe Konfiguration mit der Änderung, dass fünf Themen in der Themenextraktion extrahiert werden sollen. Der F_2-Wert ist etwas geringer, jedoch ist bei dieser Konfiguration die Quote an unproblematischen Ontologien bei eine Drittel, weshalb diese Konfiguration eine ebenbürtige Alternative darstellt.

Neben der Betrachtung der Gesamtergebnisse für die Ontologieauswahl wurde auch eine detailliertere Analyse der Ergebnisse durchgeführt. In Tabelle 7.10 sind die Ergebnisse pro Eingabetext für die Ontologieauswahl mit fünf extrahierten Themen bei der Themenextraktion, dem Schwellwertfaktor 0,10 und jeweils fünf zugeordneten Ontologiethemen. Bei diesen Ergebnissen erkennt man, dass für alle Eingabetexte, die lediglich eine Ontologie benötigen, diese auch ausgewählt wird. Problematisch sind diejenigen Eingabetexte, die mehrere Ontologien benötigen. In der Evaluation sind von fünf Eingabetexten, die mehrere Ontologien benötigen, lediglich bei einem alle Ontologien ausgewählt worden. Bei mehreren benötigten Ontologien passiert es oftmals, dass eine benötigte Ontologie einen hohen Übereinstimmungswert erzielt, die Übereinstimmungswerte der anderen Ontologie jedoch geringer ausfallen. Dabei können die Werte unterhalb des Schwellwertes zur Ontologieauswahl fallen. An dieser Stelle könnte man versuchen, über einer Erhöhung des Schwellwertfaktors die anderen benötigten Ontologien zu erhalten. Dabei würde man jedoch gleichzeitig die Präzision senken und die obige Evaluation ergaben, dass sich dadurch nur geringfügige Verbesserungen ergeben.

Um die Auswirkungen bei mehreren benötigten Ontologien gerade im Bezug auf fehlenden Ontologien zu untersuchen, wurde zusätzlich ermittelt, wie viele Bestandteile der Eingabe wie etwa Objekte, zu denen Informationen aus der Ontologie benötigt werden, sich in den ausgewählten Ontologien befinden. Damit kann abgeschätzt werden, wie sehr sich die fehlenden Ontologien auswirken. Für die Eingabetexte mit mehreren benötigten Ontologien ist diese Analyse in Tabelle 7.11 zu finden. In den unteren beiden Zeilen finden sich die Durchschnittswerte für die Eingabetexte mit den fehlenden Ontologien, sowie zum Vergleich die Durchschnitte für alle Eingabetexte. Gerade in Text E3.1 sind mehr Konzepte der ausgewählten Ontologie vertreten, da drei von vier Nomen der Eingabe dieser Ontologie zugeordnet werden konnten. Entsprechend einseitiger ist die Ontologieauswahl und die andere auszuwählende Ontologie wird nicht ausgewählt, da der Übereinstimmungswert zu gering ist.

Tabelle 7.10.: Ergebnisse der Ontologieauswahl pro Eingabetext und Gesamt mit der Konfiguration mit fünf Themen, Schwellwertfaktor 0,1 und jeweils fünf Ontologiethemen

	benötigte Ont.		zusätzliche Ont.		Metriken	
Text	gefunden	benötigt	gesamt	unprob.	Präzision	Ausbeute
1.1	1	1	0	0	100%	100%
1.2	1	1	2	1	33%	100%
2.1	1	1	0	0	100%	100%
2.2	1	1	0	0	100%	100%
3.1	1	1	0	0	100%	100%
3.2	1	1	0	0	100%	100%
4.1	1	1	1	0	50%	100%
4.2	1	1	1	0	50%	100%
5.1	1	1	0	0	100%	100%
5.2	1	1	0	0	100%	100%
6.1	1	1	0	0	100%	100%
6.2	1	1	0	0	100%	100%
7.1	1	1	0	0	100%	100%
7.2	1	1	0	0	100%	100%
8.1	1	1	0	0	100%	100%
8.2	1	1	1	0	50%	100%
E1.1	1	1	0	0	100%	100%
E1.2	1	1	0	0	100%	100%
E2.1	1	1	0	0	100%	100%
E2.2	1	1	0	0	100%	100%
E3.1	1	2	0	0	100%	50%
E3.2	1	2	0	0	100%	50%
E4.1	1	1	0	0	100%	100%
E4.2	1	1	0	0	100%	100%
E5.1	1	1	0	0	100%	100%
E5.2	1	1	0	0	100%	100%
E6.1	1	1	0	0	100%	100%
E6.2	1	1	0	0	100%	100%
E7.1	1	1	0	0	100%	100%
E7.2	1	1	0	0	100%	100%
E8.1	1	2	0	0	100%	50%
E8.2	2	2	0	0	100%	100%
E9.1	1	2	1	0	50%	50%
Summe	34	38	5	1	85,00%	89,47%
Durchschnitt	1,03	1,15	0,18	0,03	85,00%	89,47%

Tabelle 7.11.: Analyse der Eingabetexte, die mehrere Ontologien benötigen. Fokus aus die Anzahl der Bestandteile, zu denen Informationen aus der Ontologie benötigt werden.

Text	vorhanden	benötigt	Ausbeute
E3.1	3	4	75%
E3.2	1	2	50%
E8.1	3	6	50%
E8.2	3	3	100%
E9.1	3	6	50%
Durchschnitt	2,6	4,2	61,90%
Gesamtdurchschnitt	5,51	5,76	95,79%

Eine Erklärung für die Problematik der fehlenden Ontologien, wenn mehrere Domänen vertreten sind, lässt sich in der Themenextraktion finden. In Abschnitt 7.2 wurde festgestellt, dass Themengebiete in der resultierenden Themenliste unterrepräsentiert sein können. Ist ein Themengebiet unterrepräsentiert, dann erhält die zugehörige Umgebungsontologie einen entsprechend niedrigeren Übereinstimmungswert. Der Ansatz zur Ontologieauswahl setzt darauf, dass Ontologien, die ausgewählt werden sollen, einen entsprechend hohen Übereinstimmungswert besitzen, während andere Ontologien einen sehr geringen Wert erhalten. Dadurch entsteht zum einen das Problem, dass durch die Unterrepräsentation ein zu großer Unterschied zwischen den Übereinstimmungswerten existiert. Weiterhin gibt es das Problem der Prämisse des Ansatzes, dass die Umgebungsontologien sehr unterschiedliche Themen behandeln und sich nicht überschneiden, wodurch ein Thema auch nur zu einer Ontologie einen hohen Wert erhalten würde. Dies trifft aber nicht zu, das es oftmals thematische Überschneidungen in den Ontologien gibt. So befindet sich unter anderem das Themengebiet *Möbel* jeweils in den Ontologien über die Küche, das Schlafzimmer, das Kinderzimmer, den Garten und die Bar. Dadurch erhalten mehrere Ontologien einen höheren Übereinstimmungswert, wenn im Eingabetext Möbel vorkommen, was die Auswahl schwierig macht. Um dem entgegenzugehen könnte man versuchen darauf zu achten, Ontologien strikt nach Themen zu trennen. Dies ist jedoch sehr aufwändig und würde die Erstellung neuer Ontologien zu stark einschränken. Deshalb sollte eine andere Herangehensweise gefunden werden.

Aus diesen Ergebnissen kann zumindest die Voraussetzung formuliert werden, die für eine zuverlässigere Auswahl bei Eingabetexten mit mehreren benötigten Ontologien erfüllt sein sollte: Die Themen, die aus dem Eingabetext extrahiert werden, sollten möglichst gleichmäßig verteilt sein, sodass die dazugehörigen Ontologien jeweils ähnlich gute Übereinstimmungswerte erzielen können und keine Ontologie verdeckt wird. Dies ließe sich durch eine bessere und ausgeglichenere Themenextraktion sicherstellen, für die eine Idee in Kapitel 8 präsentiert wird.

Zusammenfassend lässt sich sagen, dass die Ontologieauswahl für Umgebungsontologien gut funktioniert, wenn lediglich eine Ontologie ausgewählt werden soll, was den schnellen und einfachen Wechsel der Einsatzgebiete des Systems ermöglicht. Eine Grenze des Ansatzes sind jedoch Texte, für die mehrere verschiedene Ontologien ausgewählt werden sollen. Eine korrekte Auswahl aller benötigten Ontologien ist nicht zuverlässig und hängt davon ab, ob die Themen gleichmäßig genug auf die verschiedenen Themengebiete verteilt sind, sodass zu allen benötigten Ontologien ähnlich gute Übereinstimmungswerte ermittelt werden können. Eine weitere Grenze des Ansatzes stellt die Auswahl von Akteur-Ontologien dar. Mit diesem Ansatz kann nicht festgestellt werden, welcher Akteur für die Eingabe der beste ist, da Akteure nicht in den extrahierten Themen repräsentiert sind. Daher sollte für die Bestimmung des Akteurs ein alternativer Ansatz entwickelt werden.

8. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde ein Ansatz entwickelt, mit dem Themen aus einem Eingabetext für die Programmierung in natürlicher Sprache extrahiert werden können. Zwar gibt es bereits einige Ansätze auf diesem Gebiet, diese sind jedoch nicht für die Programmierung in natürlicher Sprache ausgelegt. Nach [MCCS09] betrachtet eine gute Themenextraktion nicht nur die Eingabe an sich, sondern versucht über zusätzliche Informationen aus anderen Quellen bessere Themen zu ermitteln. Deshalb wurde ein Ansatz gewählt, in dem semantische Informationen zu den Schlüsselbegriffen der Eingabe aus der Wikipedia extrahiert und zur Ermittlung von Themen genutzt werden. Dafür musste zunächst ein Ansatz gefunden werden, mit dem die Nomen des Eingabetextes disambiguiert und zu Wikipedia-Artikeln zugeordnet werden können. Dafür wurde mit Artikeln aus der Wikipedia ein Trainingskorpus erstellt, da über die Querverweise in den Artikeln der Wikipedia eine explizite Auflösung von Mehrdeutigkeiten stattfindet. Mit dem Trainingskorpus konnte ein Naive-Bayes-Klassifikator zur Auflösung der Mehrdeutigkeiten von Nomen trainiert werden. Die Evaluation der Güte des Klassifikators auf dem PARSE-Korpus, in dem Transkripte von Anweisungen von Menschen an den Haushaltsroboter ARMAR in verschiedenen Situationen gesammelt sind, ergab eine Genauigkeit des Klassifikators von 88,03%. Mithilfe des Klassifikators können die Bedeutungen der Nomen in Form von Titeln von Wikipedia-Artikeln zugewiesen werden. Diese Informationen können dann für die Themenextraktion genutzt werden. In dieser werden Verbindungen zwischen den Konzepten innerhalb der Wikipedia ausgenutzt, um aus den Bedeutungen aus dem Eingabetext Bedeutungsgraphen zu erstellen, wobei jeder Knoten der Graphen ein Konzept aus der Wikipedia darstellt. Die Bedeutungsgraphen werden dann zu einem Themengraphen verschmolzen und die zentralen Knoten zwischen den Bedeutungen extrahiert. Die extrahierten Knoten beziehungsweise die Wikipedia-Konzepte, für die sie stehen, stellen schließlich die Themen dar. Zur Evaluation des Ansatzes wurde eine Umfrage erstellt, bei der die Teilnehmer bewerten sollten, ob zu einem Text aus dem Bereich der Programmierung in gesprochener Sprache die extrahierten Themen jeweils passend sind. Die Resultate dieser Evaluation sind vielversprechend. Bei vier oder mehr extrahierten Themen ist in über 91% der Fälle mindestens ein passendes Thema enthalten. Gleichzeitig wurden bis zu 63,6% der ersten Themen als passend eingestuft. Mit diesen Ergebnissen ist dieser Ansatz vergleichbar mit dem Ansatz von Hulpus et al. in [HHKG13], deren Ansatz als Orientierung genutzt wurde. In einigen Bereichen wurden sogar bessere Werte erzielt. Ein Vorteil des Ansatzes dieser Arbeit ist unter anderem, dass er unüberwacht abläuft, wodurch nicht extra ein Trainingskorpus annotiert werden muss. Zusätzlich basiert er auf Daten aus der Wikipedia, die ständig verbessert wird, wodurch sich auch die Ergebnisse des Ansatzes verbessern könnten.

Als konkreten Anwendungsfall der extrahierten Themen wurde ein Ansatz zur Auswahl passender Ontologien entwickelt. Dies soll ermöglichen, statt einer großen Ontologie, in der Informationen über alles enthalten sein muss, kleinere Domänenontologien zu erstellen und einzusetzen. Aus diesem Grund wurde die existierende Ontologie aus PARSE analysiert. Daraus resultierte ein Regelsatz zur Erstellung von domänenspezifischen Ontologien, um die Erstellung von neuen Ontologien zu vereinfachen und zu vereinheitlichen. Eine dieser Regeln ist die Unterscheidung von Ontologien in Akteur-Ontologien und Umgebungsontologien. Akteur-Ontologien repräsentieren das handelnde System, während Umgebungsontologien die Umgebung, in der eine Ontologie eingesetzt werden soll, abbilden. Nach dem Erstellen der Regeln wurden vier Akteur-Ontologien und achte Umgebungsontologien erstellt. Um für die Eingabetexte passende Ontologien auswählen zu können, wurde zunächst ein Ansatz zur Extraktion von Themen aus Ontologien entwickelt, bei dem der Ansatz der Themenextraktion für Texte wiederverwendet werden konnte. Daraufhin konnten die Themen des Eingabetextes jeweils mit den Themen einer Ontologie verglichen werden, wofür ein Ansatz zur Ermittlung der Übereinstimmung entworfen wurde. Mit Hilfe der Übereinstimmung der Themen aus dem Eingabetext zu den verschiedenen Ontologien konnten dann passende Ontologien ausgewählt werden. Dabei zeigte die Evaluation der Zuordnung von Umgebungsontologien, dass dieser Ansatz sehr gute Resultate erzielt, wobei F_1-Werte von bis zu 90,67% und F_2-Werte von bis zu 89,94% erreicht werden konnten. Die Ergebnisse zeigten jedoch auch, dass dieser Ansatz für Eingaben, zu denen mehr als eine Ontologie zugeordnet werden sollte, nicht gut geeignet ist. Bei lediglich einem von fünf solchen Fällen konnten alle Ontologien ausgewählt werden. An dieser Stelle ist noch Potential für Verbesserungen vorhanden.

Auch für anderen Teile dieser Arbeit gibt es Potential für Verbesserungen, obwohl die Ergebnisse jeweils vielversprechend sind. So werden bei der Auflösung von Mehrdeutigkeiten keine zusammengesetzten Nomen aufgelöst. In Abschnitt 5.1.1 wurde bereits angemerkt, dass dafür eine Erkennung von zusammengesetzen Nomen benötigt wird. Hier könnte man eventuell die Auflösung von Mehrdeutigkeiten und die Erkennung von zusammengesetzen Nomen verbinden. Man könnte einen Klassifikator trainieren, der für zwei Nomen inklusive ihrer Bedeutungen sowie Informationen zum Kontext die Bedeutung des zusammengesetzen Nomens der Bestandteile zurückliefert oder zurückgibt, dass keine Bedeutung eines zusammengesetzen Nomens wahrscheinlich ist. Dieser Klassifikator könnte dann jeweils befragt werden, wenn zwei Nomen in der Eingabe nebeneinander vorkommen.

Die Ergebnisse der Themenextraktion sind zwar ebenfalls zufriedenstellend, allerdings ist die Güte der extrahierten Themen nicht optimal. Bei fünf extrahierten Themen pro Eingabetext wurden in der Evaluation unter 40% der Themen als treffend klassifiziert. Um dies zu verbessern könnte man versuchen, durch Änderungen in den genutzten Verbindungen oder durch eine Filterung bestimmter Knoten die Erstellung der Themengraphen anzupassen. Weiterhin kann versucht werden, die Berechnung der Graphzentralität zu verändern oder für die Auswahl der Knoten andere Kriterien als die Konnektivität und Graphzentralität zu verwenden. Ebenso gibt es Verbesserungspotential für Texte, bei denen mehr als ein Themengebiet vorhanden ist. Bei diesen Texten ist in einigen Fällen ein Themengebiet dominant, während andere Themengebiete eher unterrepräsentiert sind. Man könnte versuchen den Themengraphen in zusammengehörige Gruppen zu unterteilen, die jeweils ein Themengebiet darstellen. Auf den jeweiligen Gruppen könnte dann die Themenextraktion getrennt stattfinden und die Ergebnisse der Gruppen könnten dann kombiniert werden. Damit könnte man auch sicherstellen, dass die Anzahl der Themen pro Themengebiet zwischen den Themen ausgeglichen ist, da so für jede Gruppe die gleiche Anzahl an Themen extrahiert werden könnte. Dies würde der Ontologieauswahl helfen, da damit die Auswahl mehrerer Ontologien verbessert wird.

Die Auswahl der Umgebungsontologien funktioniert mit dem Ansatz aus dieser Arbeit gut, für Akteurontologien ist dies jedoch nicht der Fall. Dies liegt auch unter anderem daran, dass für den Ansatz der Themenextraktion aus dieser Arbeit keine Eigennamen oder Verben genutzt werden. Diese liefern aber oftmals am ehesten Hinweise auf den Akteur. In dieser Arbeit wurde versucht, das Problem der fehlenden Informationen über den Akteur zu lösen, indem über einen Abgleich der Datentypen der Akteure mit den vorkommenden Objekten in den Umgebungsontologien der passendste Akteur ermittelt wird. Diese Behelfslösung kann aber in den wenigsten Fällen zuverlässige Ergebnisse liefern. An dieser Stelle kann man überlegen, ob eine automatische Ermittlung des Akteurs benötigt wird. In der Regel spricht man das System direkt an, wodurch das Zielsystem automatisch bekannt ist. Wenn das Zielsystem bekannt ist, ist die Auswahl der Akteurontologie trivial. Eine Ermittlung des Zielsystems ist lediglich notwendig, wenn tatsächlich mehrere Systeme simultan angesprochen werden sollen. Hier könnte man jedoch so verfahren, dass man über die Verben und eventuell den Kontext versucht, unabhängig von der Themenauswahl den Akteur zu bestimmen. Wenn der Akteur unabhängig von der Ontologieauswahl ermittelt wird, könnte die Auswahl an Umgebungsontologien sogar verbessert werden, indem man Informationen über den Akteur bei der Ermittlung der Übereinstimmung einfließen lässt. Beispielweise könnten so Umgebungsontologien ausgeschlossen werden, die nicht mit dem Akteur kompatibel sind, da der Akteur mit den jeweiligen Umgebungen nicht kompatibel ist.

Auch das Verschmelzen der gefundenen Ontologien kann weiter verbessert werden. Die gefundenen Ontologien werden mit einem einfachen Ansatz verschmolzen, bei dem lediglich die Strukturen der übergebenen Ontologien kombiniert werden, wobei gleichnamige Elemente aufeinander abgebildet werden. Da strikt die Regeln zur Erstellung von Ontologien eingehalten wurden und manuell auf Konformität geachtet wurde, ist der einfache Ansatz für das Verschmelzen der Ontologien dieser Arbeit angemessen. Für komplexere Strukturen und einer größeren Anzahl an Ontologien empfiehlt sich jedoch beim Verschmelzen der Ontologien auch semantische Übereinstimmungen zwischen den Ontologien zu ermitteln, um auf diese Weise ein möglichst optimales Ergebnis zu erhalten. Einige Ansätze aus dem Forschungsbereich der Ontologieabbildungen nutzen semantischen Informationen und sind entsprechend vielversprechender als der einfache Ansatz aus dieser Arbeit. Diese Ansätze hätten voraussichtlich die Ergebnisse der Verschmelzung der eingesetzten Ontologien nicht erheblich verbessert und konnten außerdem aus zeitlichen Gründen nicht umgesetzt werden. Man könnte jedoch die Konzepte der Ansätze zur Extraktion und Nutzung von semantischen Information für den Ontologieabgleich nutzen, um das Verschmelzen besser und robuster zu gestalten.

Neben der in diese Arbeit umgesetzten Auswahl von Ontologien gibt es noch weitere Gebiete und Anwendungen, bei denen die Themenextraktion eingesetzt werden könnte. Durch die Zuordnung von Themen zu einer Eingabe wird die Domäne der Eingabe bestimmt. Wenn die Domäne bekannt ist, kann dies die Ergebnisse anderer Ansätze verbessern und neue Möglichkeiten eröffnen. Ein Beispiel hierfür wäre die Auflösung von Mehrdeutigkeiten, die auch in dieser Arbeit angegangen wurde. Wenn das Themengebiet bekannt ist, könnte bei der Auflösung der Mehrdeutigkeit überprüft werden, ob eine der Bedeutungen der Domäne zugeordnet werden kann. Dies würde die Wahrscheinlichkeit für diese Bedeutung erhöhen. Dadurch könnten letztendlich falsch aufgelöste Mehrdeutigkeiten korrigiert werden. Dies wiederum könnte sich wieder positiv auf die Themenextraktion auswirken, da dadurch fehlerhafte Bedeutungen reduziert werden, wodurch die Themen etwas präzisiert werden könnten. Aber auch bei anderen Ansätze können sowohl die Themenextraktion als auch die Auflösung von Mehrdeutigkeiten eingesetzt werden, um diese zu verbessern. Neben der Verbesserung bestehender Ansätze kann die Themenextraktion auch neue Ansätze und Herangehensweisen anstoßen. Zum Beispiel werden aktuell Informationen zu Akteuren

und Umgebungen vor allem aus den Ontologien speziell für PARSE extrahiert. Hier könnte man versuchen, weitere Informationen über den Kontext der Anweisungen aus anderen Informationsquellen zu extrahieren, die sich mit den Themen beschäftigen. Eine naheliegende Option wäre weitere Informationen aus DBpedia zu extrahieren, da die Themen Konzepte aus DBpedia sind. Eine weitere Option wäre, die extrahierten Themen zu den Mikrotheorien von Cyc [Cyc], der allgemeinen Ontologie über das Weltwissen, zuzuordnen. Diese Mikrotheorien vereinen Konzepte und Fakten zu einem bestimmten Wissensbereich, wodurch weiteres Kontextwissen gewonnen werden könnte, das bei der Übersetzung von Sprache zu Quelltext hilft.

Literaturverzeichnis

[ARA+06] Asfour, Tamim ; Regenstein, Kristian ; Azad, Pedram ; Schroder, J ; Bierbaum, Alexander ; Vahrenkamp, Nikolaus ; Dillmann, Rüdiger: ARMAR-III: An integrated humanoid platform for sensory-motor control. In: *Humanoid Robots, 2006 6th IEEE-RAS International Conference on* IEEE, 2006, S. 169–175 (zitiert auf Seite 15).

[BBGV03] Beneventano, Domenico ; Bergamaschi, Sonia ; Guerra, Francesco ; Vincini, Maurizio: Synthesizing an integrated ontology. In: *IEEE Internet Computing* 7 (2003), Nr. 5, S. 42 (zitiert auf Seite 25).

[Blu04] Blunsom, Phil: Hidden markov models. In: *Lecture notes, August* 15 (2004), S. 18–19 (zitiert auf Seite 7).

[BNJ03] Blei, David M. ; Ng, Andrew Y. ; Jordan, Michael I.: Latent dirichlet allocation. In: *Journal of machine Learning research* 3 (2003), Nr. Jan, S. 993–1022 (zitiert auf Seite 19).

[BP63] Bayes, Mr. ; Price, Mr: An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfrs. In: *Philosophical Transactions (1683-1775)* (1763), S. 370–418 (zitiert auf Seite 4).

[Bri92] Brill, Eric: A Simple Rule-based Part of Speech Tagger. In: *Proceedings of the Third Conference on Applied Natural Language Processing*. Stroudsburg, PA, USA : Association for Computational Linguistics, 1992 (ANLC '92), 152–155 (zitiert auf Seite 7).

[CAS09] Cruz, Isabel F. ; Antonelli, Flavio P. ; Stroe, Cosmin: AgreementMaker: efficient matching for large real-world schemas and ontologies. In: *Proceedings of the VLDB Endowment* 2 (2009), Nr. 2, S. 1586–1589 (zitiert auf Seite 24).

[Cha97] Charniak, Eugene: Statistical techniques for natural language parsing. In: *AI magazine* 18 (1997), Nr. 4, S. 33 (zitiert auf Seite 7).

[CLRS01] Cormen, Thomas H. ; Leiserson, Charles E. ; Rivest, Ronald L. ; Stein, Clifford: Section 22.1: Representations of graphs. In: *Introduction to Algorithms* (2001), S. 527–531 (zitiert auf Seite 11).

[CSH06] Choi, Namyoun ; Song, Il-Yeol ; Han, Hyoil: A Survey on Ontology Mapping. In: *ACM SIGMOD Record* 35 (2006), September, Nr. 3, S. 34–41. `http://dx.doi.org/10.1145/1168092.1168097`. – DOI 10.1145/1168092.1168097. – ISSN 01635808 (zitiert auf den Seiten 12 und 25).

[Cyc] Cycorp: *ResearchCyc*. `http://www.cyc.com/platform/researchcyc`. – [Online; abgerufen am 21. Februar 2018] (zitiert auf den Seiten 12 und 78).

[Die00] Diestel, Reinhard: *Graph theory*. Springer-Verlag Berlin and Heidelberg GmbH, 2000 (zitiert auf Seite 10).

[DMDH02] Doan, AnHai ; Madhavan, Jayant ; Domingos, Pedro ; Halevy, Alon: Learning to map between ontologies on the semantic web. In: *Proceedings of the 11th international conference on World Wide Web* ACM, 2002, S. 662–673 (zitiert auf Seite 25).

[ES+07] Euzenat, Jérôme ; Shvaiko, Pavel u. a.: *Ontology matching*. Bd. 18. Springer, 2007 (zitiert auf Seite 12).

[Eye06] Eye, Alexander von: An alternative to Cohen's κ. In: *European Psychologist* 11 (2006), Nr. 1, S. 12 (zitiert auf Seite 63).

[Fel98] Fellbaum, Christiane: *WordNet*. Wiley Online Library, 1998 (zitiert auf den Seiten 32 und 61).

[Gü15] Günes, Zeynep: *Aufbau eines Sprachkorpus zur Programmierung autonomer Roboter mittels natürlicher Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, 2015. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/guenes_ba` (zitiert auf Seite 16).

[GD95] Gordon, Diana F. ; Desjardins, Marie: Evaluation and selection of biases in machine learning. In: *Machine learning* 20 (1995), Nr. 1-2, S. 5–22 (zitiert auf Seite 4).

[Gil05] Giles, Jim: *Internet encyclopaedias go head to head*. 2005 (zitiert auf Seite 13).

[Hav02] Haveliwala, Taher H.: Topic-sensitive pagerank. In: *Proceedings of the 11th international conference on World Wide Web* ACM, 2002, S. 517–526 (zitiert auf Seite 23).

[HCPC13] Hingmire, Swapnil ; Chougule, Sandeep ; Palshikar, Girish K. ; Chakraborti, Sutanu: Document classification by topic labeling. In: *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval* ACM, 2013, S. 877–880 (zitiert auf den Seiten 20 und 28).

[Hes02] Hesse, Wolfgang: *Ontologie(n)*. `https://www.gi.de/service/informatiklexikon/detailansicht/article/ontologien.html`. Version: 2002 (zitiert auf Seite 12).

[Hey16] Hey, Tobias: *Kontext- und Korreferenzanalyse für gesprochene Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Master's Thesis, 2016. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/hey_ma` (zitiert auf Seite 16).

[HHKG13] Hulpus, Ioana ; Hayes, Conor ; Karnstedt, Marcel ; Greene, Derek: Unsupervised graph-based topic labelling using dbpedia. In: *Proceedings of the sixth ACM international conference on Web search and data mining* ACM, 2013, S. 465–474 (zitiert auf den Seiten 20, 21, 29, 30, 34, 35, 37, 59, 62, 63, 64, 65, 66 und 75).

[HQ08] Hu, Wei ; Qu, Yuzhong: Falcon-AO: A practical ontology matching system. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 6 (2008), Nr. 3, S. 237–239 (zitiert auf Seite 24).

[Jur09] Jurafsky, Dan: *Speech & language processing*. Pearson Education India, 2009 (zitiert auf den Seiten 8 und 51).

[Kö14] Körner, Sven J.: *RECAA - Werkzeugunterstützung in der Anforderungserhebung*, Karlsruher Institut für Technologie, Fakultät für Informatik (INFORMATIK), Institut für Programmstrukturen und Datenorganisation (IPD), Diss., 2014. `http://dx.doi.org/10.5445/KSP/1000039460`. – DOI 10.5445/KSP/1000039460 (zitiert auf Seite 12).

[Kie16] Kiesel, Viktor: *Optimierung von POS-Tagger-Ergebnissen*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, 2016. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/kiesel_ba` (zitiert auf Seite 16).

[Koc15] Kocybik, Markus: *Projektion von gesprochener Sprache auf eine Handlungsrepräsentation*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, 2015. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/kocybik_ba` (zitiert auf den Seiten 16 und 17).

[KR00] Kilgarriff, Adam ; Rosenzweig, Joseph: Framework and results for English SENSEVAL. In: *Computers and the Humanities* 34 (2000), Nr. 1-2, S. 15–48 (zitiert auf Seite 32).

[KRW] Kino Coursey ; Rada Mihalcea ; William Moen: Using Encyclopedic Knowledge for Automatic Topic Identification (zitiert auf den Seiten 22, 23, 29, 30, 34, 36 und 37).

[Les86] Lesk, Michael: Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In: *Proceedings of the 5th Annual International Conference on Systems Documentation*. New York, NY, USA : ACM, 1986 (SIGDOC '86). – ISBN 978–0–89791–224–2, S. 24–26 (zitiert auf Seite 10).

[LIJ+15] Lehmann, Jens ; Isele, Robert ; Jakob, Max ; Jentzsch, Anja ; Kontokostas, Dimitris ; Mendes, Pablo ; Hellmann, Sebastian ; Morsey, Mohamed ; Kleef, Patrick van ; Auer, Sören ; Bizer, Chris: DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. In: *Semantic Web Journal* 6 (2015), Nr. 2, S. 167–195 (zitiert auf Seite 13).

[LK77] Landis, J R. ; Koch, Gary G.: The measurement of observer agreement for categorical data. In: *biometrics* (1977), S. 159–174 (zitiert auf Seite 63).

[LKT+15] Landhausser, M. ; Korner, S. J. ; Tichy, W. F. ; Keim, J. ; Krisch, J.: DeNom: a tool to find problematic nominalizations using NLP. In: *2015 IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, 2015, S. 1–8 (zitiert auf Seite 12).

[LN02] Lee, Yoong K. ; Ng, Hwee T.: An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10* Association for Computational Linguistics, 2002, S. 41–48 (zitiert auf Seite 31).

[LTLL09] Li, Juanzi ; Tang, Jie ; Li, Yi ; Luo, Qiong: RiMOM: A dynamic multistrategy ontology alignment framework. In: *IEEE Transactions on Knowledge and Data Engineering* 21 (2009), Nr. 8, S. 1218–1232 (zitiert auf Seite 24).

[McC96] McCallum, Andrew K.: *Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering.* 1996. – http://www.cs.cmu.edu/ mccallum/bow (zitiert auf Seite 49).

[MCCS09] Magatti, Davide ; Calegari, Silvia ; Ciucci, Davide ; Stella, Fabio: Automatic labeling of topics. In: *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on* IEEE, 2009, S. 1227–1232 (zitiert auf den Seiten 19, 20, 21 und 75).

[MCK04] Mihalcea, Rada ; Chklovski, Timothy ; Kilgarriff, Adam: The Senseval-3 English lexical sample task. In: *Proceedings of SENSEVAL-3, the third international workshop on the evaluation of systems for the semantic analysis of text*, 2004 (zitiert auf Seite 32).

[Mih07] Mihalcea, Rada: Using Wikipedia for Automatic Word Sense Disambiguation. In: *HLT-NAACL*, 2007, S. 196–203 (zitiert auf den Seiten 10, 30, 32, 47, 49, 52 und 60).

[Mit97] Mitchell, Tom: *Machine Learning*. McGraw Hill, 1997 (zitiert auf Seite 3).

[MMS93] Marcus, Mitchell P. ; Marcinkiewicz, Mary A. ; Santorini, Beatrice: Building a large annotated corpus of English: The Penn Treebank. In: *Computational linguistics* 19 (1993), Nr. 2, S. 313–330 (zitiert auf den Seiten xi und 7).

[MSB+14] Manning, Christopher D. ; Surdeanu, Mihai ; Bauer, John ; Finkel, Jenny ; Bethard, Steven J. ; McClosky, David: The Stanford CoreNLP Natural Language Processing Toolkit. In: *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, 55–60 (zitiert auf den Seiten ix, 7, 8, 9 und 48).

[MSZ] Mei, Qiaozhu ; Shen, Xuehua ; Zhai, Chengxian: Automatic Labeling of Multinomial Topic Models (zitiert auf den Seiten 19, 20 und 22).

[Mur06] Murphy, Kevin P.: Naive bayes classifiers. In: *University of British Columbia* (2006) (zitiert auf Seite 4).

[NM99] Noy, Natalya F. ; Musen, Mark A.: SMART: Automated support for ontology merging and alignment. In: *Proc. of the 12th Workshop on Knowledge Acquisition, Modelling, and Management (KAW'99), Banf, Canada*, 1999 (zitiert auf Seite 25).

[OAE] OAEI: *Ontology Alignment Evaluation Initiative*. `http://oaei.ontologymatching.org/`. – [Online; abgerufen am 21. Februar 2018] (zitiert auf Seite 25).

[OID] Olena Medelyan ; Ian H. Witten ; David Milne: Topic Indexing with Wikipedia (zitiert auf den Seiten 20, 21 und 28).

[Ou16] Ou, Yue: *Erkennung von Aktionen in gesprochener Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, 2016. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/ou_ba` (zitiert auf Seite 16).

[Pas15] Paskaran, Dinesh: *Evaluation unterschiedlicher Spracherkennungssysteme in der Domäne Humanoide Robotik*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, 2015. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/paskaran_ba` (zitiert auf Seite 16).

[PBMW99] Page, Lawrence ; Brin, Sergey ; Motwani, Rajeev ; Winograd, Terry: The PageRank citation ranking: Bringing order to the web. / Stanford InfoLab. 1999. – Forschungsbericht (zitiert auf den Seiten 22 und 23).

[PJ13] Pavel Shvaiko ; Jérôme Euzenat: Ontology Matching: State of the Art and Future Challenges. In: *IEEE Transactions on Knowledge and Data Engineering* 25 (2013), Januar, Nr. 1, S. 158 – 176. http://dx.doi.org/10.1109/TKDE.2011.253. – DOI 10.1109/TKDE.2011.253. – ISSN 1041–4347 (zitiert auf Seite 25).

[Pow11] Powers, David M.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. (2011) (zitiert auf Seite 5).

[PY01] Preiss, Judita ; Yarowsky, David: Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems. In: *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, 2001 (zitiert auf Seite 32).

[RA07] Rada Mihalcea ; Andras Csomai: Wikify!: Linking Documents to Encyclopedic Knowledge, 2007 (zitiert auf den Seiten 22 und 30).

[RN95] Russell, Stuart ; Norvig, Peter: *Artificial Intelligence: A modern approach*. Bd. 1. Prentice-Hall, 1995 (zitiert auf den Seiten 5 und 10).

[SA09] Seddiqui, Md H. ; Aono, Masaki: An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 7 (2009), Nr. 4, S. 344–356 (zitiert auf Seite 24).

[Sch15] Schneider, Michael: *Entwurf einer Handlungsrepräsentation für gesprochene Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, 2015. https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/schneider_ba (zitiert auf Seite 16).

[Sch16] Scheu, Sven: *Aufbereitung von Spracherkennerausgaben*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Master's Thesis, 2016. https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/scheu_ma (zitiert auf Seite 16).

[Sch17] Schlereth, Mario: *Entwicklung eines Dialogagenten für dialogbasierte Programmierung*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Master's Thesis, 2017. https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/schlereth_ma (zitiert auf Seite 16).

[SS09] Staab, Steffen ; Studer, Rudi: *Handbook on Ontologies*. 2nd. Springer Publishing Company, Incorporated, 2009. – ISBN 3540709991, 9783540709992 (zitiert auf Seite 12).

[Ste16] Steurer, Vanessa: *Strukturerkennung von Bedingungen in gesprochener Sprache*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, 2016. https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/steurer_ba (zitiert auf Seite 16).

[SW03] Stevenson, Mark ; Wilks, Yorick: Word sense disambiguation. In: *The Oxford Handbook of Comp. Linguistics* (2003), S. 249–265 (zitiert auf den Seiten 2 und 10).

[Tho99] Thorup, Mikkel: Undirected Single-source Shortest Paths with Positive Integer Weights in Linear Time. In: *J. ACM* 46 (1999), Mai, Nr. 3, 362–394. http://dx.doi.org/10.1145/316542.316548. – DOI 10.1145/316542.316548. – ISSN 0004–5411 (zitiert auf Seite 30).

[Wei] Weigelt, Sebastian: *Programming ARchitecture for Spoken Explanations (PARSE).* – `http://ps.ipd.kit.edu/163_422.php` (zitiert auf den Seiten 6 und 15).

[WFHP16] Witten, Ian H. ; Frank, Eibe ; Hall, Mark A. ; Pal, Christopher J.: *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2016 (zitiert auf Seite 5).

[Wik10] Wikinews: *Interview mit Jimmy Wales: Wie geht es weiter mit Wikipedia?* `https://de.wikinews.org/wiki/Interview_mit_Jimmy_Wales%3A_Wie_geht_es_weiter_mit_Wikipedia%3F`. Version: 2010. – [Online; abgerufen am 21. Februar 2018] (zitiert auf Seite 13).

[Wik18a] Wikipedia: *Wikipedia, The Free Encyclopedia.* `https://en.wikipedia.org/wiki/Wikipedia`. Version: 2018. – [Online; abgerufen am 21. Februar 2018] (zitiert auf Seite 13).

[Wik18b] Wikipedia: *Wikipedia:Verlinken.* `https://de.wikipedia.org/wiki/Wikipedia:Verlinken`. Version: 2018. – [Online; abgerufen am 21. Februar 2018] (zitiert auf Seite 32).

[WT15] Weigelt, S. ; Tichy, W. F.: Poster: ProNat: An Agent-Based System Design for Programming in Spoken Natural Language. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* Bd. 2, 2015. – ISSN 0270–5257, S. 819–820 (zitiert auf den Seiten ix und 15).

[XNXT04] Xie, Zhuli ; Nelson, Peter C. ; Xiao, Weimin ; Tirpak, Thomas M.: Using Noun Phrase Centrality to Identify Topics for Extraction based Summaries. In: *Proceedings of the IASTED International Conference on Knowledge Sharing and Collaborative Engineering*, 2004, S. 89–94 (zitiert auf Seite 34).

[Yng55] Yngve, Victor H.: Syntax and the problem of multiple meaning. In: *Machine translation of languages* 14 (1955), S. 208–226 (zitiert auf den Seiten 2 und 10).

Anhang

A. Training des Klassifikators

In Tabelle A.1 sind einige Statistiken zu den Attributen der Trainingsinstanzen abgebildet. In der zweiten Zeile sind die Namen der jeweiligen Attribute. In der dritte Zeile ist abgebildet, für wieviele der Instanzen für das Attribut kein Attributwert existiert. In der vierten Zeile ist angegeben, wieviele der Attributwerte einmalig sind. Die letzte Zeile zeigt die Anzahl der unterschiedlichen Attributwerte an.

Tabelle A.1.: Statistiken zu den Attributen der 5.188.470 Trainingsinstanzen für die Auflösung von Mehrdeutigkeiten

#	Name	#Fehlend	#Einmalig	#Distinktiv
0	wordSense	0 / 0%	136964 / 3%	283173
1	actualWord	0 / 0%	74658 / 1%	169176
2	actualWordPOS	0 / 0%	1 / 0%	33
3	word-3	8551 / 0%	290649 / 6%	448163
4	word-3POS	8551 / 0%	1 / 0%	36
5	word-2	6318 / 0%	225247 / 4%	334896
6	word-2POS	6318 / 0%	2 / 0%	36
7	word-1	2563 / 0%	353047 / 7%	485010
8	word-1POS	2563 / 0%	2 / 0%	35
9	word+1	5421 / 0%	158600 / 3%	261381
10	word+1POS	5421 / 0%	0 / 0%	34
11	word+2	8555 / 0%	169653 / 3%	282232
12	word+2POS	8555 / 0%	0 / 0%	35
13	word+3	10728 / 0%	153255 / 3%	258519
14	word+3POS	10728 / 0%	0 / 0%	35
15	leftNN	4390 / 0%	374988 / 7%	539089
16	leftVB	7995 / 0%	3899 / 0%	10867
17	rightNN	6100 / 0%	141563 / 3%	236368
18	rightVB	18801 / 0%	4102 / 0%	10986

B. Evaluation Themenextraktion

B.1. Fragebogen

Im Folgenden befindet sich ein Auszug aus dem Fragebogen, der den Teilnehmern vorgelegt wurde. Dafür wurde den Teilnehmern ein Eingabetext vorgelegt und zu jedem Thema Fragen wie im folgenden Beispiel zum Thema *Home* gestellt:

Beispiel

Das Thema „Home“ ist

1 passend

2 verwandt

3 unverwandt

wenn 1: Das Thema „Home“ ist passend, dabei ist es

- etwas zu allgemein
- in Ordnung
- etwas zu spezifisch

wenn 2: Das Thema „Home“ ist verwandt, denn es ist

- zu allgemein
- zu spezifisch
- anderer Grund

B.2. Evaluationskorpus

Im Folgenden sind die verschiedenen Szenarien mit den jeweiligen Eingabetexten zu finden, die für die Evaluation der Themenextraktion genutzt wurden. Satzzeichen wurden hinzugefügt, um den Studienteilnehmern das Textverständnis zu erleichtern. Bei der Aufzählung steht die erste Zahl für das Szenario, die zweite Ziffer gibt an, in welcher der beiden Umfragen der Text verwendet wurde. In Tabelle B.2 sind die Themen, die mit den verschiedenen Auswahlverfahren zu den Texten ermittelt wurden, zu finden.

1.1 Okay Armar, go to the table, grab popcorn, come to me, give me the popcorn, which is in your hand

1.2 Armar, can you please bring me the popcorn bag

2.1 hello Armar, could you go to the table and take the green cup? Please put it in the dishwasher and close it

2.2 hey Armar, please place the green cup from the kitchen table into the dishwasher

3.1 Armar, would you please go to the fridge and open it, take out the orange juice and bring it to me?

3.1 Armar, open the fridge and take the orange juice. Afterwards close the fridge and bring me the orange juice.

4.1 hey Armar, could you please have a look at these dishes? If they are dirty put them into the dishwasher. If they are not dirty, put them into the cupboard

4.2 robo, go to the table. If there are any dirty dishes, grab the dirty dishes and go to the dishwasher, open the dishwasher, and put the dirty dishes into the dishwasher. Close the dishwasher and return to the table. If there are any clean dishes, grab the clean dishes and go to the cupboard. Open the cupboard and put the clean dishes into the cupboard

5.1 hello Armar, I want to make some drinks. Go to the fridge and if there fresh oranges, bring me the fresh oranges together with vodka. Otherwise bring me just orange juice and the vodka

5.2 hello Armar, I would want to have some vodka with fresh orange, please. Go to the fridge and check if there are some fresh orange. If there are not any, please make me a vodka with orange juice

6.1 Go to the table, take the green cup standing on the table, and go to the fridge. Open the fridge. Right in front of you there is a water bottle, take the bottle, open it, fill water in the cup, and put the bottle back in the fridge. Close the fridge and then bring me the cup. Afterwards go to the dishwasher, open the dishwasher, take two red cups em from inside, bring them to the cupboard, and open the cupboard. Put the cups on the shelf and close the cupboard again

6.2 Go to the fridge, open the fridge door, then take the water bottle out. Close the fridge door, then go to the table and open the water bottle. Fill the green cup with the water and then bring the cup to me. Go to the dishwasher, ahm take the two red cups out of the dishwasher, and put them on the cupboard

7.1 Hey Armar, I'm hungry. Please, take eh a plate from the dishwasher and rinse it with water at the sink. Afterwards, go to the fridge, open the fridge, and put the instant food you find into in there on to the plate. Close the fridge and warm the plate in the eh microwave. Therefore you have to open the door first. Afterwards, when it's warm, please, bring it eh to me at the table

7.2 Go to the dishwasher and take one plate out. Wash this plate and then go to the fridge, put the em meal instant meal on the plate and bring it to the microwave. Put it into the microwave. Then, after it is warmed up, put the ehm meal on the plate and bring the plate to the table

8.1 Go to the washing machine and open it. Take out the laundry, put the laundry into the dryer, start the dryer

8.2 Go to the dryer. Open the dryer. Go to the front side of the washing machine. Grab the washing machine window handle. Put the handle to open the washing machine. Put your arms into the washing machine. Grab the laundry. Take the laundry out of the washing machine. Go to the dryer. Put the laundry into the dryer. Close the dryer. Push the start button of the dryer

E1.1 Start and accelerate as fast as possible until you fly through the gate. Then slow down and turn left by sixty degree. Accelerate again and dodge the table by ascending first and descending afterwards. Fly through the greenhouse, then turn left and fly above the pond. If you crossed the pond, break and descend down to the lawn and finally turn off

E1.2 Ascend and fly as fast as possible to the gate, turn left and ascend and start flying to the greenhouse. After you dodged the table by ascending, descend again. After passing through the greenhouse, turn left again and fly over the pond. Afterwards slow down and descend until you are on the lawn

E2.1 Follow the line on the carpet. At the end of the carpet turn until you see the rattle. Grab the rattle and afterwards release it again

E2.2 Move along the line until you are at the end of the carpet. Turn right until you see the rattle, grab it and then release it again

E3.1 Alexa, turn up the temperature of the radiator by two degrees, then start playing my favorite playlist

E3.2 Alexa, before you play my favorite playlist, turn on the radiator, because it is getting cold in here

Tabelle B.2.: Themen zu den jeweiligen Texten, die mit den beiden Varianten erzeugt wurden

	CombinedProcessor	***TopConnectivityProcessor***
1.1	Plants used in Native American cuisine	Plants used in Native American cuisine
	Furniture	Furniture
	Snack foods	Snack foods
	Hand	Crops originating from the Americas
	Upper limb anatomy	Hand
	Tables	Upper limb anatomy
1.2	Microwave popcorn	Microwave popcorn
	Tool	Tool
	Plants used in Native American cuisine	Plants used in Native American cuisine
	Snack foods	Snack foods
2.1	Home	Home
	Light fixtures	Light fixtures
	Cocktail shaker	Cocktail shaker
	Device	Infuser
	Furniture	Wine table
	Cup	Wine rack
2.2	Home	Home
	Coffee preparation	Coffee preparation
	Light fixtures	Light fixtures
	Kitchenware	Kitchenware
	Exhaust hood	Buddhist terminology
	Wine table	Bartending equipment
	Wine rack	Hoosier cabinet
	Cup	Wet bar
3.1	Grapefruit	Grapefruit
	Maraschino cherry	Maraschino cherry
	Food science	Food science
	Food preservation	Fruit
	Liquid	Tropical agriculture
	Juice	Fruits originating in Asia
3.2	Maraschino cherry	Maraschino cherry
	Food science	Food science
	Grapefruit	Grapefruit
	Food preservation	Fruit
	Liquid	Tropical agriculture
	Juice	Fruits originating in Asia
4.1	Home	Home
	Home appliances	Home appliances

	Domestic implements	Domestic implements
	Napkin holder	Napkin holder
	Cabinet	Lazy Susan
	Furniture	Sideboard
4.2	Home	Home
	Welsh dresser	Sideboard
	Sideboard	Welsh dresser
	Consumer goods	Consumer goods
	Clothes valet	Slipcover
	Device	Clothes valet
	Home automation	Coat rack
	Cleaning	Wine rack
5.1	Mixed drinks	Mixed drinks
	Tea	Tea
	Cocktails	Cocktails
	Food preservation	Food science
	Cooling technology	Wine
	Home appliances	Smoothie
	Liquid	Kraut juice
	Sugarcane juice	Palm wine
	Tomato juice	
5.2	Wine	Wine
	Maraschino cherry	Maraschino cherry
	Food science	Food science
	Grapefruit	Grapefruit
	Smoothie	Smoothie
	Kraut juice	Kraut juice
	Beverage	Orange liqueurs
	Food preservation	Cocktails
6.1	Home	Home
	Hoosier cabinet	Hoosier cabinet
	Containers	Containers
	Liquid containers	Light fixtures
	Device	Architectural elements
	Packaging	Hatstand
	Clean-in-place	Wine rack
	Oxides	Furniture
	Console table	
6.2	Home	Home
	Architectural elements	Architectural elements
	Food storage	Food storage
	Liquid containers	Liquid containers
	Can opener	Hoosier cabinet
	Castle architecture	Containers
	Cocktail shaker	Can opener
	Structure	Light fixtures
	Cup	
	Console table	
7.1	Home	Home
	Architectural elements	Architectural elements
	Food preparation appliances	Food preparation appliances
	Kitchenware	Tools

	Arab cuisine	Domestic implements
	Via fence	Preservatives
	Herb	Back boiler
	Radio technology	Cooking appliances
	Plumbing	
7.2	Home	Home
	Food preparation appliances	Food preparation appliances
	Domestic implements	Tools
	Serving and dining	Domestic implements
	Furniture	Device
	Radio technology	Home appliances
	Wireless	Home automation
	Foods	Refrigerator
8.1	Laundry	Laundry
	Combo washer dryer	Combo washer dryer
	Washing machine	Washing machine
	Tunnel washer	Tunnel washer
	Drying cabinet	Drying cabinet
	Cleaning	Cleaning
8.2	Machines	Machines
	Laundry	Laundry
	Mechanism (Push-button)	Washing machine
	Architectural elements	Combo washer dryer
	Arm	Tunnel washer
	Opening	Tools
	Upper limb anatomy	Drying cabinet
	Washing	Cleaning
E1.1	Garden features	Garden features
	Horticulture and gardening	Horticulture and gardening
	Lawns	Lawns
	Body	Duck pond
	Furniture	Koi pond
	Mathematical constants	Garden pond
	Imperial units	Picnic table
	Tables	Park furniture
E1.2	Garden features	Garden features
	Horticulture and gardening	Horticulture and gardening
	Lawns	Lawns
	Body	Duck pond
	Furniture	Koi pond
	Fluvial landforms	Garden pond
	Tables	Picnic table
	Table	Park furniture
E2.1	Instrument	Instrument
	Textiles	Textiles
	Unpitched percussion	Unpitched percussion
	Elementary geometry	Elementary geometry
	Textile arts	Textile arts
	Analytic geometry	Weaving
E2.2	Instrument	Instrument
	Textiles	Textiles
	Unpitched percussion	Unpitched percussion

	Elementary geometry Textile arts Analytic geometry	Elementary geometry Textile arts Weaving
E3.1	Temperature Heating,ventilating,and air conditioning Heat transfer Broadcasting Playlist file formats Fab 40 Degree Degree of temperature	Temperature Heating,ventilating,and air conditioning Heat transfer Units of temperature Balance point temperature Rankine scale Freeze stat Thermal mass
E3.2	Heating,ventilating,and air conditioning Plumbing Broadcasting Playlist file formats	Heating,ventilating,and air conditioning Plumbing Broadcasting Vehicle parts

C. Evaluation Ontologieauswahl

Im Folgenden sind die zusätzlichen Texte zu finden, die für die Evaluation der Ontologieauswahl genutzt wurden.

E4.1 hey armar grab the lawn mower and use it to cut the grass

E4.2 hey armar open the shed grab the mower and cut the lawn

E5.1 Go to the Fridge take the tonic water and mix it in the glass with the gin that is on the counter

E5.2 Mix a cuba libre by putting coke rum and some lime juice in a glass

E6.1 go to the closet open it and grab the sweater and the trousers and bring them to me

E6.2 move to the desk and clean it afterwards go the the nightstand and clean it as well if you finished these tasks take the book out of the shelf and bring it to me

E7.1 alexa please play my metal playlist in a random order

E7.2 alexa what are the most famous songwriters who use a piano

E8.1 hey armar, Louis Armstrong is a really good artist, he plays the trumpet so well. turn up the volume. Then walk to the fridge and bring us two beers, afterwards get us each one whisky so we can honor him

E8.2 armar, i think it is perfect to combine jazz with a nice glass of whisky. So please bring us our drinks

E9.1 go to the fridge and grab the ketchup and the water and bring it outside to our patio table at the pond. be careful not to walk over the freshly sewn lawn