Karlsruhe Institute of Technology

Bestimmung der semantischen Funktion von Sätzen in Anforderungsbeschreibungen

Bachelorarbeit
von

Dana Tomova

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

Erstgutachter:	Prof. Dr. Walter F. Tichy
Zweitgutachter:	Prof. Dr. Ralf Reussner
Betreuender Mitarbeiter:	M.Sc. Tobias Hey

Bearbeitungszeit: 01.07.2019 – 31.10.2019

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, 31.10.2019

(**Dana Tomova**)

Publikationsgenehmigung

Melder der Publikation

Hildegard Sauer

Institut für Programmstrukturen und Datenorganisation (IPD)
Lehrstuhl für Programmiersysteme
Leiter Prof. Dr. Walter F. Tichy

+49 721 608-43934
hildegard.sauer@kit.edu

Erklärung des Verfassers

Ich räume dem Karlsruher Institut für Technologie (KIT) dauerhaft ein einfaches Nutzungsrecht für die Bereitstellung einer elektronischen Fassung meiner Publikation auf dem zentralen Dokumentenserver des KIT ein.

Ich bin Inhaber aller Rechte an dem Werk; Ansprüche Dritter sind davon nicht berührt.

Bei etwaigen Forderungen Dritter stelle ich das KIT frei.

Eventuelle Mitautoren sind mit diesen Regelungen einverstanden.

Der Betreuer der Arbeit ist mit der Veröffentlichung einverstanden.

Art der Abschlussarbeit:	Bachelorarbeit
Titel:	Bestimmung der semantischen Funktion von Sätzen in Anforderungsbeschreibungen
Datum:	31.10.2019
Name:	Dana Tomova

Karlsruhe, 31.10.2019

(**Dana Tomova**)

Kurzfassung

Das Verständnis der Absicht von Softwareanforderungen ist essentiell für die automatische Generierung von Informationen zur Rückverfolgbarkeit. Funktionale Anforderungen können verschiedene semantische Funktionen, wie die Beschreibung von erwarteten Funktionalitäten oder Zuständen des Systems, beinhalten. Im Rahmen des INDIRECT-Projektes wird ein Werkzeug zur Klassifikation der semantischen Funktion der Sätze in Anforderungsbeschreibungen entwickelt. Dafür werden verschiedene maschinelle Lernverfahren (Stützvektormaschine, Logistische Regression, Random Forest und Naïve Bayes) auf ihre Eignung für diese Aufgabe überprüft. Um ihre Funktionalität zu evaluieren, werden die Verfahren auf einem Datensatz aus frei verfügbaren Anforderungsbeschreibungen getestet, welcher manuell mit semantischen Funktionen etikettiert wurde. Die Ergebnisse zeigen, dass der Random Forest-Klassifikator unter Verwendung von N-Grammen auf Zeichenebene mit einem F_1-Maß von 0,79 die beste Leistung auf unbekannten Projekten liefert. Die Lernverfahren werden zusätzlich mittels einer Kreuzvalidierung auf allen vorhandenen Daten getestet. Dabei erzielt die Stützvektormaschine mit einem F_1-Maß von 0,90 die besten Ergebnisse, während der Random Forest-Klassifikator ein F_1-Maß von 0.89 erreicht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Rückverfolgbarkeit von Anforderungen ist eine aktuelle Herausforderung im Bereich der Softwaretechnik, die Aufgaben wie Wirkungsanalyse und Wiederverwendung von Teilen eines Systems unterstützt. Die Generierung von Informationen zur Rückverfolgbarkeit ist mit einem kostspieligen manuellen Aufwand verbunden. Diese Generierung zu automatisieren und insbesondere eine Verknüpfung von Anforderungen und Quelltext zu erzeugen, würde Zeit und Kosten sparen. Um diese Abbildungen vollständig herzustellen, benötigt es ein tiefergehendes Verständnis der Semantik der Anforderungen.

Anforderungsbeschreibungen können verschiedene Informationen tragen. Abhängig von der Bedeutung der Sätze können sie unterschiedlich gehandhabt werden. Daher ist es sinnvoll, Anforderungen nach ihrer Semantik zu klassifizieren. Um einen Satz semantisch zu erfassen, ist es wichtig, eine Erkenntnis der beschriebenen Tätigkeiten und Zustände zu erlangen. In diesem Zusammenhang identifizierte Glinz [Gli05] bereits eine operative und deklarative Form der Darstellung von funktionalen Anforderungen. Die Anforderungen, die in einer operativen Form dargestellt werden, beziehen sich auf auszuführende Handlungen und das Verhalten des Systems. Sie beschreiben Aktivitäten und bestimmen mögliche Methoden im Quelltext. Ein Beispiel dafür ist:

Beispiel 1.1: Aktion

„The system shall compute the sum of all applicable deductions.“

Eine deklarative Repräsentation haben die Anforderungen, die erforderliche Merkmale beschreiben. Sie definieren bestimmte Eigenschaften des Systems und damit eventuelle Attribute im Programm. Folgendes Beispiel stellt eine Zustandsbeschreibung dar:

Beispiel 1.2: Zustand

„Current students shall be members of the library.“

Anforderungen können unter anderem auszuführende Aktionen, benötigte und bereitzustellende Daten, Zustände oder irrelevante Informationen umfassen. Es gibt Anforderungen, in denen mehrere dieser Funktionen vertreten sein können. Sie lassen sich in sinnvolle

Teilanforderungen zerlegen. Das Bestimmen der semantischen Funktion der (Teil-)Sätze würde bei der Identifikation korrespondierender Methoden, Parameter und Attribute im Quelltext helfen. Somit würde die Erkennung solcher Unterschiede im Kontext der semantischen Darstellung der Anforderungsbeschreibungen die Gewinnung von Informationen zur Rückverfolgbarkeit erleichtern.

1.1 Zielsetzung der Bachelorarbeit

Das Ziel dieser Bachelorarbeit besteht darin, einen Agent für das Projekt INDIRECT zu entwerfen, der die Teile der Anforderungen automatisiert nach ihrer Semantik klassifiziert. Dafür müssen zuerst die möglichen Klassen identifiziert werden. Der Fokus dieser Bachelorarbeit liegt auf funktionalen Anforderungen. Sie umfassen auszuführende Aktionen, benötigte und bereitzustellende Daten oder Zustände. Allerdings gibt es auch Anforderungen, die mehr als eine Verbphrase beinhalten und mehreren Klassen zugeordnet werden können. Daher ist es ebenso Teil der Aufgabe, die Bereiche eines Satzes zu bestimmen, die einer gewissen Klasse zugeordnet werden können. Basierend auf der Analyse der Klassen und der möglichen Techniken zur Umsetzung der Klassifikation soll ein passender Ansatz gewählt werden, um das Verfahren zu entwickeln.

1.2 Aufbau der Bachelorarbeit

Zu Beginn werden in Kapitel 2 wichtige Grundlagen zum Verständnis dieser Ausarbeitung erläutert. Danach wird in Kapitel 3 das Gesamtprojekt INDIRECT vorgestellt. In Kapitel 4 wird ein Überblick über die verwandten wissenschaftlichen Arbeiten gegeben. In Kapitel 5 wird analysiert, welche möglichen Klassen es für Arten von Anforderungen gibt, und welche Methoden und Techniken aus der Rechnerlinguistik passend für die Aufgabenstellung sind und umgesetzt werden können. Daraufhin wird ein Verfahren für die Klassifikation der Anforderungsbeschreibungen entworfen. In Kapitel 6 wird die Implementierung des Werkzeugs beschrieben. Auf dieser Basis wird in Kapitel 7 die Lösung evaluiert. Zum Schluss werden in Kapitel 8 die Ergebnisse zusammengefasst.

2 Grundlagen

Dieses Kapitel stellt wichtige Begriffe vor, die zum Verständnis dieser Arbeit dienen. Zuerst werden einige Grundlagen im Bereich der Sprachverarbeitung (Abschnitt 2.1) eingeführt. Des Weiteren wird auf das Konzept der Rückverfolgbarkeit (Abschnitt 2.2.2) eingegangen. Am Ende werden wichtige Grundbegriffe für die Textklassifikation (Abschnitt 2.3), sowie in dieser Arbeit verwendete Werkzeuge, vorgestellt.

2.1 Sprachverarbeitung

Die Verarbeitung natürlicher Sprache (engl. *Natural language processing, NLP*) ist ein Bereich der künstlichen Intelligenz und Linguistik [KKKS17], der sich mit der Erfassung von gesprochener Sprache oder geschriebenem Text in natürlicher Sprache befasst. Die eingesetzten Techniken und Methoden zur maschinellen Verarbeitung der natürlichen Sprache haben zum Ziel, die Sprache zu erkennen, zu analysieren und den Sinn zur weiteren Verarbeitung zu extrahieren. Einige der NLP-Anwendungsgebiete sind Stimmungsanalyse, Wortart-Markierung, Phrasenerkennung, Eigennamenerkennung und semantische Rollenerkennung.

NLP kann grundsätzlich in zwei Teilbereiche eingeteilt werden, nämlich natürliches Sprachverständnis (engl. *Natural Language Understanding (NLU)*) und natürliche Sprachgenerierung (engl.*Natural Language Generation (NLG)*), deren Aufgabe es ist, den Text zu verstehen und zu generieren. Da sich diese Arbeit auf die Klassifizierung von Anforderungsbeschreibungen bezieht, liegt hier der Fokus auf NLU - die Teildisziplin, die zum Ziel hat, die Bedeutung einer Frage oder einer Aussage im Detail zu verstehen. Zum Einsatz kommen die folgenden Phasen der internen Verarbeitung: Phonetik und Phonologie bei gesprochener Sprache, Morphologie und lexikalische Analyse, Zerlegung und syntaktische Analyse, semantische und pragmatische Analyse.

2.1.1 Syntax

Die Morphologie beschränkt sich auf die Analyse einzelner Wortformen und bildet die Grundlage für deren syntaktische Zusammensetzung [Hau14]. Die Syntax ist ein Teilgebiet der Grammatik und bezieht sich auf die Strukturbildung von Sätzen. Beachtet wird die grammatikalische Korrektheit und der Zusammenhang zwischen den einzelnen Wörtern. Syntaktische Strukturen lassen sich in zwei Hauptrichtungen einteilen [CEE+09]. Die erste Hauptrichtung ist die Dependenz- und Determinationssyntax, bei der syntaktische

Strukturen als Relationen zwischen Wörtern betrachtet werden. Die zweite Hauptrichtung ist die Konstituentensyntax. Hier werden neben Wörtern auch Phrasen, auch Konstituenten genannt, und die Relationen zwischen ihnen betrachtet. Die syntaktische Analyse beschränkt sich auf die Bestimmung der Zusammensetzung von Wortformen zu komplexen Ausdrücken, wobei die Bedeutung bzw. der Sinn der gebildeten Sätze nicht betrachtet wird. Die syntaktischen Regeln können also überprüft werden, ohne dass es nötig ist, den Inhalt zu verstehen.

2.1.2 Semantik

Eine der Teildisziplinen der Linguistik ist die Semantik. Sie befasst sich mit der Bedeutung natürlichsprachlicher Ausdrücke [CEE+09]. Es werden nicht nur die Bedeutung der sprachlichen Einheiten betrachtet, sondern auch die Bedeutungszusammenhänge von größeren strukturellen Einheiten dargestellt. Die lexikalische Semantik beschreibt die Bedeutung von Wörtern, die Satzsemantik den Zusammenhang der benutzten Wörter, und die Diskurssemantik den Zusammenhang von Sätzen. Die Semantik konzentriert sich auf die Interpretation von Äußerungen und nicht auf ihre Produktion oder auf ihre syntaktische Korrektheit. Damit einer Aussage eine Bedeutung zugewiesen werden kann, ist die syntaktische Korrektheit der Aussage erforderlich, aber nicht hinreichend.

2.1.3 Tokenisierung

Das Token ist die kleinste betrachtete Einheit eines Textes und stellt eine Folge von Buchstaben oder Ziffern dar, die von Leerzeichen oder Interpunktion umgeben ist [CEE+09]. Die Tokenisierung bezeichnet die Segmentierung eines Textes in Tokens und ist oft erforderlich für die Weiterverarbeitung und Analyse der Daten. Dabei handelt es sich aber nicht nur um die Unterteilung des Textes mittels Leerzeichen. Manchmal ist es notwendig, eine feinere Aufspaltung durchzuführen, oder zwei oder mehr durch Leerzeichen getrennte Einheiten als einen Token aufzufassen (z.B. „1 000 000“). Bei der Tokenisierung werden zuerst alle Interpunktionszeichen von Wörtern abgetrennt und danach werden bestimmte Folgen von Einheiten zu größeren Token zusammengefügt.

2.1.4 Wortart-Markierung

Ein wichtiger Unterscheidungsmerkmal der Einheiten eines Textes sind die Wortarten (engl. *Part-of-Speech, POS*), die Informationen über die grammatikalische Eigenschaften der Wörter beinhalten und zumeist auf Basis der Token zugewiesen werden. Die Wortarten lassen sich in zwei Hauptkategorien unterteilen: offene und geschlossene Klassen [Hau14]. Die offenen Klassen umfassen Zehntausende von Elementen, während die geschlossenen Klassen nur wenige hundert Wörter enthalten. Substantive, Verben, Adjektive und Adverbien bilden die so genannten offenen Wortklassen. Wortarten wie Präpositionen, Konjunktionen und Determinative definieren die geschlossenen Wortklassen.

Der Prozess der Zuweisung einer eindeutigen Wortart zu jedem Token wird als Wortart-Markierung (engl. *Part-of-Speech-Tagging*) bezeichnet. Die Darstellung der Wortarten wird mit Hilfe von definierten Mengen von Markierungen, den so genannten Tagsets, ermöglicht. In dieser Arbeit wird das Penn-Treebank-Tagset [MSM93] eingesetzt. Das folgende Beispiel stellt die Wortart-Markierungen für den Satz „The system should display both cases simultaneously“ dar.

Beispiel 2.1: Wortart-Markierungen

The/DT system/NN should/MD display/VB both/DT cases/NNS simultaneously/RB

Die ermittelten Annotationen werden in Tabelle 2.1 aufgelistet. Das Penn-Treebank-Tagset umfasst insgesamt 36 Markierungen, deren vollständige Übersicht in Anhang Abschnitt A.1 zu finden ist.

Tabelle 2.1: Ausschnitt aus dem Penn-Tagset

Markierung	Beschreibung
DT	Determinativ
MD	Modalverb
NN	Substantiv, Singular
NNS	Substantiv, Plural
RB	Adverb
VB	Verb, Infinitiv

2.1.5 Phrasenerkennung

Phrasen (engl. *chunks*) sind abgeschlossene Gruppen von Token, wie Nominalgruppen oder Verbalgruppen, die bestimmte syntaktische Informationen tragen. Der Prozess der Identifizierung und Klassifizierung dieser Teile eines Satzes wird Phrasenerkennung (engl. *Chunking*) genannt [JM18]. Es gibt zwei grundlegende Aufgaben, die bei der Phrasenerkennung anfallen - Segmentieren und Beschriften. Zuerst werden die Wörter in Gruppen mit einer gemeinsamen Rolle eingeteilt und dann werden den Phrasen die richtigen Markierungen zugewiesen. Es wird eine flache Analyse durchgeführt, wobei sich nicht überlappende Phrasen erkannt werden. Im folgenden Beispiel werden die Phrasen des Satzes aus dem vorherigen Beispiel in Klammernotation angegeben.

Beispiel 2.2: Phrasenerkennung

[$_{\text{NP}}$ The system] [$_{\text{VP}}$ should display] [$_{\text{NP}}$ both cases] [$_{\text{ADVP}}$ simultaneously]

Dieses Beispiel enthält zwei Nominalphrasen, eine Verbalphrase und eine Adverbialphrase. Die Nominalphrasen in diesem Satz bestehen aus einer Kombination aus den Wortarten Determinativ und Substantiv in Singular- oder Pluralform (DT+NN, DT+NNS), die Verbalphrase aus einem Modalverb und einem Verb im Infinitiv (MD+VB) und die Adverbialphrase aus einem Adverb (RB). Hierbei wird wieder die Penn-Treebank verwendet. Sie umfasst insgesamt 8 verschiedene Phrasentypen, die in Tabelle 2.2 aufgelistet sind.

Tabelle 2.2: Phrasen-Markierungen

Markierung	Beschreibung
NP	Nominalphrase
VP	Verbalphrase
PP	Präpositionalphrase
ADVP	Adverbialphrase
ADJP	Adjektivphrase
SBAR	Unterordnende Konjunktion
PRT	Partikel
INTJ	Einwurf

Jedem Wort einer Phrase kann zusätzlich ein Phrasenkennzeichen zugeordnet werden. Es ist üblich, Phrasenerkennung als IOB-Markierung zu modellieren. Das IOB-Format bestimmt, ob sich ein Wort am Anfang (*B - Beginning*) im Inneren (*I - Inside*) oder außerhalb (*O - Outside*) der Phrase befindet. Im Folgenden wird der Satz aus dem obigen Beispiel mit IOB-Markierungen versehen.

Beispiel 2.3: IOB-Format

The	system	should	display	both	cases	simultaneously
B-NP	I-NP	B-VP	I-VP	B-NP	I-NP	B-ADVP

2.1.6 Semantische Rollenzuteilung

Semantische Rollen, auch thematische Rollen genannt, sind eine der ältesten Klassen von Konstruktoren in der Sprachtheorie. Das führt zu einer Vielfalt von sehr spezifischen bis sehr allgemeinen Sätzen von semantischen Rollen. Thematische Rollen tragen Informationen über die Zusammenhänge in einem Satz und bezeichnen die Beziehungen zwischen dem Prädikat und seinen Argumenten. Sie beantworten Fragen wie „Wer?“, „Was?“, „Wo?“, „Wann?“, „Warum?“ usw. Daher werden sie häufig bei NLP-Aufgaben verwendet, in denen eine Art semantische Interpretation erforderlich ist, um zum Beispiel Fragen zu beantworten oder Schlussfolgerungen zu ziehen. Typische abstrakte semantische Rollen sind `AGENT`, `PATIENT` und `INSTRUMENT`. Der `AGENT` bezeichnet den Initiator einer Aktion, der in der Lage ist, eine Aktion durchzuführen. Der `PATIENT` ist ein Argument, das keine Kontrolle über die Handlung hat und die Wirkung der Aktion erlebt. Das `INSTRUMENT` ist das Mittel, mit dem eine Aktion ausgeführt wird oder etwas zustande kommt. Noch weiter verallgemeinerte thematische Rollen sind die Proto-Rollen, wie zum Beispiel die Markierung `PROTO-AGENT` [JM18]. Damit werden Argumente bezeichnet, die Eigenschaften eines Handelnden besitzen. Zum anderen können semantische Rollen domänenspezifisch definiert werden.

Der Prozess der automatischen Erkennung semantischer Rollen und der Beschriftung jedes Argumentes jedes Prädikats in einem Satz heißt semantische Rollenzuteilung (engl. *Semantic Role Labeling, SRL*). Davor findet oft eine Vorverarbeitung des Texts, wie Wortart-Markierung und Syntaxanalyse, statt. Die Markierung wird dann häufig mittels maschineller Lernverfahren, unter Verwendung von lexikalische Ressourcen wie FrameNet und PropBank, durchgeführt.

2.1.7 Abhängigkeitsbaum

Ein Abhängigkeitsbaum ist ein gerichteter azyklischer Graph, der die Abhängigkeiten zwischen Wörtern darstellt. Er hat einen einzelnen Wurzelknoten, von dem aus es einen eindeutigen, gerichteten Weg zu jedem der Wörter im Satz gibt [JM18]. In NLP analysiert ein Abhängigkeitszerteiler die Struktur des Satzes und definiert grammatikalische Beziehungen wie Subjekt, Objekt, Klauselkomplement, Substantivdeterminativ usw. Das Projekt „Universal Dependencies“ [NMG+16] stellt eine universelle Menge an anwendbaren Abhängigkeitsbeziehungen vor. Bei der Gestaltung der Beziehungen wird zwischen drei Arten von Strukturen unterschieden: Nominale, Klauseln und Modifikatoren.

In Abbildung 2.1 wird der Abhängigkeitsgraph des Satzes aus dem obigen Beispiel dargestellt. Die Nominale `nsubj` und `dobj` bezeichnen das Subjekt und das direkte Objekt von dem Prädikat „display“, während `aux` das vom Prädikat abhängige Hilfsverb „should“ identifiziert. Die Markierung `DET` ist ein Modifikator der Substantive „system“ und „cases“.

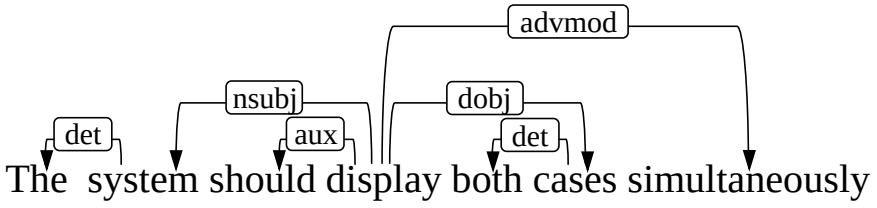

Abbildung 2.1: Abhänigkeitsbaum für einen Beispielsatz

Mit `advmod` wird ein Modifikator des Vollverbs bezeichnet. Eine vollständige Auflistung der Abhängigkeitsbezeichnungen wird in Abschnitt A.2 angegeben.

2.2 Anforderungen

Anforderungen in Software-Entwicklungsprojekten sind Aussagen über die Voraussetzungen und Fähigkeiten, die ein System erfüllen muss [VHH+12]. Anforderungsmanagement ist entscheidender und zentraler Prozess für jedes erfolgreiche Softwareentwicklungsprojekt. Dieser Prozess umfasst die Identifikation, Dokumentation, Kommunikation, Verfolgung und Verwaltung von Projektanforderungen.

2.2.1 Arten von Anforderungen

Anforderungen können nach verschiedenen Kriterien klassifiziert werden, wie z. B. nach Priorität oder Detaillierungsgrad. Um die Absichten der Anforderungen in die Softwareumgebung darstellen zu können, ist es nötig, die unterschiedlichen Arten von Anforderungen zu klären und abzugrenzen. In diesem Abschnitt wird eine der in der Praxis am häufigsten verwendeten Klassifikationsmöglichkeiten vorgestellt, und zwar die Unterteilung in funktionale und nichtfunktionale Produktanforderungen.

Die **funktionalen Anforderungen** sollen die grundlegenden Aktionen der Anwendung definieren, die bei der Annahme und Verarbeitung der Eingaben, sowie bei der Verarbeitung und Erzeugung der Ausgaben stattfinden müssen. Laut dem Buch „Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level“ [PR15] sind sie wie folgt definiert:

> *„Eine funktionale Anforderung ist eine Anforderung bezüglich des Ergebnisses eines Verhaltens, das von einer Funktion des Systems bereitgestellt werden soll.“*

Das IREB (*International Requirements Engineering Board*) [BH15] untergliedert die **nichtfunktionalen Anforderungen** in Qualitätsanforderungen und Randbedingungen. Die Qualitätsanforderungen beziehen sich auf Merkmale, die die Qualität des Systems gewährleisten sollen und somit Einfluss auf die Systemarchitektur haben. Sie erfassen Anforderungen an die Zuverlässigkeit, Sicherheit, Skalierbarkeit und Performanz der Anwendung. Die Randbedingungen sind die definierten organisatorischen, gesetzlichen und technischen Bedingungen und Einschränkungen für die Umsetzung des Programms.

2.2.2 Rückverfolgbarkeit von Anforderungen

Die Rückverfolgbarkeit von Anforderung (engl. *Requirements Traceability*) ist eine Teildisziplin des Anforderungsmanagements und bezeichnet den Prozess der Verfolgung von Anforderungen, sowohl in Vorwärts- als auch in Rückwärtsrichtung. Die Beziehungen und Abhängigkeiten zwischen Teilen der Artefakte werden durch gerichtete Rückverfolgbarkeitsverbindungen, auch Links oder Traces genannt, dargestellt. Die Nachvollziehbarkeitsbeziehungen können Links in Richtung Entwicklung, in Richtung Herkunft oder Links zwischen Anforderungen sein. Davis [Dav93] differenziert zwischen vier Typen von Rückverfolgbarkeit von Anforderungen:

1. „Rückwärts-von"-Rückverfolgbarkeit: verknüpft Anforderungen mit ihren Quellen in anderen Dokumenten.
2. „Vorwärts-von"-Rückverfolgbarkeit: verknüpft Anforderungen mit den Entwurf- und Implementierungskomponenten.
3. „Rückwärts-zu"-Rückverfolgbarkeit: verbindet die Entwurf- und Implementierungskomponente mit den Anforderungen.
4. „Vorwärts-zu"-Rückverfolgbarkeit: verknüpft andere Dokumente mit relevanten Anforderungen.

Diese Einteilung deckt viele Informationen ab und stellt ein Konzept der Rückferfolgbarkeit vor, bei dem die Informationen als Pfeile vorwärts und rückwärts aus verschiedenen Dokumenten veranschaulicht werden können. Außer den genannten Typen werden in der Praxis oft Informationen verwendet, die die Abhängigkeiten zwischen den Anforderungen selbst aufzeichnen.

2.3 Klassifikation

Die Textklassifizierung beschäftigt sich mit der inhaltlichen Analyse und der Sortierung von Dokumenten nach bestimmten Kriterien. Die einfachste Klassifikation ist die binäre Klassifizierungsaufgabe. Ein Beispiel ist die Stimmungsanalyse, bei der die Eingabe in zwei Klassen eingeteilt werden kann - positiv oder negativ. Die Eingabe kann in diesem Fall ein Datensatz aus Dokumenten, wie Rezensionen eines Films oder eines Produkts, sein, die die Einstellung des Autors zum Produkt ausdrücken.

Es gibt verschiedene Ansätze zur Klassifikation von Dokumenten. Einer davon ist die Verwendung von handgeschriebenen Regeln. Regelbasierte Klassifikatoren können zwar sehr präzise sein, aber es ist oft schwer für die Entwickler, die Regeln zu bestimmen. Außerdem können Daten im Laufe der Zeit geändert werden, was eine Anpassung der Regeln erforderlich machen kann. Eine Alternative sind die maschinellen Lernverfahren. Dabei werden Klassifikatoren darauf trainiert, neuen Daten ihre richtigen Klassen zuzuweisen.

2.3.1 Klassifikation mit mehreren Klassen

Viele Klassifizierungsaufgaben benötigen mehr als zwei Klassen. Sogar bei der Stimmungsanalyse werden normalerweise mindestens drei Klassen definiert - Positiv, Neutral und Negativ. Es gibt zwei verschiedene Arten von Klassifikation mit mehreren Klassen - einerseits die multinomiale, und andererseits die Multi-Label-Klassifikation.

Wenn ein Dokument in viele Kategorien eingeteilt werden kann, da es zum Beispiel für viele Themen gleichzeitig relevant ist, wird eine Multi-Label-Klassifikation (engl. *multilabel, multivalue, any-of classification*) eingesetzt. In diesem Fall kann ein Dokument einer, keiner oder mehreren Klassen zugewiesen werden. Es werden binäre Klassifikatoren für jede Klasse

erstellt und separat angewandt. Die Entscheidung von einem beliebigen Klassifikator hat keinen Einfluss auf die Entscheidungen der anderen.

Bei der multinomialen Klassifikation (engl. *multionomial, polytomous, multiclass, single-lable, one-of classification*) [MRS09] schließen sich die Klassen gegenseitig aus. So eine Klassifikation wird seltener als Multi-Label Klassifikation verwendet, aber ihre Einschränkungen können die Effektivität bei manchen Aufgaben, wie beispielsweise der Identifikation der Sprache eines Textes, erhöhen. Bei diesem Algorithmus werden auch binäre Klassifikatoren für alle Klassen erstellt und separat angewandt, und dann wird eine Bewertung durchgeführt, um die passendste Klasse für jedes Dokument auszuwählen.

2.4 Maschinelles Lernen

Maschinelles Lernen ist ein Teilbereich der künstlichen Intelligenz, der sich mit der Entwicklung von Algorithmen beschäftigt, die auf Basis der Beobachtungen und Erfahrungen mit Daten lernen und dadurch ihre Leistung verbessern [PS12]. Derartige Systeme entwickeln selbst einen passenden Algorithmus für die Lösung eines bestimmten Problems. Sie sollen eine Funktion lernen, die die Eingabedaten auf die gewünschte Ausgabe abbildet. Es gibt drei Haupttypen von ML-Algorithmen:

- *Überwachtes Lernen* - eine Technik, die eine Zuordnung der Eingabedaten zu einem festgelegten Satz von Bezeichnungen (der gewünschten Ausgabe) generiert. Um ein Modell zu trainieren, sollen die Trainingsdaten vorab manuell mit den entsprechenden Bezeichnungen versehen werden.
- *Unüberwachtes Lernen* - eine Technik, die versucht, eine Struktur aus einem Eingabesatz unbeschrifteter Daten zu ermitteln.
- *Semiüberwachtes Lernen* - eine Technik, die eine Zuordnung von sowohl beschrifteten als auch unbeschrifteten Daten generiert.

Für diese Arbeit ist das überwachte maschinelle Lernen relevant. Hier handelt es sich um das Trainieren eines Modells anhand von Daten in textueller Form, die vorab mit bestimmten Kategorien versehen sind. Im Folgenden werden einige häufig verwendeten Algorithmen für überwachtes Lernen vorgestellt.

2.4.1 Naïve Bayes

Der Naïve Bayes Klassifikator ermittelt die Wahrscheinlichkeit, mit der jedes Objekt einer Klasse gehört [PS12]. Die a posteriori Wahrscheinlichkeiten werden mit Hilfe der a priori Wahrscheinlichkeiten und beobachteten Daten berechnet. Der Klassifikator basiert auf dem Satz von Bayes:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

Hier bezeichnet P(A|B) die bedingte Wahrscheinlichkeit des Ereignisses A unter der Bedingung, dass B eingetreten ist, und wird anhand drei anderen Wahrscheinlichkeitsmaße berechnet, für die es möglicherweise einfacher ist, Schätzungen zu finden. Beim Naïve Bayes Klassifikator wird versucht, eine passende Kategorie (C) für die angegebenen Eingabemerkmale (F_1, ... , F_n) zu finden. Man kann die Formel unter Bezugnahme auf die Merkmale umformulieren, wobei P(A|B) als P(C|X_{F_1}, ... , X_{F_n}) dargestellt wird.

2.4.2 Entscheidungsbaum

Ein Entscheidungsbaum klassifiziert die Instanzen eines Datensatzes, indem er sie vom Stammknoten bis zu einem Blatt „entlang des Baums“ sortiert, das eine entsprechende Kategorie repräsentiert [PS12]. Die inneren Knoten stellen jeweils eine Frage dar, die von den Zweigen beantwortet wird, die aus dem Knoten ausgehen. Die Fragen können sich auf alle verfügbaren Informationen aus dem Datensatz beziehen, wie zum Beispiel N-Gramme, Bezeichnungen oder strukturabhängigen Eigenschaften.

2.4.3 Random Forest

Random Forest ist eine Ensemble-Lernmethode, die mehrere Entscheidungsbäume kombiniert, um noch leistungsstärkere Modelle zu erstellen. Die einzelnen Entscheidungsbäume werden separat trainiert, sodass das Training parallel durchgeführt werden kann. Jeder Baum im „Wald“ von Entscheidungsbäumen wird aus einer zufälligen Teilmenge der Merkmale erstellt [Fla12]. Durch Erhöhen der Tiefe jedes Baums wird das Modell i.d.R. ausdrucksvoller und leistungsfähiger. Tiefere Bäume brauchen jedoch mehr Zeit zum Trainieren und sind anfälliger für Überanpassungen. Neben der Anzahl von Parametern in einem Baum ist die Anzahl der Bäume auch ein wichtiger Parameter für den Random-Forest-Algorithmus. Die Erhöhung der Anzahl der Bäume kann die Varianz der Vorhersagen verringern und dadurch die Genauigkeit des Modells verbessern, jedoch benötigt der Algorithmus eine längere Trainingszeit.

2.4.4 Logistische Regression

Logistische Regression (engl. *logistic regression*) ist ein regressionsanalytisches Verfahren, das für Klassifikationsprobleme eingesetzt wird [JM18]. Die logistische Regression untersucht den Zusammenhang zwischen einem Ereignis (abhängige kategoriale Variable) mit einer oder mehreren Einflussfaktoren (unabhängigen Variablen). Im Gegensatz zur linearen Regression, bei der die abhängige Zielvariable intervallskaliert ist, modelliert logistische Regression die Wahrscheinlichkeiten für bestimmte Ergebnisse anstatt der Ergebnisse selbst und wird daher beim Vorhersagen von Kategorien verwendet. Es gibt insgesamt drei Arten der logistischen Regression:

- *Binäre logistische Regression* - Die Antwortvariable hat nur zwei mögliche Ergebnisse. Beispiel: positive oder negative Stimmung
- *Multinomiale logistische Regression* - Die Antwortvariable fällt in eine von mehreren Kategorien. Beispiel: Wortart-Markierung
- *Ordinale logistische Regression* - Die Antwortvariable fällt in eine von mehreren Kategorien und weist eine sinnvolle Ordnung auf. Beispiel: Filmbewertung von 1 bis 10

2.4.5 Stützvektormaschine

Stützvektormaschine (engl. *Support Vector Machine, SVM*) ist ein binärer Klassifikator, der eine Menge von Eingabedaten als Vektoren in einem Vektorraum darstellt und vorhersagt, welcher Klasse jedes Element zugewiesen werden soll [PS12]. Die Hauptidee ist es, in diesem Vektorraum eine Hyperebene zu finden, die maximal von jedem Punkt in den Trainingsdaten entfernt ist und die Objekte in zwei Klassen teilt. Bei der Bestimmung der Lage und Position der Hyperebene werden nur die Trainingsvektoren beachtet, die am nächsten liegen. Diese Vektoren werden auch Stützvektoren (engl. *support vectors*) genannt.

Eine Voraussetzung für die Einsetzung der Hyperebene ist, dass die Objekte linear trennbar sind (siehe Abbildung 2.2). Häufig wird aber diese Bedingung wird nicht erfüllt. Nichtlinear trennbare Daten können verarbeitet werden, indem eine als *Kernel-Trick* bezeichnete

Technik verwendet wird, die die Daten in einer höheren Dimension abbildet, in der sie sich linear verhalten.

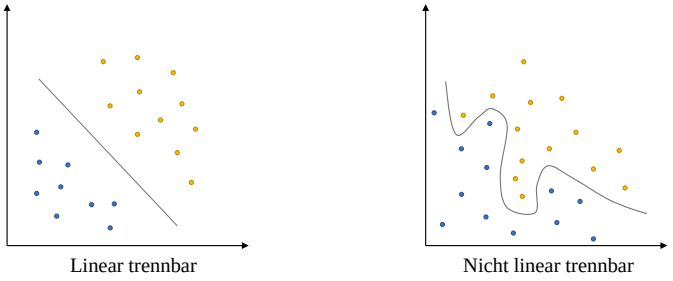

Abbildung 2.2: Lineare Trennbarkeit

3 INDIRECT

In diesem Kapitel wird das Gesamtprojekt INDIRECT (Intent-driven Requirements-to-Code Traceability) [Hey19] vorgestellt. Das in dieser Bachelorarbeit entwickelte Werkzeug soll als Agent in das Projekt eingebunden werden.

3.1 Ziel von INDIRECT

Das Projekt INDIRECT hat das Ziel, eine automatische Generierung von Informationen zur Rückverfolgbarkeit von Anforderungen und Quelltext zu ermöglichen. Zu diesem Zweck werden zuerst die Absichten der in natürlicher Sprache beschriebenen Anforderungen, sowie des Quelltextes analysiert. Die Idee ist, dass das Verständnis der Absichten und die Verwendung von Modellen einen domänenunabhängigen und präzisen Ansatz für die Erzeugung der Verknüpfungen bietet.

Die Verarbeitung der Texteingaben in INDIRECT basiert auf der Rahmenarchitektur PARSE (Programming ARchitecture for Spoken Explanations) [WHT17]. Das Projekt PARSE zielt ursprünglich darauf ab, Programmieren in gesprochener natürlicher Sprache zu ermöglichen. Ein Zielsystem ist der am KIT entwickelte humanoide Haushaltsroboter ARMAR-III [ARA+06]. Mittels gesprochener Anweisungen werden ihm neue Funktionen beigebracht. Darüber hinaus kann die Rahmenarchitektur des Projektes auf sämtliche Probleme übertragen werden, die Textverständnis behandeln.

PARSE ist ein agentenbasiertes System, das sich einfach erweitern lässt. Die Agenten arbeiten unabhängig voneinander und mit ihrer Hilfe werden verschiedene Funktionalitäten in das Projekt integriert. Außerdem können in PARSE externe Wissensquellen, wie die Datenbanken WordNet [Mil95] und Cyc [Cyc], integriert werden, um ein tieferes Sprachverständnis, vor allem mit Bezug auf die Auflösung von Mehrdeutigkeiten und die Eigennamenerkennung, zu ermöglichen.

3.2 Architektur von INDIRECT

Die Architektur von INDIRECT wird in Abbildung 3.1 dargestellt. Gleichzeitig werden zwei Absichtsmodelle erzeugt - eins für die Anforderungen und eins für den Quelltext. Sie modellieren die Absichten der Interessenvertreter und Entwickler unabhängig voneinander. Beide Modelle enthalten Wissen über zugrundeliegende Konzepte und die Beziehungen zwischen ihnen. Nach ihrer Generierung wird eine Zuordnung zwischen beiden Absichtsmodellen gelernt, anstatt direkt zwischen den ursprünglichen Artefakten abzubilden. Diese Zuordnung wird genutzt, um die Rückverfolgbarkeitsverknüpfungen zu erstellen.

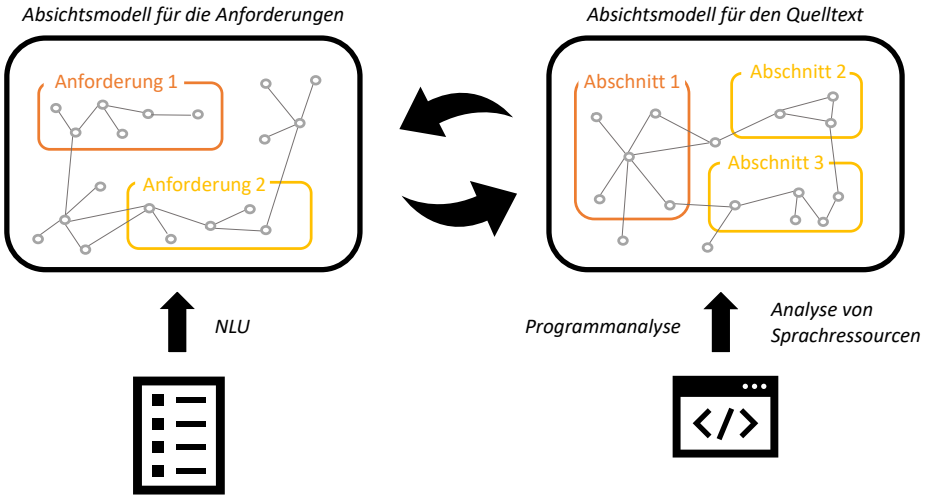

Abbildung 3.1: Architektur von INDIRECT: Ansatz zur Wiederherstellung von Rückverfolgbarkeitsverbindungen

3.2.1 Absichtsmodell für die Anforderungen

Für die Repräsentation der Anforderungsbeschreibungen werden Techniken aus PARSE und Strategien aus dem Bereich der Computerlinguistik verwendet. Die Anforderungen werden zuerst inkrementell durch seichte Sprachverarbeitung (engl. Shallow Natural Language Processing - SNLP) in eine initiale Repräsentationen [Koc15] überführt. Dadurch wird jedem Wort eine Wortart, Phrase im IOB-Format und Befehlsnummer zugewiesen. Daraufhin ergänzen Agenten diese Repräsentation um Informationen wie einzelne Entitäten, ihre semantischen Rollen und die Konzepte, zu denen sie gehören. Die Erkennung erster Beziehungen, wie zum Beispiel Teil-Ganzes, wurde bereits umgesetzt.

3.2.2 Absichtsmodell für den Quelltext

Der Quelltext wird in semantisch verwandte Abschnitte unterteilt. Diese Abschnitte haben bestimmte Absichten und sind die Bausteine des zweiten Absichtsmodells. Dieses Modell wird auch inkrementell generiert. Dabei werden zuerst die Struktur des Quelltexts und die möglichen Testfälle analysiert. Des Weiteren werden die verfügbaren Sprachressourcen, wie Dokumentation und Kommentare, in Betracht gezogen.

3.2.3 Rückverfolgbarkeitsverknüpfungen

Für die Identifizierung der möglichen Verknüpfungen werden Muster in beiden Absichtsmodellen gelernt. Mit Hilfe der wiederkehrenden Struktur der Modelle und der durch das Weltwissen hinzugefügten Informationen weisen diese Muster auf die Zuordnung der Quelltextabschnitte zu ihren jeweiligen Gegenstücken hin.

4 Verwandte Arbeiten

In diesem Kapitel werden die verwandten Arbeiten vorgestellt. Diese lassen sich in zwei Kategorien aufteilen. Zuerst werden Forschungsarbeiten beschrieben, die sich auf unterschiedliche Klassifizierungen von Anforderungen beziehen. Im zweiten Teil werden Arbeiten im Bereich der Computerlinguistik vorgestellt, die sich damit beschäftigen, Sätze zu klassifizieren.

4.1 Klassifikation von Anforderungen

Vlas und Robinson beschäftigen sich in ihrer Arbeit **„A Rule-Based Natural Language Technique for Requirements Discovery and Classification in Open-Source Software Development Projects“** [VR11] mit dem Entwurf und der Validierung eines automatisierten Anforderungsklassifikators (Requirements Classifier for Natural Language - RCNL) für Open Source-Projekte. Anforderungen für solche Projekte werden meistens informell in verschiedenen Informationsressourcen, wie beispielsweise Anfragen, Foren, Chats oder E-Mails, beschrieben. Aus diesem Grund ist die Anforderungsermittlung und die Erkennung von unnötigen Informationen ein wichtiger Bestandteil in dieser Domäne. Die verwendeten Quellen sind auf die Beschreibungen von Arbeitsaufgaben (engl. *work items*) in Diensten wie SourceForge beschränkt. Nachdem die Anforderungen identifiziert werden, werden sie nach ihrer Art (funktionale oder nichtfunktionale Anforderungen – Zuverlässigkeit, Effizienz, Integrität u.a.) klassifiziert. Ein Schlüsselelement für die Klassifikation ist die mehrstufige Ontologie, bei der die unteren Ebenen auf Grammatik und die oberen Ebenen auf Anforderungen basieren. Für jede Ebene legen die Regeln fest, wie der Text mit den Konzepten dieser Ebene gekennzeichnet wird. Die Regeln werden in einer Pipeline von Stufe 0 bis Stufe 5 verarbeitet und jeder Text kann mehrere Tags aus mehreren Ebenen enthalten. Die Implementierung des RCNL-Klassifikators besteht aus etwa 205 Regeln für alle Ebenen. Um den Klassifikator zu evaluieren, wurden zwei Arten von Validierungsmethoden angewandt. Nach der Generierung von Metriken aus der automatischen Klassifizierung der Daten werden die Ergebnisse zusätzlich mit denen eines Experten verglichen. Mit einem F-Maß von 76% für die Anforderungserkennung durch das entwickelte Werkzeug sind die Ergebnisse vielversprechend.

Sicherheitsanforderungen sind bei Anforderungsspezifikationen nicht immer explizit definiert und werden daher möglicherweise von Entwicklern übersehen. Deshalb wird das Forschungsprojekt **„Security Discoverer“** [RKSW14] entwickelt. Als Eingabe werden Anforderungsartefakte in natürlicher Sprache verwendet und mithilfe von Techniken aus

dem maschinellen Lernen werden die sicherheitsrelevanten Sätze automatisch identifiziert. Sie werden dann nach Sicherheitszielen, wie Vertraulichkeit, Integrität und Verfügbarkeit, klassifiziert. Damit Gemeinsamkeiten der Sätze identifiziert werden können, werden gemeinsame Muster und Themen in der Satzstruktur, sowie Schlüsselwörter in den Sätzen definiert, und eine Gruppierung von Sätzen (k-mediods / LDA) durchgeführt. Um die wichtigsten 20 Schlüsselwörter für jedes Sicherheitsziel zu extrahieren, wird der Informationsgewinn (engl. *information gain*) verwendet. Eine Vielzahl von Klassifikatoren wird unter der Verwendung einer zehnfachen Kreuzvalidierung auf dem Datensatz bewertet - ein k-Nächste-Nachbarn-Klassifikator (k-NN) (k=1), ein multinomialer Naïve-Bayes-Klassifikator (NB) und ein sequentieller Minimaloptimierungsklassifikator (engl. *sequential minimal optimization classifier, SMO*). Außerdem wird ein „kombinierter“ Klassifikator entwickelt, der die Ergebnisse des k-NN-Klassifikators verwendet, wenn relativ nahe Sätze gefunden wurden. Andernfalls verwendet er den Mehrheitsentscheid der drei Klassifikatoren. Es werden 10.963 Sätze in Dokumenten aus dem Gesundheitsbereich klassifiziert und die entsprechende Sicherheitsziele extrahiert. Die manuelle Analyse ergibt, dass 28% aller sicherheitsrelevanten Sätze explizit angegeben sind, während 72% der Sätze funktionale Anforderungen mit Auswirkungen auf die Sicherheit sind. Die Evaluation zeigt, dass beim Vorhersagen der Sicherheitsziele der kombinierte Klassifikator eine Präzision von 82% und eine Ausbeute von 79% erreicht.

Kurtanović und Maalej stellen in ihre Arbeit **„Automatically Classifying Functional and Non-Functional Requirements Using Supervised Machine Learning“** [KM17] einen Ansatz zur Klassifikation von Anforderungen mithilfe von überwachtem Lernen vor. Zum Einsatz kommt eine Stützvektormaschine (engl. *Support-Vector Machine, SVM*), sowie die Verwendung von Metadaten und lexikalischen und syntaktischen Merkmalen. Für die Textvorverarbeitung werden Entfernung von Satzzeichen, Entfernung von Stoppwörtern und Lemmatisierung, sowie die Extraktion von Klassifizierungsmerkmalen, wie Text-N-Gramme, Wortart-Markierung und CP-Unigramme, eingesetzt. Als Erstes wird untersucht, wie genau die Anforderungen automatisch in funktionale (F) und nichtfunktionale (NF) klassifiziert werden können. Darüber hinaus werden die spezifischen Arten von NFs, insbesondere Nutzbarkeit, Sicherheit, Betrieb und Leistung, identifiziert. Dabei werden vier Binärklassifikatoren entworfen, um die vier häufigsten Arten von nichtfunktionalen Anforderungen zu erkennen. Es werden Unter- und Überabtastungsstrategien angewendet, um eine ausgewogene Klassenverteilung zu erreichen. Zusätzlich wird ein Multi-Label-Klassifikator verwendet, um die vier Klassen vorherzusagen. Alle Klassifikatoren werden mittels 10-facher Kreuzvalidierung bewertet. Der F/NF Binärklassifikator erreicht eine Ausbeute und Präzision von 92% für beide Klassen. Bei der Identifizierung von spezifischen nichtfunktionalen Anforderungen erzielt der Binärklassifikator für Leistungsanforderungen die besten Ergebnissen mit einem F-Maß von 90%.

In der Arbeit **„Using Frame Embeddings to Identify Semantically Related Software Requirements“** [ABZ19] wird eine Methode zur Ermittlung von semantisch verwandten oder ähnlichen Softwareanforderungen vorgestellt, um die Zusammenhänge zwischen ihnen zu verstehen und ihre Rückverfolgbarkeit in Anforderungsdokumenten zu verbessern. Die semantischen Beziehungen zwischen Anforderungen werden auf Basis der semantischen FrameNet-Rahmen gemessen, die ihre lexikalische Einheiten (engl. *lexical units*) hervorrufen. Das FrameNet-Projekt baut eine lexikalische Datenbank auf, die mehr als 200 000 manuell markierte Sätze enthält, die mit mehr als 1 200 semantischen Rahmen verknüpft sind. Die Rahmeneinbettungen werden auf einer Sammlung von mehr als drei Millionen Nutzerrezensionen für mobile Anwendungen aus unterschiedlichen Themenbereichen trainiert. Die Vorverarbeitung der Trainingsdaten umfasst die folgenden Techniken: Satzteilung (engl. *sentence splitting*), Tokenisierung, Entfernen von Stoppwörtern (engl. *stop-word removal*), Wortart-Markierung und Lemmatisierung. Unter Verwendung eines

vorverarbeiteten Datensatzes werden Worteinbettungen mit der kontinuierlichen Bag-Of-Words-Lernmethode von `Word2Vec`[1] trainiert. Aus den Worteinbettungsvektoren wird eine einbettungsbasierte Darstellung von semantischen Rahmen gebildet. Die erzeugten Rahmeneinbettungen werden zur Bestimmung der Ähnlichkeiten und der Beziehungen zwischen einem Paar von Anforderungsaussagen verwendet. Für die Evaluation des Verfahrens wird ein manuell markierter Korpus verwendet. Zum Vergleich wird ein Basissystem mit vorab trainierten Worteinbettungen implementiert. Die vorgeschlagene Methode erzielt eine Leistung in Bezug auf das F-Maß von 86% und übertrifft somit das Basissystem um 24%.

Die Arbeit **„Using semantic roles to improve text classification in the requirements domain“** [RMD18] zielt darauf ab, die Klassifizierung von Texten zu verbessern, indem Informationen über semantische Rollen genutzt werden. Die Autoren entwickeln Multi-Label-Klassifikatoren zur Identifizierung spezieller Konzepte, so genannter Domain-Aktionen, in Anforderungsdokumenten. Eine Domain-Aktion (DA) erfasst die Absicht eines bestimmten Satzes und bietet ein tieferes Verständnis seiner Semantik. Die Kategorisierung von DA's wird als Mehrfachetiketten-Klassifizierungsproblem behandelt, da zwei oder mehr Verhaltensweisen innerhalb desselben Satzes enthalten werden können und eine Anforderung mit verschiedenen DAs versehen werden kann. Insgesamt werden 25 DA-Klassen definiert und in einer dreistufigen Hierarchie angeordnet. Die hierarchische Struktur trägt dazu bei, die Leistung der Textklassifizierung zu verbessern. Ein wichtiger Aspekt des vorgeschlagenen Ansatzes ist die semantische Anreicherung von Datensätzen mit Informationen über semantische Rollen, die dem Text mittels Techniken zur Kennzeichnung von semantischen Rollen (engl. *semantic role labeling - SRL*) hinzugefügt werden. Es wird ein neuer Datensatz unter Verwendung von Anforderungsspezifikationen aus echten Softwaresystemen definiert. Insgesamt werden ca. 900 Sätze gesammelt und manuell mit DA's versehen (markiert). Für die Klassifizierung werden drei Algorithmen untersucht - Entscheidungsbäume (J48), Naïve Bayes (NB) und SVM. Außerdem werden zwei Transformationen behandelt, um mehrere Bezeichnungen und Klassen zu unterstützen: Label-Powerset (LP) und Binärrelevanz (engl. *binary relevance, BR*). Zusätzlich wird wegen der hierarchischen Struktur der DA-s noch ein Algorithmus angewendet, um die „ist-ein“-Beziehungen zwischen DA-s zu berücksichtigen. Das Ziel der Evaluierung ist, die Vor- und Nachteile einer mit SRL angereicherten Klassifizierung im Vergleich zu einem reinen Textklassifizierung zu ermitteln und diese Ergebnisse mit anderen Anreicherungsstrategien zu vergleichen. Die Auswertung mehrerer Klassifikatorkonfigurationen mit realen Anforderungen ergibt aufgrund der SRL-Anreicherung konsistente Verbesserungen der Präzision und Ausbeute, wobei der NB-Klassifikator und die SVM-LP-Konfiguration die besten Verbesserungen erzielen - 12 bzw. 18% für beide Metriken. Der Klassifikator mit der besten Leistung ist die Kombination aus BR und SVM mit einer Präzision von 76%. Des Weiteren wird die Leistung von Anreicherungsstrategien für Textklassifizierer wie WordNet und Wikipedia evaluiert. Im Vergleich zum Training mit reinen Daten können jedoch diese Strategien die Ergebnisse nicht wesentlich verbessern. Daraus lässt sich die Schlussfolgerung ziehen, dass herkömmliche Anreicherungsstrategien wie WordNet und Wikipedia nicht für Softwareanforderungsspezifikationen geeignet sind und im Gegensatz dazu das Anreichern von Textanforderungen mit semantischen Rollen die Vorhersage der Klassen erheblich verbessern kann.

4.2 Satzklassifikation

Eine weit verbreitete Technik in NLP ist das Darstellen von Wörtern als niedrigdimensionale Vektoren, auch als Worteinbettungen bekannt, da sie allgemeine semantische und

[1] Quelle: https://code.google.com/p/word2vec/, zuletzt besucht am 06.10.2019

syntaktische Eigenschaften der Wörter erfassen können. Deshalb wird in die Arbeit **„Dependency Based Embeddings for Sentence Classification Tasks“** [KM16] die Wirksamkeit von verschiedenen Arten von Worteinbettungen für Wortähnlichkeits- und Satzklassifizierungsaufgaben verglichen. Verglichen werden ein fensterbasiertes, ein abhängigkeitsbasiertes und ein erweitertes abhängigkeitsbasiertes Skipgramm-Modell. Ersteres ist ein Standard-Skipgramm-Modell, das den Text in einem Fenster von 5 Wörtern rechts und links vom Zielwort berücksichtigt. Das zweite Modell ersetzt Kontextwörter in einem Fenster durch Abhängigkeitskontexte (engl. *dependency context*), wobei ein Abhängigkeitskontext ein diskretes Symbol ist, das ein Wort und seine syntaktische Rolle in einem Abhängigkeitsanalyse-Diagramm bezeichnet. Das erweiterte Modell ist eine Variation des Skipgramms basierend auf Abhängigkeitsgraphen, wobei Informationen über zusätzliche gemeinsame Auftreten genutzt werden. Jedes Zielwort wird als Knoten im Abhängigkeitsgraphen dargestellt und die Wahrscheinlichkeit für andere Wörter im Abstand eins und zwei im Graphen wird maximiert, um die Worteinbettungen zu optimieren. Drei häufige Satzklassifizierungsaufgaben werden betrachtet, nämlich Fragentypklassifizierung, binäre Stimmungsvorhersage und Beziehungsidentifikation. Die Experimente zielen darauf ab, die Auswirkung verschiedener Kontextmerkmale für Worteinbettungen bei der Satzklassifizierung zu beurteilen. Für jede Aufgabe wird die Nützlichkeit syntaxbasierter Worteinbettungen und Abhängigkeitskontext-Einbettungen mit drei verschiedenen Satzklassifizierungsmethoden bewertet - SVM-BoE (Bag-Of-Embeddings) mit gemittelten Einbettungen von Satzmerkmalen (Wörter und Abhängigkeitskontexte), einem faltenden neuronalen Netzwerk (engl. *Convolutional Neural Network, CNN*) und LSTM (Long Short Term Memory Network). Die Auswertung zeigt, dass Abhängigkeitskontext-Einbettungen wertvolle syntaktische Informationen für Satzklassifizierungsaufgaben liefern können. Die Fragenklassifizierung und die Beziehungsidentifikation erzielen große Verbesserungen bei der Verwendung von Einbettungen im Abhängigkeitskontext im Vergleich zur Ausgangsbasis, während die Stimmungsklassifizierung nur mäßige Verbesserungen aufweist. Von allen drei Klassifizierungsmethoden profitiert die SVM-BoE am meisten vom Hinzufügen von Einbettungen im Abhängigkeitskontext, da dies die einzige Quelle für strukturelle Informationen ist. Obwohl der Zweck der Experimente ein Vergleich der Einbettungen ist, liegen die Ergebnisse des CNN unter Verwendung des erweiterten abhängigkeitsbasierten Skipgramm-Modells für Fragenklassifizierung (F-Maß von 95%) und Abhängigkeitsidentifikation (84.3%) in der Nähe der besten berichteten Ergebnisse für dieselben Aufgaben (96% bzw. 85.6%).

In der Arbeit **„Convolutional Neural Networks for Sentence Classification“** [Kim14] wird die Durchführung von Experimenten mit faltenden neuronalen Netzwerken beschrieben. Vier Modellvariationen werden vorgestellt: ein Basismodell, bei dem alle Wörter zufällig initialisiert und beim Training geändert werden, ein statisches Modell mit vorab trainierten Worteinbettungen aus `Word2Vec`, bei dem alle Wörter statisch bleiben und nur andere Parameter des Modells gelernt werden, ein nicht-statisches Modell, bei dem vorab trainierte Vektoren für jede Aufgabe feinabgestimmt werden, und ein Mehrkanalmodell mit zwei Sätzen von Worteinbettungen, bei dem ein Satz fein abgestimmt werden kann, während der andere statisch bleibt. Die Modelle verbessern den Stand der Technik bei vier von sieben Aufgaben, einschließlich Filmkritiken und Kundenrezensionen zu verschiedenen Produkten. Das Basismodell an sich erzielt schlechte Ergebnisse, dagegen weist aber sogar das einfache Modell mit statischen Vektoren eine gute Leistung auf. Es erreicht eine Verbesserung der Genauigkeit (engl. *accuracy*) um 2,4% bei der Erkennung der Polarität einer Meinung. Die Feinabstimmung der vorab trainierten Vektoren für jede Aufgabe führt zu weitere Leistungssteigerungen und erzielt die besten Ergebnissen bei der Klassifizierung von Filmkritiken mit Genauigkeit von 81,5%.

Die Arbeit **„Experiments with Sentence Classification“** [KMA06] befasst sich mit Satzklassifizierungsaufgaben für Textdaten und vergleicht anhand verschiedener Experi-

mente die Verwendung von einigen gängigen Algorithmen zur Klassifizierung mit der Verwendung von aktuellen Methoden zur Merkmalsselektion (engl. *feature selection*). Der Korpus besteht aus 160 E-Mail-Dialogen zwischen Kunden und Betreibern, wobei der Fokus auf die Antwort-E-Mails liegt, da diese wohlgeformte grammatikalische Sätze enthalten. Insgesamt werden 1486 Sätze klassifiziert, die eine Vielzahl von Themen behandeln, wie z.B. Anfragen nach technischer Unterstützung, Produktanfragen und Fragen zur Rücksendung fehlerhafter Produkte. Sie werden in die folgenden Klassen eingeteilt: `Apology, Instruction, Instruction-Item, Saluation, Suggestion, Specification, Statement, Request, Question, Signature, Thanking, Response-Ack, Url, Others`. Sätze, die aus mehreren unabhängigen Klauseln bestehen, die durch Konjunktionen verbunden sind, werden nur einer Klasse zugeteilt, wobei eine dominante Klausel gewählt und berücksichtigt wird. Es werden dann drei Klassifizierungsalgorithmen angewendet - NB, Entscheidungsbaum und SVM. Zunächst wird eine Untersuchung mit dem binären Bag-Of-Words-Ansatz ohne weitere Verarbeitung durchgeführt. Bei dieser Basisrepräsentation weist der Mikro-F_1-Durchschnitt (engl. *micro-average*), aufgrund der Existenz sehr kleiner Klassen, durchweg eine bessere Leistung als der Makro-F_1-Durchschnitt (engl. *macro-average*). Bei beiden Kennzahlen übertrifft SVM die anderen Klassifikatoren. Nach Experimentieren mit unterschiedlichen Darstellungstechniken wird herausgestellt, dass die Verwendung von Tokenisierung und Gruppierung von bestimmten Arten von Wörtern, die einzelne Merkmale darstellen, die besten Ergebnissen erzielt. Diese Methode führt zu einer Reduzierung der Anzahl der Merkmale von ca. 40%. Darüber hinaus wird das F-Maß für alle Klassifikatoren verbessert. Der Mikro-F_1-Durchschnitt beträgt bei SVM 88%. Ein überraschendes Ergebnis der Untersuchung ist, dass sich zwei der häufigsten Vorverarbeitungstechniken, das Entfernen von Stoppwörtern und die Lemmatisierung, schädlich für die Leistung erweisen. Des Weiteren werden vier Merkmalsauswahlmethoden bewertet - Chi-Quadrat, Informationsgewinn, Bi-Normal Separation und Satzhäufigkeit (engl. *sentence frequency*), indem die Leistung der Klassifikatoren beim Training mit einer zunehmenden Anzahl an Merkmalen überprüft wird. Die SVM- und Entscheidungsbaum-Klassifizierer sind bei der Merkmalsauswahl weniger empfindlich als NB. Wenn mindestens 300 Merkmale erhalten bleiben, verhalten sich die vier Methoden sich ähnlich. Da Sätze viel kleiner als Dokumente und weniger reich an Textinformationen sind, wird aus den Beobachtungen geschlossen, dass das Ausmaß der Nützlichkeit der Merkmalsauswahl bei Satzklassifikation nicht so groß ist wie bei der Textklassifizierung.

In der Arbeit **„TimeML Events Recognition and Classification: Learning CRF Models with Semantic Roles“** [LSNC10] wird der Beitrag semantischer Rollen zur Erkennung und Klassifizierung von Ereignissen analysiert, mit Fokus auf die TimeML-Ansicht von Ereignissen. Time ML ist eine Spezifikationssprache für Ereignisse und zeitliche Ausdrücke in natürlicher Sprache. Es werden sieben Klassen definiert: `Reporting, Perception, Aspectual, I_Action, I_State, State, Occurrence`. Das Hauptziel der Arbeit ist es, den Beitrag der semantischen Rollen als zusätzliches Merkmal unter Verwendung von bedingten Zufallsfeldern (engl. *conditional random fields, CRFs*) bei der Erkennung und Klassifizierung von Ereignissen zu analysieren. Semantische Rollen liefern nicht nur strukturelle und syntaktische Beziehungen der Prädikate, sondern auch weitere Informationen über die semantische Beziehungen zwischen den Argumenten eines Prädikats. Darüber hinaus können verschiedene Einstellungen der semantischen Rollen bestimmte Ereignisklassen darstellen. Für den Lernprozess wird der CRF-L2 Algorithmus mit Hyperparameter C=1 durchgeführt. Die verwendeten Merkmale lassen sich in zwei Gruppen aufteilen - einerseits die allgemeinen Merkmale und andererseits die semantischen Rollenmerkmale. Die allgemeinen Merkmale umfassen morphologische, syntaktische und ontologische Informationen. Zuerst werden Tokenisierung, Wortart-Markierung und Lemmatisierung durchgeführt. Die syntaktischen Informationen werden berücksichtigt, da in bestimmten Arten von Phrasen und syntaktischen Abhängigkeiten verschiedene Ereignisse enthalten sind. Um die lexikali-

sche Semantik der Wörter zu bestimmen, werden vier WordNet Top-Ontologie-Klassen für jedes Wort erhalten. Die spezifischen Rollenmerkmale werden unter Berücksichtigung des PropBank-Rollensatzes entwickelt. Die Autoren haben sich für PropBank entschieden, weil hier eine höhere Abdeckung als bei FrameNet geboten wird. Die Rollenmerkmale enthalten die Berücksichtigung der semantischen Rolleninformationen für jedes Token, die Bestimmung des Leitverbs (engl. *governing verb*), zu dem das aktuelle Token eine bestimmte Rolle hat, die Bestimmung der Rolle+Verb Kombinationen, um die Beziehung zwischen den ersten zwei Merkmalen zu erfassen und damit neue Klassifizierungsinformationen zu bekommen, und die Festlegung der Rollenkonfigurationen, die aus dem Satz von Rollen abhängig von einem Verb bestehen. Mithilfe der Rollenkonfigurationen können verschiedene Satzeinstellungen unterschieden werden und damit kann entschieden werden, ob ein Token ein Ereignis in einer bestimmten Satzart bezeichnet. Alle berichteten Ergebnisse der Auswertung werden mit einer 5-fachen Kreuzvalidierung erzielt. Das System erreicht ein F_1 von 81,4% bei der Erkennung und 64% bei der Klassifizierung von Ereignissen. Von allen Klassen erzielt `Reporting` die besten Ergebnisse, da 80% der dazugehörigen Ereignissen durch Lemmata „sagen“ und „berichten“ definiert werden. `I_Action` und `State` zeigen die schlechtesten Ergebnisse. Die Autoren ziehen den Schluss, dass eine bessere Identifizierung der `I_Action`-Klasse durch Verwendung von Wortsinn-Disambiguierungstechniken (engl. *word sense disambiguation*) erzielt werden kann, die aber im vorgeschlagenen Ansatz nicht integriert sind. Die schlechte Leistung für die `State`-Klasse ost auf die Vielfalt an Lemmata, Wortart-Markierung und Phrasen zurückzuführen. Um die Auswirkung der semantischen Rollen zu bewerten, wird zusätzlich eine Version des Ansatzes ohne Rolleninformationen evaluiert. Diese Version verwendet nur die allgemeinen Merkmale. Beim voll ausgestatteten Ansatz verbessert sich insbesondere die Ausbeute – um 6% bei der Erkennung bzw. 10% bei der Klassifizierung. Um den Beitrag der CRFs zu messen, wird außerdem ein Experiment mit SVM durchgeführt. Es wird bestätigt, dass CRF bessere Ergebnisse als SVM mit einem um 5% höheren F_1-Maß liefert.

In der Arbeit **„Spatial Pyramid Pooling Mechanism in 3D Convolutional Network for Sentence-Level Classification“** [OGZ18] wird zum ersten mal ein tiefes neuronales Netzwerk vorgeschlagen, das 3D-Faltung und SPP-Schichten (*Spatial Pyramid Pooling*) integriert, um NLP-Probleme zu lösen. Das ursprüngliche CNN wird auf 3D-CNN verbessert, um mit der n-Gramm-Eingabe umzugehen. Die Originalsätze werden durch n-Gramm-Sequenzen ersetzt, wobei die Eingabedimension erweitert wird. Die neue Dimension wird durch die Aufteilung von Sätzen in Phrasen von k-Wörtern definiert. Anschließend werden 3D-Faltungsschichten angewandt, um lokale Wort-Ebenen-Korrelationen, die hauptsächlich in den Phrasen enthalten sind, zu extrahieren. Daher kann die vorgeschlagene Methode kompliziertere interne Beziehungen in Sätzen als 2D-CNN erfassen. Darüber hinaus wird die ursprüngliche SPP-Struktur in eine 3D-Phasenpoolingoperation aktualisiert, damit Sätze mit unterschiedlichen Längen ohne großen Aufwand von Null-Padding akzeptiert werden können. Das Rauschen bei kurzen Sätzen wird erheblich reduziert, indem Sätze in verschiedene Abschnitte entsprechend ihrer tatsächlichen Länge eingeteilt werden. Sie werden mit Nullen bis zur oberen Grenze des Abschnitts aufgefüllt, wobei die Längen der einzelnen Abschnitte [17, 24, 44, 53] sind. Es werden Experimente für die Klassifikation von Sätzen und von Beziehungen durchgeführt. Die Modelle werden an fünf Datensätzen für Satzklassifikation getestet, die mehrere Filmkritiken (SST-1, SST-2), Fragenarten (TREC), subjektive/objektive Sätze (Subj) und Yelp-Bewertungen enthalten. Bei der Vorverarbeitung werden Word2Vec-Worteinbettungen mit dem kontinuierlichen Bag-of-Words-Ansatz trainiert. Ein weiterer Schritt ist die Initialisierung von Faltungskernen. Da Sätze in überlappende Phasen aufgeteilt werden, werden die 3D-Kerne mit gleichmäßiger Verteilung auf zwei Dimensionen außer der Dimension der sequentiellen Länge initialisiert. 3D-Kerne werden also als mehrere in der dritten Dimension verkettete 2D-Kerne behandelt. Die Autoren vergleichen die Ergebnisse des Werkzeugs mit diesen

anderer hochmodernen Basislinien (CNN-MC, RNTN, MGNC-CNN, MVCNN, BLSTM-2DP, GRA, S-LSTM) und zeigen, dass die mittlere Genauigkeit des 3D-SPP-CNN-Modells (k=3) den Stand der Technik auf vier von fünf Datensätzen übertrifft. Der vorgeschlagene Ansatz erreicht eine höhere Genauigkeit als andere Modelle in SST-2, TREC-, Subj- und Yelp-Datensätzen um bis zu 7,6%, 3,6%, 1,3% bzw. 5,9%. Zusätzlich werden Hilfsexperimente auf dem SemEval-2010-Datensatz für die Aufgabe der Beziehungsklassifizierung durchgeführt, um die universelle Wirksamkeit des Modells bei Merkmalsextraktion zu demonstrieren. Dabei wird Wortpositionseinbettung integriert, um die relativen Abstände zwischen dem i-ten Wort und den markierten Entitäten wiederzugeben. Zur Bewertung verschiedener Methoden wird der Makro-F_1-Durchschnitt verwendet. Das Modell wird mit früheren Methoden verglichen, einschließlich solchen, die sich auch stark auf Vorkenntnisse stützen – SVM, RNN, MVRNN, FCM, DRNNs, CNN+softmax, CR-CNN. Das 3D-SPP-Netzwerk erreicht einen Makro-F_1-Durchschnitt von 87,2% und übertrifft damit alle bisherige Methoden.

5 Analyse und Entwurf

Software-Anforderungen stellen die erforderlichen Bedingungen und Fähigkeiten eines Systems dar. Damit die fachlichen und technischen Leistungsmerkmale erfüllt werden, ist ihre detaillierte Beschreibung besonders wichtig. Obwohl die Produktanforderungen die zu implementierenden Systemfunktionalitäten nicht konkret beschreiben, können sie als eine Grundlage für die Ausschreibung der Entwicklung dienen. Die Anforderungen bestimmen unter anderem die erwarteten Funktionen und Eigenschaften der Anwendung, die Systemgrenzen und Beschränkungen, sowie die Schnittstellen zu externer Software. Daraus ergibt sich die Struktur, sowie die möglichen Interaktionen eines Systems. Da Anforderungen verschiedene Arten von Informationen tragen können, unterscheiden sich auch ihre Repräsentationen im Quellcode.

Die semantische Klassifikation der Anforderungen dient dazu, die automatische Generierung eines Absichtsmodells der Anforderungen zu ermöglichen, um dieses für die Rückverfolgbarkeit der Anforderungen einzusetzen. Die Aufgabe dieser Bachelorarbeit ist es, passende Satzkategorien für die verschiedenen Semantiken, die Anforderungen ausdrücken können, zu finden und einen Agenten von INDIRECT (siehe Kapitel 3) zu entwickeln, der diese vorhersagt. Dabei werden nur relevante Arten von Anforderungen berücksichtigt, die bestimmte Qualitätskriterien erfüllen. Da die Spezifikation dieser Arten erforderlich für die Auseinandersetzung mit der weiteren Analyse ist, werden sie in Abschnitt 5.1 definiert. Die Identifizierung der semantischen Kategorien wird dann in Abschnitt 5.2 beschrieben. Abschnitt 5.3 befasst sich mit den möglichen Vorgehensweisen für ihre Erkennung.

5.1 Arten von Anforderungen und Qualitätskriterien

Da sich INDIRECT auf die Rückverfolgbarkeit der Anforderungen in Richtung Implementiertung bezieht, wird der Fokus auf funktionale Anforderungen (siehe Abschnitt 2.2.1) gelegt. Dennoch ist es möglich, dass Anforderungen, die als nichtfunktional bezeichnet wurden, funktionsrelevante Merkmale spezifizieren und somit auch von Belang für die Implementierung sind. Beispiele dafür sind die Einschränkung „*The system shall not allow for a product to exist in more than one submenu.*“ und die Sicherheitsanforderung „*Only authorized users shall have access to clinical site information.*“. Es wird also angenommen, dass auch nichtfunktionale Anforderungen auftreten können, die aber für die nähere Spezifikation der Funktionen hilfreich sind.

Damit die geforderten Leistungen und das Verhalten des Systems durch die funktionalen Anforderungen gut dargestellt werden, ist es wichtig, dass gewisse Qualitätskriterien

bei der Erstellung der Anforderungsdokumentation beachtet werden. Hoffmann [Hof13] fasst die qualitätsbestimmenden Eigenschaften von Anforderungen zusammen. Eine hochwertige Anforderung soll vollständig, korrekt, konsistent, bewertbar, prüfbar, eindeutig, modifizierbar, verfolgbar, vollständig, notwendig, realisierbar, abgestimmt, klassifizierbar, gültig, aktuell und atomar sein. Für diese Bachelorarbeit ist die Atomarität der Anforderungsbeschreibungen besonders wichtig. Um eine semantische Sortierung zu ermöglichen, ist es empfehlenswert, dass die einzelnen Sätze einen Sachverhalt isoliert beschreiben. Ein Gegenbeispiel ist die Anforderung „*The system must provide search functionality, which must include the ability to search by the dispute case number, the cardmember account number and the issuer number and must further allow the user to limit the results of the search by a date range and the case status (open closed or all).*“. Anforderungen, die aus mehreren Sätzen bestehen oder mehr als eine Funktionalität oder Eigenschaft der Anwendung beschreiben, sind nicht atomar und sollten in mehrere Anforderungen aufgeteilt werden. Aufgrund der uneindeutigen Semantik der Teile der nicht atomaren Anforderungen werden im Rahmen dieser Arbeit nur atomare Anforderungen betrachtet.

5.2 Satzkategorien

Das Ziel der funktionalen Anforderungen besteht darin, das zu entwickelnde System und dessen Funktionen zu beschreiben. Basierend auf den Anforderungen können bestimmte Prozesse innerhalb des Systems, sowie mögliche Zustände, in denen sich das System befinden kann, erkannt werden. Das Verständnis der verschiedenen Absichten der Sätze in Anforderungsbeschreibungen hilft dabei, das erwartete Verhalten des Zielsystems im Quelltext abzubilden. Nach einer Untersuchung von mehreren funktionalen Anforderungen wurden zwei Hauptarten von Informationen detektiert. Dadurch können Sätze in Anforderungen als Erstes in zwei Hauptgruppen aufgeteilt werden. Die erste Gruppe von Anforderungen beschreibt Vorgänge und Zustandsänderungen, die im Zielsystem auftreten können. Das sind zum Beispiel Nutzerinteraktionen wie „*The user shall be able to pay with a credit card*“, die einen möglichen Prozess (*to pay*) darstellen und Funktionen im Zielsystem beschreiben können. Die zweite Gruppe von Anforderungen bezieht sich auf die Gestalt des Systems, Eigenschaften und Beziehungen zwischen Entitäten, die keine Übergänge oder Interaktionen betreffen. Beispiele dafür sind Anforderungen wie „*The credit card number must be valid.*“, die ein mögliches Ergebnis oder eine Voraussetzung beschreiben, ohne eine dazu führende oder daraus resultierende Operation zu definieren.

In der Arbeit „TimeML Events Recognition and Classification: Learning CRF Models with Semantic Roles“ [LSNC10], die sich auf die TimeML-Ansicht von Ereignissen konzentriert, wird eine ähnliche Differenzierung zwischen „Events“ durchgeführt. TimeML definiert „Events“ als Situationen, die geschehen oder auftreten, oder als Elemente, die Zustände oder Umstände beschreiben. In der Forschungsarbeit wird u. a. die Klasse `STATE` abgegrenzt, die verschiedene Umstände repräsentiert. Die Autoren stellen fest, dass diese Umstände oft an dem Verb „*to be*“ und einem variablen Merkmal erkannt werden können.

Beispiel 5.1: STATE und OCCURANCE

It’s <EVENT class=„OCCURRENCE“>turning</EVENT> out to be another <EVENT class=„STATE“>bad</EVENT> financial week.

Da auftretende Ereignisse hauptsächlich durch Verben bezeichnet werden, schlagen die Autoren eine Grundlinie vor, die alle Verben als `OCCURRENCE` annotiert. In der Forschungsarbeit wird der Beitrag semantischer Rollen zur Verbesserung der Leistung des gesamten Systems analysiert. Die Autoren argumentieren, dass die Verwendung von Informationen

über semantische Rollen zur Erkennung und Klassifizierung von TimeML-Events hilfreich sein kann. Zum Beispiel führt die AM-MNR-Rolleninformation zu einer korrekten Erkennung der `STATE`-Klasse, die ohne die Anwendung von semantischen Rollen nicht identifiziert wurde. Semantische Rollen stellen strukturelle Beziehungen der Prädikate zu ihren Argumenten bereit (siehe Abschnitt 2.1.6) und können bestimmte Event-Klassen darstellen. Obwohl hier einzelne Wörter und nicht ganze Sätze klassifiziert werden, ist das Konzept der Differenzierung von `STATE`- und `OCCURENCE`-Klassen auch für die Klassifikation von Anforderungen relevant, da sie unterschiedliche Zustands- und Ereignisbeschreibungen darstellen können.

Bei Produktanforderungen kann eine weitere Spezifizierung und Unterteilung der Zustands- und Prozessbeschreibungen definiert werden. Die Absichten der Sätze werden in Hinblick auf ihre Darstellung im Quelltext betrachtet. Ein UML-Diagramm besteht aus Klassen mit ihren Attributen und Methodendefinitionen, sowie Beziehungen zwischen Klassen.

Zustandsbeschreibungen können zur Erkennung von Attributen der Entitätsklassen verwendet werden. Zustände stellen keine Transformationen oder Prozesse, sondern mögliche Eigenschaften oder Merkmale des Subjekts dar. Aus diesem Grund können sie wie in [LSNC10] oft an dem Verb *„to be“*, gefolgt von einem Adjektiv oder Substantiv, erkannt werden.

Der Zustand, in dem sich das Zielsystem befindet, betrifft außerdem auch Beziehungen zwischen Klassen. Eine oft auftretende Beziehung zwischen Entitäten in Anforderungsbeschreibungen ist die Aggregation – eine binäre Assoziation, die eine Teil-Ganzes-Beziehung definiert. Sätze, die sich auf Aggregationen beziehen, enthalten sowohl ein Subjekt, als auch ein Objekt. Sie stellen aber weder einen Zustand einer konkreten Klasse, noch eine Handlung oder Zustandsänderung dar, und müssen deshalb von Prozessen und Zustandsbeschreibungen abgegrenzt werden.

Ein erheblicher Teil der Anforderungen stellen nämlich Prozesse dar. Um mögliche Funktionen im Zielsystem erkennen zu können, ist es wichtig, diese Prozesse zu erfassen. Durch Prozessbeschreibungen werden nützliche Informationen angegeben, die bei der Identifikation der aufrufenden und aufgerufenen Klasse, sowie des Methodennamens, helfen können. Im Gegensatz zu Zustands- und Aggregationsbeschreibungen, beziehen sich Prozesse nicht auf die Zustände, sondern auf mögliche Änderungen bzw. Zustandsübergänge innerhalb des Systems. Prozesse selbst werden oft mittels Vollverben der Sätze ausgedrückt. Bei der Untersuchung von Anforderungen konnten zwei Prozesstypen identifiziert werden – „passive“ und „aktive“ Transformationen. Passive Transformationen sind Ereignisse, die zum Beispiel eine Zustandsänderung der Subjektentität beschreiben und kein direktes Objekt enthalten. Im Gegensatz dazu stellen aktive Transformationen Aktionsbeschreibungen dar, bei denen es immer ein Subjekt und ein Objekt gibt. Diese Unterscheidung gibt an, ob eine konkrete Entität die Funktion hervorruft oder nicht, und kann für die weitere Identifikation von Beziehungen zwischen Klassen im Zielsystem nützlich sein.

Um ein Verständnis für die Bedeutung einer Aussage zu bekommen, kann man die Satzstruktur und die Beziehungen, die durch Syntax und Lexik definiert werden, betrachten. Die Komponenten eines Satzes können als ein Array dargestellt werden, das aus einem Verb und einer Reihe von Substantivphrasen besteht [Fil68], die mit semantischen Rollen gekennzeichnet sind. Da in dieser Arbeit Sätze betrachtet werden, die atomar sind bzw. einen Sachverhalt isoliert beschreiben, können semantische Rollen hilfreich zur Erkennung von möglichen Kategorien bei der Satzklassifikation sein. Ein Verb wird durch die semantischen Rollen definiert, die es als Argument besitzt. Zum Beispiel besitzt das Verb „select“ immer ein Agens (Bezeichnung: A0) und ein Patiens (Bezeichnung: A1) als Argument, während das Verb „appear“ nur das Vorhandensein von einem Experiencer (von engl. *to experience* „erfahren, erleben, erleiden“) (Bezeichnung: A1) erfordert.

Beispiel 5.2: „select“

[The user $_{A0}$] [selects $_{V}$] [a configuration $_{A1}$].

Beispiel 5.3: „appear“

[The default configuration $_{A1}$] [appears $_{V}$] [first $_{AM-TMP}$].

Die semantische Rolle eines Arguments kann durch die lexikalische Eigenschaften, die ihm durch das Verb auferlegt werden, bestimmt werden. Laut dem Proto-Rollen-Ansatz von Dowty [Dow91] sind die sogenannten `Proto-Agent` und `Proto-Patient` die einzigen zwei thematischen Rollen, die für die Argumentrealisierung relevant sind. Diese Proto-Rollen erfassen wichtige Verallgemeinerungen über die Realisierung von Argumenten. Die `Proto-Agent`-Merkmale umfassen u. a. „freiwillige Beteiligung an einem Ereignis oder einem Zustand“ (engl. *„volitional involvement in the event or state“*) und „Verursachen eines Ereignisses oder einer Zustandsänderung bei einem anderen Teilnehmer“ (engl. *„causing an event or change of state in another participant“*), während `Proto-Patient` Merkmale wie „erfährt eine Zustandsänderung“ (engl. *„undergoes change of state“*) und „ist von einem anderen Teilnehmer kausal betroffen“ (engl. *„causally affected by another participant“*) beinhaltet. Darüber hinaus existiert ein `Proto-Agent` unabhängig von der Aktion des Verbs, im Gegensatz zum `Proto-Patient`. In Bezug auf die Entwicklung eines Programms kann hier eine Parallele zu möglichen Entitätsklassen gezogen werden. Ein `Proto-Agent` kann durch eine Klasse repräsentiert werden, die eine Methode aufruft und auch unabhängig von der jeweiligen Methode existiert. `Proto-Patient`-Klassen enthalten spezifische Methoden, die aufgerufen werden können, und Attribute (Zustände), die sich ändern lassen. Im ersten Beispiel stellt das Wort „*user*“ einen `Proto-Agent` und das Wort „*configuration*“ einen `Proto-Patient` dar, da der Nutzer eine Funktion auf die Konfigurationen anwendet. Im zweiten Beispiel wird der `Proto-Patient` durch das Wort „*configuration*“ repräsentiert und der Satz weist keinen `Proto-Agent` auf, da es keine Entität gibt, die unabhängig von der Aktion ist und diese hervorruft. Nicht alle Sätze erfordern ein Agens. Dies ermöglicht es, passive und aktive Transformationen zu erkennen. Sätze, die kein Agens enthalten und nur ein Patiens angeben, das durch die Funktion behandelt wird, beschreiben eine „passive“ Transformation, die hier auch ein Ereignis genannt wird. Die „aktiven“ Transformationen sind Aktionen, die, im Gegensatz zu Ereignissen, ein Agens enthalten, der die Funktion ausführt.

Ein weiterer wichtiger Faktor für die Semantik eines Satzes, der in Betracht gezogen werden muss, ist seine Polarität. Manchmal werden in funktionalen Anforderungen Satznegationen verwendet, um bestimmte Einschränkungen auszudrücken. Diese Anforderungen enthalten Wörter wie „*not*“, „*neither*“ und „*never*“ und bezeichnen einen negativen Zustand, die Unmöglichkeit oder das Verbot für die Ausführung einer Aktion oder für das Auftreten eines Ereignisses. Die Verneinung weist auf nicht vorkommende Situationen hin und hat einen starken Einfluss auf die Absicht des gesamten Satzes. Daher werden negierte Formen der verschiedenen Kategorien als eigene Klassen betrachtet. Negierte Anforderungsbeschreibungen beinhalten oft eine Nebenbedingung, bei deren Erfüllung die negative Einschränkung gültig ist.

Beispiel 5.4: Negation

The estimator shall not apply recycled parts to the collision estimate if no available parts are returned.

Der obere Satz bezeichnet den Abbruch einer Funktion „*apply recycled parts to the collision estimate*“, der in Abhängigkeit von einem Ereignis „*no available parts are returned*“ stattfindet. Diese Anforderungsbeschreibung besteht aus zwei Teilen – einem Hauptsatz und einem Nebensatz. Sie tragen unterschiedliche Semantik und sollen daher separat klassifiziert werden. Eine der Herausforderungen der Satzklassifikation sind nämlich die Ausdrücke, die Nebensätze enthalten. Ein Beispiel dafür sind Sätze, die einen Bedingungssatz, einen Hauptsatz und eventuell einen alternativen Hauptsatz enthalten. Bedingungssätze erkennt man an Schlüsselwörtern, wie *„if“*, *„then“* und *„else“*. Eine Bedingung kann entweder wahr oder falsch sein und wird meistens durch ein Ereignis oder eine Zustandsbeschreibung definiert. Bedingungen, die Aktionen beschreiben, treten bei Produktanforderungen selten auf, da sie typisch für die Anwendungsszenarien sind. Die Strukturerkennung von Bedingungssätzen wurde schon von Weigelt, Hey und Steurer in ihrer Arbeit „Detection of Conditionals in Spoken Utterances“ [WHS18] behandelt. Die betrachteten Schlüsselwörter werden in Tabelle 5.1 angegeben.

Tabelle 5.1: Schlüsselwörter der Bedingungssätze

Befehlstyp	Schlüsselwörter
Bedingung	if, when, supposing that, whenever, in case, in the case that, unless, on condition that, providing that, provided that, as long as, else if
Folgeanweisung	then, please, if so, you can, you have to, could you, would you
Alternativ-Anweisung	else, if not, otherwise, elseways, alternatively, instead, either, rather, oppositely

Obwohl hier Anweisungen betrachtet werden, kann das Verfahren auch zur Bedingungserkennung bei Anforderungsbeschreibungen hilfreich sein. Alle drei Befehlstypen *(Bedingung, Folgeanweisung, Alternativ-Anweisung)* können unterschiedliche semantische Merkmale *(Zustandsbeschreibung, Ereignis, Aktionsbeschreibung oder Aggregation)* tragen. Außer den oben genannten Schlüsselwörtern zur Erkennung von Bedingungen können Nebensätze durch weitere untergeordneten Konjunktionen, wie zum Beispiel *„after“*, *„although“*, *„as“*, *„because“*, *„before“*, *„unless“*, *„until“*, *„when“*, *„where“*, *„while“*, oder durch Relativpronomen, wie *„that“*, *„which“*, *„whichever“*, *„who“*, *„whoever“*, *„whom“*, *„whomever“*, *„whose“*, gebildet werden. Untergeordnete Klauseln bieten informative Unterstützung für das Hauptereignis des Satzes. Da sie selbst auch ein Prädikat enthalten und daher einen Prozess oder Zustand beschreiben, ist es wichtig, sie separat vom Hauptsatz zu klassifizieren.

Die Verwendung einer einheitlichen Sprache erleichtert die Identifizierung verschiedener Typen von Anforderungen. Die benutzten Modalverben, wie „*shall*“, „*should*“ und „*may*“, können zum Beispiel die Priorität einer Anforderung implizieren. Es existieren aber verschiedene Formen von textueller Darstellung der Anforderungen, wie Prosa in Reinform, Satzschablonen oder Vorlagen zur Strukturierung, wie die Formulierung von Anwendungsfällen. In dieser Arbeit werden nur Sätze in Reinform behandelt, ohne bestimmte Satzschablonen oder Vorlagen zu verwenden. Zudem wird zwischen vier grundlegenden Satz-

kategorien differenziert, die die Absichten der Anforderungsbeschreibungen erfassen – Zustandsbeschreibung (Abschnitt 5.2.1), Ereignis (Abschnitt 5.2.2), Aktionsbeschreibung (Abschnitt 5.2.3) und Aggregation (Abschnitt 5.2.4). Um ein besseres Verständnis für die gewählten Satzkategorien zu erlangen, werden sie im Folgenden näher spezifiziert und anhand verschiedener Beispiele erklärt.

5.2.1 Zustandsbeschreibung

Als erstes werden die Anforderungen betrachtet, deren Hauptziel es ist, einen Zustand des Subjekts zu bestimmen. Häufig kann das an dem Verb „*to be*“, gefolgt von einem Adjektiv, erkannt werden. Das Adjektiv stellt das Merkmal dar, das die Entität kennzeichnet.

Beispiel 5.5: Zustandsbeschreibung (1)

Students shall be eligible for the cohort program of study.

Dieses Beispiel beschreibt keine Operation, sondern einen Status oder Eigenschaft, die das Subjekt erfüllen muss. Hier wird der Zustand der Studenten durch das Adjektiv *„eligible“* und die Zusatzinformationen *„for the cohort program of study“*, die den Zustand näher spezifizieren, definiert. Da sich der Zustand auf das Subjekt des Satzes bezieht, ist er in Abhängigkeitsbäumen mit ihm durch eine `nsubj`-Bezeichnung direkt verbunden. Das Kopulaverb *„to be“* wird manchmal bei einer semantischen Rollenzuteilung nicht als ein selbständiges Prädikat identifiziert, da dadurch nur eine Eigenschaft deklariert und keine Transformation durchgeführt wird. Ein Zustand lässt sich außerdem nicht nur durch ein Adjektiv bestimmen. Im folgenden Beispiel wird der Zustand durch ein Substantiv definiert:

Beispiel 5.6: Zustandsbeschreibung (2)

Current students shall be members of the library.

Das Verb *„to be“*, gefolgt von einem Substantiv, kann entweder einen Zustand des Subjektes oder eine Generalisierung beschreiben. Da diese Unterscheidung aber vom Kontext, sowie von den Entwurfsentscheidungen, abhängig ist, wird sie in dieser Arbeit nicht betrachtet. Allerdings deklarieren beide Fälle einen Zustand, in dem sich das System (nicht) befinden soll. Wie bereits erwähnt wurde, kann eine Negation die Semantik des Satzes verändern. Im folgenden Beispiel wird die Negation eines existierenden Zustands angezeigt:

Beispiel 5.7: Negierte Zustandsbeschreibung

Former students shall not be members of the library.

5.2.2 Ereignis

Ereignisse sind Prozessbeschreibungen, die kein konkretes Agens beinhalten. Sie stellen eine passive Transformation dar. Es wird eine Funktion einer Entitätsklasse beschrieben, die nicht unbedingt von einer anderen Entitätsklasse aufgerufen wird. Der Satz enthält also kein Agens, sondern nur ein Patiens oder ein Theme. Eine passive Handlung wird oft durch das Hilfsverb „*to be*“ zusammen mit einem Verb im Partizip Perfekt ausgedrückt.

Beispiel 5.8: Ereignis (1)

The room equipment shall be displayed according to search parameters.

Aus dem Satz kann man die Schlussfolgerung ziehen, dass die Entitätsklasse „*room equipment*“ die Funktion „*display*“ besitzt. Die zusätzliche Information „*according to search parameters*“ beantwortet die Frage „Wie?“ und gibt eine nähere Spezifikation des Ereignisses an. Ein Ereignis kann außerdem nicht nur durch eine Kombination aus dem Verb „*be*“ und einem Verb im Partizip Perfekt ausgedrückt werden, sondern auch durch den Mangel an einer Entität, die den Prozess hervorruft. Ein Beispiel dafür ist:

Beispiel 5.9: Ereignis (2)

New students shall become members of the library.

Im oberen Satz wird das Subjekt durch das Substantiv „*students*“ repräsentiert. Studenten repräsentieren in diesem Fall die Entität, die eine konkrete Funktion aufweist. Es wird nicht explizit angegeben, ob eine andere Entität diese Funktion aufruft. Durch Anwendung von semantischer Rollenzuteilung kann festgestellt werden, dass diese Anforderungsbeschreibung kein Agens enthält und die Studenten das Patiens des Satzes repräsentieren. Im Gegensatz zu dem zweiten Beispiel für eine Zustandsbeschreibung, wird hier ein Zustandsübergang und nicht ein Zustand der Entitätsklasse beschrieben. Ereignisbeschreibungen können ebenfalls negiert werden. Im folgenden Beispiel wird eine mögliche passive Transformation („*be saved*“) beschrieben, die aber in Abhängigkeit von einer temporalen Einschränkung (das Ereignis „*after closing the application*“) nicht durchgeführt werden soll.

Beispiel 5.10: Negiertes Ereignis

The data shall not be saved after closing the application.

5.2.3 Aktionsbeschreibung

Aktionen beschreiben Operationen, die von einem Subjekt an einem Objekt ausgeführt werden. Aktionsbeschreibungen sind Sätze, die in Tätigkeitsform (*active voice*) geschrieben sind. Das Vorhandensein eines Subjekts und eines direkten Objekts kann zum Beispiel durch Anwendung von Abhängigkeitsbäumen erkannt werden, wobei das Subjekt durch eine `nsubj`- und das Objekt durch eine `dobj`-Beziehung mit dem Verb verbunden ist. Die beschriebene Funktion ist eine Fähigkeit oder Methode der Objektentität, die von der Subjektentität aufgerufen werden kann.

Beispiel 5.11: Aktionsbeschreibung

The adjuster shall be able to override the preferred repair facility on the estimate assignment.

Obwohl in dem Beispiel wieder das Verb „*to be*“ auftritt, definiert es hier nicht die Absicht des Satzes, da es nur ein Teil der Verbphrase „*shall be able to override*“ ist. Das Verb „*override*“ bezieht sich auf das Objekt, also auf die Reparatureinrichtung („*repair facility*“), die durch den Einsteller („*adjuster*“) überschrieben werden kann.

Ein Spezialfall der Aktionsbeschreibungen sind die Sätze, die ein Subjekt enthalten, das so abstrakt ist, dass es nicht als eine Entitätsklasse kategorisiert werden kann. Hierbei werden Sätze gemeint, die zum Beispiel mit „*The system shall*“ oder „*The product shall*“ anfangen und ausschließlich neben Modal- und Hilfsverben ein Verb in Normalform enthalten. Es wird keine Aussage darüber getroffen, welche Entität die Funktion aufrufen kann. In diesem Fall besteht beispielsweise die Möglichkeit, dass die Funktion durch einen `Observer` gesteuert wird. Ein Beispiel dafür ist die Aktualisierung von Daten.

Beispiel 5.12: Allgemeine Aktionsbeschreibung

The system shall update existing room equipment.

Um solche „abstrakten“ Aktionsbeschreibungen zu erkennen, kann man nach Schlüsselwörtern suchen, die das Subjekt des Satzes definieren. Eine vollständige und allgemeingültige Liste solcher Schlüsselwörter aufzustellen ist nicht einfach, da nicht von der Einhaltung eines Standards zum Schreiben von Anforderungen ausgegangen werden kann. Beispiele für solche Schlüsselwörter sind „*system*“, „*program*“, „*product*“, „*project*“, „*framework*“ und „*application*“. Außerdem kann das Agens auch der Eigenname des Programms sein. Auch wenn der Klassifikator einen Eigennamen an der Stelle des Subjektes erkennt, kann er nicht mit Sicherheit feststellen, dass es um das ganze System geht. Der Eigenname kann zum Beispiel eine externe Schnittstelle repräsentieren und in diesem Fall soll die Aktionsbeschreibung nicht als „allgemein“ bezeichnet werden. Aus diesem Grund gibt es hier zwei Möglichkeiten: die konkreten Eigennamen zu der Liste der Schlüsselwörter hinzuzufügen oder sie gar nicht als Identifikatoren für allgemeine Aktionsbeschreibungen zu verwenden.

Aus allen definierten Operationstypen tritt eine Negation am häufigsten bei Aktionsbeschreibungen auf. Negationen müssen immer berücksichtigt werden, um wichtige Einschränkungen des Systems festzustellen. Im folgenden Beispiel wird eine Aktion („*display a customer's password*“) beschrieben, die unter keinen Umständen auftreten soll.

Beispiel 5.13: Negierte Aktionsbeschreibung

The customer's web browser shall never display a customer's password.

5.2.4 Aggregation

Sätze, die einen Besitz ausdrücken, beschreiben in der Regel eine Aggregation. Üblicherweise wird das Verb „have“ für die Darstellung einer Teil-Ganzes-Beziehung verwendet. Der Satz soll immer ein Subjekt und ein oder mehrere Objekte enthalten.

Beispiel 5.14: Aggregation (1)

The menu shall have 3 submenus.

Das obere Beispiel stellt eine Relation zwischen dem Subjekt „*menu*“ und dem direkten Objekt „submenus“ dar. Daher enthält der Abhängigkeitsbaum des Satzes sowohl eine `nsubj`, als auch eine `dobj`-Abhängigkeit. Diese Struktur ist die gleiche wie bei Aktionsbeschreibungen. Hier wird aber nicht eine Funktion des direktes Objekts, sondern einen Besitz des Subjekts und eine Beziehung zwischen Entitäten beschrieben. So eine Beziehung kann auch durch die Verwendung anderer Verben definiert werden. Ein Beispiel dafür ist:

Beispiel 5.15: Aggregation (2)

The vehicle location shall include street, address, city, state, zipcode.

Hier besitzt das Subjekt „*vehicle location*“ Instanzen der Objekte „*street*“, „*address*“, „*city*“, „*state*“ und „*zipcode*“. Um eine Aggregation darzustellen, können außerdem Verben wie „*contain*“, „*possess*“ und „*own*“ benutzt werden. Zwischen einer Aggregation und einer Komposition wird in dieser Arbeit nicht unterschieden, da für ihre Differenzierung ein Verständnis des Kontextes nötig ist.

Eine negierte Aggregation stellt in der Regel eine Einschränkung dar. Dies wird durch das folgende Beispiel gezeigt.

Beispiel 5.16: Negierte Aggregation

The menu shall not have more than 3 submenus.

Jedoch treten negierte Aggregationen selten auf, da sie auch als positive Sätze formuliert werden können. Zum Beispiel kann der obere Satz ohne Negation so geschrieben werden kann: „*The menu shall have a maximum of 3 submenus.*“. Hier ist es wichtig zu beachten, dass die verschiedene Ausdrucksweise die Semantik zwar nicht ändert, aber Einfluss auf die Darstellung des Satzes haben kann. Im ersten Fall wird also der Satz als eine negative und im zweiten Fall - als eine positive Aggregation klassifiziert.

5.3 Klassifikationsverfahren

Die Lösungsstrategie des entwickelten Agenten lässt sich in zwei Teilaufgaben aufteilen - Erkennung von Satzkategorien (siehe Abschnitt 5.2) und Angabe der Polarität (positiv oder negativ). Der Grund dafür ist, dass negative Formen der Sätze im Allgemeinen selten auftreten. Bei manchen Ansätzen zur Klassifikation, wie die meisten Verfahren des maschinellen Lernen, werden große Datenmengen benötigt. Dabei ist für die richtige Vorhersage einer Kategorie notwendig, dass viele Beispiele für diese Kategorie in den Trainingsdaten vorkommen. Durch Aufteilung der Aufgaben wird erwartet, dass bessere Ergebnisse erzielt werden. Dieser Abschnitt befasst sich mit der Analyse von möglichen Vorgehensweisen und mit der Bestimmung von einem geeigneten Ansatz zur Lösung beider Teilprobleme. Da in dieser Arbeit atomare Sätze betrachtet werden, fällt ein Satz nur unter eine Kategorie nach Art und kann entweder positiv oder negativ sein. Aus diesem Grund ist bei der Festlegung der Satzkategorien eine multinomiale Klassifikation besser geeignet als eine Multi-Label-Klassifikation.

5.3.1 Analyse

Maschinelles Lernen und regelbasierte Systeme werden häufig verwendet, um Rückschlüsse auf Daten zu ziehen. Beide Ansätze haben ihre Stärken und Schwächen und daher ist es wichtig, die Unterscheidung zwischen ihnen zu verstehen und die Stellen auszuwerten, an denen sie für den Lösungsentwurf dieser Arbeit wertvoll sein können.

5.3.1.1 Regelbasiertes Verfahren

Bei regelbasierten Systemen wird ein Algorithmus durch Anwendung von expliziten Regeln und einer Faktenbasis, die das Wissen zu einem Problemkreis repräsentiert, erstellt. Diese

Regeln sind einfach zu implementieren, wenn die konkreten Schritte zum Lösen des Problems bekannt sind und der Entwickler alle Ausnahmen berücksichtigen kann. In Bezug auf die Aufgabenstellung dieser Arbeit muss erst festgestellt werden, welche Angaben über die Sätze ermittelt und verwendet werden können. Um Informationen über die Satzstruktur und die Beziehungen zwischen Entitäten zu gewinnen, können Wortart-Markierungen, Phrasenerkennung, semantische Rollenzuteilung und Abhängigkeitsbäume verwendet werden.

Wie bereits erwähnt, können semantische Rollen hilfreich bei der Erkennung mancher Satzkategorien sein. Das Vorhandensein eines Agens impliziert z.B. eine Aktionsbeschreibung, während das Fehlen eines Agens ein Ereignis angeben kann. Laut der Arbeit von Gelhausen [Gel10] können sogar Aggregationsbeziehungen an den thematischen Rollen `OMN` und `PARS` erkannt werden. Die Verwendung von semantischer Rollenzuteilung in einem heuristischen Ansatz kann aber auch Nachteile mit sich bringen. Die Markierungen hängen vom Rahmen eines Verbs ab und wenn ein Satz mehrere Verben enthält, können manche Wörter mehrere Rollen haben. Außerdem ist die semantische Rollenzuteilung eine komplexe NLP-Aufgabe, die nur bis zu einem gewissen Grad korrekte Ergebnisse liefert. Zum Beispiel ist „SENNA“ (*Semantic/syntactic Extraction using a Neural Network Architecture*) [CWB$^+$11] ein System, das beim Vorhersagen von semantischen Rollen ein F-Maß von 75,49% erreicht. Die Leistung des entwickelten Werkzeugs zur Satzklassifikation würde bei der Verwendung von Informationen über semantischen Rollen stark von der Genauigkeit ihrer Erkennung abhängig sein.

Verschiedene Abhängigkeitszerleger [CMJM10] erzielen ein bis zu 13,6% höheres F-Maß als „SENNA“ und sind geeignet, um Kategorien wie Aktionsbeschreibung, Ereignis und Zustandsbeschreibung zu erkennen. Aktionsbeschreibungen enthalten immer `nsubj` und `dobj` Abhängigkeiten, was das Vorhandensein eines Proto-Agens und eines Proto-Patiens impliziert. Abhängigkeitsbäume von Erreignisbeschreibungen beinhalten dagegen nur eine `nsubj`-Markierung, die das Proto-Patiens in der Rolle des Subjektes bezeichnet. Diese `nsubj`-Abhängigkeit stammt in diesem Fall aus dem Vollverb und verbindet es mit dem Subjekt des Satzes. Sie ist bei Zustandsbeschreibungen auch vorhanden, wo sie aber nicht das Verb, sondern den Zustand (Adjektiv oder Substantiv) mit der Subjektentität verbindet. Zusätzlich ist der Zustand auch mit einem Kopulaverb wie *„be“* (Bezeichnung: `cop`) verbunden. Informationen über Wortart-Markierungen und Abhängigkeitsgraphen sind aber nicht genug, um Aggregationen zu erkennen. Die Struktur der Sätze, die Aggregationen beschreiben, ist dieselbe wie die von Aktionsbeschreibungen. Um Vollverben wie *„have“*, *„own“* und ihre Synonyme zu erkennen, ist eine lexikalische Datenbank wie WordNet notwendig.

Abgesehen von verschiedenen Satzkategorien sind auch Negationen in Anforderungsbeschreibungen zu beachten. Eine Negation kann man entweder durch das Vorhandensein von Wörtern wie *„not“* und *„never“* erkennen, oder durch Verwendung von Abhängigkeitsbäumen – die Markierung `neg` bezeichnet genau die Negation-Modifikatoren.

Regelbasierte Systemen sind deterministisch und für ihre Erstellung ist häufig mehr Arbeitsaufwand notwendig als für maschinelle Lernsystemen. Das Fehlen der richtigen Regeln kann die Ergebnisse negativ beeinflussen. Wenn bestimmte Regeln vergessen wurden oder nicht alle Ausnahmen gefunden und berücksichtigt werden, ist das entwickelte System nicht optimal, da es möglicherweise nicht alle Konstrukte erkennen kann. Eine Herausforderung besteht darin, dass so ein deterministisches System bestimmte Satzstrukturen erwartet und neue Eingabedaten davon abweichen können.

5.3.1.2 Maschinelles Lernen

Maschinelles Lernen ist ein alternativer Ansatz, mit dem einige Schwächen von regelbasierten Methoden angegangen werden können. Im Gegensatz zu regelbasierten Methoden ist maschinelles Lernen probabilistisch und verwendet statistische Modelle anstatt deterministischer Regeln. Anstatt zu versuchen, den ganzen Entscheidungsprozess vollständig zu implementieren, wird der Fokus auf die Endergebnisse gelegt. Für ein maschinelles Lernverfahren ist es nicht wichtig, wie der Experte bei der manuellen Markierung eine Entscheidung trifft, sondern was die Entscheidung ist.

Datensatz

Maschinelle Lernmodelle zur Satzklassifikation müssen auf von Menschen gekennzeichnete Sätze trainiert werden, um die Markierungen neuer Daten vorhersagen zu können. Der Mangel an ausreichenden Trainingsdaten kann daher ein Problem sein und die Leistung des entwickelten Verfahrens einschränken. Es wurde kein Datensatz mit funktionalen Anforderungen gefunden, der groß genug und für diese Arbeit geeignet ist. Aus diesem Grund muss bei der Anwendung von maschinellem Lernen zuerst ein Datensatz aus verschiedenen Quellen, die frei zur Verfügung stehen, aufgebaut und beschriftet werden. Außerdem wird durch die Zusammensetzung verschiedener Datensätze eine Vielfalt an Anforderungen erreicht.

Vorverarbeitung

Vor dem Trainieren eines Vorhersagemodells müssen die Daten von einem Rohzustand zu einem für die Modellierung geeigneten Zustand umgewandelt werden. Häufig verwendete Vorverarbeitungstechniken umfassen Umwandlung des Textes in Kleinbuchstaben, Entfernung von nicht aussagekräftiger Zeichensetzung, Tokenisierung, Lemmatisierung, Stammformreduktion (engl. *stemming*) und Entfernung von Stoppwörtern. Stoppwörter werden oft als Vorverarbeitungsschritt bei Stimmungserkennung verwendet, da Wörter, wie „*I“, „this“, „there“, „be“, bei solchen Aufgaben nicht relevant sind. Durch Verwendung von Stoppwörtern wird aber die Satzstruktur geändert, wobei wichtige Informationen verloren gehen können. Daher ist diese Vorverarbeitungsmethode für die Bestimmung der semantischen Funktion von Sätzen nicht geeignet. Die Anwendung von Lemmatisierung und Stammformreduktion ist hier auch nicht empfehlenswert, da die Form der Wörter und die Grammatik der Sätze behalten werden soll.

Vektorrepräsentation

Nachdem die Eingabedaten normalisiert und tokenisiert wurden, müssen sie in numerische Werte konvertiert werden. Mittels Vektorisierung lassen sich so Zusammenhänge zwischen Wörter beschreiben und neue Erkenntnisse gewinnen. Dieser Schritt ist erforderlich für die Anwendung der Klassifikatoren. Um die Informationen als Punkte in einem hochdimensionalen, metrischen Vektorraum zu repräsentieren, können verschiedene Textdarstellungsmodelle wie Bag-of-Words, Bag-Of-N-Grams, Tf-Idf (*Term Frequency - Inverse Document Frequency*), Worteinbettungen oder Satzeinbettungen verwendet werden. Jedes dieser Modelle gibt lediglich eine numerische Vektordarstellung der Textdaten wieder. Der resultierende Vektor fungiert als der vom Klassifikationsverfahren verwendete Merkmalsvektor und bestimmt, wie viele Merkmale vorhanden sind.

Eine häufig verwendete Technik zur Darstellung der Textdaten als Vektoren mit fester Länge ist das Bag-Of-Words-Modell, das errechnet, wie oft ein Wort im Dokument auftritt. Wegen seiner Einfachheit, Effizienz und oft überraschender Genauigkeit ist es ein beliebtes Verfahren bei manchen Klassifikationsproblemen wie Klassifikation von Dokumenten und Themen Modellierung. Allerdings hat Bag-Of-Words (BoW) zwei bedeutende Nachteile:

die syntaktische Reihenfolge der Wörter geht verloren und ihre Semantik wird ignoriert. Bag-Of-N-Grams betrachtet Wortphrasen der Länge N, um die Texteingaben als Vektoren fester Länge darzustellen. Dabei wird die lokale Reihenfolge der Wörter erfasst und es gehen weniger Informationen als bei BoW verloren. Obwohl das Bag-of-N-Grams-Modell die Wortreihenfolge in einem kurzen Kontext betrachtet, leidet es unter Datenknappheit und hoher Dimensionalität. Da Bag-Of-Words und Bag-Of-N-Grams nicht in der Lage sind, die Semantik und die Abstände zwischen Wörtern zu erfassen, sind sie für komplexe Textklassifizierungsproblemen nicht geeignet.

Tf-Idf ist ein Vektorraum-Retrieval-Verfahren (engl. *Vector Space Model*), mit dem die Wörter gewichtet werden. Im Gegensatz zu BoW misst der Tf-Idf-Modell ihre Relevanz und nicht nur ihre Häufigkeit, wobei Wert auf die im Korpus selten vorkommenden Wörter gelegt wird. Jedoch basiert Tf-Idf auf dem BoW-Modell und erfasst daher nicht die Position im Text, die Semantik oder das gemeinsame Auftreten in verschiedenen Dokumenten.

Um semantisch ähnliche Wörter und Zusammenhänge zwischen Wörtern zu erkennen, können Worteinbettungen verwendet werden. Sie kommen oft zum Beispiel bei Verfahren zur Stimmungsanalyse zum Einsatz, um die Assoziation eines Wortes mit anderen Wörtern herzustellen oder um Dokumente zu gruppieren und nach Themen zu klassifizieren. Hierfür ist `Word2Vec` eine bekannte Technik, mit der ein großer Textkorpus aufgenommen wird und für jedes Wort ein Vektorraum (in der Regel mit Hunderten von Dimensionen) erzeugt wird. Für die Implementierung von `Word2Vec` stehen zwei Varianten zur Auswahl: Continuous Bag-Of-Words (CBOW) oder Continuous Skip-Gram (SG) [MCCD13]. Der Unterschied besteht darin, dass CBOW versucht, die Ausgabe (das Zielwort) von seinen Nachbarwörtern (den Kontextwörtern) zu erraten, während SG die Kontextwörter von einem Zielwort errät. Laut Mikolov[1] funktioniert SG gut mit wenigen Trainingsdaten und repräsentiert auch selten auftretende Wörter und Phrasen, während CBOW um ein Vielfaches schneller zu trainieren ist und etwas bessere Genauigkeit für häufig auftretende Wörter erzielt. Damit genaue Schätzungen über die Bedeutung eines Wortes gemacht werden können, ist eine ausreichend große Datenmenge zum Trainieren der Worteinbettungen notwendig. Wenn der zur Verfügung stehende Datensatz klein ist, ist es keine gute Idee, ihn zum Trainieren von Worteinbettungen zu verwenden. Aufgrund der begrenzten Anzahl an Wörtern können die Kodierungen von Kontext und Vokabeln nicht ordnungsgemäß gelernt werden und die Einbettungsmethode kann beim Vorhersagen der Kategorien von ungesehenen Testdaten keinen großen Einfluss haben. Beliebte Methoden, um damit umzugehen, sind ein `Word2Vec`-Modell auf große Datenmengen wie Wikipedia zu trainieren und ihn bei der Verarbeitung des eigenen Datensatzes einzusetzen oder ein bereits vortrainierten Modell zu verwenden. Der Nachteil bei Verwendung von vortrainierten Modellen ist, dass diese Modelle nur Wörter aus ihrem Korpus kennen und alle Wörter, die beim Training nicht vorhanden waren, aber im eigenen Datensatz vorkommen, als einen Null-Vektor repräsentiert werden.

Eine alternative Möglichkeit, die Eingabedaten zu repräsentieren, ist die Anwendung von Satzeinbettungen. `Doc2Vec` von Gensim[2] modifiziert den `Word2Vec`-Algorithmus so, dass fortlaufende Darstellungen für größere Textblöcke wie Sätze, Absätze oder ganze Dokumente unüberwacht gelernt werden. Im Vergleich zu anderen Modellen wie Tf-Idf, die nur Frequenz-basiert sind, ist `Doc2Vec` deutlich besser zum Verständnis der kontextuellen Bedeutung von Text geeignet. Vektoren werden durch Trainieren eines neuronalen Netzwerks gegen ein Nachbarschaftswort eines zufällig abgetasteten Wortes und umgekehrt erzeugt. Die erlernten Gewichte der verborgenen Ebene werden als `Doc2Vec`-Werte verwendet. Hier

[1] Quelle: https://groups.google.com/forum/#!searchin/word2vec-toolkit/c-bow/word2vec-toolkit/NLvYXU99cAM/E5ld8LcDxlAJ, zuletzt besucht am 06.10.2019

[2] Quelle: https://radimrehurek.com/gensim/models/doc2vec.html, zuletzt besucht am 06.10.2019

gibt es wieder zwei möglichen Techniken - Distributed Memory (DM) und Distributed Bag-of-Words (DBOW). Dabei ähnelt DM stark dem CBOW-Modell von `Word2Vec`, während DBOW auf dem SG-Modell basiert. `Doc2Vec` profitiert von vielen variierenden domänenspezifischen Daten und daher muss ein passender Trainingsdatensatz gefunden werden, der relevant für die Aufgabenstellung und groß genug ist.

Klassifikator

Für das Klassifikationsproblem können verschiedene Lernverfahren für überwachtes maschinelles Lernen verfolgt werden. Um besser abschätzen zu können, welcher Ansatz für die Satzklassifikation am besten geeignet ist, sollen einige Modelle erstellt und miteinander verglichen werden. Dabei können bestimmte Hyperparameter der Lernmodelle gewählt werden, um bessere Genauigkeit bei der Vorhersage von unbekannten Daten zu erreichen. Durch Anwendung von Kreuzvalidierung wird versucht, eine optimale Leistung der Modelle zu erzielen. Die Ergebnisse werden anhand von Metriken wie Genauigkeit, Präzision, Ausbeute und F-Maß evaluiert und miteinander verglichen. Die klassischen Lernverfahren umfassen u. a. Naïve Bayes, Entscheidungsbäume, Random Forest, logistische Regression und Stützvektormaschine. Da es keine universellen Algorithmen für bestimmte Optimierungsprobleme gibt, werden verschiedene Verfahren betrachtet. Ihre Leistung hängt von der Größe des Datensatzes, sowie von den Modellparametern ab.

Der Naïve Bayes-Klassifikator (siehe Abschnitt 2.4.1) ist vielleicht der einfachste Klassifikator für maschinelles Lernen. Bei Klassifikationsproblemen mit kleinen Trainingssätzen können bessere Ergebnisse erzielt werden als bei anderen Klassifikatoren, da die Neigung zur Überanpassung gering ist. Aufgrund der linearen Skalierung, des schnellen Trainierens der Daten, des geringen Speicherbedarfs und der geringen CPU-Auslastung ist Naïve Bayes eine praktikable Lösung für einfache Klassifikationsprobleme.

Entscheidungsbäume (siehe Abschnitt 2.4.2) werden häufig bei Klassifikationsaufgaben verwendet, da sie einfach zu interpretieren sind, kategorische Merkmale handhaben, keine Merkmalsskalierung erfordern und Nichtlinearitäten und Merkmalsinteraktionen erfassen können. Dabei lässt sich die maximale Tiefe eines Baums festlegen. Obwohl tiefere Entscheidungbäume manchmal eine höhere Genauigkeit ermöglichen, können sie zu Überanpassung führen. Daher sind bei kleinen Datensätzen kürzere Entscheidungsbäume zu bevorzugen. Random Forests (siehe Abschnitt 2.4.3) erweitern das Verfahren von Entscheidungsbäumen, wobei viele Entscheidungsbäume parallel trainiert und dann kombiniert werden. Aufgrund der Varianzreduzierung durch Mittelwertbildung mehrerer Bäume im Entscheidungswald wird das Risiko einer Überanpassung verringert.

Logistische Regression (siehe Abschnitt 2.4.4) wird oft für Klassifikationsprobleme eingesetzt. Sie ist ein Sonderfall von verallgemeinerten linearen Modellen, die die Wahrscheinlichkeit der Ergebnisse vorhersagen. Multinomiale logistische Regression ist eine Klassifizierungsmethode, die die logistische Regression auf Mehrklassenprobleme verallgemeinert. Sie ist eine weit verbreitete Technik, die sehr effizient und leicht interpretierbar ist, und nicht zu viele Rechenressourcen erfordert.

Der Stützvektormaschine-Klassifikator (siehe Abschnitt 2.4.5) hat sich bei der Klassifizierung von hochdimensionalen Daten als gut geeignet erwiesen [RMD18] [KM17] [KMA06]. SVM basiert auf der Idee, eine Hyperebene zu finden, die die Merkmale am besten in verschiedene Domänen unterteilt. Dieser Klassifikator ist einer der beliebtesten Algorithmen für maschinelles Lernen, da er häufig die besten Ergebnisse liefert, wenn Klassen sich nicht überlappen. Er wird für kleinere Datenmengen verwendet, da die Verarbeitung großer Datensätze zu lange dauert.

Obwohl regelbasierte Systeme manchmal schnelle Lösungen bieten können, kann maschinelles Lernen wegen der Komplexität der natürlichen Sprache und des damit verbundenen

Aufwands für die Erstellung der Regeln langfristig eine bessere Wahl für die Erkennung der Satzkategorien sein. Durch Datenaufbereitung, Algorithmenauswahl und Optimierung von Parametern kann das maschinelle Lernen flexibler als heuristische Ansätze sein und ist außerdem zugänglicher für kontinuierliche Anpassungen und Verbesserungen. Aus diesen Gründen wird ein maschinelles Lernverfahren als Vorgehensweise für die Satzklassifikation in dieser Arbeit gewählt.

5.3.2 Entwurf

In diesem Abschnitt wird auf Basis der Analyse ein Gesamtkonzept für die Lösung der Teilaufgaben des Zielsystems entworfen, das als Grundlage für die anschließende Implementierung dienen soll. Hierfür wird eine Kombination aus maschinellem Lernverfahren für die Vorhersage der Satzkategorien und einem heuristischen Ansatz zur Bestimmung der Polarität der Sätze angewendet. Für die Negationserkennung wird kein maschinelles Lernverfahren eingesetzt, weil INDIRECT Informationen über Abhängigkeitsbäume bereitstellen kann und es ausreicht, für jeden Satz zu prüfen, ob der Negationsmodifikator im Graphen vorhanden ist. In Abbildung 5.1 ist der Ablauf der einzelnen Schritte, die vom Agenten ausgeführt werden, dargestellt.

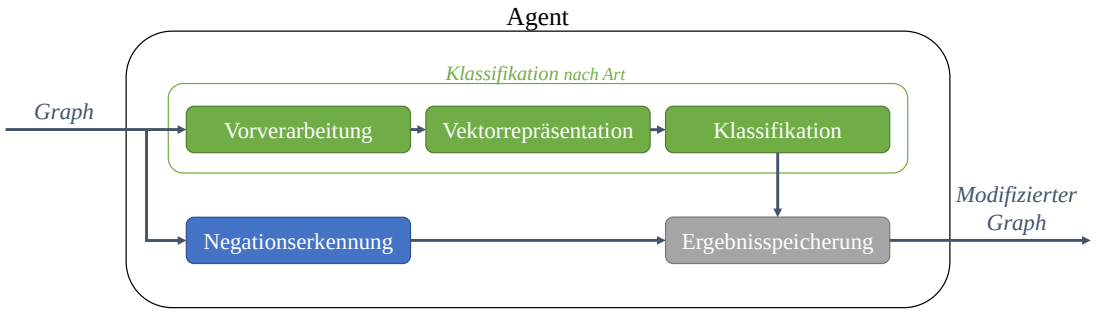

Abbildung 5.1: Ansatz zur Klassifikation der Nutzereingabe

Zu Beginn wird die Klassifikation nach Art der Sätze betrachtet. In Abschnitt 5.3.2.1 wird ein Lösungsansatz für den gesamten Ablauf dieses Teilproblems angegeben. Danach wird in Abschnitt 5.3.2.2 die Vorgehensweise zur Erkennung von Negationen in Anforderungsbeschreibungen eingeführt. Der Agent soll die ausgewählten Teillösungen in einem Algorithmus zusammenfassen und nach der Ausführung die Ergebnisse im System speichern. Die Eingabedaten werden durch das Projekt INDIRECT (siehe Kapitel 3) in Form eines Graphen repräsentiert. Die verschiedenen Agenten können auf diesen Graphen zugreifen und ihn mit weiteren Informationen anreichern. Die Modifikation des Graphen durch den in dieser Arbeit entwickelten Agenten wird in Abschnitt 5.3.2.3 vorgestellt.

5.3.2.1 Satzkategorien

Damit der bestmögliche Klassifikator zur Erkennung der Satzkategorien ermittelt werden kann, werden Experimente mit vier Klassifikatoren durchgeführt - Naïve Bayes, Random Forest, Stützvektormaschine und logistische Regression. Dabei werden verschiedene Kombinationen aus Strategien zur Datenrepräsentation und multinomialen Lernverfahren untersucht. Nachdem alle Ergebnisse evaluiert und verglichen wurden, wird entschieden, welcher Klassifikator für die Lösung der Aufgabenstellung verwendet wird. Das Modell,

das die beste Leistung erbringt, wird dann auf dem gesamten Datensatz trainiert und für weitere Nutzung gespeichert.

Nachdem ein bestimmtes Verfahren gewählt wird, soll für die Umsetzung der Klassifikation ein Agent im Projekt INDIRECT erzeugt werden. Als Eingabe bekommt er einen Graphen, in dem jedes Wort durch einen Knoten und der Wortfluss durch die Kanten repräsentiert wird. Zusätzlich sind in den Wortknoten Informationen über die Satznummer enthalten. Unter Berücksichtigung der Satznummer wird jeweils eine Wortliste für jeden Satz erstellt. Diese Wortlisten werden in einer Datei gespeichert und dienen als Eingabe für das entwickelte Verfahren zur Klassifikation nach Art. Die Vorverarbeitung der Sätze beginnt mit der Entfernung von allen Satzzeichen. Anschließend werden eine Umwandlung der Texte in Kleinbuchstaben und eine Tokenisierung der Sätze durchgeführt. Danach werden die Daten in eine Vektorrepräsentation fester Länge überführt und an den Klassifikator übergeben, der dann die wahrscheinlichste Kategorie für jeden Satz vorhersagt. Schließlich können die neu gewonnenen Informationen nach Ausführung beider Teillösungen im Graphen gespeichert werden.

5.3.2.2 Negation

Um Negationen zu erkennen, ist es ausreichend, den Abhängigkeitsgraphen (siehe Abschnitt 2.1.7) eines Satzes und die dadurch bereitgestellten Beziehungen zu betrachten. Der Negationsmodifikator (`neg`) ist die Beziehung zwischen einem Negationswort und dem Wort, das es modifiziert. Er wird sowohl für die Negation von Prädikaten (siehe Abbildung 5.2), als auch für die nominelle Negation (siehe Abbildung 5.3) verwendet.

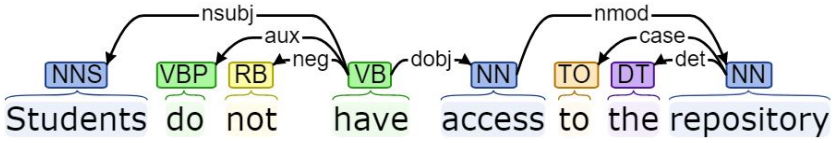

Abbildung 5.2: Negation von Prädikat; CoreNLP - Basic Dependencies[3]

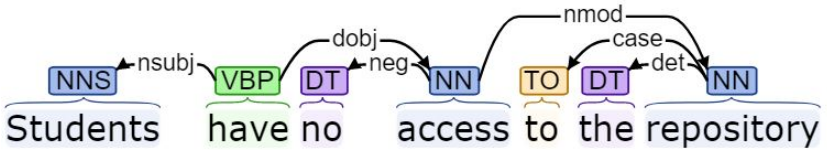

Abbildung 5.3: Nominelle Negation; CoreNLP - Basic Dependencies

Im Rahmen des INDIRECT-Projekts werden die Eingabedaten auf eine logische Graphenrepräsentation abgebildet. Der Agent `DepParse` modifiziert diesen Graphen durch Anreicherung mit Informationen über Abhängigkeitsbäume. Die durch diesen Agent bereitgestellte Datenstruktur kann also verwendet werden, um festzustellen, ob die Eingabesätze Negationen enthalten. Hierfür wird vorausgesetzt, dass der Agent `DepParse` vorab ausgeführt wird. Dann soll geprüft werden, ob eine `neg`-Abhängigkeit aus dem Graphen extrahiert werden kann. Dadurch wird entschieden, ob der Satz als positiv oder negativ markiert werden soll.

5.3.2.3 Ergebnisspeicherung

Nachdem die Klassifikation durchgeführt wurde, werden die Ergebnisse im Graphen gespeichert. Es gibt verschiedene Möglichkeiten, den Graphen zu modifizieren. Man kann

[3]Quelle: https://corenlp.run/, zuletzt besucht am 06.10.2019

entweder neue Knoten oder Kanten einfügen, oder neue Attribute zu den bestehenden Knoten oder Kanten hinzufügen. Im Folgenden werden zwei alternative Darstellungen des Graphen angegeben.

Die erste Möglichkeit besteht darin, die Informationen über die Semantik in den Wortknoten zu speichern. Hierfür wird ein neues Attribut (*„semantics“*) zu den Token-Knoten hinzugefügt. Diese Repräsentation wird anhand eines Beispiels in Abbildung 5.4 angegeben. Dabei wird das neue Attribut in blau dargestellt und enthält den gleichen Wert in allen Wortknoten eines Satzes. Die Polarität (*„positive“* oder *„negative“*) und die Art (*„action“*, *„state“*, *„occurrence“* oder *„aggregation“*) der Sätze werden in einem String zusammengestellt und im Feld *„semantics“* gespeichert. Dieser Sachverhalt erzielt einen schnellen Zugriff auf die Semantik bei Untersuchung der Wortknoten.

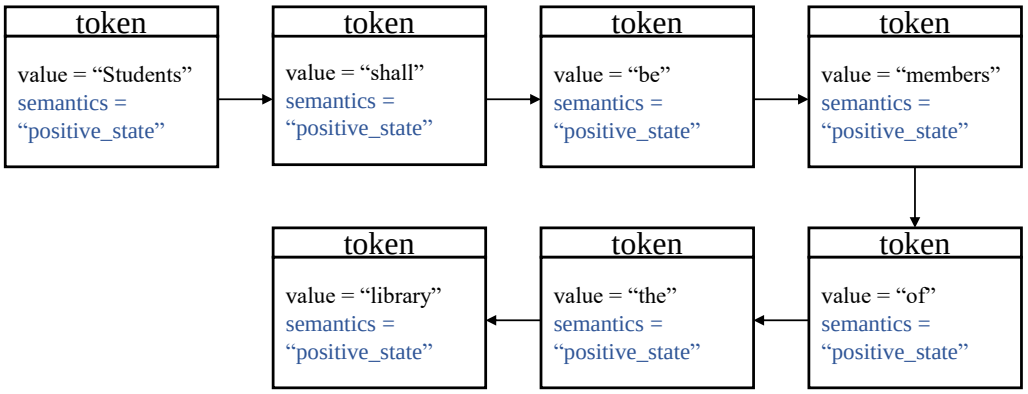

Abbildung 5.4: Modifikation des Graphen durch Hinzufügen von neuen Attributen zu den bestehenden Knoten

Die zweite Möglichkeit wäre es, für jeden Satz im Graphen einen neuen Knoten einzufügen. Hierzu wird ein neuer Knotentyp (*„semantics“*) definiert, der zuständig für die Speicherung der Informationen über die Art und Polarität der Sätze ist. Der so modifizierte Graph wird in Abbildung 5.5 illustriert. Der zusätzliche Knoten ist mit allen Wortknoten eines konkreten Satzes verbunden und gibt im Feld *„value“* die Semantik an. Diese Struktur ermöglicht eine schnelle Lieferung der Ergebnisse bei einer Abfrage an die Semantik der Sätze. Dabei wird geprüft, ob der entsprechende Knotentyp im Graphen enthalten ist, und der Attributwert direkt ausgelesen werden kann. Da hier die Semantik nicht einfach als Information der einzelnen Knoten, sondern als Merkmal des ganzen Satzes gespeichert wird, ist diese Modifikation ein sinnvoller Ansatz für die Repräsentation der Ergebnisse.

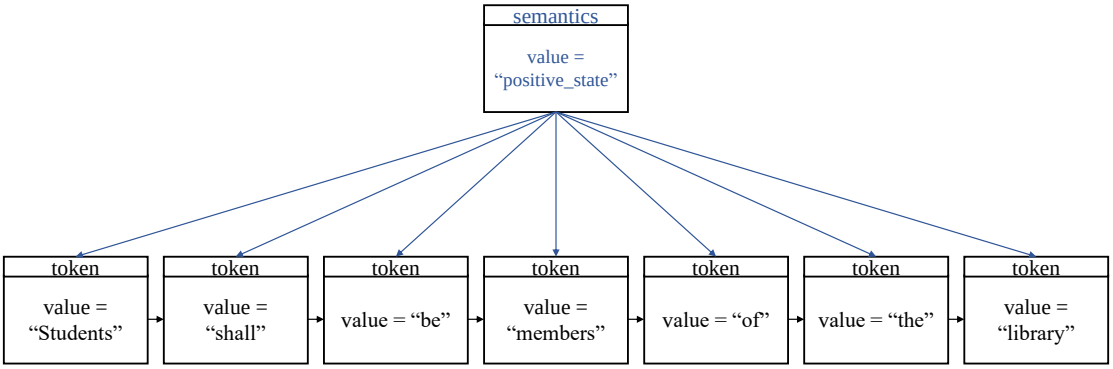

Abbildung 5.5: Modifikation des Graphen durch Einfügen von neuen Knoten

6 Implementierung

In diesem Kapitel wird die Realisierung der im Kapitel 5 betrachteten Verfahren beschrieben. Zunächst wird in Abschnitt 6.1 ein Überblick über den verwendeten Datensatz gegeben. Dann werden im Abschnitt 6.2 die entworfenen Klassifikatoren und ihre inneren Komponenten dargelegt. Abschließend wird auf die Integration der Klassifikatoren und den allgemeinen Programmablauf eingegangen.

6.1 Datensatz

Für die Durchführung von Experimenten mit überwachten Modellen wird ein annotierter Datensatz benötigt, der die Zuordnung von Anforderungsbeschreibungen zu ihren entsprechenden Kategorien enthält. Der Datensatz wird durch die Kombination und Beschriftung von Anforderungsbeschreibungen aus unterschiedlichen Quellen erstellt. Die ausgesuchten Quellen stehen frei zur Verfügung und enthalten funktionale Anforderungen für Zielsysteme in verschiedenen Bereichen. Der gesamte Korpus besteht aus 700 annotierten Klauseln, wobei 202 davon aus dem *NFR*-Datensatz [CHMLP07], 98 aus dem *Software Requirement Risk Prediction*-Datensatz [sNZ18], 230 aus dem *iTrust*-, 100 aus dem *IceBreaker*- und 70 aus dem *WARC*-Datensatz von CoEST[1] stammen. CoEST ist eine Forschungsgesellschaft, die sich in verschiedenen Projekten im Bereich Rückverfolgbarkeit von Anforderungen engagiert. Sie stellt nützliche Ressourcen bereit, damit Wissenschaftler die Herausforderungen der Rückverfolgbarkeit angehen können. Da sich INDIRECT mit Rückverfolgbarkeit von Anforderungen beschäftigt, sind die von CoEST angebotenen Datensätze auch für diese Bachelorarbeit relevant.

Sätze, die Konjunktionen oder Bedingungen enthalten und aus mehreren Klauseln bestehen, werden aufgeteilt, um Atomarität zu erreichen. Im folgenden Beispiel 6.1 wird der erste Teil des Satzes als eine Zustandsbeschreibung (*state*), und der zweite Teil - als eine Aktionsbeschreibung (*action*) annotiert.

Beispiel 6.1: Anforderung aus dem iTrust-Datensatz

[If answer to security question is correct, $_{\text{state}}$] [the user can change their password. $_{\text{action}}$]

[1] Quelle: http://www.coest.org/, zuletzt besucht am 06.10.2019

Die Anforderungsbeschreibungen werden in die vier in der Analyse definierten Satzkategorien eingeordnet. In Abbildung 6.1 wird eine Übersicht über die Aufteilung nach Polarität und Art der Sätze in Form von einem Balkendiagramm angegeben. Dabei werden Aktionsbeschreibungen als *action*, Ereignisse als *occurrence*, Zustandsbeschreibungen als *state* und Aggregationen als *aggregation* bezeichnet. In rot wird die Anzahl der negativen Klauseln dargestellt, während die blaue Farbe den Anteil der positiven Sätze bezeichnet. In blau wird die Anzahl der positiven und in rot die Anzahl der negativen Klauseln dargestellt. Der Datensatz beinhaltet insgesamt 54 negative Beschreibungen.

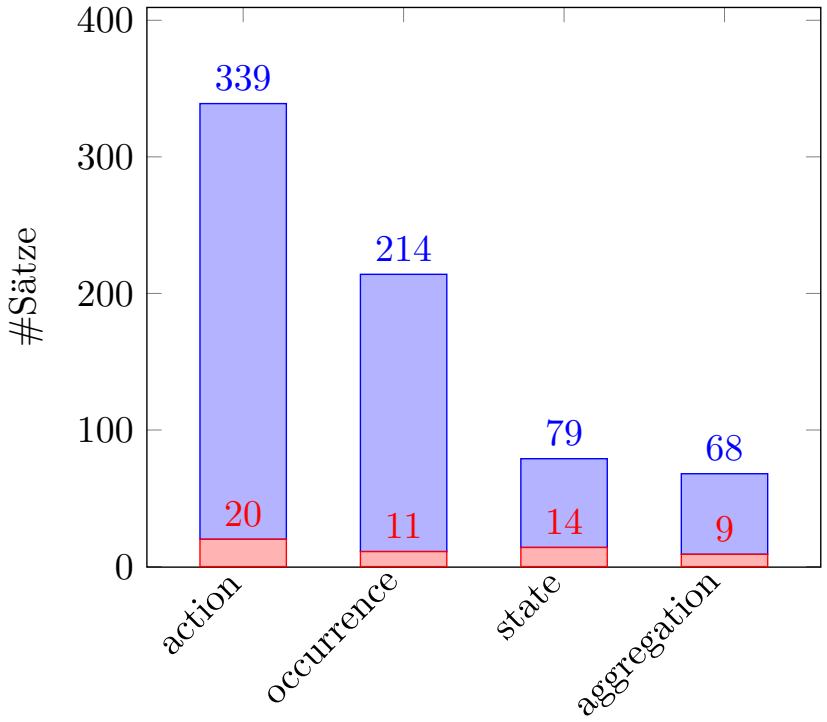

Abbildung 6.1: Aufteilung nach Kategorien: positive (in blau) und negative (in rot) Aktionsbeschreibungen (*action*), Ereignisse (*occurrence*), Zustandsbeschreibungen (*state*), Aggregationen (*aggregation*)

6.2 Klassifikation

In diesem Abschnitt werden die verwendete Programmierumgebung, sowie die verschiedenen Bibliotheken und Modulen für Datenverarbeitung, Klassifikation und Evaluierung vorgestellt.

6.2.1 Klassifikation nach Art der Sätze

Für die Implementierung der multinomialen Klassifikation wird ein Skript in Python (Version 3.7) geschrieben und bereitgestellt. Das erfolgt durch die Einsetzung eines Modells, das auf dem gesamten Datensatz vorab trainiert wurde. Das Skript, das in INDIRECT verwendet wird, soll für die Vorverarbeitung und Repräsentation der Nutzereingabe, sowie für die Vorhersage der entsprechenden Kategorien der Sätze durch das vortrainierte Modell zuständig sein. Der allgemeine Ablauf wird in Abbildung 6.2 dargestellt.

Die Nutzereingabe wird von dem INDIRECT-Agenten als eine CSV-Datei zur Verfügung gestellt, wobei jeder Satz auf eine neue Zeile verteilt wird. Anschließend wird das Skript ausgeführt. Es führt alle notwendigen Schritte der Klassifikation durch – Vorverarbeitung,

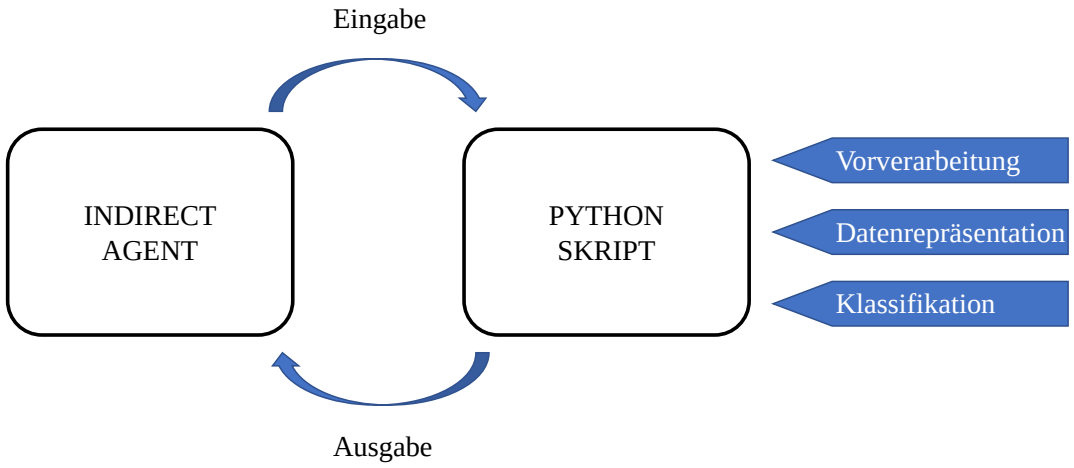

Abbildung 6.2: Ansatz zur Klassifikation nach Art der Sätze

Vektorrepräsentation der Daten und Vorhersage der möglichen Satzkategorien. Nachdem der Gesamtprozess korrekt und vollständig ausgeführt wurde, liegen die Ergebnisse in einer neuen Datei gespeichert vor. Nach Auslesen dieser Datei soll der Agent den Graphen mit den neu gewonnenen Informationen anreichern.

Datensatz

Zunächst wird das Korpus, das in einem CSV-Format gespeichert ist, vom Pythonskript ausgelesen. Dafür wird die Software-Bibliothek *Pandas*[2] benutzt, die das Auswerten und Bearbeiten tabellarischer Daten ermöglicht. *Pandas* steht für „Python Data Analysis Library“ und ist eine der am häufigsten verwendeten Python-Bibliotheken für Datenmanipulation und Datenanalyse. Der Datensatz wird als *Dataframe* repräsentiert - ein Objekt, das aus einer zweidimensionalen Tabelle besteht. Dabei hat jede Spalte einen Index und einen eindeutigen Datentyp. Beim Prozess des Trainierens der Klassifikatoren wird der Datensatz durch eine Tabelle mit zwei Spalten repräsentiert. Diese Tabelle enthält sowohl die Sätze, als auch ihre manuell beschrifteten Markierungen. Bei der tatsächlichen Anwendung in INDIRECT wird dem Pythonskript eine einspaltige Tabelle als Eingabe übergeben, wobei jeder Satz in eine neue Zeile verteilt ist. Das Skript erweitert die Tabelle um eine neue Spalte, in welcher die vorhergesagten Kategorien gespeichert werden.

Pandas ist auf *NumPy*[3] aufgebaut, was bedeutet, dass *NumPy* eine Voraussetzung für die Verwendung von Pandas ist. *NumPy* ist eine Bibliothek, mit der sich mehrdimensionale Arrays verarbeiten lassen.

Vorverarbeitung

Die Vorverarbeitung der Eingabedaten beginnt mit Entfernung von Satzzeichen und Umwandlung der Daten in Kleinbuchstaben. Vor der Anwendung von allen betrachteten Merkmalskonstruktionsmethoden außer Bag-Of-N-Grams auf Zeichen wird eine Tokenisierung der Sätze durchgeführt. Die Tokenisierung wird mittels *NLTK*[4] (*The Natural Language*

[2]Quelle: https://pandas.pydata.org/, zuletzt besucht am 06.10.2019
[3]Quelle: https://numpy.org/, zuletzt besucht am 06.10.2019
[4]Quelle: http://www.nltk.org/, zuletzt besucht am 06.10.2019

ToolKit) umgesetzt. *NLTK* ist eine Zusammenstellung von Bibliotheken für Sprachverarbeitung in Python, die Schnittstellen zu über 50 Korpora- und Lexikonressourcen wie WordNet sowie Textverarbeitungsbibliotheken bietet.

Vektorrepräsentation

Nach der Normalisierung und Tokenisierung sollen die Daten verarbeitet und als Vektoren repräsentiert werden. Für die Bag-Of-Words-, Bag-Of-N-Grams- und TfIdf-Ansätze (siehe Abschnitt 5.3.1.2) werden die Klassen *CountVectorizer* bzw. *TfIdfVectorizer* von *scikit-learn*[5] verwendet. *Scikit-learn* ist ein Python-Modul mit Werkzeugen für Datenverarbeitung, sowie für Standardaufgaben des maschinellen Lernens. Es basiert auf *NumPy* und *SciPy*[6] - eine weitere Kernbibliothek für wissenschaftliches Rechnen, die die Fähigkeiten von *NumPy* erweitert.

Für die Umsetzung von Word2Vec und Doc2Vec bietet sich *Gensim*[7] an. *Gensim* ist eine Python-Bibliothek für semantische Analysen, Themenmodellierung und Vektorraummodellierung, die auch auf *NumPy* und *SciPy* basiert. *NumPy* wird außerdem beim Berechnen des Durchschnitts der Worteinbettungsvektoren verwendet, das bei Anwendung von Word2Vec für Satzklassifikation nötig ist. Die in dieser Arbeit angewendeten Klassifikatoren erfordern, dass die modifizierten Eingabemerkmale dieselbe Länge haben. Da alle Sätze unterschiedliche Längen haben können, ist es nicht möglich, nur ein Array von Worteinbettungen ohne weitere Verarbeitung an die Klassifikatoren als Eingabe zu übergeben. Zuerst werden also die Worteinbettungen für alle Wörter erzeugt und danach wird ihr Durchschnitt berechnet, der den Satzvektor repräsentiert.

Um bestmögliche Ergebnisse durch Verwendung von Worteinbettungen zu erzielen, werden zwei bekannte vortrainierte Modelle angewendet und verglichen - das *GloVe*- und das *Google-News-Word2Vec*-Modell. *GloVe* [PSM14] ist ein unüberwachtes Lernverfahren zum Erhalten von Vektordarstellungen für Wörter. Das *GloVe*-Modell übertrifft andere Modelle (CBOW, Skip-Gram) hinsichtlich Wortanalogie, Wortähnlichkeit und Aufgaben zur Eigennamenerkennung. Es werden 50-, 100-, 200- und 300-dimensionale mit *GloVe* vortrainierte Wortvektoren zur Verfügung gestellt. Die Worteinbettungen wurden auf Wikipedia 2014 und Gigaword 5[8] aufgebaut. Das gesamte Korpus besteht aus sechs Milliarden Tokens, wobei ein Vokabular aus den 400.000 häufigsten Wörtern erzeugt wird. Das zweite fertige Modell, das für die Datenrepräsentation in dieser Arbeit verwendet wird, ist das *Google-News*-Modell[9]. Es enthält 300-dimensionale Vektoren für drei Millionen Wörter und Phrasen, die auf einem Teil des Google News-Datensatzes (etwa 100 Milliarden Wörter) trainiert wurden.

Da keine vortrainierten Modelle mit Satzeinbettungen gefunden wurden, wird Doc2Vec auf dem eigenen Datensatz trainiert. Hier muss beachtet werden, dass der verwendete Datensatz eine kleine Anzahl an Wörtern enthält und daher bei Klassifikation von ungesehenen Daten zu schlechten Ergebnissen führen kann. Es besteht noch die Möglichkeit, Doc2Vec auf einem großen Korpus wie Wikipedia zu trainieren, worauf im Rahmen der vorliegenden Arbeit aus technischen Gründen verzichtet wurde.

Klassifikatoren

Wie bereits oben angedeutet, bietet die *Scikit-learn*-Bibliothek Algorithmen für maschinelles Lernen. Die folgenden Klassifikatoren werden mit Hilfe von *Scikit-learn* trainiert,

[5] Quelle: https://scikit-learn.org/stable/, zuletzt besucht am 06.10.2019
[6] Quelle: https://www.scipy.org/, zuletzt besucht am 06.10.2019
[7] Quelle: https://radimrehurek.com/gensim/, zuletzt besucht am 06.10.2019
[8] Quelle: https://catalog.ldc.upenn.edu/LDC2011T07, zuletzt besucht am 13.10.2019
[9] Quelle: https://code.google.com/archive/p/word2vec/, zuletzt besucht am 13.10.2019

angewendet und evaluiert. Zusätzlich werden die Hyperparameter für jeden Klassifikator angegeben, die bei der Optimierung in dieser Arbeit betrachtet werden.

- Multinomial Naïve Bayes[10] (siehe Abschnitt 2.4.1)
  `alpha`: Glättungskoeffizient
- Random Forest[11] (siehe Abschnitt 2.4.3)
  `n_estimators`: Die Anzahl der Bäume im Wald
  `max_depth` : Die maximale Tiefe eines Baumes
- Logistische Regression[12] (siehe Abschnitt 2.4.4)
  `C`: Strafparameter für falsch klassifizierte Stichproben
- Stützvektormaschine[13] (siehe Abschnitt 2.4.5)
  `kernel`: linear, polynomial oder Gauss-Kernel
  `C`: Strafparameter für falsch klassifizierte Stichproben

Um optimale Werte für die Hyperparameter zu erhalten, wird eine 10-fache Kreuzvalidierung auf dem Trainingsdatensatz durchgeführt. Die Einschätzung der Leistungsfähigkeit der Modelle erfolgt durch Tests auf ungesehenen Daten. Nachdem der meist passende Klassifikator gewählt wurde, wird er auf dem gesamten Datensatz trainiert und für die weitere Anwendung durch den INDIRECT-Agenten zur Verfügung gestellt. Zum Speichern und Laden des Klassifikators wird das *pickle*-Modul[14] verwendet.

Klassifikatorintegration

Nachdem die Eingabedaten als eine CSV-Datei vorbereitet wurden, wird der Python-Prozess vom Agenten gestartet. Die Ausführung erfolgt mit Hilfe der Klasse `ProcessBuilder`[15] des Apache-Commons-Exec-Projekts. Das Pythonskript umfasst die Umsetzung aller einzelnen Schritte der Klassifikation – Datenverarbeitung und Vorhersage der Kategorien. Die Kategorien werden in eine neue CSV-Datei gespeichert. Sobald die Ausführung des Klassifikation-Prozesses fertig ist, wird die Ausgabe vom Agenten ausgelesen.

6.2.2 Negation

Für die Negationserkennung werden die durch den Agenten `DepParse` von INDIRECT bereitgestellten Informationen verwendet. Hierfür wird für jeden Satz geprüft, ob eine `neg`-Abhängigkeit als eine Kantenannotation gespeichert ist. Die Erstellung der Abhängigkeitsbäume wird durch den syntaktischen Abhängigkeitszerleger von Stanford CoreNLP[16] ermöglicht. Eine Voraussetzung für die Verwendung dieses Zerlegers ist eine vorherige Wortartmarkierung.

[10] Quelle: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html, zuletzt besucht am 20.10.2019
[11] Quelle: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, zuletzt besucht am 20.10.2019
[12] Quelle: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, zuletzt besucht am 20.10.2019
[13] Quelle: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html, zuletzt besucht am 20.10.2019
[14] Quelle: https://docs.python.org/3/library/pickle.html, zuletzt besucht am 29.10.2019
[15] Quelle: https://docs.oracle.com/javase/1.5.0/docs/api/java/lang/ProcessBuilder.html, zuletzt besucht am 06.10.2019
[16] Quelle: https://stanfordnlp.github.io/CoreNLP/depparse.html, zuletzt besucht am 06.10.2019

6.2.3 Repräsentation der Ergebnisse

Nachdem die Art und Polarität der Sätze bestimmt wurde, wird der Graph mit den Ergebnissen angereichert. Die Semantik der Sätze wird durch eine Zeichenkette dargestellt, die beide Informationen enthält. Für jeden Satz wird im Graphen jeweils ein neuer Knoten erzeugt, in dem die Semantik des Satzes gespeichert wird. Diese Repräsentation wurde bereits in Abbildung 5.5 vorgestellt.

7 Evaluation

In diesem Kapitel wird die Funktionalität des entwickelten Werkzeugs zur Klassifikation von Anforderungsbeschreibungen analysiert und bewertet. In Abschnitt 7.1 wird der für die Evaluation verwendete Datensatz näher betrachtet. Weiterhin wird in Abschnitt 7.2 die Vorgehensweise präsentiert. Anschließend werden in Abschnitt 7.3 die Ergebnisse der Evaluation vorgestellt und diskutiert. Dabei werden die Ergebnisse beim Testen auf ungesehenen Daten erläutert und zusätzlich wird eine Kreuzvalidierung auf dem gesamten Datensatz durchgeführt.

7.1 Datensatz

Der Korpus, der für die Evaluierung des in dieser Bachelorarbeit entwickelten Werkzeugs benutzt wurde, wurde bereits in Abschnitt 6.1 vorgestellt. Er enthält händisch annotierte Anforderungsbeschreibungen aus fünf verschiedenen Projekten. In Tabelle 7.1 wird ein Überblick über die Aufteilung der Anforderungsbeschreibungen in den einzelnen Projekten nach Satzkategorien gegeben.

Ziel der Bewertung der Leistungen der Klassifikatoren ist es, einen Klassifikator zu wählen, der möglichst gute Ergebnisse auf ungesehenen Daten erzielt. Dafür wird ein Testdatensatz benötigt, der unabhängig vom Trainingsprozess ist. Bei der Evaluation der Ergebnisse möchte man eine Schlussfolgerung ziehen, wie gut die verschiedenen Klassen vorhergesagt wurden. Aus diesem Grund soll der Testdatensatz genug Beispiele für alle möglichen Kategorien enthalten. Da aus allen fünf Projekten im iTrust-Datensatz am häufigsten Beispiele für Ereignisse, Aggregationen und Zustandsbeschreibungen vorkommen, wird er hier als

Tabelle 7.1: Aufteilung des Datensatzes

Projekt	Aktion	Ereignis	Zustand	Aggregation	Gesamt
NFR	139	43	14	15	202
SR Risk prediction	56	16	6	17	98
iTrust	88	83	37	22	230
Warc	24	23	15	8	70
IceBreaker	32	55	7	6	100

Testdatensatz verwendet. Dadurch kann geprüft werden, wie gut die Ergebnisse der Klassifikatoren auf neue Projekte sind.

7.2 Vorgehensweise

In diesem Abschnitt wird über die Vorgehensweise bei der Durchführung der Evaluation berichtet. Zuerst wird nur die Klassifizierung nach Satzkategorien betrachtet, ohne die Polarität zu berücksichtigen. Hierfür wird eine Optimierung der Lernverfahren durchgeführt, um den bestmöglichen Klassifikator zu ermitteln. Jedem Satz wird genau eine Klasse zugeordnet, wobei als Klassen die in Abschnitt 5.2 definierten Satzkategorien und ihre entsprechenden Markierungen (*action, occurrence, state, aggregation*) verwendet werden. Die Erkennung von Negationen hängt von der Leistung des Abhängigkeitszerlegers ab, der durch den `DepParse`-Agenten (siehe Abschnitt 6.2.2) bereitgestellt wird. Nachdem der beste Klassifikator für die Erkennung von Satzkategorien gefunden wurde, wird durch eine Ende-Zu-Ende Evaluation die Leistung des implementierten Agenten bewertet.

Die Ergebnisse werden durch Berechnung der Metriken Präzision (engl. *precision*), Ausbeute (engl. *recall*), Genauigkeit (engl. *accuracy*) und F_1-Maß (engl. F_1-*Score*) bewertet [JM18]. Dafür werden zuerst die vier möglichen Ergebnisse bei der Klassifikation nach Satzkategorien anhand der Klasse Aktionsbeschreibung (*action*) vorgestellt:

- **Richtig positives Ergebnis** (engl. *true positive, tp*): Es wurde die Klasse *action* erwartet und der Satz wurde als *action* markiert.
- **Richtig negatives Ergebnis** (engl. *true negative, tn*): Es wurde nicht die Klasse *action* erwartet und der Satz wurde nicht als *action* markiert.
- **Falsch positives Ergebnis** (engl. *false positive, fp*): Es wurde nicht die Klasse *action* erwartet, jedoch wurde der Satz als *action* markiert.
- **Falsch negatives Ergebnis** (engl. *false negative, fn*): Es wurde die Klasse *action* erwartet, jedoch wird der Satz nicht als *action* markiert.

Mittels der Kennzahlen für die richtig und fälschlicherweise zugeordneten Sätze werden die oben genannten Metriken ermittelt. Eine Möglichkeit zur Bestimmung der Leistung des entwickelten Werkzeugs ist die Genauigkeit. Sie gibt an, wie viel Prozent aller Sätze vom System korrekt erkannt wurden. Die Genauigkeit lässt sich durch die folgende Formel berechnen:

$$Genauigkeit = \frac{tp + tn}{tp + fp + tn + fn} \tag{7.1}$$

Die Genauigkeit ist ein intuitives Leistungsmaß, das das Verhältnis von korrekt vorhergesagter Beobachtung zu den gesamten Beobachtungen zeigt. Sie funktioniert gut bei einer näherungsweise gleichen Verteilung der Klassen in dem Datensatz. Wenn aber die Klassen unausgewogen sind, kann ein hohes Genauigkeitsniveau auch bei Anwendung von sehr einfachen Klassifikatoren erzielt werden. Zum Beispiel möchte man Tests auf dem NFR-Datensatz durchführen, wobei jeder Satz entweder als „Zustandsbeschreibung“ oder als „keine Zustandsbeschreibung“ klassifiziert werden sollte. Hierfür wird ein einfacher Klassifikator erstellt, der jedem Satz die Klasse „keine Zustandsbeschreibung“ zuordnet. Die Anzahl der korrekt zugeordneten negativen Ergebnisse (tn) ist in diesem Fall gleich 188, und die Anzahl der fälschlicherweise zugeordneten negativen Ergebnissen (fn) nur 14. Der Klassifikator hat also eine Genauigkeit von 93 Prozent. Er wird aber trotzdem nutzlos sein, wenn man genau die Zustandsbeschreibungen im Datensatz finden möchte.

Weitere Metriken zur Evaluierung der Leistung eines Klassifikators sind seine Präzision und Ausbeute. Die Präzision impliziert den Prozentsatz der korrekt identifizierten positiven Fälle aus allen vorhergesagten positiven Fällen. Sie berechnet sich durch die Formel:

$$\text{Präzision} = \frac{\text{Anzahl der korrekt erkannten Sätze pro Klasse}}{\text{Gesamtanzahl der erkannten Sätze pro Klasse}} = \frac{tp}{tp + fp} \tag{7.2}$$

Die Ausbeute ist ein Maß für die korrekt identifizierten positiven Fälle aus allen tatsächlich positiven Fällen. Die Ausbeute berechnet sich durch die Formel:

$$\textit{Ausbeute} = \frac{\text{Anzahl der korrekt erkannten Sätze pro Klasse}}{\text{Gesamtanzahl der zu findenden Sätze pro Klasse}} = \frac{tp}{tp + fn} \tag{7.3}$$

Bei einer ungleichmäßigen Klassenverteilung wird anstatt Genauigkeit häufig das F_1-Maß verwendet, um ein Modell zu bewerten. Das F_1-Maß misst den gewichteten Durchschnitt der Präzision und Ausbeute:

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

Das F_1-Maß berücksichtigt sowohl *fp*, als auch *fn*. Wenn *fp* und *fn* sehr unterschiedliche Werte haben, ist das F_1-Maß nützlicher als die Genauigkeit.

In dieser Arbeit wird eine multinomiale Klassifikation der Sätze betrachtet. Hierfür werden binäre Klassifikatoren für alle Klassen erstellt, wobei jeder Klassifikator auf positiven Beispielen von einer Klasse und negativen Beispielen von allen anderen Klassen trainiert wird. Um die Kategorien der Sätze vorherzusagen, werden alle Klassifikatoren ausgeführt und für jeden Satz wird die Kategorie aus dem Klassifikator mit der höchsten Punktzahl gewählt. Dabei ist die resultierende Konfusionsmatrix (engl. *confusion matrix*), die die Häufigkeiten der vorhergesagten und tatsächlichen Ergebnisse für alle Klassen angibt, eine 4x4-Matrix, da es hier vier möglichen Kategorien gibt. Diese Matrix zeigt, wie viele Sätze korrekt oder fälschlicherweise in die verschiedenen Kategorien eingeordnet wurden. Mit Hilfe der Konfusionsmatrix wird für jede Klasse ein eindeutiger Präzisions- und Ausbeutewert berechnet. Diese Werte können auf zwei Arten kombiniert werden. Die erste Möglichkeit besteht darin, den **Makro-Durchschnitt** zu ermitteln. Dabei wird zunächst die Leistung für jede Klasse und dann der Durchschnitt über alle Klassen berechnet. Die zweite Möglichkeit ist die Berechnung des **Mikro-Durchschnitts**. Hierfür wird zuerst jede der vier Kategorien (tp, tn, fp, fn) über die Klassen summiert und dann werden die Summen in die Definition eingesetzt. Die Metriken können Werte zwischen 0 und 1 annehmen. Je höher der Wert einer Kennzahl ist, desto zufriedenstellender ist die Modellgüte.

7.3 Ergebnisse

In diesem Abschnitt werden die Ergebnisse der Evaluierung erläutert. Zunächst wird eine Kreuzvalidierung auf dem Trainingsdatensatz durchgeführt, um passende Hyperparameter der verschiedenen Klassifikatoren zu finden. Dadurch wird die Generalisierungsfähigkeit der Lernmodelle getestet, die von großer Bedeutung für die Einschätzung der Voraussagequalität unbekannter Daten ist. Nachdem die bestmöglichen Parameter für alle Klassifikatoren gewählt wurden, werden die Modellen auf dem gesamten Trainingssatz trainiert. Die Performanz der Klassifikatoren wird anschließend auf Basis der Testdaten evaluiert. Wie bereits in Abschnitt 7.1 erwähnt, wird als Testdatensatz der iTrust-Datensatz verwendet. Zum Schluss werden die optimierten Klassifikatoren durch eine Kreuzvalidierung auf den gesamten Datensatz getestet.

7.3.1 Ergebnisse beim Testen auf iTrust

In diesem Abschnitt werden die Evaluationsergebnisse, die auf dem iTrust-Korpus erzielt wurden, dargestellt und analysiert. Dabei werden die Metriken Genauigkeit und F_1-Maß betrachtet. Für das Training wird dabei ein Datensatz verwendet, der die Sätze aus den anderen vier Projekten enthält.

Hyperparameteroptimierung

Bei der Auswertung verschiedener Hyperparameter der Klassifikatoren wird oft Teil des Trainingsdatensatzes verwendet, um das Risiko einer Überanpassung des Testdatensatzes zu vermeiden. Durch die Aufteilung der verfügbaren Daten in drei Sätze (für das Training, die Validierung und das Testen) wird jedoch die Anzahl der Stichproben, die zum Erlernen des Modells verwendet werden können, reduziert. Außerdem hängen die Ergebnisse von einer bestimmten Zufallsauswahl des Trainings- und Validierungsdatensatzes ab. Um die Varianz der Abschätzung zu verringern, wird eine Kreuzvalidierung angewendet. Durch dieses Verfahren können einerseits geeignete Hyperparameter der Lernmodelle gewählt und andererseits verschiedene Modelle miteinander verglichen werden. Bei einer k-fachen Kreuzvalidierung wird der Trainingsdatensatz in k ungefähr gleich große Partitionen aufgeteilt. Hierfür dient eine Partition als Testdatensatz und die übrigen k-1 Partitionen werden für das Training des Modells verwendet. Nachdem dieses Vorgehen mit allen Partitionen wiederholt wurde, wird der Durchschnitt der Testfehler der Partitionen ermittelt.

Vor der Kreuzvalidierung wird ein zufälliges Mischen der Trainingsdaten durchgeführt. Anschließend wird die Anzahl der Partitionen bestimmt. Der Fall k = N (N = Anzahl der Einträge im Datensatz) ist als eine *Leave-One-Out*-Kreuzvalidierung bekannt. Dieses Verfahren erreicht eine hohe Verlässlichkeit in der Qualität der Schätzung der Ergebnisse, aber kann sehr rechenintensiv sein. Im Vergleich dazu ist eine Kreuzvalidierung mit minimalem k = 2 schnell berechenbar, dafür aber ungenauer. Eine typische Wahl für die Anzahl der Partitionen ist k = 5 oder k = 10 [HTF01]. In dieser Arbeit wird eine 10-fache Kreuzvalidierung auf dem Trainingsdatensatz angewandt. Die Validierungsstichproben repräsentieren jeweils 10 Prozent des Datensatzes und der Prozess der Modellbildung basiert auf den restlichen 90 Prozent. Im Folgenden werden die Ergebnisse der 10-fachen Kreuzvalidierung erläutert. Die Genauigkeitswerte werden in Tabelle 7.2 dargestellt. In Tabelle 7.3 werden die Werte des F_1-Maßes (Makro-Durchschnitt) angegeben. Dabei steht „LR“ für logistische Regression, „SVM“ für Stützvektormaschine, „RF“ für Random Forest und „NB“ für Naïve Bayes. Es werden die Techniken Bag-Of-Words („BoW“), N-Gramme auf Zeichenebene („N-Gramme: Zeichen“), N-Gramme auf Wortebene („N-Gramme: Wörter“), Vorkommenshäufigkeit und inverse Dokumenthäufigkeit („TfIdf“, engl. *term frequency–inverse document frequency*), GloVe-Worteinbettungen („GloVe“), GoogleNews-Worteinbettungen („GoogleNews“) und Satzeinbettungen („Doc2Vec“) untersucht.

Bei multinomialen Klassifikationsaufgaben entspricht der Micro-F_1-Durchschnittswert der Genauigkeit. Da der Mikro- und Makro-Durchschnitt unterschiedlich berechnet werden, werden sie auch unterschiedlich interpretiert. Ein Makro-Durchschnitt behandelt alle Klassen gleich, während ein Mikro-Durchschnitt die Beiträge aller Klassen zur Berechnung des durchschnittlichen Wertes berücksichtigt. Wie man aus den Tabellen sehen kann, ist der Mikro-Durchschnitt überall höher als der Makro-Durchschnitt. Das liegt daran, dass es ein Klassenungleichgewicht gibt. Zustandsbeschreibungen treten sehr selten auf und ihre Erkennung ist eine Herausforderung für alle betrachteten Klassifikatoren. Wegen der schlechten Vorhersage von Zustandsbeschreibungen sinkt der gesamte Makro-Durchschnittswert.

Mit Hilfe der Kreuzvalidierung werden die Parameter, mit deren Einstellung die Modelle die geringsten Fehler aufweisen, ausgewählt. Diese optimalen Parameter werden dann zur

Tabelle 7.2: Mikro-F_1-Durchschnitt (Genauigkeit), 10-fache Kreuzvalidierung

Technik	LR	SVM	RF	NB
BoW	0.86	0.87	0.84	0.75
N-Gramme: Zeichen	0.91	**0.92**	0.9	0.79
N-Gramme: Wörter	0.88	0.90	0.80	0.78
TfIdf	0.84	0.84	0.84	0.71
GloVe	0.83	0.84	0.73	-
GoogleNews	0.86	0.87	0.74	-
Doc2Vec	0.85	0.87	0.86	-

Tabelle 7.3: Makro-F_1-Durchschnitt, 10-fache Kreuzvalidierung

Technik	LR	SVM	RF	NB
BoW	0.77	0.78	0.66	0.59
N-Gramme: Zeichen	0.85	**0.87**	0.78	0.71
N-Gramme: Wörter	0.74	0.79	0.57	0.66
TfIdf	0.74	0.76	0.63	0.55
GloVe	0.70	0.72	0.46	-
GoogleNews	0.76	0.77	0.46	-
Doc2Vec	0.78	0.81	0.83	-

endgültigen Modellbildung verwendet. Hier muss beachtet werden, dass nicht alle möglichen Parameter der Klassifikatoren optimiert wurden, da eine Modellselektion, die durch die Anwendung von vielen verschiedenen Kombinationen von Parametern getroffen wurde, sehr rechenintensiv ist. Im Folgenden werden die betrachteten Parameter und ihre Auswirkung auf die Ergebnisse der Kreuzvalidierung vorgestellt.

Die Stützvektormaschine und die logistische Regression erzielen bessere Ergebnisse als die anderen zwei Klassifikatoren, wobei SVM einen kleinen Vorteil aufweist. Ein Parameter der Stützvektormaschine ist der Kernel. Dieser Parameter wählt den Typ der Hyperebene aus, die zur Trennung der Daten verwendet wird. Dabei bezieht sich der Wert *linear* auf eine lineare Hyperebene, während *rbf* und *poly* eine nicht-lineare Hyperebene implizieren. Bei allen Modellen hat sich eine lineare Hyperebene als die bessere Einstellung erwiesen.

Ein weiterer Konstruktorparameter, der sowohl bei Stützvektormaschinen, als auch bei logistischer Regression angepasst werden kann, ist der Strafparameter C. Er definiert die Größe der Strafe, wenn eine Stichprobe falsch klassifiziert wird. Der Standardwert für C ist 1. Bei allen Experimenten hat ein größerer Wert für C (C=10 oder C=100) zu besseren Ergebnissen geführt, da dadurch die Modelle für jede falsche Klassifizierung strenger bestraft werden. Allerdings kann die Erhöhung von C eine Überanpassung verursachen.

Einer der wichtigsten Parameter des Random Forest-Klassifikators ist `n_estimators`. Er definiert die Anzahl der Bäume im Wald. Bei den meisten Vektorisierungstechniken wurde `n_estimators`=100 gewählt. Nur bei der Anwendung von Doc2Vec war ein größerer Wert (`n_estimators`=200) besser geeignet. Eine höhere Anzahl von Bäumen hat die Ergebnisse nicht deutlich verbessert, verlangsamt jedoch die Berechnung. Zusätzlich wurde die maximale Tiefe jedes Baumes im Wald durch den Hyperparameter `max_depth` angepasst.

Je tiefer ein Baum ist, desto mehr Aufteilungen hat er und desto mehr Informationen erfasst er über die Daten. Ein großer Wert der maximalen Baum-Tiefe kann aber auch zu einer Überanpassung führen. Bei der Verwendung von Bag-Of-N-Grams auf Zeichenebene, Word2Vec und Doc2Vec wurde eine Baum-Tiefe von 20, und bei allen anderen – von 25 vorgezogen. Bei zunehmender Tiefe der Bäume ist die Performanz der Algorithmen nahezu identisch.

Der multinomiale Naïve Bayes-Klassifikator bietet die Möglichkeit an, den Glättungskoeffizienten `alpha` zu bestimmen. Der Standardwert von `alpha` ist 1. Dies impliziert eine Laplace-Glättung (engl. *add-one smoothing*). Bei diesem Glättungsverfahren wird die Worthäufigkeit je Klasse um eins erhöht, um Nullwahrscheinlichkeiten zu vermeiden. Unter Verwendung von TfIdf und Bag-Of-N-Grams auf Zeichen hat eine Lidstone-Glättung ($\alpha < 1$) die Ergebnisse verbessert, wobei hier `alpha` $= 0.1$ gewählt wurde.

Aus den Tabellen kann man entnehmen, dass alle vier Modelle am besten mittels des Bag-Of-N-Grams-Ansatzes auf Zeichen zur Merkmalskonstruktion funktionieren. Hierfür wurden die folgenden Parameter der Klassifikatoren verwendet:

LogisticRegression (C=100, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, l1_ratio=None, max_iter=100, multi_class='warn', n_jobs=None, penalty='l2', random_state=None, solver='warn', tol=0.0001, verbose=0, warm_start=False)

SVC (C=100, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)

RandomForestClassifier (bootstrap=True, class_weight=None, criterion='gini', max_depth=20, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False)

MultinomialNB (alpha=0.1, class_prior=None, fit_prior=True)

Bei dem Bag-Of-N-Grams-Ansatz auf Zeichen werden alle Zeichenketten der Länge n betrachtet, wobei sich eine untere Grenze von 2 ($2 \leq n$) und eine obere Grenze von 5 ($n \leq 5$) Zeichen als optimal erwiesen haben. Dieser Bereich wird mittels des `ngram_range`-Parameters der `CountVectorizer`-Klasse[1] von *Scikit-learn*, die für die Vektorisierung der Daten verwendet wird, definiert. Im Vergleich zu n-Grammen von Wörtern, die nur die Identität eines Wortes und seine möglichen Nachbarn erfassen, sind n-Gramme von Zeichen zusätzlich in der Lage, die morphologische Zusammensetzung eines Wortes zu erkennen. Beim Bag-Of-N-Grams-Ansatz auf Wörtern wurden alle Unigramme, Bigramme und Trigramme berücksichtigt. Der einfache BoW-Ansatz, der nur Unigramme betrachtet, verliert viele Feinheiten einer möglichen guten Darstellung, wie die Wortfolge. Das ist der Fall auch bei der Anwendung von Worteinbettungen, wo der Mittelwert von allen Wortvektoren berechnet wird, um den Satzvektor zu bekommen. In den Tabellen sind die Ergebnisse für die vortrainierten 300-dimensionale GloVe-Worteinbettungen angezeigt, da sie die 50-, 100- und 200-dimensionalen Vektoren übertreffen. Obwohl vortrainierte Word2Vec-Modelle eine große Basis an Vokabular haben, kommen im Datensatz auch Wörter vor, die nicht durch das Modell repräsentiert werden können und daher als Null-Vektoren abgebildet werden. Während die Wortvektoren nur das Konzept eines Wortes darstellen, versucht Doc2Vec, das Konzept eines Textes zu erfassen. Hierfür wird ein Doc2Vec-Modell auf dem

[1] Quelle: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html, zuletzt besucht am 22.10.2019

Tabelle 7.4: Mikro-F_1-Durchschnitt (Genauigkeit), Tests auf dem *iTrust*-Datensatz

Technik	LR	SVM	RF	NB
BoW	0.71	0.72	0.62	0.61
N-Gramme: Zeichen	0.72	0.72	**0.79**	0.57
N-Gramme: Wörter	0.68	0.69	0.51	0.63
TfIdf	0.68	0.66	0.63	0.61
GloVe	0.67	0.69	0.56	-
GoogleNews	0.74	0.71	0.56	-
Doc2Vec	0.61	0.60	0.62	-

Tabelle 7.5: Makro-F_1-Durchschnitt, Tests auf dem *iTrust*-Datensatz

Technik	LR	SVM	RF	NB
BoW	0.68	0.69	0.52	0.49
N-Gramme: Zeichen	0.69	0.70	**0.75**	0.50
N-Gramme: Wörter	0.61	0.64	0.31	0.51
TfIdf	0.64	0.63	0.56	0.52
GloVe	0.58	0.60	0.36	-
GoogleNews	0.68	0.66	0.32	-
Doc2Vec	0.49	0.50	0.52	-

gesamten Trainingsdatensatz trainiert. Durch Satzeinbettungen werden relativ gute Vorhersagen der Satzkategorien erzielt. Allerdings ist der Datensatz zu klein, um generell ein gutes Doc2Vec-Modell zu erstellen.

Tests auf dem iTrust-Datensatz

Nach der Optimierung der Parameter erfolgt die Evaluierung der Modelle auf Basis der Testdaten. Der Mikro- und Makro-F_1-Durchschnitt beim Testen auf den iTrust-Datensatz werden in Tabelle 7.4 und Tabelle 7.5 angegeben. Zusätzlich werden in Abschnitt B die Ergebnisse beim Testen auf den anderen vier Projekte dargestellt. Dabei werden die Klassifikatoren immer auf vier Projekten trainiert und auf einem getestet.

Der Prädiktionsfehler, der durch die Kreuzvalidierung geschätzt wurde, ist viel kleiner als der Generalisierungsfehler auf Basis der Testdaten. Das liegt daran, dass dieselben Daten verwendet werden, um die Modelle anzupassen und ihre Fehler zu bewerten. Die Modelle passen sich typischerweise den Trainingsdaten an, so dass der Trainingsfehler eine zu optimistische Schätzung des Generalisierungsfehlers ist. Bei der Einsetzung von Doc2Vec zur Merkmalskonstruktion werden um fast 30 Prozent schlechtere Ergebnisse erzielt. Im Allgemeinen ist Doc2Vec ein passender Ansatz zur Repräsentation von Texten, weil er nicht nur die Semantik von Wörtern erfasst, sondern auch die Reihenfolge der Wörter berücksichtigt. Da das Doc2Vec-Modell hier aber nur auf einen kleinen Trainingsdatensatz trainiert wurde, ist er auf ungesehene Daten schwer übertragbar.

Aus den Tabellen kann man erkennen, dass unter Verwendung von Bag-Of-N-Grammen auf Zeichen der Random Forest-Klassifikator die Stützvektormaschine übertrifft. Um eine detaillierte Übersicht über die Ergebnisse zu bekommen, werden in Abbildung 7.1 die

Konfusionsmatrizen für beide Verfahren dargestellt. Zusätzlich wurde das Werkzeug `classification_report`[2] von *Scikit-learn* verwendet, um die einzelnen Metriken zu berechnen. In Tabelle 7.6 und Tabelle 7.7 werden die Kennzahlen pro Klasse und die Mittelwerte angegeben.

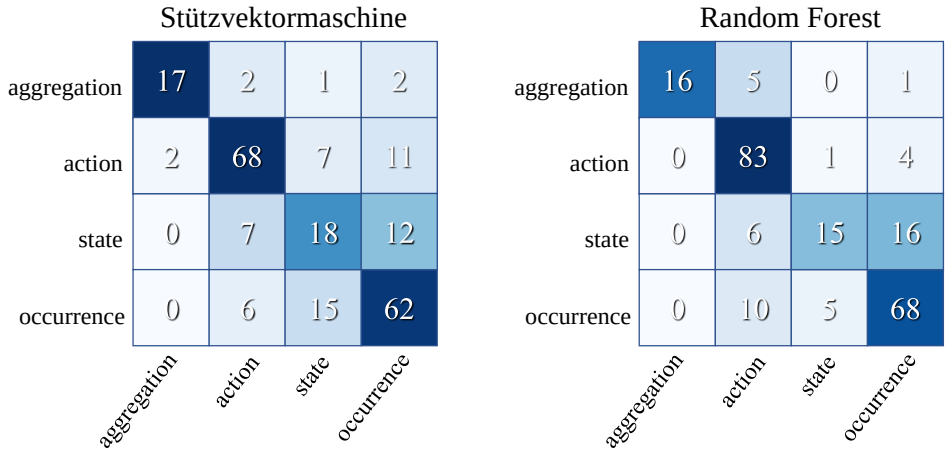

Abbildung 7.1: Konfusionsmatrizen für Stützvektormaschine und Random Forest (Zeile = erwartet, Spalte = erkannt)

Tabelle 7.6: Stützvektormaschine: Klassifizierungsbericht; Tests auf dem *iTrust*-Datensatz

Klasse	Präzision	Ausbeute	F_1-Maß	Anzahl
action	0.82	0.77	0.80	88
occurrence	0.71	0.75	0.73	83
aggregation	0.89	0.77	0.83	22
state	0.44	0.49	0.46	37
Genauigkeit			0.72	230
Makro-Durchschnitt	0.72	0.69	0.70	230

Auffällig ist die sehr niedrige Trefferquote für Zustandsbeschreibungen. Obwohl SVM mehr Zustandsbeschreibungen (18) als RF (15) korrekt erkannt hat, erzielt der RF-Klassifikator eine höhere Präzision (0.71), da er generell wenige Sätze als Zustandsbeschreibungen markiert hat. Die Stützvektormaschine erzielt bei der Erkennung der Zustandsbeschreibungen eine Präzision von nur 0.44, da er viele Sätze, die in andere Kategorien gehören, als `state` markiert. Ein möglicher Grund dafür ist, dass manche Sätze zusätzliche Informationen beinhalten, die als Zustände interpretiert werden können. Damit werden zum Beispiel Sätze gemeint, die eine Aktion angeben und eine Zustandsbeschreibung in Form eines Adjektivs enthalten, die sich auf die Subjektentität bezieht. Da bei manchen Anforderungsbeschreibungen einzelne Satzteile unterschiedliche Klassen darstellen können, kann

[2] Quelle: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metrics.classification_report, zuletzt besucht am 23.10.2019

Tabelle 7.7: Random Forest: Klassifizierungsbericht; Tests auf dem *iTrust*-Datensatz

Klasse	Präzision	Ausbeute	F_1-Maß	Anzahl
action	0.80	0.94	0.86	88
occurrence	0.76	0.82	0.79	83
aggregation	1.00	0.73	0.84	22
state	0.71	0.41	0.52	37
Genauigkeit			0.79	230
Makro-Durchschnitt	0.82	0.72	0.75	230

Tabelle 7.8: Zusammenfassung der Ergebnisse

Klasse	Präzision	Ausbeute	F_1-Maß	Anzahl
positive_action	0.82	0.96	0.89	80
negative_action	0.60	0.75	0.67	8
positive_occurrence	0.82	0.82	0.82	77
negative_occurrence	0.42	0.83	0.56	6
positive_aggregation	1.00	0.80	0.89	15
negative_aggregation	1.00	0.57	0.73	7
positive_state	0.67	0.38	0.49	26
negative_state	0.83	0.45	0.59	11
Genauigkeit			0.79	230
Makro-Durchschnitt	0.77	0.70	0.70	230

bei einer Weiterentwicklung des Agenten die Einsetzung einer sorgfältigen Rahmenklassifikation überlegt werden.

Beide Verfahren haben Schwierigkeiten damit, Ereignisse und Zustandsbeschreibungen zu unterscheiden. Dies kann daran liegen, dass zu wenige Beispiele für Zustandsbeschreibungen im Trainingsdatensatz vorhanden sind. Außerdem kommt das Verb „*to be*“ sowohl in Zustands-, als auch in Ereignisbeschreibungen oft vor, was zu Ungenauigkeiten bei der Vorhersage führen kann. Bezüglich Aktionsbeschreibungen und Ereignissen weist RF eine um 6 Prozent bessere Prädiktionsgüte als SVM auf. Bei der Erkennung von Aggregationen erreichen beide Klassifikatoren ähnliche Ergebnisse.

Negationserkennung

Die negativen Beschreibungen werden anhand von den durch `DepParse` bereitgestellten Abhängigkeitsbäumen erkannt. Der Negationsmodifikator (`neg`) wird bei allen negativen Beispielen im ganzen Datensatz korrekt erfasst, wodurch eine Präzision und Ausbeute von 1 erzielt werden. Die Negationswörter, die im Datensatz vorkommen, sind *„no“*, *„not“*, *„never“*, sowie die abgekürzte Form von *„not“* – *„n't“* in *„doesn't“* und *„shouldn't“*.

Zusammenführung der Ergebnisse

Da der entwickelte Agent negative Beschreibungen als eigene Klassen darstellt, ergeben sich insgesamt acht mögliche Kategorien: *positive_action*, *negative_action*, *positive_occurrence*,

negative_occurrence, *positive_aggregation*, *negative_aggregation*, *positive_state* und *negative_state*. In Tabelle 7.8 werden die Ergebnisse der Klassifikation mit Random Forest unter Berücksichtigung auf die Polarität der Klassen angegeben.

Aufgrund der Korrektheit der Negationserkennung ist kein Unterschied zwischen den Mikro-F_1-Durchschnittswerten in Tabelle 7.7 und Tabelle 7.8 zu beobachten. Allerdings führt die Unterteilung in acht Klassen zu einer Abnahme des Makro-F_1-Durchschnittswertes um 5 Prozent. Es fällt auf, dass bei der Vorhersage von Aktionsbeschreibungen, Ereignissen und Aggregationen ein höheres F_1-Maß für positive als für negative Beschreibungen erzielt wird. Außerdem gibt es eine ungleichmäßige Verteilung der Klassen wegen der kleinen Anzahl von negativen Beschreibungen. Das hat eine negative Auswirkung auf den gesamten Makro-F_1-Durchschnitt, da er nicht gewichtet ist und alle Klassen gleich behandelt.

7.3.2 Kreuzvalidierung auf dem gesamten Datensatz

Um eine Einschätzung der Leistungsfähigkeit der Klassifikatoren auf dem gesamten Datensatz zu erhalten, wird eine 10-fache Kreuzvalidierung eingesetzt. Dabei werden die in Abschnitt 7.3.1 bestimmten Parameter verwendet. In Tabelle 7.9 und Tabelle 7.10 werden die Mikro- und Makro-Durchschnittswerte dargestellt.

Mit einem Mikro-F_1-Durchschnitt von 0.90 und einem Makro-F_1-Durchschnitt von 0.85 weist die Stützvektormaschine die besten Ergebnisse auf. Ähnliche Ergebnisse zeigen sich bei der Anwendung von logistischer Regression und dem Random Forest-Klassifikator, die jeweils eine Genauigkeit von 0.89 erreichen. Wieder hat sich Bag-Of-N-Grams auf Zeichen als der am besten geeignete Ansatz zur Merkmalskonstruktion erwiesen. An zweiter Stelle hinsichtlich des Makro-F_1-Durchschnitts steht Doc2Vec. Hierfür wurde Doc2Vec vorab auf dem gesamten Datensatz trainiert. Obwohl dadurch kein generelles Modell erzeugt werden kann, stellt die hohe Leistung der Klassifikatoren mittels Doc2Vec klar, dass die Reihenfolge der Wörter von Bedeutung für die Klassifikation ist.

Da Anforderungen eine Vielzahl von Themen abdecken können, werden die Anforderungsbeschreibungen für verschiedene Zielprojekte mittels unterschiedlicher Fachwortschätze definiert. Aus diesem Grund ist es wichtig, dass die Modelle auf vielen Beispielen trainiert werden. Zusätzlich muss bei der Merkmalskonstruktion sichergestellt werden, dass möglichst wenig Informationen über die Semantik der Wörter verloren gehen. Einfache Bag-Of-Words- und TfIdf-Modelle sind für die Informationsgewinnung vieler Anwendungsbereiche nicht ausreichend, da sie die Semantik, die Wortreihenfolge und die Grammatik nicht berücksichtigen. Obwohl sie bei der Kreuzvalidierung auf dem Datensatz eine gute Leistung erbringen, ist eine schlechte Übertragbarkeit der Ergebnisse auf andere Kontexte zu erwarten. Die Semantik der Wörter kann durch Worteinbettungen ermittelt werden, die auf einem großen Korpus trainiert wurden. Da die vortrainierten Word2Vec- und GloVe-Modelle nur Wörter aus ihrem Korpus kennen, können sie nicht immer eine sinnvolle Vektorrepräsentation von Sätzen, die Fachausdrücke enthalten, generieren. Daher muss sichergestellt werden, dass Fachbegriffe, die häufig in Softwareanforderungen vorkommen, repräsentiert werden können, um die Erzeugung von Null-Vektoren zu vermeiden. Da in dieser Arbeit nur Verfahren zum überwachten maschinellen Lernen untersucht werden, wird für jeden Satz der Durchschnitt aller Wortvektoren berechnet, wodurch die Reihenfolge verloren geht. Eine alternative Möglichkeit ist es, die generierten Worteinbettungen als Eingabe für ein neuronales Netz zu verwenden, sodass die Wörter in der richtigen Reihenfolge eingelesen werden. Um eine Klassifikation mittels eines neuronalen Netzes einzusetzen, wird eine Vielzahl von Trainingsdaten benötigt.

Tabelle 7.9: Mikro-F_1-Durchschnitt (Genauigkeit), 10-fache Kreuzvalidierung

Technik	LR	SVM	RF	NB
BoW	0.87	0.87	0.84	0.75
N-Gramme: Zeichen	0.89	**0.90**	0.89	0.79
N-Gramme: Wörter	0.87	0.88	0.78	0.77
TfIdf	0.86	0.85	0.84	0.74
GloVe	0.83	0.83	0.72	-
GoogleNews	0.85	0.84	0.73	-
Doc2Vec	0.84	0.87	0.86	-

Tabelle 7.10: Makro-F_1-Durchschnitt, 10-fache Kreuzvalidierung

Technik	LR	SVM	RF	NB
BoW	0.79	0.80	0.63	0.62
N-Gramme: Zeichen	0.84	**0.85**	0.83	0.71
N-Gramme: Wörter	0.79	0.79	0.58	0.71
TfIdf	0.77	0.78	0.73	0.64
GloVe	0.74	0.74	0.46	-
GoogleNews	0.76	0.78	0.47	-
Doc2Vec	0.78	0.82	0.82	-

8 Zusammenfassung und Ausblick

Diese Arbeit beschäftigt sich im Rahmen des INDIRECT-Projektes mit der Klassifikation von Anforderungsbeschreibungen nach ihrer Semantik. Hierfür wurde ein Werkzeug entwickelt, das die semantische Funktion der Sätze und ihre Polarität vorhersagt. Bei der Analyse der Problemstellung wurden vier grundlegende Satzkategorien identifiziert: Aktionsbeschreibungen, Ereignisse, Zustandsbeschreibungen und Aggregationen. Dabei stellen Aktionsbeschreibungen eine aktive und Ereignisse eine passive Transformation dar. Aktive Transformationen sind Prozesse, bei denen eine Funktionalität der Objektentität von einer Subjektentität aufgerufen werden kann. Im Gegensatz dazu wird mit passiver Transformation eine Zustandsänderung der Subjektentität gemeint, die nicht unbedingt von einer anderen Entität aufgerufen wird. Zustandsbeschreibungen definieren mögliche Zustände und Eigenschaften des Zielsystems. Aggregationen repräsentieren Sätze, die eine Teil-Ganzes-Beziehung ausdrücken. Ein Bestandteil der Absicht einer Anforderung ist außerdem ihre Polarität. Durch Satznegationen können bestimmte Einschränkungen des Zielsystems definiert werden. Aus diesem Grund wurde zusätzlich für jeden Satz angegeben, ob er eine Negation enthält. In der Entwurfsphase wurde entschieden, die Klassifikation nach Satzkategorien mittels eines maschinellen Lernverfahrens umzusetzen und die Erkennung von Negationen mit Hilfe von Abhängigkeitsbäumen, die durch INDIRECT bereitgestellt werden, zu realisieren.

In der Evaluationsphase wurden diverse Verfahren zur multinomialen Klassifikation von Sätzen bewertet. Hierfür wurden Experimente mit einigen maschinellen Lernverfahren unter Verwendung von verschiedenen Techniken zur Datenrepräsentation durchgeführt. Zudem wurde ein Datensatz aufgebaut und manuell annotiert. Als Testdatensatz wurden 230 (Teil-)Sätze aus dem *iTrust*-Datensatz von CoEST verwendet. Die Trainingsdaten bestehen aus insgesamt 470 Beispielen, die aus vier anderen Projekten stammen. Als Erstes wurde anhand der Trainingsdaten versucht, durch eine Kreuzvalidierung optimale Werte für die Hyperparameter zu erhalten. Anschließend wurde untersucht, wie gut die optimierten Modelle die Kategorien der Beispiele im Testdatensatz vorhersagen können, um eine realistische Einschätzung der Leistungsfähigkeit der Klassifikatoren zu ermitteln. Die Verwendung von N-Grammen auf Zeichenbasis erzielte bei allen Klassifikatoren die besten Ergebnisse. Dabei übertrifft Random Forest die anderen Klassifikatoren, indem er eine Genauigkeit von 0.79 erreicht. Alle Verfahren erzielten einen besseren Mikro-F_1-Durchschnitt als Makro-F_1-Durchschnitt, was an der ungleichmäßigen Verteilung der Klassen liegt. Von allen Klassen hat `action` mit einem F_1-Wert von 0.86 die besten Ergebnisse erzielt. Die Vorhersage der `state`-Klasse hat die schlechteste Leistung erbracht, was auf den Mangel an

Beispielen in den Trainingsdaten zurückzuführen ist. Des Weiteren wurden die Leistungsfähigkeiten der verschiedenen Verfahren mittels einer Kreuzvalidierung über den gesamten Datensatz ausgewertet. Dabei wurden deutlich bessere Ergebnisse als beim Testen auf den *iTrust*-Datensatz erzielt. Als bester Klassifikator hat sich hier die Stützvektormaschine mit einem F_1-Maß von 0.90 erwiesen. Der Random Forest-Klassifikator und die logistische Regression erzielten mit einem F_1-Maß von 0.89 die zweitbesten Ergebnisse. Dabei zeigte sich, dass das vorgestellte Verfahren grundsätzlich in der Lage ist, Satzkategorien zu erkennen, jedoch Raum für Verbesserungen lässt.

Der Fokus dieser Arbeit wurde auf die Klassifikation von atomaren Sätzen gelegt, wobei der verwendete Datensatz sowohl einfache Sätze, als auch Teile von zusammengesetzten Sätzen beinhaltet. Jedoch wurde die Erkennung von Satzteilen, die isoliert einen Sachverhalt beschreiben, vom entwickelten Agenten nicht betrachtet. Die Unterteilung komplexer Sätze kann bei einer Weiterentwicklung des Agenten berücksichtigt werden, um bessere Übertragbarkeit des Verfahrens zu erreichen. Des Weiteren kann eine Anreicherung des Klassifikators mit Merkmalen wie semantischen Rollen und Abhängigkeitsbäumen zu einer Verbesserung der Ergebnisse führen. Dadurch lassen sich viele Beziehungen zwischen den Wörtern untersuchen, die zur Erkennung der entsprechenden Kategorien hilfreich sein können.

Um die Güte der Vorhersage der Klassen zu steigern, wird eine Vielzahl an Trainingsbeispielen benötigt. Dabei ist eine möglichst gleichmäßige Verteilung der Klassen zu empfehlen. Sollte in Zukunft ein großer Datensatz vorliegen, könnte die Verwendung eines neuronalen Netzes zu besseren Ergebnissen führen, wobei als Eingabe Worteinbettungen genutzt werden. Dadurch kann eine sinnvolle Repräsentation der Bedeutung von Wörtern erfasst werden, indem nicht nur die Semantik, sondern auch die Wortreihenfolge berücksichtigt wird.

Literaturverzeichnis

[ABZ19] Alhoshan, Waad ; Batista-Navarro, Riza ; Zhao, Liping: Using Frame Embeddings to Identify Semantically Related Software Requirements. (2019), S. 9 (zitiert auf Seite 16).

[ARA+06] Asfour, T. ; Regenstein, K. ; Azad, P. ; Schroder, J. ; Bierbaum, A. ; Vahrenkamp, N. ; Dillmann, R.: ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control. In: *2006 6th IEEE-RAS International Conference on Humanoid Robots*. University of Genova, Genova, Italy : IEEE, Dezember 2006. – ISBN 978–1–4244–0199–4 978–1–4244–0200–7, 169–175 (zitiert auf Seite 13).

[BH15] Bühne, Stan ; Herrmann, Andrea: Handbuch Requirements Management nach IREB Standard. (2015) (zitiert auf Seite 7).

[CEE+09] Carstensen, Kai-Uwe (Hrsg.) ; Ebert, Christian (Hrsg.) ; Ebert, Cornelia (Hrsg.) ; Jekat, Susanne (Hrsg.) ; Klabunde, Ralf (Hrsg.) ; Langer, Hagen (Hrsg.): *Computerlinguistik und Sprachtechnologie: Eine Einführung*. 3. Heidelberg : Spektrum, 2009. – ISBN 978–3–8274–2023–7 (zitiert auf den Seiten 3 und 4).

[CHMLP07] Cleland-Huang, Jane ; Mazrouee, Sepideh ; Liguo, Huang ; Port, Dan: *nfr*. `http://dx.doi.org/10.5281/zenodo.268542`. Version: März 2007 (zitiert auf Seite 41).

[CMJM10] Cer, Daniel ; Marneffe, Marie-Catherine de ; Jurafsky, Daniel ; Manning, Christopher D.: Parsing to Stanford Dependencies: Trade-offs between speed and accuracy, 2010, S. 5 (zitiert auf Seite 32).

[CWB+11] Collobert, Ronan ; Weston, Jason ; Bottou, Leon ; Karlen, Michael ; Kavukcuoglu, Koray ; Kuksa, Pavel: Natural Language Processing (almost) from Scratch, 2011. – arXiv: 1103.0398 (zitiert auf Seite 32).

[Cyc] Cycorp, Inc.: *ResearchCyc*. `https://www.cyc.com/researchcyc/` (zitiert auf Seite 13).

[Dav93] Davis, Alan M.: *Software Requirements: Objects, Functions, and States*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1993. – ISBN 0–13–805763–X (zitiert auf Seite 8).

[Dow91] Dowty, David: Thematic Proto-roles and Argument Selection, 1991, S. 547–619 (zitiert auf Seite 26).

[Fil68] Fillmore, Charles J.: The case for case. In: Bach, E. (Hrsg.) ; Harms, R. (Hrsg.) ; Holt, Rinehart & Winston (Veranst.): *Universals in Linguistic Theory*. New York : Holt, Rinehart & Winston, 1968 (zitiert auf Seite 25).

[Fla12] Flach, Peter: *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012. http://dx.doi.org/10.1017/CBO9780511973000. http://dx.doi.org/10.1017/CBO9780511973000 (zitiert auf Seite 10).

[Gel10] Gelhausen, Tom: *Modellextraktion aus natürlichen Sprachen: Eine Methode zur systematischen Erstellung von Domänenmodellen*. Karlsruhe, KIT Scientific Publishing, PhD Thesis, April 2010. http://digbib.ubka.uni-karlsruhe.de/volltexte/1000019366. – bibtex: Gelhausen2010 (zitiert auf Seite 32).

[Gli05] Glinz, Martin: Rethinking the Notion of Non-Functional Requirements. In: *in Proceedings of the Third World Congress for Software Quality (3WCSQ'05*, 2005, S. 55–64 (zitiert auf Seite 1).

[Hau14] Hausser, Roland R.: *Foundations of Computational Linguistics: Human-Computer Communication in Natural Language*. Third edition. Heidelberg : Springer, 2014. – ISBN 978–3–642–41430–5. – OCLC: ocn870509088 (zitiert auf den Seiten 3 und 4).

[Hey19] Hey, Tobias: Intent-driven Requirements-to-Code Traceability. In: *ICSE'19 DS: Doctoral Symposium of the 2019 IEEE/ACM 41th International Conference on Software Engineering*, 2019, S. 4 (zitiert auf Seite 13).

[Hof13] Hoffmann, Axel: Anforderungsmuster im Requirements Engineering, 2013, S. 58 (zitiert auf Seite 24).

[HTF01] Hastie, T. ; Tibshirani, R. ; Friedman, J.: The Elements of Statistical Learning. (2001), S. 764 (zitiert auf Seite 50).

[JM18] Jurafsky, Daniel ; Martin, James H.: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2018 https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf (zitiert auf den Seiten 5, 6, 10 und 48).

[Kim14] Kim, Yoon: Convolutional Neural Networks for Sentence Classification, 2014 (zitiert auf Seite 18).

[KKKS17] Khurana, Diksha ; Koli, Aditya ; Khatter, Kiran ; Singh, Sukhdev: Natural Language Processing: State of The Art, Current Trends and Challenges. (2017), 08 (zitiert auf Seite 3).

[KM16] Komninos, Alexandros ; Manandhar, Suresh: Dependency Based Embeddings for Sentence Classification Tasks. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California : Association for Computational Linguistics, Juni 2016, 1490–1500 (zitiert auf Seite 18).

[KM17] Kurtanovic, Zijad ; Maalej, Walid: Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. Lisbon, Portugal : IEEE, September 2017. – ISBN 978–1–5386–3191–1, 490–495 (zitiert auf den Seiten 16 und 35).

[KMA06] Khoo, Anthony ; Marom, Yuval ; Albrecht, David: Experiments with Sentence Classification. In: *Proceedings of the Australasian Language Technology Workshop 2006*. Sydney, Australia, November 2006, S. 18–25 (zitiert auf den Seiten 18 und 35).

[Koc15] Kocybik, Markus: *Projektion von gesprochener Sprache auf eine Handlungsrepräsentation*, Karlsruher Institut für Technologie (KIT) – IPD Tichy, Bachelor's Thesis, Juli 2015. `https://code.ipd.kit.edu/weigelt/parse/wikis/Theses/kocybik_ba` (zitiert auf Seite 14).

[LSNC10] Llorens, Hector ; Saquete, Estela ; Navarro-Colorado, Borja: TimeML events recognition and classification: learning CRF models with semantic roles. (2010), S. 9 (zitiert auf den Seiten 19, 24 und 25).

[MCCD13] Mikolov, Tomas ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: Efficient Estimation of Word Representations in Vector Space. In: *arXiv:1301.3781 [cs]* (2013), Januar. `http://arxiv.org/abs/1301.3781`. – arXiv: 1301.3781 (zitiert auf Seite 34).

[Mil95] Miller, George A.: WordNet: A Lexical Database for English. In: *Commun. ACM* 38 (1995), November, Nr. 11, 39–41. `http://dx.doi.org/10.1145/219717.219748`. – DOI 10.1145/219717.219748. – ISSN 0001–0782 (zitiert auf Seite 13).

[MRS09] Manning, Christopher ; Raghavan, Prabhakar ; Schuetze, Hinrich: Introduction to Information Retrieval. (2009), S. 581 (zitiert auf Seite 9).

[MSM93] Marcus, Mitchell ; Santorini, Beatrice ; Marcinkiewicz, Mary A.: Building a large annotated corpus of English: The Penn Treebank. (1993) (zitiert auf Seite 4).

[NMG+16] Nivre, Joakim ; Marneffe, Marie-Catherine de ; Ginter, Filip ; Goldberg, Yoav ; Hajic, Jan ; Manning, Christopher D. ; McDonald, Ryan ; Petrov, Slav ; Pyysalo, Sampo ; Silveira, Natalia ; Tsarfaty, Reut ; Zeman, Daniel: Universal Dependencies v1: A Multilingual Treebank Collection, 2016, S. 8 (zitiert auf Seite 6).

[OGZ18] Ouyang, Xi ; Gu, Kang ; Zhou, Pan: Spatial Pyramid Pooling Mechanism in 3D Convolutional Network for Sentence-Level Classification. In: *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 26 (2018), November, Nr. 11, 2167–2179. `http://dx.doi.org/10.1109/TASLP.2018.2852502`. – DOI 10.1109/TASLP.2018.2852502. – ISSN 2329–9290 (zitiert auf Seite 20).

[PR15] Pohl, K. ; Rupp, C.: *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*. dpunkt.verlag, 2015 `https://books.google.de/books?id=WdZNDAAAQBAJ`. – ISBN 978–3–86491–674–8 (zitiert auf Seite 7).

[PS12] Pustejovsky, J. ; Stubbs, A.: *Natural Language Annotation for Machine Learning: A Guide to Corpus-Building for Applications*. O'Reilly Media, 2012. – ISBN 978–1–4493–5976–8 (zitiert auf den Seiten 9 und 10).

[PSM14] Pennington, Jeffrey ; Socher, Richard ; Manning, Christopher D.: GloVe: Global Vectors for Word Representation. In: *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, 1532–1543 (zitiert auf Seite 44).

[RKSW14] Riaz, M. ; King, J. ; Slankas, J. ; Williams, L.: Hidden in plain sight: Automatically identifying security requirements from natural language artifacts. In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 2014, S. 183–192 (zitiert auf Seite 15).

[RMD18] Rago, Alejandro ; Marcos, Claudia ; Diaz-Pace, J. A.: Using Semantic Roles to Improve Text Classification in the Requirements Domain. In:

Language Resources and Evaluation 52 (2018), September, Nr. 3, S. 801–837. http://dx.doi.org/10.1007/s10579-017-9406-7. – DOI 10.1007/s10579–017–9406–7. – ISSN 1574–0218 (zitiert auf den Seiten 17 und 35).

[sNZ18] SHAUKAT, Zain ; NASEEM, Rashid ; ZUBAIR, Muhammad: *Software Requirement Risk Prediction Dataset*. http://dx.doi.org/10.5281/zenodo.1209601. Version: März 2018 (zitiert auf Seite 41).

[VHH+12] VERSTEEGEN, G. ; HESSELER, A. ; HOOD, C. ; MISSLING, C. ; STÜCKA, R.: *Anforderungsmanagement: Formale Prozesse, Praxiserfahrungen, Einführungsstrategien und Toolauswahl*. Springer Berlin Heidelberg, 2012 (Xpert.press). https://books.google.de/books?id=X5IlngEACAAJ. – ISBN 978–3–642–62388–2. – bibtex*[lccn=2005433677] (zitiert auf Seite 7).

[VR11] VLAS, R. ; ROBINSON, W. N.: A Rule-Based Natural Language Technique for Requirements Discovery and Classification in Open-Source Software Development Projects. In: *2011 44th Hawaii International Conference on System Sciences*, 2011, S. 1–10 (zitiert auf Seite 15).

[WHS18] WEIGELT, S. ; HEY, T. ; STEURER, V.: Detection of Conditionals in Spoken Utterances. In: *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, 2018, S. 85–92 (zitiert auf Seite 27).

[WHT17] WEIGELT, Sebastian ; HEY, Tobias ; TICHY, Walter F.: Context Model Acquisition from Spoken Utterances. In: *The 29th International Conference on Software Engineering & Knowledge Engineering,*. Pittsburgh, PA, Juli 2017, S. 201–206 (zitiert auf Seite 13).

Anhang

A Markierungssätze

A.1 Penn Treebank Tagset

A.1.1 Offene Klassenkategorien

Tag	Description
FW	Foreign word
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present

A.1.2 Geschlossene Klassenkategorien

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
IN	Preposition or subordinating conjunction

LS	List item marker
MD	Modal
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

A.2 Abhängigkeitsbezeichnungen

Tag	Description
acl	finite and non-finite clausal modifier
acomp	adjectival complement
advcl	adverbial clause modifier
advmod	adverbial modifier
agent	agent
amod	adjectival modifier
appos	appositional modifier
attr	attribute
aux	auxiliary
auxpass	auxiliary (passive)
case	case marker
cc	coordinating conjunction
ccomp	clausal complement
compound	compound nouns/numbers
complm	complementizer
conj	conjunct
csubj	clausal subject
csubjpass	clausal subject (passive)
dative	dative case
dep	unclassified dependent
det	determiner
dobj	direct object
expl	expletive
hmod	modifier in hyphenation
hyph	hyphen
infmod	infinitival modifier
intj	interjection
iobj	indirect object
mark	marker
meta	meta modifier
neg	negation modifier
nummod	numeric modifiers
nmod	modifier of nominal

nn	noun compound modifier
npadvmod	noun phrase as advmod
nsubj	nominal subject
nsubjpass	nominal subject (passive)
num	numeric modifier
number	number compound modifier
oprd	object predicate
parataxis	parataxis
partmod	participial modifier
pcomp	complement of a preposition
pobj	object of a preposition
poss	possession modifier
possessive	possessive modifier
preconj	pre-correlative conjunction
predet	predeterminer
prep	prepositional modifier
prt	particle
punct	punctuation
quantmod	quantifier phrase modifier
rcmod	relative clause modifier
relcl	relative clause modifiers
root	root
xcomp	open clausal complement

B Tests auf einzelne Projekte

B.1 NFR

Tabelle B.4: Mikro-F_1-Durchschnitt (Genauigkeit), Tests auf dem *NFR*-Datensatz

Technik	LR	SVM	RF	NB
BoW	0.89	0.90	0.87	0.81
N-Gramme: Zeichen	0.92	**0.93**	0.91	0.87
N-Gramme: Wörter	0.88	0.90	0.86	0.82
TfIdf	0.89	0.89	0.89	0.81
GloVe	0.80	0.79	0.77	-
GoogleNews	0.83	0.79	0.77	-
Doc2Vec	0.79	0.80	0.80	-

Tabelle B.5: Makro-F_1-Durchschnitt, Tests auf dem *NFR*-Datensatz

Technik	LR	SVM	RF	NB
BoW	0.62	0.66	0.63	0.50
N-Gramme: Zeichen	0.83	**0.84**	0.76	0.72
N-Gramme: Wörter	0.61	0.66	0.55	0.51
TfIdf	0.71	0.76	0.65	0.59
GloVe	0.58	0.58	0.38	-
GoogleNews	0.65	0.59	0.38	-
Doc2Vec	0.48	0.51	0.52	-

B.2 Risk prediction

Tabelle B.6: Mikro-F_1-Durchschnitt (Genauigkeit), Tests auf dem *Risk prediction*-Datensatz

Technik	LR	SVM	RF	NB
BoW	0.87	0.86	0.78	0.71
N-Gramme: Zeichen	0.85	0.84	0.87	0.81
N-Gramme: Wörter	0.86	0.86	0.74	0.73
TfIdf	0.85	**0.88**	0.84	0.70
GloVe	0.85	0.84	0.68	-
GoogleNews	0.85	0.85	0.69	-
Doc2Vec	0.73	0.78	0.77	-

Tabelle B.7: Makro-F_1-Durchschnitt, Tests auf dem *Risk prediction*-Datensatz

Technik	LR	SVM	RF	NB
BoW	0.76	0.74	0.60	0.48
N-Gramme: Zeichen	0.75	0.72	**0.78**	0.57
N-Gramme: Wörter	0.76	0.76	0.49	0.50
TfIdf	0.73	**0.79**	0.73	0.48
GloVe	0.72	0.71	0.36	-
GoogleNews	0.75	0.76	0.38	-
Doc2Vec	0.53	0.57	0.56	-

B.3 Warc

Tabelle B.8: Mikro-F_1-Durchschnitt (Genauigkeit), Tests auf dem *Warc*-Datensatz

Technik	LR	SVM	RF	NB
BoW	0.74	0.74	0.69	0.59
N-Gramme: Zeichen	**0.81**	0.77	0.77	0.61
N-Gramme: Wörter	0.73	0.76	0.69	0.60
TfIdf	0.74	0.69	0.73	0.54
GloVe	0.66	0.63	0.56	-
GoogleNews	0.79	0.70	0.60	-
Doc2Vec	0.57	0.59	0.60	-

Tabelle B.9: Makro-F_1-Durchschnitt, Tests auf dem *Warc*-Datensatz

Technik	LR	SVM	RF	NB
BoW	0.72	0.72	0.63	0.55
N-Gramme: Zeichen	**0.82**	0.77	0.71	0.59
N-Gramme: Wörter	0.68	0.72	0.56	0.56
TfIdf	0.70	0.65	0.68	0.50
GloVe	0.58	0.56	0.33	-
GoogleNews	0.73	0.64	0.36	-
Doc2Vec	0.52	0.54	0.53	-

B.4 IceBreaker

Tabelle B.10: Mikro-F_1-Durchschnitt (Genauigkeit), Tests auf dem *IceBreaker*-Datensatz

Technik	LR	SVM	RF	NB
BoW	0.86	0.82	0.85	0.62
N-Gramme: Zeichen	**0.96**	0.95	0.91	0.76
N-Gramme: Wörter	0.90	0.90	0.86	0.73
TfIdf	0.83	0.83	0.84	0.58
GloVe	0.8	0.81	0.76	-
GoogleNews	0.90	0.85	0.74	-
Doc2Vec	0.82	0.83	0.78	-

Tabelle B.11: Makro-F_1-Durchschnitt, Tests auf dem *IceBreaker*-Datensatz

Technik	LR	SVM	RF	NB
BoW	0.80	0.76	0.74	0.40
N-Gramme: Zeichen	**0.91**	0.88	0.82	0.66
N-Gramme: Wörter	0.81	0.81	0.66	0.68
TfIdf	0.75	0.75	0.73	0.48
GloVe	0.61	0.65	0.4	-
GoogleNews	0.86	0.82	0.39	-
Doc2Vec	0.51	0.64	0.41	-