Karlsruhe Institute of Technology

Ontologie-unterstützte Klassifikation von Software-Anforderungen

Masterarbeit
von

Vitali Chaiko

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

Erstgutachter:	Prof. Dr. Walter F. Tichy
Zweitgutachter:	Prof. Dr. Ralf H. Reussner
Betreuender Mitarbeiter:	Dipl.-Inform. Sebastian Weigelt
Zweiter betr. Mitarbeiter:	Dr. Michael Dorna (Robert Bosch GmbH)

Bearbeitungszeit: 01.05.2018 – 31.10.2018

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

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

Karlsruhe, 31.10.2018

. . . . . .

(**Vitali Chaiko**)

Kurzfassung

Die Analyse der Lastenhefte für die Planung der Software bringt einen erheblichen manuellen Aufwand mit sich. Bei Robert Bosch GmbH werden die Anforderungen aus den Lastenheften der Kunden auf eine V-Prozessmodell-Datenbank abgebildet. Diese Datenbank besteht aus den sogenannten internen Anforderungen, die Richtlinien für Hardware- und Softwareentwicklung enthalten. Jede Kundenanforderung muss von den Mitarbeitern manuell auf eine oder mehrere interne Anforderungen abgebildet werden.

In Rahmen dieser Arbeit wurde ein automatisiertes Verfahren entwickelt, welches den Mitarbeiter bei dem Abbildungsprozess unterstützen kann. Dafür wurde aus den Textdaten der Kundenanforderungen eine Ontologie automatisch generiert, die Fachbegriffe und ihre Beziehungen enthält. Aus dieser Ontologie wurden Merkmale erzeugt, welche mit einem unüberwachten Verfahren des maschinellen Lernens, nämlich hiearchisches Clustering gruppiert wurden. Dadurch war es möglich eine neue Kundenanforderung in ein bestehendes Cluster einzuordnen und, basierend auf die Kundenanforderungen in dem Cluster, Vorschläge für die zutreffenden internen Anforderungen zu erhalten.

Um die entstandene Ontologie zu evaluieren, wurde diese auf falsch extrahierte Konzepte und Beziehungen überprüft. 16% der Konzepte und 24% der Relationen erwiesen sich als falsch. Die Voraussage der Vorschläge erreichte ein F-Maß von 58%, bei den Evaluationsmetriken Präzision@5 und Ausbeute@5.

Inhaltsverzeichnis

1. Einleitung

Bei der Entwicklung neuer Softwaresysteme im Automobilbereich ist die Anzahl der Anforderungen in den letzten Jahren rasant gewachsen. Besonders im Zuge der Digitalisierung erwartet man heutzutage vieles von einem Fahrzeug, sei es ein automatisches Bremssystem oder ein Einparkassistent. All diese Anforderungen müssen in einem Lastenheft festgehalten werden und das Lastenheft für die Planung der Software analysiert werden. Dies bringt einen erheblichen manuellen Aufwand mit sich.

Beispielsweise werden bei der Robert Bosch GmbH die Lastenhefte der Kunden auf eine V-Prozessmodell-Datenbank abgebildet. Diese Datenbank besteht aus sogenannten internen Anforderungen, die Richtlinien für Hardware- und Softwareentwicklung enthalten. Jede Kundenanforderung muss von den Mitarbeitern manuell auf eine oder mehrere interne Anforderungen abgebildet werden. Dabei schaut sich der zuständige Mitarbeiter den Textinhalt und die Metainformationen der Kundenanforderung an und durchsucht darauf die Datenbank nach einer passenden internen Anforderung. Offensichtlich ist dieser Prozess fehlerbehaftet und redundant. Auch nimmt er viel Zeit in Anspruch und sorgt für beachtliche Kosten. Ein solcher Prozess bietet eine gute Grundlage für die Optimierung. Diese Masterarbeit befasst sich mit der Automatisierung der Abbildung der Kundenanforderungen auf die internen Anforderungen.

Das Ziel ist die Erstellung von einem System, welches einen Mitarbeiter bei dem Abbildungsprozess einer neuen Anforderung unterstützen kann. Da die Algorithmen der künstlichen Intelligenz heutzutage in Bereichen der Textklassifikation signifikante Erfolge vorweisen, soll ihre Eignung als automatische Abbildungsunterstützung evaluiert werden. So kann ein solcher Algorithmus darauf trainiert werden, für die Informationen aus einer bestimmten Kundenanforderung eine passende interne Anforderung vorauszusagen.

Um einen solchen Algorithmus verwenden zu können, müssen aus den Anforderungen sinnvolle Eingabedaten, auch Merkmale genannt, erzeugt werden. Diese Merkmale dienen dazu, dem Algorithmus Muster beizubringen, anhand deren eine Voraussage vorgenommen wird. Die Wahl solcher Merkmale ist entscheidend für die Qualität der Voraussage. Im Rahmen dieser Arbeit werden Verfahren entwickelt und evaluiert, welche die Erzeugung von Merkmalen aus den Anforderungen behandeln. Ein besonderer Fokus wird auf die Erstellung einer Wissensbasis aus den Textinformationen der Anforderungen, in Form einer Ontologie, gelegt. Mithilfe einer solchen Ontologie können mögliche Merkmale identifiziert und für eine Verwendung in einem Klassifikationsalgorithmus aufbereitet werden.

2. Grundlagen

In diesem Abschnitt werden die Themen und Technologien erklärt, welche signifikant für das Verständnis dieser Arbeit sind. Es werden Grundlagenkenntnisse in Bereichen Anforderungsmanagement, maschinelles Lernen und Computerlinguistik vermittelt.

2.1 Software-Anforderungen

In der Softwareentwicklung stellen die Anforderungen die formale Form der Kommunikation zwischen einem Kunden und einem Auftragnehmer dar. In einer Anforderung beschreibt der Kunde eine Bedingung, die das Softwaresystem erfüllen muss, um ein bestimmtes Ziel zu erreichen. Die Beschreibung erfolgt meist im Fließtext. Des Weiteren existieren Techniken wie Satzschablonen, um die Klarheit einer Anforderung zu gewährleisten. Ein Beispiel für eine Satzschablone ist „Das System soll X sein“, wobei X mit einer Bedingung vom Kunden ausgefüllt wird, wie „24 Stunden am Tag erreichbar“.

Heute ist Anforderungsmanagement eine Wissenschaft mit definierten Standards. Software-Anforderungen können in mehrere Arten aufgeteilt werden. Es gibt die funktionalen Anforderungen, die die Funktionsweise des Systems beschreiben, wie beispielsweise „Das System berechnet die Summe der Ableitungen bis auf 4 Nachkommastellen genau“. Des Weiteren gibt es die nicht-funktionalen Anforderungen, die die Qualität des Systems definieren. Ein Beispiel hierfür ist ”Das System soll für gelegentliche Benutzer einfach zu benutzen sein”. Auch können die funktionalen und nicht-funktionalen Software-Anforderungen weiter unterteilt werden, wobei mehrere Modelle der Aufteilung existieren. Beispielsweise können sich die funktionalen Anforderungen in Berechnung-Anforderungen, welche die mathematischen Wege der Berechnung definieren, und Datenmanipulation-Anforderungen, welche die Änderungen an der Datenbank beschreiben, aufteilen.

Viele Projekte scheitern aufgrund von falsch spezifizierten oder fehlerhaften Anforderungen. Bekannte Beispiele aus dem deutschen Raum sind das Bewerbungsportal hochschulstart.de und das Mauterfassungssystem Toll Collect.

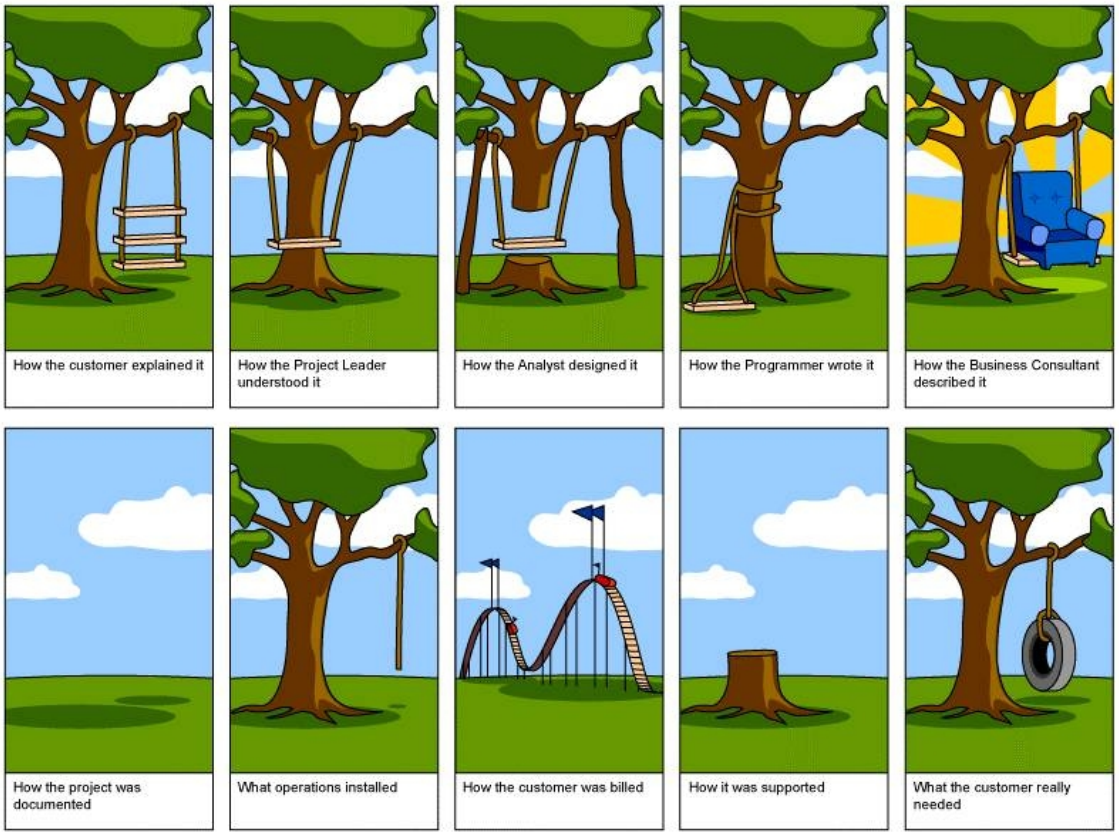


Abbildung 2.1: How Projects Really Work[1]

Die Schwierigkeiten, die während der Definition der Anforderungen und Kommunikation auftreten, können durch die metaphorische Bildreihe in der Abbildung 2.1 verdeutlicht werden. So werden die Anforderungen von jedem Teilnehmer der Kommunikation subjektiv aufgefasst und dadurch kann es zu gravierenden Missverständnissen kommen. [ER03]

2.2 V-Prozess-Modell

Das V-Prozess-Modell in der Abbildung 2.2 ist ein Vorgehensmodell. Es unterteilt den Prozess der Softwareentwicklung in Phasen und ist als eine Richtlinie während der Umsetzung eines Softwareprojekts zu betrachten. Es zeigt außerdem, wie die einzelnen Phasen zueinander in Beziehung stehen. Auf der linken Seite werden die Phasen von der Anforderungsdefinition bis zur Implementierung der Software festgelegt. Auf der rechten Seite werden die Tests festgelegt, die für den Einsatz der Software benötigt werden. Jeder der Tests bezieht sich auf eine Phase des Prozesses von der linken Seite. Zum Beispiel wird die Software-Architektur mit Unit-Tests getestet, während die Anforderungsdefinition mit einem Abnahmetest seitens des Kunden geprüft wird. Die Definition der Software-Anforderungen finden dabei links oben im Modell statt. Danach werden die Anforderungen verfeinert, bis schließlich die Anforderungen für den exakten Software-Entwurf spezifiziert sind und die Entwicklung stattfinden kann. [ER03]

2.3 Maschinelles Lernen

Als maschinelles Lernen werden Algorithmen bezeichnet, die aus Daten Muster bzw. Gesetzmäßigkeiten extrahieren und diese auf unbekannte Daten anwenden, um sie zu beur-

[1] Quelle: `http://www.projectcartoon.com/cartoon/2`, zuletzt besucht am 23.10.2018

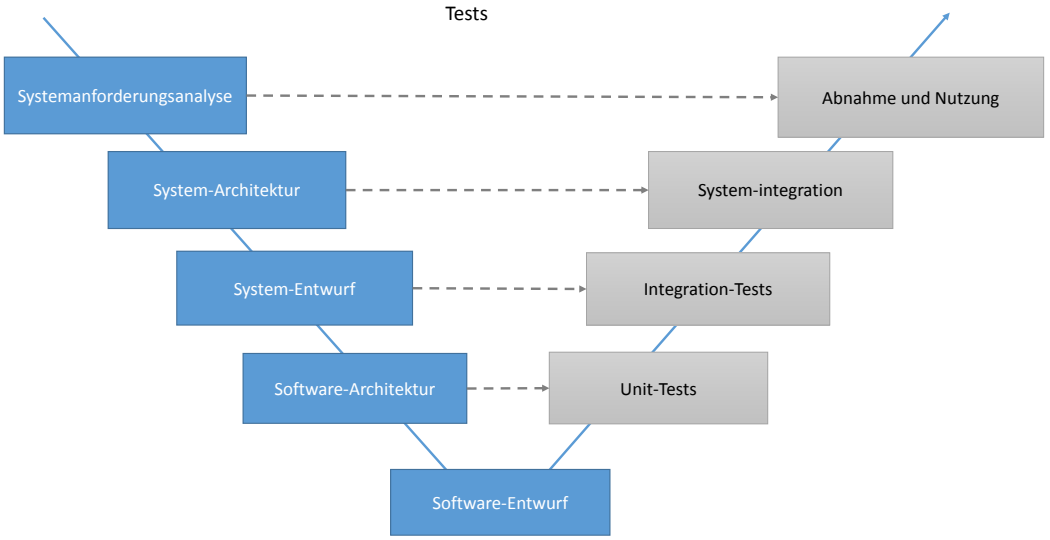


Abbildung 2.2: V-Prozess-Modell [ER03]

teilen. Das Extrahieren der Muster aus den Lerndaten wird als Trainingsphase bezeichnet, während die Anwendung auf neue Daten als Klassifikationsphase bezeichnet wird. Je mehr Lerndaten man zu Verfügung hat, desto besser sind in der Regel die Ergebnisse der Klassifikationsphase. Beispielsweise verwendet man maschinelles Lernen bei der Kreditbeurteilung. Als Lerndaten verwendet man die Merkmale der Personen wie das Alter, Einkommen und den Wohnort. Zu jeder Person gibt es in den Lerndaten die Angabe, ob ein Kredit gewährt wurde oder nicht. Diese Angabe wird als Klasse bezeichnet und wird von dem Algorithmus in der Klassifikationsphase anschließend vorausgesagt. Bei einer Voraussage wendet der Algorithmus die zuvor gelernten Muster an. Beispielsweise wird aus den Lerndaten statistisch ein Entscheidungsbaum erzeugt, welches das Alter, Einkommen und den Wohnort als Entscheidungszweige für die Kreditvergabe enthält. Ein Beispiel für den Entscheidungsbaum befindet sich in der Abbildung 2.3. Ein Zweig wird dabei aus den Daten ausgerechnet. Falls man beispielsweise bei Mehrheit der Personen über 30 Kreditwürdigkeit vorfindet, wird daraus ein Entscheidungszweig erzeugt.

Wenn man das Alter, Einkommen und den Wohnort einer bisher unbekannten Person erfasst, kann man voraussagen, ob diese Person einen Kredit bekommen soll oder nicht. Dieses Beispiel gehört zum sogenannten überwachten maschinellen Lernen. Beim überwachten maschinellen Lernen ist die Klasse in den Lerndaten bekannt. Die andere Art ist unüberwachtes Lernen, dort sind die Klassen nicht bekannt und werden von dem Algorithmus vor der Trainingsphase ermittelt, beispielsweise durch Clustering-Verfahren. [Mur12]

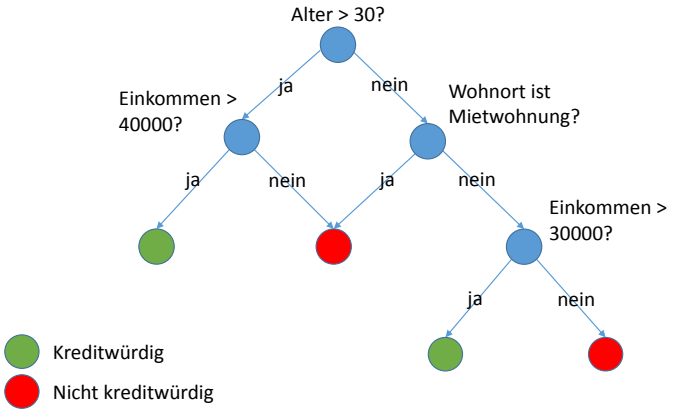


Abbildung 2.3: Entscheidungsbaum für die Kreditvergabe

2.3.1 Naïve-Bayes-Klassifikator

Der Naïve-Bayes-Klassifikator ist ein überwachtes Verfahren des maschinellen Lernens. Das Verfahren geht von der naiven Annahme aus, dass jedes Merkmal in den Lerndaten nur von der Klassenzugehörigkeit abhängt. Im obigen Beispiel mit den Kreditdaten bedeutet es, dass die Merkmale Alter, Einkommen und Wohnort voneinander unabhängig anzunehmen sind. Um eine Voraussage zu treffen, wird die Bayes-Regel angewendet. Die Bayes-Regel ist in der Gleichung 2.1 dargestellt.

$$P(Klasse_i \mid Merkmal_i) = \frac{P(Merkmal_i \mid Klasse_i)\, P(Klasse_i)}{P(Merkmal_i)} \tag{2.1}$$

Der rechte Teil der Gleichung lässt sich dabei aus den Lerndaten abzählen. Zum Beispiel ist die Wahrscheinlichkeit für $P(Merkmal_i \mid Klasse_i)$ gleich der Anzahl der Vorkommnisse der $Klasse_i$ mit dem Merkmal $Merkmal_i$, geteilt durch die Gesamtmenge der Lerndaten. Durch die naive Annahme ist der Naïve-Bayes-Klassifikator schnell und bei einfachen Problemen gut einsetzbar. [Mur12]

2.3.2 Random-Forest-Klassifikator

Der Random-Forest-Klassifikator ist ein überwachtes Verfahren des maschinellen Lernens. Es basiert auf den Entscheidungsbäumen, die in der Abbildung 2.3 gezeigt wurden. Bei dem Random-Forest-Klassifikator werden eine zuvor festgelegte Anzahl von Entscheidungsbäumen aus den Lerndaten erzeugt. Als Vorgehen der Erzeugung können die einzelnen Merkmale als Zweige des Baums zufällig gewählt werden. Um eine Voraussage zu treffen, werden die Ergebnisse aller erzeugten Entscheidungsbäume gesammelt und die finale Klasse durch einen Mehrheitsentscheid ausgewählt. Der Random-Forest-Klassifikator eignet sich gut für Datenmengen mit vielen Merkmalen, wie zum Beispiel Textmengen. [Mur12]

2.3.3 Neuronales Netzwerk

Das neuronale Netzwerk ist dem menschlichen Gehirn nachempfunden und hat einen Graphenaufbau. In der Abbildung 2.4 sieht man den Aufbau eines dreischichtigen neuronalen

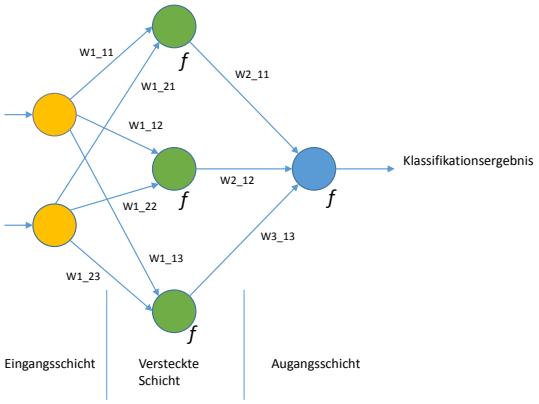


Abbildung 2.4: Neronales Netzwerk

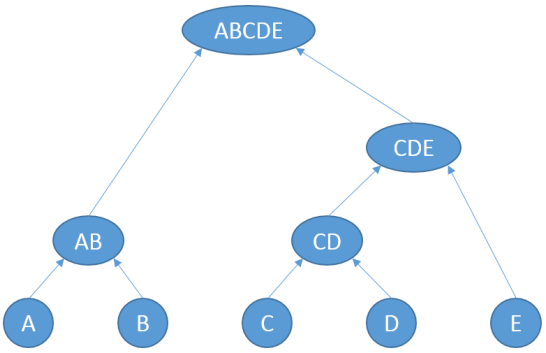
Netzwerks. Jede Schicht besteht dabei aus mehreren Knoten (Neuronen genannt), welche durch gewichtete Kanten miteinander verbunden sind. Die Neuronen von der Eingangsschicht nehmen dabei die Merkmale auf, beispielsweise könnte das erste Neuron das Merkmal Alter einer Person aufnehmen und das zweite das Merkmal Einkommen. Die Merkmale x_i werden mit den Kanten w_i gewichtet und an die versteckte Schicht geschickt, wo jedes Neuron eine Aktivierungsfunktion f implementiert. Ein einfaches Beispiel für die Aktivierungsfunktion befindet sich in der Gleichung 2.2.

$$f(x) = \begin{cases} 0 & \sum_i w_i x_i \leq t \\ 1 & \sum_i w_i x_i > t \end{cases} \tag{2.2}$$

Diese Aktivierungsfunktion gibt eine 1 aus, falls das Produkt aus allen eingehenden Merkmalen und Gewichten über einem festgelegten Schwellenwert t liegt, sonst 0. Das Ergebnis der Aktivierungsfunktion jedes Neurons in der versteckten Schicht wird mit seinen Ausgangskanten gewichtet und an die Ausgabeschicht weitergeleitet. Die Neuronen der Ausgabeschicht geben das endgültige Ergebnis der Klassifikation aus. Die Trainingsphase eines neuronalen Netzwerks besteht darin, die Kantengewichte anhand der Lerndaten zu ermitteln. Dafür existieren mehrere Verfahren, beispielsweise die Rückpropagierung. Die neuronalen Netzwerke eignen sich gut für komplexe Probleme, da man durch eine große Anzahl an Neuronen und mehrere versteckte Schichten (auch Deep-Learning genannt) komplexe Muster aus den Daten extrahieren kann. [Mur12]

2.3.4 Hierarchisches Clustering

Clustering bedeutet eine automatische Strukturentdeckung in den Daten. Dabei versucht man die Daten durch vordefinierte Distanzmaße nach ihren Merkmalen zusammenfassen. Ein mögliches Distanzmaß ist die euklidische Distanz zwischen zwei n-dimensionalen Vektoren x und y. Sie ist definiert durch die Gleichung 2.3.



Person	Alter	Einkommen	Wohnort
A	24	20000	Wohnung(1)
B	65	21000	Wohnung(1)
C	35	45000	Haus(0)
D	26	35000	Haus(0)
E	39	60000	Haus(0)

Abbildung 2.5: Hierarchisches Clustering von Personendatensätzen

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.3}$$

Bei hierarchischem Clustering handelt es sich um ein unüberwachtes Verfahren des maschinellen Lernens. Auf obiges Beispiel bezogen, würde man die Kreditwürdigkeit der Personen in den Lerndaten nicht kennen. Man versucht die Personen nach ihren Merkmalen Alter, Einkommen und Wohnort zu strukturieren und darauf basierend Gruppen für die Kreditvergabe zu identifizieren.

Bei dem Von-Unten-Nach-Oben-Verfahren würde man jede Person zunächst als ein einzelnes Cluster betrachten. Danach würde man die nächst ähnliche Person suchen, die beinahe das gleiche Alter, Einkommen und Wohnort hat, und die beiden Personen zu einem Cluster zusammenfassen. Nachdem alle Paare Cluster ergeben, würde man alle Paare miteinander zu neuen Clustern zusammenfassen. Am Ende ergibt sich eine Hierarchie von Clustern, an deren Wurzel alle Personen in einem Cluster liegen.

Bei dem Von-Oben-Nach-Unten-Verfahren würde man zuerst alle Daten als ein Cluster betrachten und es danach in zwei Cluster aufteilen. Dafür würde man beispielsweise zwei sich am wenigsten ähnliche Personen suchen und alle restlichen Personen zu einer von den beiden Personen zuordnen. Die entstandenen Cluster würde man wiederum aufteilen, bis man am Ende eine hierarchische Struktur mit jeder einzelnen Person am Ende besitzt oder ein Abbruchkriterium erreicht ist.

In der Abbildung 2.5 sieht man ein mögliches Clustering für 5 Personen mit dem euklidischen Distanzmaß. Die Cluster „AB“ (niedriges Einkommen, Wohnung) und „CDE“ (hohes Einkommen, eigenes Haus) können als Gruppen für die potentielle Kreditwürdigkeit identifiziert werden. [Mur12]

Tabelle 2.1: Konfusionsmatrix

		Realität besagt	
		Korrekt	Falsch
Voraussage besagt	Korrekt	wahr korrekt	wahr fälschlich
	Falsch	falsch fälschlich	falsch korrekt

2.3.4.1 Calinski-Harabaz-Maß

Das Calinksi-Harabaz-Maß hilft die Anzahl der Cluster für die Lerndaten festzulegen. Es ist ein Maß für die Trennschärfe zwischen den Clustern. Es wird berechnet durch die Gleichung 2.4,

$$CH(k) = [B(k)/W(k)] \times [(n-k)/(k-1)] \tag{2.4}$$

wobei k die Anzahl der Cluster darstellt und n die zuvor festgelegte maximale Clusteranzahl. Bei $W(k)$ in der Gleichung 2.5 handelt es sich um die Intra-Cluster-Varianz, also der quadratischen Distanz jedes Punktes x_i des Clusters C_k von dem Mittelpunkt r_k des C_k .

$$W(k) = \sum_{k=1}^{n} \sum\nolimits_{x_i \in C_k} d(x, r_k)^2 \tag{2.5}$$

$B(k)$ in der Gleichung 2.6 ist die Inter-Cluster-Varianz bzw. die quadratische Distanz zwischen den Mittelpunkten aller Cluster:

$$B(k) = \sum\nolimits_{1 \leq j < k \leq K} d(r_j, r_k) \tag{2.6}$$

Je höher das Calinski-Harabaz-Maß ausfällt, desto eindeutiger sind die entstandenen Cluster voneinander getrennt. [Mur12]

2.4 Evaluationsmetriken

Um die Qualität der Vorhersagen der Klassifikatoren zu evaluieren existieren vordefinierte Metriken. Die Konfusionsmatrix in der Tabelle 2.1 definiert die möglichen Ausgänge einer Klassifikation.

Die "wahr korrekt"-Zelle der Matrix definiert ein Ergebnis, das von dem Klassifikator als korrekt erkannt wurde und in der Wirklichkeit auch korrekt ist. Die "wahr fälschlich"-Zelle bedeutet, dass der Klassifikator ein Ergebnis als korrekt klassifiziert, welches aber in der Wirklichkeit falsch ist. Die Ergebnisse, welche falsch sind und auch als falsch eingestuft wurden, werden als "falsch korrekt" bezeichnet. Die "falsch fälschlich"-Zelle bezieht sich auf die Ergebnisse, welche in Wirklichkeit korrekt sind, von dem Klassifikator aber als falsch eingestuft wurden.

Auf die Konfusionsmatrix beziehen sich die Metriken Präzision, Ausbeute und F-Maß. Die Metrik Präzision in der Gleichung 2.7 gibt an, wie viele Klassen aus der Menge der vorausgesagten korrekten Klassen wirklich korrekt sind.

$$\text{Präzision} = \frac{\text{wahr korrekt}}{\text{wahr korrekt} + \text{falsch korrekt}} \tag{2.7}$$

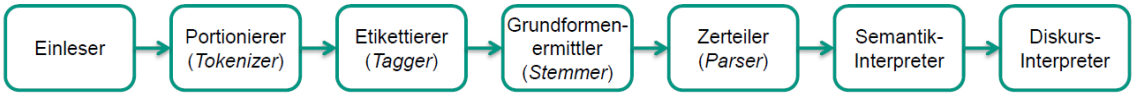


Abbildung 2.6: Computerlinguistik-Fließband [CEE+09]

Beispielsweise möchte man 10 Personen auf ihre Kreditwürdigkeit überprüfen. Wenn der Klassifikator 6 Personen als kreditwürdig klassifiziert, davon aber in Wirklichkeit nur 3 kreditwürdig sind, dann beträgt die Präzision $\frac{3}{6} = 0.5$.

Die Metrik Ausbeute in der Gleichung 2.8 gibt an, wie viele korrekte Klassifikationen insgesamt vorgenommen wurden. Beispielsweise möchte man 10 Personendatensätze auf ihre Kreditwürdigkeit überprüfen. Wenn der Klassifikator 6 Personen als kreditwürdig klassifiziert, davon aber in Wirklichkeit nur 3 kreditwürdig sind, dann beträgt die Ausbeute $\frac{3}{10} = 0.3$.

$$\text{Ausbeute} = \frac{\text{wahr korrekt}}{\text{wahr korrekt} + \text{falsch fälschlich}} \tag{2.8}$$

Das F-Maß in der Gleichung 2.9 ist das harmonische Mittel aus Präzision und Ausbeute.

$$F1 = \frac{2 \cdot \text{Präzision} \cdot \text{Ausbeute}}{\text{Präzision} + \text{Ausbeute}} \tag{2.9}$$

Der Wertebereich aller drei Metriken liegt im Intervall $\{0, 1\}$. Je näher der Wert an 1 liegt, desto besser ist die Voraussage. [Mur12]

2.5 Computerlinguistik

Die Computerlinguistik ist ein Fachgebiet, welches die maschinelle Verarbeitung von natürlicher Sprache behandelt. Offensichtlich können Computer keine natürliche Sprache verstehen, explizite Verfahren sind nötig um Verständnis zu schaffen und Syntax, Semantik sowie Pragmatik extrahieren zu können. Beispiele für Computerlinguistik-Anwendungen sind Textklassifikation und semantische Rollenzuweisung. Typisch bei der Verarbeitung von Texten ist ein Fließband. Das Fließband verarbeitet die Informationen des Textes mithilfe von einzelnen Stationen, welche in der Abbildung 2.6 zu sehen sind. Im Einleser werden die Informationen des Textes eingelesen und im Portionierer nach Sätzen oder Wörtern zerteilt. Im Etikettierter weist man den zuvor portionierten Entitäten bestimmte Etiketten zu, beispielsweise will man alle Nomen als solche kennzeichnen. Der Grundformenermittler ermittelt die Grundform eines Wortes. Im Zerteiler werden die Sätze nach gewünschten Eigenschaften zerteilt, beispielsweise erstellt man aus einem Satz einen Syntax-Baum. Der Semantik-Interpreter sucht nach den möglichen Bedeutungen des Satzes und der Diskursinterpreter nach den möglichen Bedeutungen für den ganzen Text. [CEE+09]

2.5.1 Bag-of-Words

Bag-of-Words ist ein Verfahren zur Konstruktion von Merkmalen aus einem Text. Dabei wird eine Liste, der im Dokument vorkommenden Wörter, erstellt. Diese Liste wird nach der Häufigkeit der einzelnen Wörter sortiert. Oft werden nicht alle Wörter betrachtet, sodass Wörter mit wenig Information wie Artikel und Präpositionen weggelassen werden. Aus der Liste werden anschließend Merkmale für die Sätze oder einzelne Wörter aus dem

Tabelle 2.2: Beispielhaftes Dokument für das Bag-of-Words-Verfahren

Nr.	Satz
1	Ein Auto hat ein automatisches Bremssystem.
2	Der Fahrer von dem Auto fängt an zu bremsen.
3	Das Bremssystem von dem Auto ist defekt.

Tabelle 2.3: Vektoren für Nomen aus Bag-of-Words

Nr.	Auto	Bremssystem	Fahrer
1	1	1	0
2	0	0	1
3	1	1	0

Dokument generiert. Dafür wird aus der Liste ein Vektor erstellt, welcher an den Stellen der betrachteten Wörter eine 1 enthält und an den anderen Stellen eine 0. In der Tabelle 2.2 wird ein beispielhaftes Dokument, bestehend aus drei Sätzen, gezeigt. Es wird eine Liste erzeugt, welche alle Nomen von dem Dokument enthält. Für die Nomen aus den einzelnen Sätzen ergeben sich folgende Vektoren, welche in der Tabelle 2.3 gezeigt werden. So kommt im ersten Satz das Nomen Auto vor, was durch eine 1 gekennzeichnet wird. Die 1 steht an erster Position, da das Nomen Auto im Dokument am Häufigsten vorkommt. [Mur12]

2.5.2 Vorkommenshäufigkeit-und-inverse-Dokumenthäufigkeit

Die Vorkommenshäufigkeit tf gibt an, wie oft ein Wort t in einem Dokument d vorkommt. Für die Normalisierung wird die Anzahl der Vorkommen des Wortes durch die Anzahl der Vorkommen des am häufigsten aufgetretenen Wortes unter allen Wörtern geteilt. Das Vorgehen wird in der Gleichung 2.10 gezeigt.

$$tf_{i,d} = \frac{\#(t,d)}{\max_{t' \in d} \#(t',d)} \tag{2.10}$$

Die inverse Dokumenthäufigkeit idf steht für den Informationsgehalt von einem Wort innerhalb einer Dokumentensammlung an und durch die folgende Gleichung 2.11 berechnet.

$$idf(t) = \log \frac{N}{\sum_{D:t \in D} 1} \tag{2.11}$$

Die beiden Maße werden in der Gleichung 2.12 kombiniert und geben den Vorkommenshäufigkeit-und-inverse-Dokumenthäufigkeit-Maß $tfidf_{i,d}$ eines Wortes in einer Dokumentensammlung an. Mithilfe dieses Maßes lassen sich die wichtigen Wörter eines Textes identifizieren. Auch ist es möglich, den Bag-of-Words-Vektor durch Vorkommenshäufigkeit-und-inverse-Dokumenthäufigkeit-Maße zu erweitern, indem man statt der binären Kodierung der Wörter ihre Vorkommenshäufigkeit-und-inverse-Dokumenthäufigkeit-Maße benutzt. Dadurch ergibt sich die Möglichkeit die Merkmale zu erweitern. [Mur12]

$$tfidf_{i,d} = tf_{i,d} \cdot idf_i \tag{2.12}$$

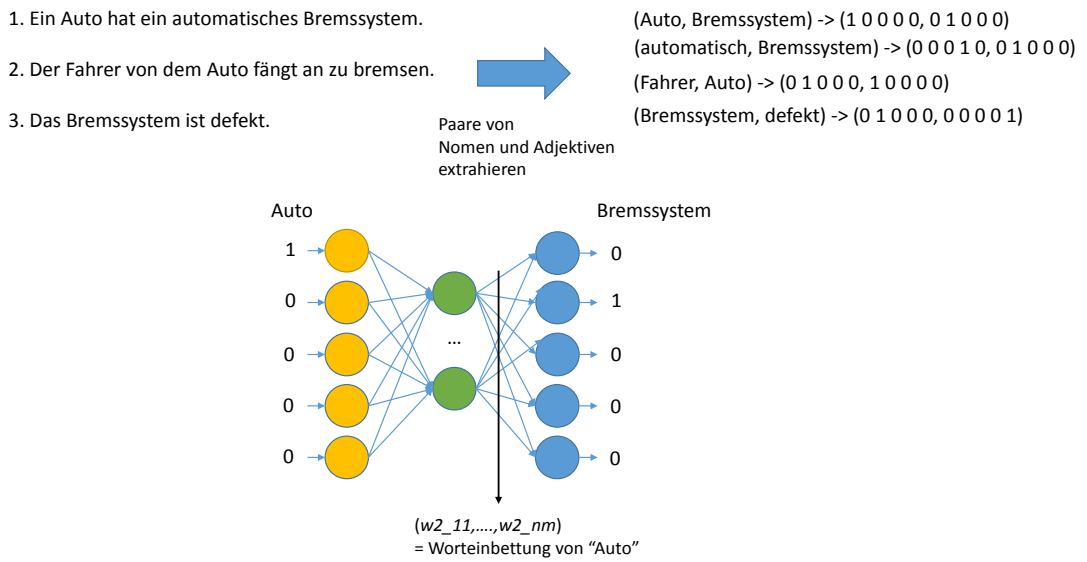


Abbildung 2.7: Worteinbettungen

2.5.3 Konstruktion der Worteinbettungen

Eine weitere Möglichkeit zur Konstruktion von Merkmalen aus Texten sind Worteinbettungen. Um Worteinbettungen zu ermitteln, wird beispielsweise ein neuronales Netzwerk trainiert. Man trainiert dabei auf Paare von Wörtern. In der Abbildung 2.7 sieht man die extrahierten Paare von Nomen bzw. Adjektiven und das dazugehörende neuronale Netzwerk, welches die Worteinbettungen berechnet. Die Paare werden durch Bag-of-Words-Vektoren repräsentiert, bei welchen nur die Stelle des vorkommenden Wortes besetzt ist. Der Vektor des ersten Wortes des Paares dient als Eingangsmerkmale des neuronalen Netzwerks. Der Vektor des zweiten Wortes des Paares wird als Ausgangsklasse behandelt. Der Vektor der Worteinbettung wird nach der Lernphase aus den Gewichten der versteckten Neuronen extrahiert. Weit verbreitet sind die 300-stelligen Worteinbettungen, extrahiert durch eine versteckte Schicht mit 300 Neuronen.

Der große Vorteil von Worteinbettungen ist die Möglichkeit des Vergleichs von Wörtern, denn ähnliche Wörter bekommen ähnliche Gewichte der versteckten Schicht. Der Vergleich von Wörtern ist dabei signifikant schneller als mit den Bag-of-Words-Vektoren, welche Hunderttausend oder mehr Stellen bei großen Dokumenten erreichen können. Ein Nachteil ist die benötigte Menge an Lerndaten, da man nach dem Training keine Möglichkeit besitzt neue Wörter einzufügen und die Lerndaten möglichst viele Wörter abdecken müssen. [MCCD13]

2.6 Ontologie

Ursprünglich stammt der Begriff Ontologie aus der Philosophie und bedeutet systematische Darstellung der Existenz. In der Informatik bedeutet eine Ontologie eine formale Repräsentation einer bestimmten Wissensdomäne. Eine Ontologie besteht aus Konzepten dieser Domäne und Beziehungen zwischen diesen Konzepten. Ein Beispiel für eine Ontologie ist in der Abbildung 2.8 dargestellt. Es ist eine Ontologie der Domäne Fahrzeug. Es handelt sich um einen Auszug. Um die gesamte Wissensdomäne des Autos zu modellieren, würde man tausende Konzepte benötigen. Die Ontologie stellt vier Konzepte dar: Auto,

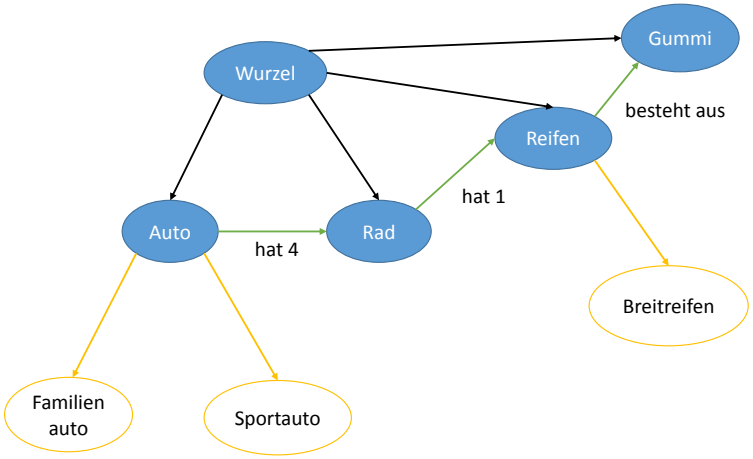


Abbildung 2.8: Ausschnitt der Ontologie der Fahrzeugdomäne

Rad, Reifen und Gummi (dargestellt durch blaue Kreise). In der Ontologie können Oberkonzepte definiert werden, in dem Beispiel sind die vier Konzepte an das Oberkonzept Wurzel angehängt. Die Beziehungen zwischen den Konzepten sind mithilfe grüner Pfeile dargestellt. Das Konzept Auto hat eine Beziehung „hat 4“ zum Konzept Rad, was bedeuten soll, dass ein Auto vier Räder besitzt. Auch ist es möglich in der Ontologie Instanzen von Konzepten anzulegen. Familienauto und Sportauto sind Instanzen des Konzepts Auto und Breitreifen ist eine Instanz des Konzepts Reifen. Eine Instanz bekommt alle Eigenschaften und Beziehungen von ihren Konzepten, dementsprechend besitzt ein Familienauto auch 4 Räder. Die Möglichkeiten eine Wissensdomäne zu modellieren sind sehr vielfältig. So kann man die Instanzen Familienauto und Sportauto auch als Unterkonzepte von dem Konzept Auto darstellen. Es kommt auf den Ersteller an, welche Informationen in welcher Form benötigt werden. [Mur12]

3. Verwandte Arbeiten

Nachfolgend werden Arbeiten aus den letzten Jahren vorgestellt, welche für die Zielsetzung und Vorgehensweise dieser Arbeit relevant sind.

Die Klassifikation von Software-Anforderungen wurde 2016 von Winkler und Vogelsang [WV16] behandelt. Dort werden die Texte der Software-Anforderungen binär klassifiziert in zwei mögliche Klassen „Information“ und „Anforderung“. Als Klassifikator wird ein Neuronales Netzwerk verwendet, mit diskreter Faltung als Aktivierungsfunktion. Als Lerndaten werden die Texte der Anforderungen verwendet. Zuvor werden die Texte der Anforderungen alle manuell untersucht und darauf basierend werden Filter definiert. Aus den Anforderungen werden alle Tabellen, Formeln, Aufzählungen, Referenzen und Abbildungen entfernt. Es bleiben natürlichsprachliche Wörter. Um Merkmale zu erzeugen, werden mit einem neuronalen Netzwerk eigene Worteinbettungen aus den Texten der Anforderungen erstellt. Die Worteinbettungen werden dann von dem neuronalen Netzwerk mit diskreter Faltung in zwei mögliche Klassen klassifiziert. Aus der Evaluation folgt, dass der Ansatz mit 70% Präzision und 85% Ausbeute gut funktioniert. Des Weiteren werden neue Anforderungen mit unbekannten Wörtern für die Worteinbettungen evaluiert. Um die Präzision und Ausbeutemaße zu behalten, müssen mindestens 40% der Wörter durch die Worteinbettungen bekannt sein. Die Möglichkeit des Einsatzes der Worteinbettungen als Merkmale ist interessant für diese Arbeit, allerding besitzt Winkler weitaus mehr Textdaten als die Anzahl der Anforderungen, die für diese Arbeit zur Verfügung steht.

Rashwan et al. [ROW13] stellten die Klassifikation von nicht-funktionallen Anforderungen vor. Die Anforderungen wurden mithilfe eines Stützvektormaschine-Klassifikators in sieben mögliche Klassen klassifiziert. Die Klassen sind die Unterarten von nicht-funktionallen Anforderungen wie zum Beispiel Operationsseinschränkung oder Qualitätsanforderung. Außerdem wurde eine Ontologie manuell erstellt und während des Klassifikationsprozesses angereichert. Die erstellte Ontologie enthält dabei diese Klassen als Konzepte. Als Lerndaten wird ein eigenerstellter Korpus verwendet, bei welchem die Anforderungen von mehreren Studenten händisch annotiert und auf Fehler untersucht wurden. Als Merkmale für die Klassifikation werden die Unigramme der Wörter der Anforderungen verwendet, wobei Stoppwörter entfernt werden. Nach der Klassifikation werden mithilfe des Computerlinguistik-Fließbands Fachbegriffe aus den Anforderungen extrahiert und als Konzepte in der Ontologie angehängt. Die Klassifikation liefert eine Präzision von 77% und eine Ausbeute von 60%. Die Klassifikation in mehrere Klassen ist auch für diese Arbeit von Bedeutung, allerdings liegt bei den Anforderungen von Bosch kein bereinigter und annotierter Korpus vor. Der Aspekt der Ontologie als Wissensbasis ist auch in der

Fahrzeugdomäne von Bedeutung. Es wäre vorteilhaft zu wissen, welche Begriffe für welche Systeme und Anforderungen relevant sind.

2013 untersuchten Wijewickrema und Gamage [WG13], wie man eine Ontologie in die Klassifikation von Textdokumenten einbeziehen kann. Das Klassifikationsproblem besteht aus mehreren Klassen, geisteswissenschaftliche Texte werden nach ihren Thematiken klassifiziert. Eine Ontologie wird im Vorfeld manuell erstellt und dient dazu die Ausgabe eines Vorkommenhäufigkeit-und-inverse-Vorkommenhäufigkeit-Klassifikators zu verfeinern. Als Merkmale werden die Vorkommenshäufigkeit-und-inverse-Dokumenthäufigkeit-Vektoren und die Bag-of-Words-Vektoren der Texte verwendet. Die Ontologie enthält alle möglichen Thematiken und Begriffe. Die von dem Klassifikator vorausgesagte Klasse wird in der Ontologie gesucht und alle Konzepte, die damit in Beziehung stehen extrahiert. Diese Konzepte werden anschließend im Eingabetext gesucht. Die Klasse mit den meisten Suchtreffern ist schließlich die vorausgesagte Klasse. Das Verfahren kommt dabei auf 73% Genauigkeit. Die Idee der Suche der Konzepte ist auch in der Anforderungsdomäne von Bedeutung, falls sich aus den Texten geeignete Konzepte und Hierarchien erstellen lassen.

Die automatische Erstellung einer Ontologie aus den Software-Anforderungen wurde von Roth und Klein [RK15] behandelt. Die Ontologieerstelllung erfolgt dabei mithilfe von Syntaxanalyse und semantischer Rollenerkennung des Computerlinguistik-Fließbands. So werden drei Konzepte definiert: Akteur, Aktion und Objekt. Diese Konzepte werden in den Texten identifiziert und anschließend in die Ontologie übernommen. Das Konzept Aktion definiert dabei eine Beziehung zwischen Akteur und Objekt. Die semantische Rollenerkennung wird mithilfe des Werkzeugs mate durchgeführt, das auf Nachrichtentexte trainiert wurde. Die Ansätze von Roth und Klein sind hilfreich für die automatische Erstellung der Ontologie für diese Arbeit, wobei die Anforderungen der Fahrzeugdomäne viele ungrammatikalische Sätze wie Aufzählungen und Tabellen enthalten. Das könnte ein Problem für die semantische Rollenerkennung werden.

Ein weiterer automatischer Ansatz der Repräsentation von Software-Anforderungen wird von Verma und Beg [VB13] vorgeschlagen. Dort wird die Syntaxanalyse angewendet, um einen Graphen aus dem Text einer Anforderung zu erzeugen. Die Erzeugung geschieht mithilfe zuvor manuell erstellter Regeln, welche die Syntaxetiketten jedes Wortes betrachten. Ein besonderer Fokus wird dabei auf die Erkennung von Zahlenwerten gelegt, da die betrachteten Anforderungen allesamt funktionalle Anforderungen sind. Der Ansatz von Verma eignet sich gut für die schnelle Erzeugung von Graphstrukturen. Die Frage ist, wie sich der Ansatz bei nicht-funktionallen Anforderungen verhält, welche in der Fahrzeugdomäne vertreten sind.

Bei Taylor et al. [TMKW] werden Informationen aus Texten automatisch in die CYC-Ontologie eingeordnet. Die Einordnung geschieht mithilfe von Klassifikatoren. Die CYC-Ontologie zielt darauf ab, das Weltwissen zu repräsentieren und setzt sich aus einer Vielzahl von Mikrotheorien zusammen. Jede Mikrotheorie steht für einen Fachgebiet, beispielsweise gibt es Mikrotheorien für Haustiere, Fahrzeuge oder Raumfahrt. Die Namen der Wurzelkonzepte der Mikrotheorien repräsentieren die möglichen Klassen. Die Texte werden mithilfe von manuell erstellen Regeln zu möglichen Konzepten und Beziehungen umgewandelt. Danach werden diese in ein Bag-of-Words-Vektor übersetzt und als Merkmale verwendet. Für die Klassifikation wird der Naïve-Bayes-Klassifikator verwendet, die Wahrscheinlichkeiten dafür werden aus den Konzepten der Mikrotheorien und der Gesamtontologie berechnet. Der Ansatz erreicht 95% Präzision und 83% Ausbeute. Die guten Ergebnisse der Klassifikation können ein Indiz für die Eignung des Naïve-Bayes-Klassifikators für die Textklassifikation sein. Bei dem Abbildungsproblem zwischen internen und externen Anforderungen handelt es sich um so ein Problem. Die Frage ist, ob sich die manuell erstellten Regeln von Taylor et al.[TMKW] auch auf die Fahrzeugdomäne übertragen lassen, da die

Sprache der CYC-Ontologie Englisch ist.

Smith et al. [STHH17a] beschäftigten sich mit der Transformation der Worteinbettungen von einer Sprache zu einer anderen. Urprünglich wurden von Facebook Worteinbettungen für 78 Sprachen veröffentlicht, die auf der Grundlage von Nutzerkommentaren erstellt wurden. Mithilfe der Singulärwertzerlegung der Matrix aller Worteinbettung einer Sprache erstellten Smith et al. eine Transformationsmatrix, um die Worteinbettung für ein Wort in einer Sprache in eine andere Sprache zu transformieren. Dadurch lassen sich Wörter oder sogar Texte in zwei verschiedenen Sprachen vergleichen. Bei der Transformation von Deutsch nach Englisch erreicht der Ansatz eine Präzision von 73%. In der Fahrzeugdomäne von Bosch sind die Sprachen Deutsch und English beide stark vertreten. Eine Möglichkeit, sprachübergreifend Texte zu vergleichen, ist hilfreich bei der Analyse von Anforderungen.

Manku et al. [MJDS07] beschäftigen sich mit der Erkennung von Fast-Duplikaten in den Webdokumenten. Der Inhalt der Webdokumente wird dabei extrahiert und danach mit dem deterministischen Hashing-Verfahren Simhash in einen Hashwert umgewandelt. Die enstandenen Hashwerte lassen sich nach ihren Hamming-Distanz sortieren, Hashwerte mit geringer Distanz haben dabei einen sehr ähnlichen Inhalt. Das Verfahren eignet sich nicht nur für Webdokumente, sondern auch für Sätze. Die Präzision des Verfahrens liegt bei 90%. Auch in der Anforderungsdomäne ist die Erkennung von Fast-Duplikaten von Bedeutung, da dadurch Kopien und gegebenenfalls ähnliche Anforderungen sich identifizieren lassen.

4. Analyse und Entwurf

Das Ziel dieser Arbeit ist die Erstellung einer automatischen Abbildung der externen Anforderungen eines Kunden der Fahrzeugindustrie auf die internen Anforderungen von Bosch, sodass bei einer neuen externen Anforderung dem Mitarbeiter Vorschläge für eine sinnvolle interne Abbildung geliefert werden können. Zurzeit läuft der Vorgang manuell ab. Bosch-Mitarbeiter mit breitem Domänen-Wissen bearbeiten ein Kundenlastenheft, extrahieren daraus die Anforderungen und bilden die Anforderungen ab. Nachfolgend wird der Abbildungsprozess beschrieben und die vorliegenden Daten der Anforderungen analysiert.

4.1 Beschreibung des Lastenhefts

Am Anfang erhält der zuständige Mitarbeiter ein Lastenheft von dem Kunden im DOORS-Datenbankformat, welches informative Richtlinien und Anforderungen für ein Projekt enthält. In dieser Arbeit werden die Projekte der Bremssysteme für den Kunden namens A betrachtet. Bei einem Projekt handelt es sich um einen bestimmten Fahrzeugtyp, beispielsweise beschreibt das Projekt 714123 die Anforderungen der Bremssysteme für das Fahrzeugmodell A1 . Das Lastenheft ist unter anderem in die Abschnitte wie „Einleitung“ oder „Einzulesende PDU/ Signale“ gegliedert. In den einzelnen Abschnitten befinden sich die Anforderungen als Textblöcke. Zusätzlich zu den Abschnitten sind die Anforderungen in Module gegliedert. Ein Modul steht für einen bestimmten Funktionsumfang eines Systems.

Die Tabelle 4.1 zeigt einen Ausschnitt des Lastenheftes. In der Spalte „Extern Heading“ werden die Überschriften der Abschnitte angezeigt. Die Spalte „Extern Text“ zeigt den Textblock einer Anforderung. In der Spalte „Extern Module“ ist das Modul einer Anforderung enthalten. So gehören die 10 ersten Anforderungen in dem Abschnitt „Einzulesende PDU/ Signale“ zu dem Modul „A1_714123_Start Stopp_ESC_Signale“ und beschreiben somit die Start-Stopp-Funktion. Die Struktur der Module wird vom Kunden bei jedem Lastenheft gesondert erstellt.

4.2 Abbildungsprozess

Das Lastenheft wird von einem Mitarbeiter in die Bosch-interne DOORS-Datenbank eingelesen. Danach prüft der Mitarbeiter die einzelnen Anforderungen nacheinander und weißt jeder Anforderung einen Status zu. Der Status einer Anforderung bestimmt die weitere Vorgehensweise des Mitarbeiters. Es gibt mehrere Statusarten. Der Status *no_evaluation*

Tabelle 4.1: Auszug des Lastenheftes

Extern Heading	Extern Text	Extern Module
1. Einleitung		
	Für alle nachfolgend beschriebenen Anforderungen müssen dem Auftraggeber auf Anfrage detaillierte Funktionsbeschreibungen der Umsetzung übergeben werden.	A1_714123_Funktionen_Spezifikation
	Bei Bedarf sind zusammen mit dem Auftraggeber Reviews durchzuführen.	A1_714123_Funktionen_Spezifikation
1.1 Ein- und Ausgaben (Busschnittstellen)		
	Die nachfolgend angegebenen Signale sind der aktuelle Stand der Vernetzungsumfänge	A1_714123_Funktionen_Spezifikation
	Die Weiterentwicklung des Vernetzungsumfangs erfolgt in der Kommunikationsmatrix für das entsprechendes Fahrzeugprojekt	A1_714123_Funktionen_Spezifikation
1.2 Einzulesende PDU / Signale		
	Die einzulesenden Signale sind der jeweils gültigen Kommunikationsmatrix zu entnehmen.	A1_714123_Start Stopp_ESC_Signale
	Die bisher in diesem Lastenheft enthaltene Auflistung der PDUs und Signale wird verworfen	A1_714123_Start Stopp_ESC_Signale
1.2.1 PDU_A11_101		
	ZZA_01_QA	A1_714123_Start Stopp_ESC_Signale
	ZZA_01_QB	A1_714123_Start Stopp_ESC_Signale
	ZZA_01_QC	A1_714123_Start Stopp_ESC_Signale
	ZZA_01_QD	A1_714123_Start Stopp_ESC_Signale
	ZZA_01_QE	A1_714123_Start Stopp_ESC_Signale
	ZZA_01_QG	A1_714123_Start Stopp_ESC_Signale
	ZZA_02_QA	A1_714123_Start Stopp_ESC_Signale
	ZZA_02_QB	A1_714123_Start Stopp_ESC_Signale
	ZZA_02_QC	A1_714123_Start Stopp_ESC_Signale
	ZZA_02_QD	A1_714123_Start Stopp_ESC_Signale
	ZZA_02_QE	A1_714123_Start Stopp_ESC_Signale
1.2.2 PDU _A11_102		
	BBN_03_04	A1_714123_Start Stopp_ESC_Signale

bedeutet, dass es sich bei der Anforderung um eine informative Richtlinie handelt, welche nicht implementiert und somit nicht abgebildet werden muss. Der Status *no_req* besagt, dass die Anforderung für die Funktionsweise des Systems nicht von Bedeutung ist. Wenn der Status auf *accepted* gesetzt wird, dann handelt es sich um eine Anforderung, welche abgebildet werden muss. Wenn der Status auf *accepted_w_remarks* gesetzt wird, wird zu der Anforderung ein Kommentar des Mitarbeiters erstellt und die Anforderung muss abgebildet werden. Nachdem der Status gesetzt wurde, weist der Mitarbeiter jeder Anforderung einen Typ zu. Eine Anforderung kann drei mögliche Typen besitzen: System, Hardware oder Software. Aus dem Typ ergibt sich die zuständige Abteilung, welche die Abbildung der Anforderung übernimmt. Zusätzlich wird eine sogenannte FN-ID für jede Anforderung gesetzt, welche auf einen bestimmten Mitarbeiter innerhalb dieser Abteilung verweist, der die Abbildung auf eine oder mehrere interne Anforderungen vornehmen muss. Die internen Anforderungen befinden sich in der internen DOORS-Datenbank und besitzen ebenfalls Text und Modulinformationen. Dabei trennen sich die internen Anforderungen wie auch die externen Anforderungen in drei Typen: System, Hardware oder Software. Diese Typen beziehen sich auf das V-Modell. Die Anforderungen der Systemebene definieren dabei die System-Architektur. Die Anforderungen der Hardwareebene definieren den System-Entwurf und die Anforderungen der Softwareebene definieren die Software-Architektur. Um eine Abbildung vorzunehmen, muss der zuständige Mitarbeiter die DOORS-Datenbank nach einer passenden internen Anforderung durchsuchen und die DOORS-Verweise der externen und der internen Anforderungen miteinander verknüpfen. In der Tabelle 4.2 sieht man einen Ausschnitt des Lastenheftes mit externen Anforderungen und den dazugehörigen abgebildeten internen Anforderungen. Die Spalten der externen Anforderungen sind die Gleichen wie im Kapitel 4 beschrieben. Die Informationen der zugehörigen internen Anforderungen sieht man in den Spalten „Intern Text“ und „Intern Module“.

4.3 Datenanalyse

Insgesamt liegen 648 851 Anforderungen vor, welche aus der DOORS-Datenbank von Bosch exportiert wurden. Die Anforderungen kommen aus 10 Projekten, wobei alle Projekte Systeme für mehrere Fahrzeugmodelle des Kunden A behandeln. Bei den Anforderungen handelt es sich bei 176 112 um externe Anforderungen, alle anderen sind interne Anforderungen. Die Verteilung der Anforderungen ist in der Abbildung 4.1 als ein Kreisdiagramm zu sehen. Es gibt insgesamt 160 139 Abbildungen. Bei der Betrachtung der Abbildungen wurde festgestellt, dass auch interne Anforderungen auf andere interne Anforderungen abgebildet werden. Dies ist durch die Strukturierung als V-Modell zu begründen. Die internen Anforderungen der Systemebene sind auf die internen Anforderungen der Hardware- und Softwareebene abgebildet. Dabei sind die Abbildungen von internen auf interne Anforderungen sogar ein Großteil, 129 546 der Gesamtabbildungen sind interne. Im Rahmen dieser Arbeit sind die Abbildungen von internen auf interne Anforderungen nicht von Bedeutung, da dies auch nicht dem Ziel der automatischen Abbildung entspricht. 30 593 externe Anforderungen sind abgebildet. Die Zahl der internen Anforderungen, auf welche die externen Anforderungen abgebildet wurden, beträgt 15 093. In der Abbildung 4.2 wird die Häufigkeit der Abbildungen gezeigt. Dafür wurde für jede externe Anforderung aus den 30 593 gezählt, wie viele Abbildungen auf die internen Anforderungen diese Anforderung besitzt. So existieren 12 084 1-zu-1-Abbildungen, wo eine externe Anforderung auf eine interne Anforderung abgebildet wird. Maximal wurde eine externe Anforderung auf 132 verschiedene interne Anforderungen abgebildet. So eine 1-zu-132-Abbildung kommt allerdings nur einmal vor.

Extern Heading	Extern Text	Extern Module	Intern Text	Intern Module
1. Einleitung				
	Für alle nachfolgend beschriebenen Anforderungen müssen dem Auftraggeber auf Anfrage detaillierte Funktionsbeschreibungen der Umsetzung übergeben werden.	A1_714123_Funktionen_Spezifikation	Link to Matrix 1100	A_111111_Funktionsanforderungen_Spezifikation
	Bei Bedarf sind zusammen mit dem Auftraggeber Reviews durchzuführen.	A1_714123_Funktionen_Spezifikation	Link to Matrix 1100	A_111111_Funktionsanforderungen_Spezifikation
1.1 Ein- und Ausgaben (Busschnittstellen)				
	Die nachfolgend angegebenen Signale sind der aktuelle Stand der Vernetzungsumfänge	A1_714123_Funktionen_Spezifikation	A central table should define the sensitivity of all monitorings. Each monitoring shall be manipulated separatly.	A_111111_Start_Stopp_ESC_Signals
	Die Weiterentwicklung des Vernetzungsumfangs erfolgt in der Kommunikationsmatrix für das entsprechendes Fahrzeugprojekt	A1_714123_Funktionen_Spezifikation	If there is no possibility to request the gearbox park position, the value AX must be displayed.	A_111111_Start_Stopp_ESC_Signals
1.2 Einzulesende PDU / Signale				
	Die einzulesenden Signale sind der jeweils gültigen Kommunikationsmatrix zu entnehmen.	A1_714123_Start Stopp_ESC_Signale		
	Die bisher in diesem Lastenheft enthaltene Auflistung der PDUs und Signale wird verworfen	A1_714123_Start Stopp_ESC_Signale		
1.2.1 PDU_A11_101				
	ZZA_01_QA	A1_714123_Start Stopp_ESC_Signale	ZZA_01_QA	A_111111_Start_Stopp_ESC_Signals
	ZZA_01_QB	A1_714123_Start Stopp_ESC_Signale	ZZA_01_QB	A_111111_Start_Stopp_ESC_Signals
	ZZA_01_QC	A1_714123_Start Stopp_ESC_Signale	ZZA_01_QC	A_111111_Start_Stopp_ESC_Signals
	ZZA_01_QD	A1_714123_Start Stopp_ESC_Signale	ZZA_01_QD	A_111111_Start_Stopp_ESC_Signals
	ZZA_01_QE	A1_714123_Start Stopp_ESC_Signale	ZZA_01_QE	A_111111_Start_Stopp_ESC_Signals
	ZZA_01_QG	A1_714123_Start Stopp_ESC_Signale	ZZA_01_QG	A_111111_Start_Stopp_ESC_Signals
1.2.2 PDU _A11_102				
	BBN_03_04	A1_714123_Start Stopp_ESC_Signale	BBN_03_04	A_111111_Start_Stopp_ESC_Signals

Tabelle 4.2: Auszug des Lastenheftes mit internen Anforderungen

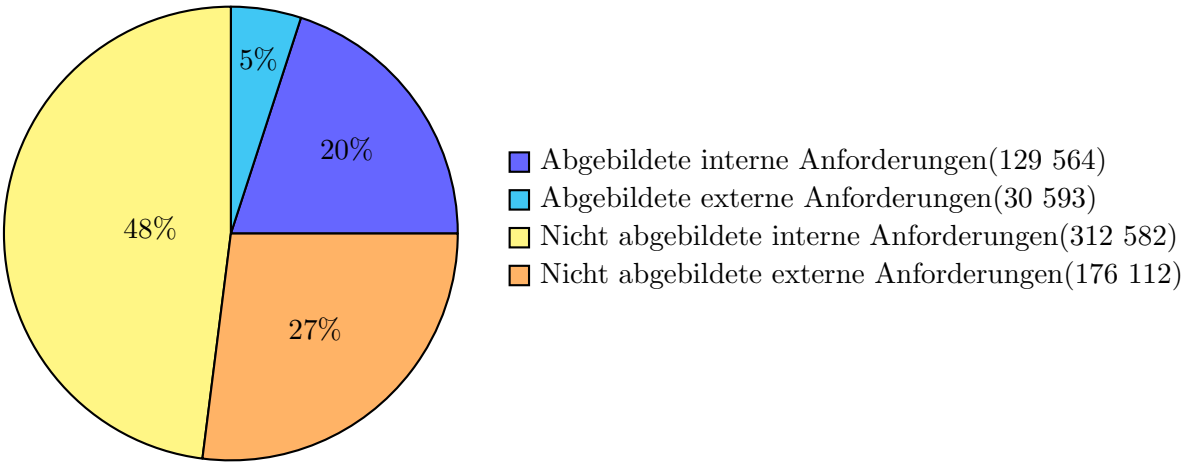


Abbildung 4.1: Verteilung der Anforderungen

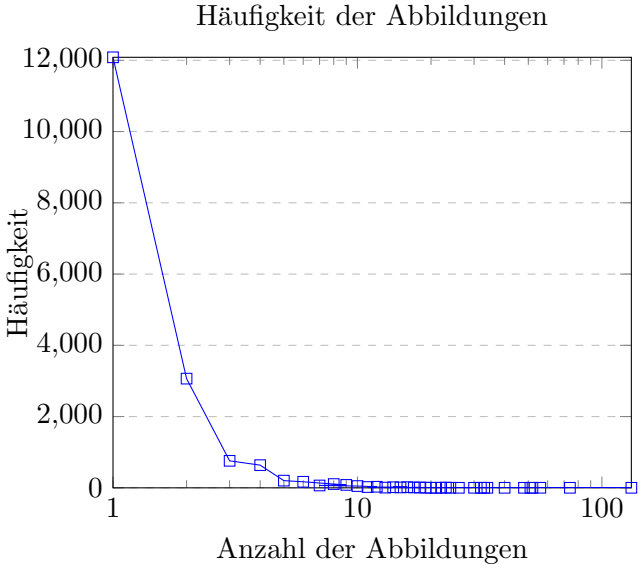


Abbildung 4.2: Häufigkeit der Abbildungen

4.3.1 Wichtige Felder

Die Daten wurden mit den Mitarbeitern von Bosch besprochen, gemeinsam wurden Felder für weitere Arbeit an der automatischen Abbildung identifiziert. Es stellte sich heraus, dass viele Felder in der DOORS-Datenbank Duplikate darstellen oder für bestimmte Anforderungen als Metadaten angelegt wurden. Die Tabelle 4.3 zeigt einen Auschnitt der Anforderungen mit den wichtigen Feldern. Die Felder, die in jeder Anforderung vorkommen und Aussagekraft besitzen, sind:

Created On

Dieses Feld enthält das Datum der Anforderung, als diese in dem DOORS-System von Bosch angelegt wurde. Die Zeitangabe besteht aus **Tag Monat Jahr**. Dieses Feld wird von dem DOORS-System automatisch erzeugt. Dieses Feld ist von Bedeutung, da man dadurch die zeitliche Abfolge der Abbildung sehen kann.

Beispiel: 29 Sep 2009

Created By

Das Feld **Created By** enthält den Kürzel des Mitarbeiters, der im DOORS-System angemeldet war und die Anforderung dort erstellt hat. Jeder Mitarbeiter hat eine eindeutige Kennung. Die Kennung des Mitarbeiters ist nützlich, um mögliche Gruppen der Anforderungen zu identifizieren, da ein Mitarbeiter eine bestimmte Menge von Anforderungen abbildet.

Beispiel: abx4rng

Object Heading

Bei den Anforderungen kann es sich um Überschriften handeln. Falls es eine Überschrift ist, so steht sie in diesem Feld.

Beispiel: Einleitung

Object Level

Dieses Feld zeigt mithilfe einer Nummer, wo sich die Anforderung ursprünglich im Lastenheft befand. **Object Level** ist eine ganzzahlige Nummer, beginnend bei 1.0. Dieses Feld ist wichtig, da es die Anforderungen zu ihren Überschriften zuordnet und diese somit in Abschnitte trennt.

Beispiel: 1.1.2

Object Text

In diesem Feld steht der Text einer Anforderung. Der Text ist nicht einheitlich, er kann nur aus einem Begriff bestehen. Es ist ebenso möglich, dass dort Tabellen, Formeln oder Verweise auf Dateien als Inhalte gespeichert sind.

Beispiel: ZZA Grenzsteigung: ZZA_Grenzsteig = 20 Prozent

Module

Bei dem Feld **Module** handelt es sich um das Modul einer Anforderung. Alle Anforderungen sind einem bestimmten Modul zugeordnet. Ein Modul repräsentiert einen Funktionsumfang. Eine Modulbezeichnung besteht aus dem Präfix des Fahrzeugs, danach folgt die Projektnummer. Danach folgt die Bezeichnung des Bremssystems für das Fahrzeug.

Beispiel: A1_714123_Funktionsanforderungen_Spezifikation

URL

Bei dem Feld **URL** handelt es sich um einen Identifikationspfad in der DOORS-Datenbank. Jede Anforderung hat eine eigene **URL** und diese ist eindeutig. Damit kann jede Anforderung abgerufen werden.

Beispiel: doors.de.bosch.com:12345?1-123456789

Status

Das Feld **Status** repräsentiert den Status der Anforderung. Es sagt aus, ob die Anforderung abgebildet werden muss. Die einzelnen Statuswerte werden in dem Abschnitt 4.2 beschrieben.

Beispiel: accepted_w_remarks

FN-ID

In diesem Feld steht die ID des zuständigen Mitarbeiters, der die Anforderung abbilden muss. Dabei handelt sich viel mehr um eine Mitarbeiterrolle, sodass mehrere Mitarbeiter einer Abteilung die gleiche ID bekommen können. Durch die Betrachtung der **FN-ID** würden sich möglicherweise Gruppen von Anforderungen finden lassen.

Beispiel: 234

Type

Das Feld **Type** zeigt den Typ der Anforderungen. Hier werden die Anforderungen in System, Hardware oder Software unterteilt.

Beispiel: System

4.4 Herausforderungen des Abbildungsprozesses

Für das Vorgehen der automatischen Abbildung ergeben sich mehrere Herausforderungen. So sind die externen Anforderungen alle in deutscher Sprache verfasst, während die internen Anforderungen auf Englisch geschrieben sind. Dies ist für einen Mitarbeiter wenig problematisch, da die zuständigen Mitarbeiter Deutsch und Englisch beherrschen. Für eine automatische Abbildung kann sich ein Übersetzungsproblem ergeben. Wenn man beispielsweise gemeinsame Begriffe wie „Bremse“ und „Brake“ in den Paaren von internen und externen Anforderungen automatisch identifizieren möchte, verursachen zwei unterschiedliche Sprachen eine zusätzliche Schwierigkeit. Um die Anforderungen sinnvoll zu übersetzen, müsste man auf die Online-Übersetzungssysteme wie Google, Bing oder DeepL zugreifen. Bei den Lastenheften handelt es sich aber um aktuelle Fahrzeugprojekte, die noch nicht auf dem Markt sind. Die Texte unterliegen daher der Geheimhaltung und können nicht ohne Weiteres an die Übersetzungsserver gesendet werden. Es ergibt sich die Frage, ob man mit Offline-Übersetzern ein gutes Ergebnis erzielen kann.

Des Weiteren sind die textuellen Informationen der Anforderungen oft mit anderen Informationen vermischt. So sind in manchen Anforderungen Tabellen von Microsoft Excel manuell herein kopiert worden. Manche Anforderungen enthalten Verweise auf PDF-Dateien oder Bilder, die aber nicht im DOORS-System enthalten sind. Diese Informationen sind dabei nicht strukturiert und können überall im Text auftauchen. Solche Informationen erschweren die automatische Abbildung, da man zusätzliche Regeln und Filter definieren muss, um diese Informationen von den textuellen Informationen der Anforderungen zu trennen.

Außerdem werden die Anforderungen nicht auf Grammatik und Rechtschreibung überprüft. Bei der Analyse sind mehrmals Rechtschreibfehler aufgefallen, wie zum Beispiel

Created By	Created On	Object Heading	Object Level	Object Text	Module	URL	Status	FNID	Type
add8rng	29 Sep 09		1.1.2	IX:QA2251	A1_7951111...	doors.de...	accepted	2	Software
acc7rng	28 Oct 2009	Programmierablauf	1.1.3		A1_7951111...	doors.de...	no req		
cc2rng	28 Oct 2009		1.1.3	Wird die verbindliche Reihenfolge des Programmierablaufes nicht eingehalten...	A1_7951111...	doors.de...	accepted	234	System
cc2rng	28 Oct 2009		1.1.3	Der Server befindet sich nach dem Hochlauf...	A1_7951111...	doors.de...	accepted_w_remarks	234	System
cc2rng	28 Oct 2009		1.1.3	Die Programmier-Phase1 lässt sich wiederum in...	A1_7951111...	doors.de...	accepted	234	System
cc2rng	28 Oct 2009		1.1.3	Diese Phase ist optional und kann im Anschluss an Programmier-Phase...	A1_7951111...	doors.de...	accepted	234	System
cc2rng	28 Oct 2009		1.1.3	Applikations-SW 1 und/oder...	A1_7951111...	doors.de...	accepted_w_remarks	234	System
cc2rng	28 Oct 2009		1.1.3	Applikations-SW 2 und/oder...	A1_7951111...	doors.de...	accepted	234	System
cc2rng	28 Oct 2009		1.1.3	Applikations-SW 3 und/oder...	A1_7951111...	doors.de...	accepted	234	System
acc7rng	28 Oct 2009	Pre-Programming	1.1.4		A1_7951111...	doors.de...	no req		

Tabelle 4.3: Ein Beispiel mit wichtigen Feldern aus einer Anforderung

Tabelle 4.4: Abstraktionsstufen zwischen externen und internen Anforderungen

Extern	Intern
A1-Fahrzeuge mit Automatik-Getriebe (Stoppgeschwindigkeit 10km/h): - Sensorgeschwindigkeit Default: 10 km/h	Values for A1: AStop10 = 10km/h, AStop34 = 5km/h
	ADelta10 and ADelta35 are calibratable. Default values: - ADelta10 = 5,5 km/h - ADelta35 = 4,5 km/h
	If an emergency brake is detected, the stop prohibition must be set for the calibratable duration D. Default value: 50s. Note: if the engine is off, the timer will be updated.

„Farhzeugbremse“ statt „Fahrzeugbremse“. Diese könnten problematisch werden, da die automatischen Extraktoren wie das Computerlinguistik-Fließband mit Modellen arbeiten, die keine Rechtschreibfehler berücksichtigen können. Dadurch können wichtige Informationen einer Anforderungen verloren gehen.

Ein weiteres Problem ist die unterschiedliche Abstraktionsstufe der Anforderungen. Es gibt Anforderungen, die genau ein Signal beschreiben wie „ZZA Grenzsteigung: ZZA_Grenzsteig = 20 Prozent“, aber auch Anforderungen, die sehr allgemein beschrieben sind wie „Ist der vom Fahrer angetretene Bremsdruck zu gering und das Fahrzeug rollt an, darf kein aktiver Druckaufbau erfolgen“. Dies kann den automatischen Abbildungsprozess erheblich erschweren. So kann man zwar mithilfe des Fließbands der Computerlinguistik die wichtigen Wörter im Text identifizieren, aber ihre Abstraktionsstufen verfälschen die Aussagekraft. Ein Wort, welches nur auf der untersten Abstraktionsstufe auftaucht, ist für die Anforderung der höheren Abstraktionsstufe als Merkmal bedeutungslos. Um die Abstraktionsstufen zu verstehen, benötigt ein Mitarbeiter viel Hintergrundwissen. Dieses Wissen muss in einem automatischen Prozess berücksichtigt werden. Eine externe Anforderung mit einer sehr genauen Beschreibung kann auf mehrere interne Anforderungen mit verschiedenen Abstraktionsstufen abgebildet werden. Ein Beispiel hierfür befindet sich in der Tabelle 4.4. Eine externe Anforderung wird dort auf drei verschiedene interne Anforderungen abgebildet. Für einen Laien ist der Zusammenhang schwer erkennbar. Es stellt sich die Frage, ob nicht nur Hintergrundwissen, sondern auch die Texte und die Überschriften der Umgebung der Anforderung für die Entscheidung einbezogen werden.

Des Weiteren erhalten die Anforderungen sehr viele Fachwörter und Abkürzungen. Die Abkürzungen auf der externen Seite sind oftmals andere als die Abkürzungen auf der internen Seite der Anforderungen. Für die Fachwörter und Abkürzungen gibt es Lexika seitens Bosch; diese enthalten jedoch nicht alle Begriffe. Somit muss die automatische Abbildung mit unbekannten Wörtern umgehen können.

Bei der Abbildung von Anforderungen handelt es sich um eine m-zu-n-Abbildung. Eine externe Anforderung kann auf mehrere interne Anforderungen abgebildet werden und mehrere externe Anforderungen können auf eine interne Anforderung abgebildet werden. Eine Minimum- bzw. Maximumanzahl der Abbildungen ist nicht definiert. Es gibt externe Anforderungen, die auf 100 oder mehr verschiedene interne Anforderungen abgebildet

wurden. Dies bedeutet einen zusätzlichen Aufwand für die automatische Abbildung, da man Informationen aus einer einzigen externen Anforderung extrahieren muss, welche repräsentativ für eine Vielzahl der internen Anforderungen sind.

Zuletzt ist die Konsistenz der Anforderungen ein Problem. Viele der externen und internen Anforderungen wurden kopiert, wenn sie in anderen Projekten benötigt wurden, oder leicht umgeändert, falls die vorherigen Formulierungen fehlerhaft waren. Bei dem Kopiervorgang werden allerdings keine Referenzen zu der ursprünglichen Anforderung hergestellt, sodass man den Ursprung der Kopie nicht aus den Attributen der Kopie bestimmen kann. Es existiert daher eine unbekannte Menge von Duplikaten oder Fast-Duplikaten. Solche Duplikate können die Ergebnisse einer automatischen Abbildung verfälschen.

4.5 Lösungsansätze

Durch die Herausforderungen im Abschnitt 4.4 ist das Problem mit einfachen Mitteln wie händisch erstellen Regeln schwer bis unmöglich zu lösen. Die m-zu-n Abbildungen erfordern die Definition von komplexen Regeln. Beispielsweise müsste man bei einer 100-zu-1 Abbildung 100 mögliche Regeln definieren. Zusätzlich dazu muss man die Regeln nach ihren Abstraktionsstufen gruppieren, wobei dafür keine Trennungen in den Daten gegeben sind.

Für den Einsatz automatischer Lösungsmöglichkeiten muss die Beschaffenheit der Anforderungen berücksichtigt werden. Die Tabellen, Formeln und Verweise auf Dateien müssen mithilfe geeigneter Verfahren von Textinformationen der Anforderungen getrennt werden. Das Betreiben von umfassender Fehleranalyse und Filterung ist aber nicht das Ziel dieser Arbeit. Somit müssen Verfahren ausgewählt werden, die robust gegen Rauschen in den Daten sind. Der Einsatz von Sperrlisten könnte bei der Auswahl von wichtigen Informationen helfen. Des Weiteren könnte man möglicherweise mit wenigen annotierten Lerndaten Klassifikatoren trainieren, welche die Informationen der Texte als wichtig oder nicht klassifizieren.

Eine Lösungsmöglichkeit für die automatische Abbildung ist der Einsatz von Textähnlichkeiten. So ist zwar die Möglichkeit der Online-Übersetzung nicht gegeben, doch gibt es bei Smith et al. [STHH17b] eine Offline-Möglichkeit der Transformation der Worteinbettungen von Deutsch nach Englisch. Dadurch ist man in der Lage die Worteinbettungen von den deutschen Texten der Anforderungen zu erstellen und diese in englische Worteinbettungen zu transformieren. Die Ähnlichkeit von beiden Worteinbettungen kann berechnet werden. Dadurch können die richtigen Paare von den externen und internen Anforderungen identifiziert werden. Es stellt sich allerdings die Frage, wie sich die Textähnlichkeiten bei m-zu-n-Abbildungen verhalten. Die unbekannten Begriffe des Modells stellen auch ein erhebliches Problem dar, da man für die Fachbegriffe der Fahrzeugdomäne keine Worteinbettungen besitzt.

Eine mögliche Lösung des Problems ist die automatische Klassifikation der Anforderungen mit einem Verfahren des maschinellen Lernens. Dabei muss man aus den Informationen der externen Anforderungen sinnvolle Merkmale extrahieren und diese zum Trainieren eines Klassifikators verwenden. Hierbei ergibt sich die Frage, was als Klasse für die Voraussage verwendet werden soll. Man könnte versuchen, die zutreffenden Begriffe der internen Anforderungen vorauszusagen, aber auch die zutreffenden Module oder die Verweise auf die internen Anforderungen. Der Mitarbeiter würde bei der Eingabe einer neuen externen Anforderung diese Klasse als Ausgabe erhalten und daraufhin die endgültige Entscheidung der Abbildung treffen. Deshalb muss die Klasse von der Seite der Bedienbarkeit verständlich und schnell lesbar sein, um für den Mitarbeiter keinen zusätzlichen Aufwand zu erzeugen. Die Wahl der Merkmale ist ebenfalls entscheidend. Die Ansätze der Anforderungsklassifikation von Rashwan et al. [ROW13] extrahieren die Merkmale aus den Anforderungstexten

mithilfe des Fließbands der Computerlinguistik. Des Weiteren können die Merkmale auch aus den Metadaten wie Überschriften, Autoren, Zeitstempeln und Modulinformationen entnommen werden. Dadurch kann die Klassifikation gegebenenfalls präziser werden.

Für die Klassifikation kann zum Beispiel den Naïve-Bayes-Klassifikator verwenden. Dadurch lässt sich ein Ergebnis schnell herstellen. Man nimmt dabei aber auch an, dass alle Merkmale unabhängig sind, was aber aufgrund der Wortreihenfolge sowie bestimmten Metadaten (Ein bestimmter Autor bildet bestimmte Anforderungen zum bestimmten Zeitpunkt ab) die Ergebnisse verfälschen kann. Für die Arbeit mit hochdimensionalen Daten wie Textmengen eignet sich außerdem der Random-Forest-Klassifikator. Wie bei Winkler und Vogelsang [WV16] kann ein neuronales Netzwerk verwendet werden. Dadurch können auch komplexe Beziehungen in den Daten erkannt werden.

Eine weitere Lösung ist das Clustering der Anforderungen. Hierbei ist die Wahl der Merkmale ebenfalls entscheidend. Anstatt anhand der Merkmale eine Voraussage für die Abbildung zu treffen, werden durch Clustering die bestehenden Abbildungen nach Merkmalen gruppiert. Beispielsweise durch Clustering der externen Anforderungen bekommt man Gruppen dieser Anforderungen, die ähnliche Merkmale aufweisen. Danach kann man eine neue Anforderung einer Gruppe zuordnen und dadurch die Abbildungskandidaten erhalten. Ebenso kann man die internen Anforderungen gruppieren und darauf abgebildeten Merkmale der externen Anforderungen betrachten. Dadurch könnte man eine externe Anforderung direkt einer Gruppe mit internen Anforderungen zuordnen.

Wünschenswert ist eine Möglichkeit das Domänenwissen repräsentieren zu können. Eine Datenstruktur, welche die relevanten Informationen wie Bezeichnungen der Signale, Systeme und Beziehungen zwischen ihnen repräsentiert, könnte für die Wahl der Merkmale verwendet werden. Auch für einen Mitarbeiter kann diese Struktur wertvolle Informationen liefern, um beispielsweise neue Anforderungen zu definieren oder nach Begriffen in den bestehenden zu suchen. Für die Erstellung einer Datenstruktur können Lexika von Bosch verwendet werden. Weitere Begriffe können mithilfe des Computerlinguistik-Fließbands aus den Anforderungen extrahiert werden.

Um eine solche Datenstruktur zu repräsentieren eignet sich kein relationales Model [SSH10]. Wenn ein Begriff aus den Anforderungen Beziehungen zu n anderen Begriffen besitzt, müsste man in einem relationalen Modell n Tabellen mit Fremdschlüsseln anlegen. Dadurch würde die Anzahl der Tabellen schnell unübersichtlich werden und viel Speicherplatz für die Verarbeitung benötigen. Ein nicht-relationales Model wie eine graphbasierte Datenbank wäre als Darstellung der Datenstruktur besser geeignet. Beispielsweise kann eine Ontologie als Modell verwendet werden.

Um die Ontologie mit den Informationen aus den Anforderungen anreichern zu können, muss zuvor die Struktur der Ontologie definiert werden. Für jede Art der extrahierten Begriffe muss ein Konzept oder eine Instanz angelegt werden. Für die Beziehungen zwischen den Begriffen werden in der Ontologie Relationen definiert, welche die extrahierten Begriffe miteinander verknüpfen. Diese Beziehungen können möglicherweise mit der Syntaxanalyse des Computerlinguistik-Fließbands erkannt werden, wie die Adjektiv-Nomen-Beziehungen. Außerdem ist es wichtig, jede Anforderung als ein eigenes Konzept oder eine eigene Instanz in der Ontologie zu besitzen und die daraus extrahierten Begriffe und Entitäten darauf zu verweisen. Damit bleibt für jede Anforderung die Information über ihre Ontologie-Entitäten erhalten.

4.6 Textähnlichkeiten

Um die Schwierigkeit des Abbildungsproblems einzuschätzen, wird in diesem Abschnitt die Textähnlichkeit zwischen externen und darauf abgebildeten internen Anforderungen

betrachtet. Die Textähnlichkeiten werden mithilfe von den Worteinbettungen berechnet. Als Worteinbettungen werden die vortrainierten Worteinbettungen von Facebook verwendet, welche auch bei Smith et al. [STHH17b] zum Einsatz kamen. Ein eigenes Modell wird nicht erstellt, da die Texte der Anforderungen eine geringe Menge an Trainingsdaten darstellen und ungefiltert vorliegen.

Um das bestehende Übersetzungsproblem von Deutsch nach Englisch zu lösen, wurden die Transformationsmatrizen von Smith et al. [STHH17b] verwendet. Die einzelnen Wörter einer externen Anforderungen werden mithilfe des Computerlinguistik-Fließbands extrahiert. Für jedes extrahierte Wort w_i wird es nach einer Worteinbettung in den deutschen Worteinbettungen gesucht. Falls ein Wort für das Modell unbekannt ist, wird das Wort ignoriert. Dies ist beispielsweise bei den Fachbegriffen und Akronymen aus der Fahrzeugdomäne der Fall. Die resultierenden Worteinbettungen werden mithilfe der Transformationsmatrix in ihre englische Repräsentation v_i übersetzt. Danach wird der Wert $tfidf_i$ jedes Wortes berechnet und mit den dazugehörigen transformierten Worteinbettungen multipliziert. Das sorgt dafür, dass den seltenen Wörtern mehr Gewicht verliehen wird. Die so resultierenden Vektoren einer Anforderung werden addiert und durch die Anzahl der Wörter n in dieser Anforderung geteilt. Daraus ergibt sich der Textvektor $textv$ einer Anforderung. Die Gleichung, welche diese Vorgehensweise beschreibt, befindet sich in der Gleichung 4.1.

$$textv = \frac{\sum_{i=1}^{n} tfidf_i \cdot v_i}{n} \tag{4.1}$$

Die Texte der internen Anforderungen werden ebenfalls durch die Gleichung 4.1 in einen Vektor transformiert. Der einzige Unterschied ist, dass die Worteinbettung nicht in die englische Repräsentation übersetzt wird, sondern direkt in der Formel als v_i verwendet wird. Nach der Erstellung aller Textvektoren ergeben sich Paare von externen und internen Vektoren der Texte. Um die Ähnlichkeit zwischen den Textvektoren der Paare messen zu können, wird das Maß Kosinusähnlichkeit verwendet. Dieses Maß bestimmt den Kosinus des Winkels zwischen zwei Vektoren. Der Maß ist nah bei 1, wenn zwei Vektoren in einem Raum gleich orientiert sind, 0, wenn die Vektoren orthogonal sind, und -1, falls die Vektoren entgegengesetzt orientiert sind. Die Kosinusähnlichkeit wird nach folgender Gleichung 4.2 berechnet.

$$\text{Kosinusähnlichkeit} = \cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2} = \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} (a_i)^2} \cdot \sqrt{\sum_{i=1}^{n} (b_i)^2}} \tag{4.2}$$

Die Tabelle 4.5 zeigt fünf externe Anforderungen mit den dazugehörigen internen Anforderungen und den Kosinusähnlichkeit-Maßen. In dem ersten Beispiel beträgt die Ähnlichkeit etwa 0.52, die Texte des Paares haben dabei keinen für Menschen sichtbaren Zusammenhang. Das zweite Beispiel zeigt ein Paar mit gleicher Thematik, die Ähnlichkeit beträgt dabei etwa 0.70. Beim dritten Beispiel ergibt sich für einen Menschen kein sichtbarer Zusammenhang, obwohl die Ähnlichkeit etwa 0.73 beträgt. Das vierte Beispiel zeigt zwar ein Paar an, aber die Ähnlichkeit zwischen den beiden Texten beträgt nur etwa 0.27. Beim fünften Beispiel besteht wieder kein sichtbarer Zusammenhang, obwohl die Ähnlichkeit etwa 0.71 beträgt.

Durch die Ergebnisse der Ähnlichkeiten kann keine eindeutige Textähnlichkeit als Weg für die Abbildungen hergestellt werden. Es wird angenommen, dass die Fachbegriffe in den Anforderungen eine sehr große Rolle spielen. Insbesondere Texte mit geringer Länge und niedriger Abstraktionsstufe hängen sehr stark von den Fachbegriffen ab. Daher kann der Ansatz der Textähnlichkeiten nur dann brauchbare Ergebnisse liefern, wenn man für die

Tabelle 4.5: Kosinusähnlichkeit zwischen externen und internen Anforderungen

Extern	Intern	Kosinusähnlichkeit
Der für das Fahrzeugprojekt gültige A-IX mit den zugrundeliegenden Vernetzungsumfängen und Parametern sind für die Vernetzungsumsetzung bindend.	Link to A-IX 3.0: tcm: SW-Archives/A-1/Start-Stop/122	0.52870374
Für sich während der Fahrzeugnutzungsdauer verändernden Fahrzeugparameter ist ein Langzeitabgleich vorzusehen (z.B. für Lenkwinkelsensor, Reifentoleranzabgleich, ...).	The tolerance compensation shall compensate differences in wheel circumference, so that all wheel speeds of a straight ahead free rolling vehicle are the same.	0.70501932
Die wichtigsten Parameter unterliegen während der Entwicklung ständigen Änderungen, deshalb müssen diese ohne Logikänderung verstellbar sein.	In case the combustion engine has already reached a speed threshold, then the holding phase shall be finished immediately and the release phase shall be started.	0.7340458
ZZA Grenzsteigung: ZZA_Grenzsteig = 20	The internal information ZZA_Trigger is set to False if all the above mentioned permanent activation conditions and all of the following conditions are fulfilled.	0.2650704
Mindestmotordrehzahl nach Motorstopp, bis Motor sicher läuft: A1_min = 270 1/min	In case the combustion engine has already reached a speed threshold, then the holding phase shall be finished immediately and the release phase shall be started.	0.71462857

Fachbegriffe ebenfalls Worteinbettungen besitzen würde. Für das Trainieren eines Modells benötigt man allerdings eine umfassende Datenbereinigung und eine größere Menge an Daten. Desweiteren müssten die Transformationsmatrizen neu erstellt werden.

4.7 Erkennung von Fast-Duplikaten

Manche externe und interne Anforderungen werden in dem DOORS-System kopiert. Beispielsweise wenn Fehler in den Anforderungen gefunden werden, werden diese oftmals neu erstellt, statt verbessert. Zusätzlich werden bei neuen Projekten für interne Anforderungen neue Module erzeugt und die bestehenden Anforderungen werden dort hineinkopiert. Dadurch entstehen in der Datenbank redundante Informationen. Was das Problem besonders kritisch macht, ist das Fehlen des Verweises zu der ursprünglichen Anforderung. Das bedeutet, dass sich in den Daten eine unbekannte Anzahl von Fast-Duplikaten befindet. Insbesondere für interne Anforderungen kann daraus für die automatische Abbildung ein großes Problem werden, da man mehrere gleiche Vorschläge als unterschiedlich behandelt. Um dieses Problem zu vermeiden, wird das Simhash-Verfahren von Manku et al. [MJDS07] verwendet, um Duplikate in den internen Anforderungen zu finden. Dafür werden die zuvor analysierten 30 593 Anforderungspaare betrachtet. Für jede interne Anforderung wird iteriert und jeder Text der internen Anforderung wird mithilfe von Simhash in einen Hashwert umgewandelt. Bei Simhash werden aus einem Text alle Trigramme bestimmt und es wird ein deterministischer Gesamt-Hashwert für diese Menge an Trigrammen gebildet. Der Gesamt-Hashwert wird basierend auf die Hamming-Distanz zu den Gesamt-Hashwerten der anderen Texte in eine Liste einsortiert. Die Hamming-Distanz ist ein Maß für das Messen des Unterschieds zweier Zeichenketten x und y und wird nach folgender Gleichung 4.3 berechnet.

$$h(x, y) := \| \{j \in \{1, \ldots, n\} \mid x_j \neq y_j\} \| \tag{4.3}$$

In der enstandenen Liste befinden sich die Gesamt-Hashwerte mit kleinen Hamming-Distanzen nah beieinander. Danach definiert man einen Schwellenwert für maximale zulässige Hamming-Distanz für ein Duplikat. Laut Manku et al. eignet sich der Schwellenwert von 3 für die Texte von kleiner bis mittlerer Länge. Alle Gesamt-Hashwerte mit Hamming-Distanz weniger als 3 werden als Duplikate betrachtet.

Die Tabelle 4.6 zeigt repräsentative Resultate der Duplikatenerkennung. Im Feld „Intern Text 1“ befindet sich ein Text einer internen Anforderung. Im Feld „Intern Text 2“ befindet sich der Text einer anderen Anforderung, welche später eingefügt wurde, aber gemäß

Tabelle 4.6: Texte mit gleichem Simhashwert

Intern Text 1	Intern Text 2
Reference -03123	Reference 03123
In case a gearstep to neutral position is recognized the deactivation timer A1_gear_th is set to a tunable time.	In case a gearstep from backward to neutral position is recognized the deactivation timer A1_gear_th is set to a tunable time.
ZZA can be active if velocity is greater than Parameter: P_ABB_Max_1 (Current value in DCMs is 1) An active ABB will be aborted when the velocity is goes below Parameter: P_ABB_Max_1 (Current value in DCMs is 0)	SZZA and Velocity threshold: ZZA can be active if velocity is greater than Parameter: P_ABB_Max_1 (Current value in DCMs is 1) An active ABB will be aborted when the velocity is goes below Parameter: P_ABB_Max_1 (Current value in DCMs is 0)"
- The signal ESC_Konsistenz_1=Konsistent in feedback channel has to be set.	ZZA and Velocity threshold: -The signal ESC_Konsistenz_1=Konsistent in feedback channel has to be set.
Byte 10 Actuator ID $123 - Brake Pedal Appliance	Format: Byte 10 Actuator ID $123 - Brake Pedal Appliance

der Hamming-Distanz mit dem Schwellenwert 3 als ein Duplikat erkannt wurde. Im ersten Beispiel wurde im Nachhinein beim Kopieren der Anforderung der Bindestrich gestrichen. Im zweiten Beispiel wurde die Beschreibung der Anforderung leicht umgeändert, dadurch, dass der Startzustand "from backward" dazugekommen ist. Das dritte Beispiel zeigt die Präzisierung der Anforderung, indem eine Überschrift eingefügt wurde. Im vierten Beispiel wurde der Bindestrich um ein Leerzeichen verschoben. Das fünfte Beispiel zeigt eine Präzisierung mit einer Überschrift.

Bei allen von den Beispielen handelt es sich um Duplikate. Die Duplikatenerkennung mithilfe des Simhash-Ansatzes funktioniert dementsprechend zuverlässig. Dadurch ergibt sich die Möglichkeit die Zielmenge der internen Anforderungen zu verkleinern, indem die Duplikate zusammengefasst werden.

5. Implementierung

In diesem Abschnitt wird zuerst die Vorgehensweise definiert. Danach wird die Implementierung der einzelnen Schritte der Vorgehensweise erklärt.

5.1 Vorgehensweise

Zuerst wird die Zielmenge der internen Anforderungen verkleinert. Dies geschieht durch die Entfernung der Duplikate und danach durch die Zusammenfassung von 1-zu-n Abbildungen, welche nur einmalig in den Daten vorkommen.

Danach werden die Verfahren des maschinellen Lernens auf den bestehenden Datenfeldern aus dem Abschnitt 4.3.1 getestet, um einen Überblick über die Möglichkeiten der Klassifikation zu verschaffen und erste Richtlinien zu erstellen. Die Merkmale sollen dabei aus den Datenfeldern der externen Anforderungen generiert werden, während eine Klasse aus den Informationen der internen Anforderungen generiert wird. Als mögliche Klassen eignen sich beispielsweise die Modulinformationen, Identifikationsnummern oder die Begriffe der internen Anforderungen. Die Klassifikationen werden mit den Naïve-Bayes- und Random-Forest-Klassifikatoren durchgeführt. Danach soll mithilfe einer Datenstruktur die Wahl der Merkmale verfeinert werden und die Klassifikation mit den bestehenden Richtlinien verglichen werden.

Die Datenstruktur wird in Form einer Ontologie modelliert. Um die Ontologie zu erstellen werden aus den Anforderungen mögliche Konzepte und Beziehungen extrahiert. Um die Fachbegriffe zu extrahieren, werden die Lexika von Bosch verwendet und die Anforderungen nach Begriffen aus den Lexika durchgesucht. Die gefundenen Fachbegriffe werden nach ihren Arten gruppiert und als Konzepte in der Ontologie modelliert. Zusätzlich zu den Fachbegriffen werden mithilfe des Computerlinguistik-Fließbands weitere Begriffe extrahiert, welche nicht in den Lexika vorkommen. Um die Beziehungen der Begriffe zu finden, werden mithilfe des Computerlinguistik-Fließbands Adjektive und Verben gesucht, welche auf die gefundenen Fachbegriffe zutreffen. Zusätzlich zu den Fachbegriffen und ihren Beziehungen sollen die Metainformationen in der Ontologie gespeichert werden. Die Module, Überschriften und Identifikationsnummern der Anforderungen werden als Konzepte modelliert.

Nach der Erstellung der Ontologie werden die extrahierten Konzepte nach gemeinsamen Wortanteilen gruppiert. Dafür werden neue Konzepte erstellt, die unter sich Konzepte mit

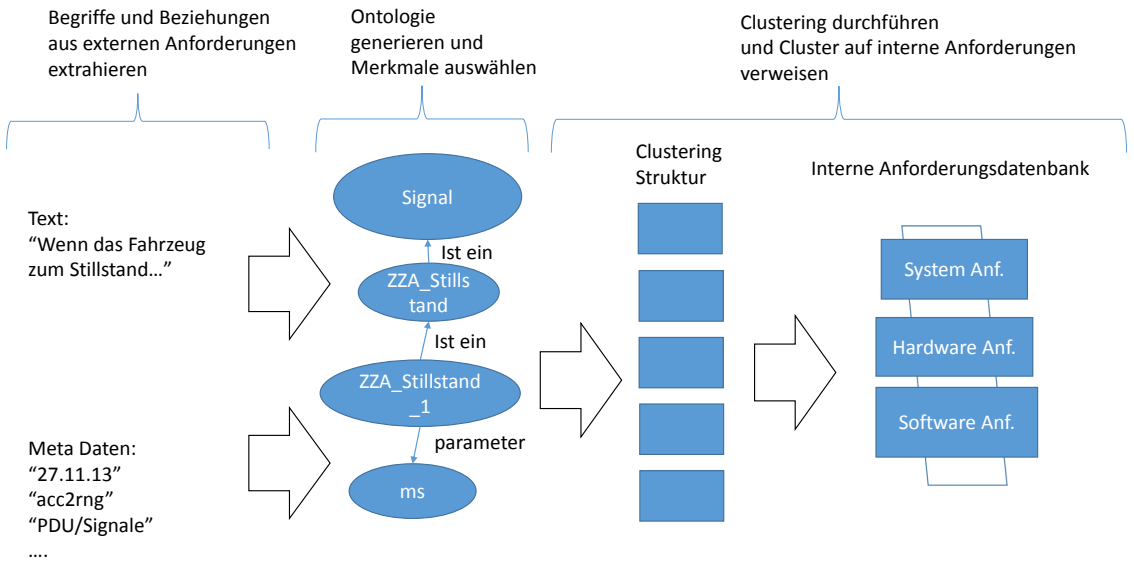


Abbildung 5.1: Vorgehensweise der Implementierung

ähnlichen Bestandteilen repräsentieren wie beispielsweise alle Ausprägungen eines Signals. Diese neuen Konzepte sollen als neue Merkmale verwendet werden.

Danach wird Hierarchisches Clustering für die Erzeugung der Vorschläge der Abbildungen verwendet. Dafür werden für jede externe Anforderung ihre zutreffenden Konzepte und Relationen aus der Ontologie als Merkmale kodiert. Mit diesen Merkmalen wird das Clustering durchgeführt. Dadurch lassen sich Gruppen von ähnlichen externen Anforderungen generieren, bezogen auf ihre Fachbegriffe und Beziehungen. Wenn man eine neue externe Anforderung enthält, werden daraus alle Fachbegriffe und Beziehungen extrahiert. In der Ontologie werden die passenden Oberkonzepte gesucht. Anschließend werden für diese Anforderung Merkmale generiert und die Anforderung wird einem Cluster zugeordnet. Danach werden für jede externe Anforderung innerhalb des Clusters ihre zutreffenden internen Anforderungen ausgegeben. Die Ausgabe der internen Anforderungen stellt die Vorschläge für den Mitarbeiter dar.

Die Abbildung 5.1 zeigt die Vorgehensweise. Die Textinformationen der externen Anforderungen werden in die Ontologie extrahiert und die Metainformationen werden damit verknüpft. Aus der Ontologie werden Merkmale generiert, die mithilfe von Clustering gruppiert werden. Danach werden für jedes Cluster die internen Anforderungen aus den darin enthaltenen externen Anforderungen bestimmt.

5.2 Behandlung von Duplikaten

In dem Abschnitt 4.7 wurde die Erkennung von Duplikaten in den Daten analysiert. Die Erkennung funktionierte zuverlässig, deshalb werden die Duplikate in den internen Anforderungen zusammengefasst. Zuerst wird für jede interne Anforderung $Intern_i$ eine Menge gebildet, welche ihre Duplikate und Fast-Duplikate enthält. Jede Anforderung in dieser Menge wird durch $Intern_i$ ersetzt und die darauf zutreffenden Abbildungen angepasst, sodass die externen Anforderungen auf $Intern_i$ abbilden. Durch dieses Verfahren verkleinert sich die, im Abschnitt 4.3 ermittelte, Zielmenge der internen Anforderungen von 15 093 auf 9 866.

Tabelle 5.1: 1-zu-100-Abbildung

Extern Text	Intern Text
Wird bei aktiver ABB - Funktion ein zurückrollen des Fahrzeugs entgegen der Fahrtrichtung...	BBC shall prevent the vehicle from rolling...
	While A is active, the active pressure...
	A must be able to detect a sliding vehicle during drive...
	B must be able to detect skidding when making...
	BBC must detect skidding of the vehicle during...
	Hold functions must be able to detect skidding...
	Skid should be detected by O...
	When a skidding situation is detected...
	BBC must be able to detect...
	BBC must be able to detect...
	BBC must be able to detect...
	BBC must be able to detect...
	If a valve failure occurs then the pressure...
	If a valve failure occurs then the pressure...
	If a valve failure occurs then the pressure...
	When AA-S holds actively a vehicle - it must...
	When AA-S holds actively a vehicle - it must...
	Use case: Vehicle is rolling due to...

5.3 Behandlung von einmaligen 1-zu-n Abbildungen

Bei der Analyse der Daten in dem Abschnitt 4.3 fielen externe Anforderungen auf, die auf eine Vielzahl von internen Anforderungen abbilden. Auf der Tabelle 5.1 wird ein Ausschnitt einer externen Anforderung gezeigt, welche mehr als auf 100 interne Anforderungen abbildet. Die Spalte „Extern Text“ enthält den Text von der externen Anforderung. Die Spalte „Intern Text“ enthält die Texte der internen Anforderungen, auf welche die externe Anforderung abgebildet wurde. Die 1-zu-100-Abbildung ist dadurch zu erklären, dass der Text der externen Anforderung lang ist und die Thematik von mehreren internen Anforderungen abdeckt. Solche Fälle treten in den Daten allerdings nicht oft auf, sodass nur eine bestimmte externe Anforderung auf bestimmte 100 interne Anforderungen abgebildet wird. Diese 100 interne Anforderungen tauchen bei keiner anderen Abbildung in den Daten auf. Somit können diese 100 interne Anforderungen als eine einzige interne Anforderung betrachet werden. Um solche einmalig vorkommenden Gruppen zu finden, werden alle Abbildungen überprüft. Falls eine einmalige Gruppe gefunden wird, so werden alle Identifikationsnummern von den internen Anforderungen der Gruppe gesammelt. Die Gruppe bekommt eine eigene Identifikationsnummer und wird als eine neue interne Anforderung behandelt. Die Abbildungen werden dementsprechend angepasst. Demnach wird die Abbildung aus der Tabelle 5.1 von 1-zu-100- zu einer 1-zu-1-Abbildung. Durch die Identifizierung von solchen Gruppen wird die Menge der internen Anforderungen von 9 866 weiter auf 7 210 verkleinert.

5.4 Richtlinienerstellung

Die folgenden Klassifikationsansätze zielen darauf ab, Vergleichswerte für die spätere Ontologiebasierende Erzeugung der Vorschläge zu schaffen.

Tabelle 5.2: Klassifikation, basierend auf die URL, mithilfe der Merkmale aus allen Feldern

Klassifikator	Präzision	Ausbeute	F-Maß
Naïve-Bayes	0.07	0.08	0.07
Random-Forest	0.15	0.18	0.16

5.4.1 Alle Felder, URL-Klassen

Bei diesem Ansatz wird versucht, aus den Informationen der Felder der externen Anforderungen auf die darauf abgebildete URLs der internen Anforderungen zu schließen. Die zur Verfügung stehenden Daten bestehen aus insgesamt 30 593 externen Anforderungen, welche auf 7 210 interne Anforderungen abgebildet werden. Um die Lerndaten zu erstellen, werden die 30 593 Anforderungen nach ihren Projekten getrennt und zeitlich nach den Informationen aus dem Feld **Created On** sortiert.

Insgesamt existieren zehn Projekte. Die ersten sieben Projekte werden als Lerndaten verwendet, was einer Anforderungszahl von 24 115 entspricht. Die aktuellsten drei Projekte mit insgesamt 6 478 Anforderungen werden als Testdaten verwendet. Diese Unterteilung ist realistischer als eine zufällige Wahl der Testdaten aus der Gesamtmenge, da die Anforderungen aus den zeitlich früheren Projekten keine Informationen über die zeitlich späteren Projekte enthalten können. Die Mitarbeiter beziehen ihr Wissen aber aus den früheren Projekten, um aktuelle Projekte abzubilden.

Für die Klassifikation wird die **python**-Bibliothek **scikit-learn** [PVG+11] verwendet. Als Klassifikationsverfahren werden der Naïve-Bayes-Klassifikator und der Random-Forest-Klassifikator verwendet. Als Merkmale werden die Informationen aus allen Feldern verwendet, die im Abschnitt 4.3.1 präsentiert wurden. Um die Merkmale für die Klassifikatoren verwenden zu können, müssen diese als Zahlenwerte repräsentiert werden. Die kategorischen Daten werden in binäre Vektoren übersetzt. Beispielsweise werden drei mögliche Autoren acc1rng, bcc2rng und ccd3rng im Feld **Created By** zu Vektoren 100, 010 und 001 kodiert. Um die Textdaten der Anforderungen für die Klassifikation zu berücksichtigen, werden diese ebenfalls in Vektoren transformiert. Dafür werden die Texte aus dem Feld **Object Text** mit Bag-of-Words- und Vorkommenshäufigkeit-und-inverse-Dokumenthäufigkeit-Verfahren zu Vektoren kodiert. Dabei werden keine Wörter ausgelassen. Die URLs der internen Anforderungen werden als Klassen für die Voraussage verwendet.

Um die Klassifikation zu evaluieren wurden die vordefinierten Maße Präzision, Ausbeute und F-Maß verwendet. Die Ergebnisse befinden sich in der Tabelle 5.2. Es zeigen sich F-Maß Werte von 0.07 und 0.16, wobei der Random-Forest-Klassifikator bessere Werte liefert. Diese Werte sind dennoch schlecht, da nicht mal 20% der Anforderungen korrekt klassifiziert werden. Die späteren Implementationen sollten diese Werte übertreffen.

5.4.2 Alle Felder, Modul-Klassen

Statt den URLs als Klassen wie im Abschnitt 5.4.1, wird das **Module**-Feld als Klasse verwendet. Das **Module**-Feld enthält das Modul einer internen Anforderung. Mehrere interne Anforderungen besitzen das gleiche Modul, was die Klassenmenge sehr stark von 7 210 auf 514 einschränkt. Die ausgewählten Merkmale der Anforderungen bleiben die gleichen wie im Abschnitt 5.4.1 beschrieben. Die Klassifikation wird erneut durchgeführt mit folgenden Ergebnissen: Tabelle 5.3. Die Daten werden dabei wie im Abschnitt 5.4.1 als Lern- und Testdaten getrennt.

Tabelle 5.3: Klassifikation basierend auf die Module, mithilfe der Merkmale aus allen Feldern

Klassifikator	Präzision	Ausbeute	F-Maß
Naïve-Bayes	0.61	0.59	0.60
Random-Forest	0.69	0.68	0.68

Tabelle 5.4: Anzahl interner Anforderungen innerhalb eines Moduls

Module	Anzahl
A1_111111_Functions_ESC	3919
A1_1111111_Functions_Diagnosis	1837
A1_111111_Functions_Hold_functions	1663
A1_111111_Functions_Information_Management	1171
A1_1111111_Functions_Spezifikation	833
A1_2222222_Functions_ABB	33
A1_3333333_Functions_ABB	33
A1_3333333_Functions_IX	15

Trotz der guten Ergebnisse ist die automatische Klassifikation basierend auf das Modul als Unterstützung für einen Mitarbeiter oft nicht hilfreich. Wie in der Tabelle 5.4 zu sehen, enthalten manche Module bis zu 4 000 interne Anforderungen. Somit ist ein Vorschlag eines Moduls bei einer neuen externen Anforderung nicht sinnvoll, wenn es auf ein Modul mit Vielzahl von internen Anforderungen zeigt. Des weiteren fiel auf, dass das Ergebnis der Klassifikation stark von den Attributen **Created On** und **Created By** abhängt. Wenn man diese Attribute aus dem Merkmalsvektor entfernt, dann fällt der F-Maß unter 50%. Das zeigt eine starke Abhängigkeit der Voraussage von dem Autor und dem Datum einer Anforderung. Das ist nicht vorteilhaft, da diese beiden Werte als Metadaten durch den DOORS-Import entstehen und deshalb nicht zu den Informationen einer neuen Anforderung gehören.

5.4.3 Eigennamen, URL-Klassen

Die dritte Richtlinie wird durch die Verfeinerung der Merkmale erzeugt. Dafür werden die Eigennamen aus den Anforderungen extrahiert und als Merkmale verwendet. Dadurch lässt sich möglicherweise das Rauschen in den Texten der Anforderungen umgehen.

5.4.3.1 Eigennamenerkennung

Die Fachbegriffe in den Anforderungen enthalten viel Information und sind einer der Gründe, warum die Ansätze der Textähnlichkeiten in Abschnitt 4.6, welche mit der Alltagssprache trainiert sind, nicht funktionieren. Um die Fachbegriffe automatisch zu erkennen, wird mithilfe der **python**-Bibliothek **spacy** [HM17] ein eigenes Model erstellt. Bei spacy gibt es die Möglichkeit, einen Übereinstimmer zu definieren, welcher alle Wörter des Textes markiert, die einem gewissen Muster folgen. Von Bosch wurde ein Lexikon bereitgestellt, das die Fachbegriffe enthält, die in den Anforderungen des Kunden A benutzt werden. Aus dem Lexikon lassen sich die Abkürzungen einzelner Systeme, Processing Digital Units(PDU) genannt, rauslesen und auch die dazugehörigen Signale. Um die PDU und Signale zu identifizieren, werden für den Übereinstimmer automatisch Wortlisten definiert. Dafür werden alle Begriffe der PDU und Signale des Lexikons extrahiert und in Form einer Liste be-

Tabelle 5.5: Eigennamen aus den Anforderungen

Bezeichnung	Regel	Beispiel
Akronym	Ein Wort ist ein Akronym, falls es mindestens drei hintereinander folgende Großbuchstaben enthält	ESP
Akronym	Ein Wort ist ein Akronym, falls es zusätzlich durch Unterstrich oder Trennstrich getrennt ist	ABB1-Anforderung v_B_max
PDU	Ein Wort ist ein PDU, falls es in dem PDU-Lexikon vorkommt	A1_ESP_2200
Signal	Ein Wort ist ein Signal, falls es in dem Signal-Lexikon vorkommt	ZZA_aktiv_1

Tabelle 5.6: Evaluation des Modells der Eigennamenerkennung

Modell	Präzision	Ausbeute	F-Maß
Eigennamenerkennung	0.91	0.85	0.87

reitgestellt. Da nicht alle Abkürzungen mit PDU und Signalen abgedeckt werden, werden händisch weitere Regeln definiert. Diese Regeln sind in der Tabelle 5.5 zu sehen.

Mit den Übereinstimmer-Mustern lassen sich die Texte der Anforderungen automatisch mit den Bezeichnungen markieren. Um ein eigenes Modell für **spacy** zu erzeugen, werden die markierten Texte als Lerndaten verwendet. Mit diesen Lerndaten trainiert man ein **spacy**-internes neuronales Netz, welches die Eigennamenerkennung durchführt. Dies ist von Vorteil, da die Nachbarschaft der Begriffe gelernt wird und dadurch mehr gefunden werden kann als nur mit einem Übereinstimmer. Die Abbildung 5.2 zeigt die erkannten Entitäten der Eigennamenerkennung in einer Anforderung. So wurden mehrere Akronyme identifiziert und jeweils ein Signal und ein PDU.

Um das entstandene Modell zu evaluieren, werden die vordefinierten Maße Präzision, Ausbeute und F-Maß verwendet. Die 30 593 vorliegenden Texte der Anforderungen wurden dabei in 70% Training- und 30% Testdaten zufällig aufgeteilt. Die Ergebnisse befinden sich in Tabelle 5.7. Die Eigennamenerkennung zeigt gute Resultate, was das F-Maß von 0.87 deutlich zeigt.

Eigennamenerkennung

Die hydraulische Druckunterstützung ist eine Erweiterung der Schutzfunktion mit Hilfe der A1_ABB-Notbrems für folgende Anwendung: Wird das Fahrzeug am Hang durch die Betriebsbremse gehalten und die eingelegte Gangstufe verhindert nicht das Zurückrollen beim Lösen der Betriebsbremse und der Schalter der ABB wird bis zu einer Grenzgeschwindigkeit in Richtung „schließen“ betätigt, muss durch den aktiven ZZA_aktiv_1 die Parkbremse vor Überlastung geschützt werden. Die ABB sendet eine D-Verzögerungsanforderung an das ESP.

- Akronyme
- Signale
- PDU

Abbildung 5.2: Eigennamenerkennung in einer Anforderung

Tabelle 5.7: Klassifikation basierend auf Eigennamen und URLs

Klassifikator	Präzision	Ausbeute	F-Maß
Naïve-Bayes	0.41	0.49	0.44
Random Forest	0.53	0.55	0.54

5.4.3.2 Klassifikation

Als Merkmale der Klassifikation werden die Eigennamen aus dem Abschnitt 5.4.3.1 verwendet. Die Eigennamen werden als ein Bag-of-Words-Vektor kodiert. Zusätzlich zu den Eigennamen werden alle Nomen einer Anforderung mithilfe des Computerlinguistik-Fließbands von **spacy** identifiziert. Diese Nomen werden als Bag-of-Words-Vektor kodiert und als Merkmale verwendet. Die Klassen sind die URLs der internen Anforderungen. Die Daten werden wie im Abschnitt 5.4.1 als Lern- und Testdaten getrennt. Die Ergebnisse befinden sich in der Tabelle 5.7. Durch die Beschränkung der Merkmale auf die Eigennamen und Nomen wird ein F-Maß von 0.54 erreicht. Das ist erheblich besser als bei dem Ansatz aus dem Abschnitt 5.4.1, bei welchem alle Felder als Merkmale verwendet werden. Es ist vorteilhafter, die Merkmale einzuschränken, statt alle Felder als Merkmale zu verwenden.

5.5 Ontologie-Erstellung

Für die Erstellung der Ontologie werden mithilfe des Fließbands der Computerlinguistik aus den Texten der externen Anforderungen Konzepte und ihre Beziehungen extrahiert. Hierbei werden die Texte der Anforderungen der ersten sieben Projekte für die Erstellung der Ontologie genommen, sodass die Anzahl der verwendeten externen Anforderungen 24 115 beträgt. Die restlichen 6 478 Anforderungen werden für die Evaluation der Ontologie-basierenden Erzeugung der Vorschläge zurückgehalten.

Die Texte der internen Anforderungen werden nicht betrachet. Um die Texte der internen Anforderungen zu betrachten, muss das Problem der Zweisprachigkeit zuerst gelöst werden, was mit bisherigen Ansätzen in Abschnitt 4.4 und Abschnitt 4.6 aufgrund der Beschaffenheit der Anforderungen nicht möglich ist. Die Metainformationen der internen Anforderungen werden aber für die Ontologie verwendet, beispielsweise ihre Identifikationsnummern. Damit werden die Informationen der Abbildungen auch in der Ontologie aufrechterhalten.

5.5.1 Konzeptextraktion

Ähnlich wie bei Roth et al. [RK15] werden im ersten Schritt mögliche Konzepte für die Ontologie identifiziert und extrahiert. Die definierten Eigennamen aus dem Abschnitt 5.4.3.1 werden als Konzepte verwendet: Akronym, Signal und PDU. Diese drei Konzepte werden an den Wurzelknoten angehängt. Alle mithilfe der Eigennamenerkennung extrahierten Begriffe werden unter den drei Konzepten als Unterkonzepte angehängt.

Danach wird ein Konzept Term erstellt, welches als Oberkonzept für alle Nomen in den Anforderungen verwendet wird, welche nicht durch die Eigennamenerkennung identifiziert werden können. Dieses Konzept wird an den Wurzelknoten angehängt. Um die Nomen zu finden, werden mithilfe des Etikettierers von **spacy** alle Nomen in der Anforderung identifiziert. Von den identifizierten Nomen wird mithilfe des Grundformerermittlers von spacy die Grundform ermittelt. Danach wird das Konzept als Unterkonzept von Term angehängt. Die Ermittlung der Grundform ist wichtig, um Duplikate in den Konzepten zu vermeiden. Beispielsweise identifiziert der Etikettierer die Nomen „Fahrzeugen“ und „Fahrzeugs“. Durch die Ermittlung der Grundform ergibt sich daraus das Nomen „Fahrzeug“, welches dann als Konzept verwendet wird.

Aus den Metainformationen einer Anforderung werden die Überschrift und die Modulbezeichnung extrahiert. In der Ontologie werden zwei Konzepte Überschrift und Modul erstellt, welche an den Wurzelknoten angehängt werden. Die extrahierten Überschriften und Module werden unter diese beiden Konzepte angehängt.

Des Weiteren werden zwei Konzepte ”Externe Anforderung” und ”Interne Anforderung” erstellt. Diese Konzepte werden an den Wurzelknoten angehängt und werden später im Abschnitt 5.5.3 für die Instanzen verwendet.

Alle bisher beschriebenen Konzepte werden in dem Abbildung 5.3. Am Anfang befindet sich der Wurzelknoten, von dem acht Unterkonzepte PDU, Signal, Akronym, Term, Überschrift, Modul, „externe Anforderungen“ und „interne Anforderungen“ ausgehen. In jedem der acht Konzepte werden die extrahierten Begriffe untergeordnet. Beispielsweise werden durch die Eigenamenerkennung zwei Signale „ZZA_aktiv_1“ und „ZZA_aktiv_2“ extrahiert und dem Konzept Signal untergeordnet. Zwei durch den Etikettierer erkannten Nomen „Fahrzeug“ und „Bremsstärke“ werden dem Konzept Term untergeordnet. Die Identifikationsnummern der Anforderungen, aus welchen die Begriffe extrahiert wurden, werden als Instanzen von Konzepten „externe Anforderung“ und „interne Anforderung“ modelliert.

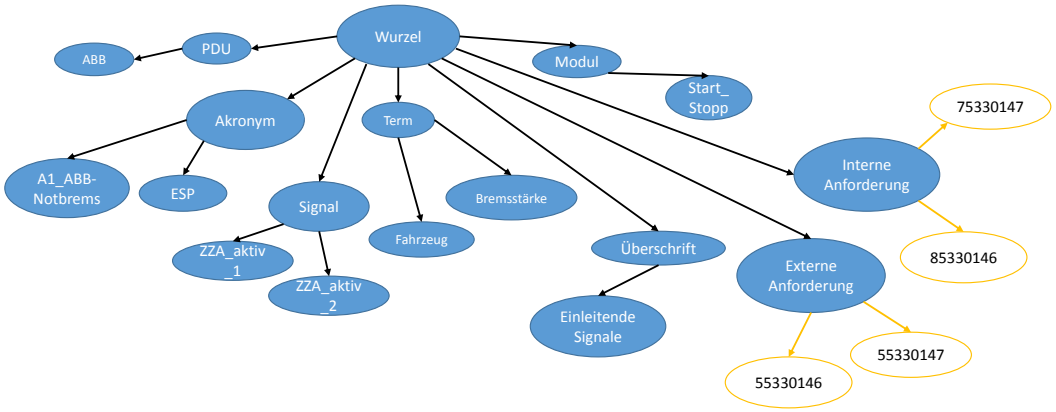


Abbildung 5.3: Ontologie-Konzepte mit extrahierten Begriffen

5.5.2 Relationenextraktion

Im zweiten Schritt werden die Relationen zwischen den Konzepten identifiziert. Nach Roth et al. [RK15] besitzen die Verben zwischen den Subjekten und Objekten eine starke Aussagekraft. Auch in der Domäne der Fahrzeuganforderungen beschreiben Verben wichtige Informationen, wie zum Beispiel die Bedingungen für die Aktivierung eines Signals. Um die Verben zu extrahieren, werden mit dem Etikettierter alle Verben in einer Anforderung identi

fiziert.Danach wird mithilfe der Syntaxanalyse von **spacy** geprüft, ob sich ein Verb auf ein Subjekt oder ein Objekt bezieht.

Um die Subjekte und Objekte zu finden, werden die extrahierten Nomen aus dem Abschnitt 5.5.1 mithilfe der Syntaxanalyse verarbeitet. Die Syntaxanalyse weißt jedem Nomen seine syntaktische Bedeutung zu, beispielsweise Subjekt oder Prädikat.

Um die Verben in der Ontologie darzustellen, wird ein Konzept Verb als ein Unterkonzept von der Wurzel erzeugt und alle extrahierten Verben werden darunter angehängt. Als Name des Verbs wird seine Grundform verwendet. Danach werden zwei Relationen erstellt, „Akteuer von“ und „Agiert auf“. Alle Subjekt-Verb-Beziehungen werden als Unterrelationen von „Akteuer von“ modelliert, indem diese Relation zwischen dem Konzept des Subjekts und dem Konzept des Verbs erstellt wird. Alle Verb-Objekt-Beziehungen werden als Unterrelationen von „Agiert auf“ modelliert, indem diese Relation zwischen dem Konzept des Verbs und dem Konzept des Objekts erstellt wird. Dadurch erhält man eine eindeutige Zuordnung zwischen den Nomen und Verben.

Die Abbildung 5.4 zeigt einen Ausschnitt mit diesen Relationen. Hierbei wird zu der bestehenden Ontologie aus der Abbildung 5.3 das Konzept Verb hinzugefügt. Zwischen den Konzepten PDU, Signal, Term und Akronym und dem Konzept Verb werden die Relationen „Akteuer von“ und „Agiert auf“ erstellt. Im Auschnitt wird nur die Modellierung für die Konzepte PDU und Term gezeigt. Aus einer Anforderung wurde das Verb „aktivieren“ extrahiert. In dieser Anforderung ist ein PDU „ABB“ als Subjekt enthalten, welches die Beziehung „Akteur von“ zum „aktivieren“ bekommt. Des Weiteren ist dort der Term „Notbremsung“ als Objekt enthalten, welches die Beziehung „Agiert auf“ zum „aktivieren“ bekommt.

Des Weiteren sind die Adjektive von Bedeutung. In der Anforderungsdomäne beschreiben die Adjektive die Eigenschaften der Begriffe, wie zum Beispiel die möglichen Zustände eines

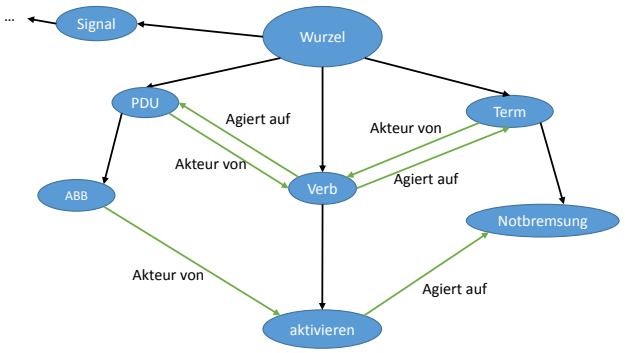


Abbildung 5.4: Ontologie-Konzepte mit Verb-Relationen

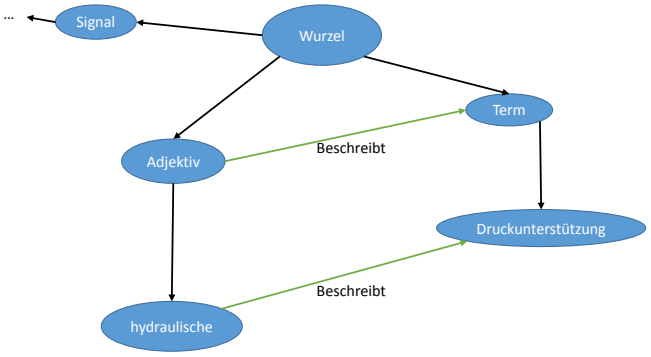


Abbildung 5.5: Ontologie-Konzepte mit Adjektiv-Relationen

Signals. Um die Adjektive zu identifizieren, werden diese mit dem Etikettierter erkannt. Danach wird mithilfe der Syntaxanalyse ihre Beziehung zu einem bestimmten Subjekt oder Objekt überprüft. Die Adjektive werden in der Ontologie ähnlich wie die Verben dargestellt. Es wird ein Konzept Adjektiv erstellt, welches alle extrahierten Adjektive als Unterkonzepte enthält. In den Relationen wird eine Relation „Beschreibt“ angelegt. Alle Adjektiv-Subjekt- und Adjektiv-Objekt-Beziehungen werden als Unterrelationen von „Beschreibt“ modelliert, indem diese Relation zwischen dem Adjektiv und dem Subjekt bzw. Objekt erstellt wird.

Die Abbildung 5.5 zeigt die Adjektiv-Relationen. Zwischen dem Konzept Adjektiv und den Konzepten PDU, Signal, Term und Akronym wird die Relation „Beschreibt“ modelliert. Aus einer Anforderung wurde das Adjektiv „hydraulisch“ extrahiert, welches auf den Term „Druckunterstützung“ zutrifft. Es wird eine Relation „Beschreibt“ zwischen „hydraulisch“ und „Druckunterstützung“ erstellt.

Zuletzt wird eine Relation „Abbildet auf“ erstellt, welche zwischen den Instanzen in dem Abschnitt 5.5.3 erstellt wird und die Abbildung von einer externen Anforderung zu einer internen Anforderung repräsentiert. Die Information hierfür wird aus den Abbildungsverweisen zwischen den Anforderungen aus den Daten extrahiert.

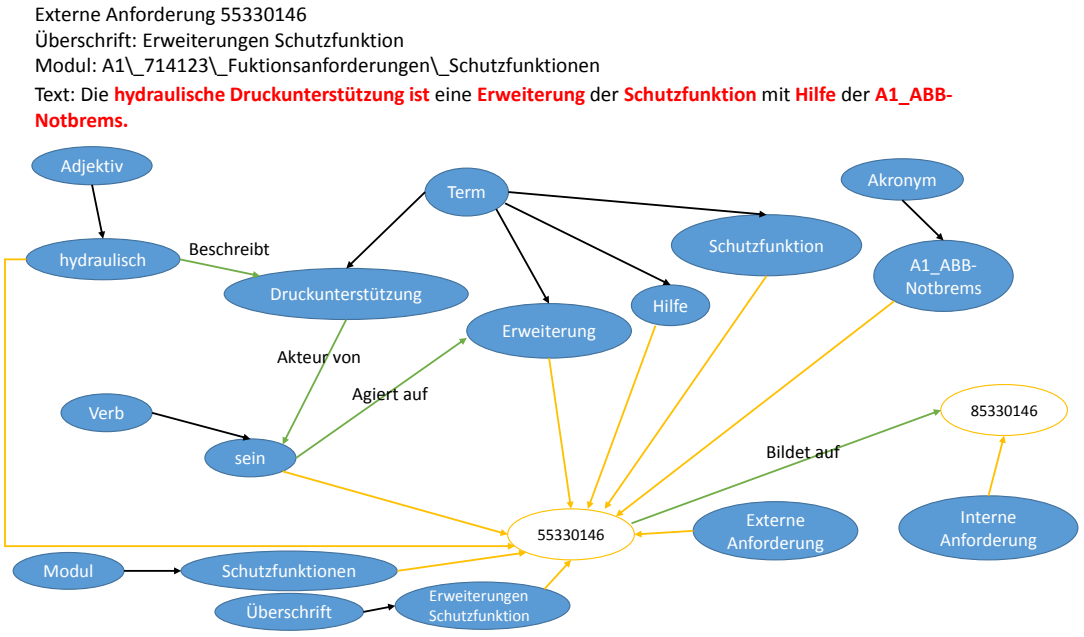


Abbildung 5.6: Ontologie-Auschnitt mit der Instanz einer externen Anforderung

5.5.3 Instanzen

Die bisher extrahierten Konzepte und Relationen aus dem Abschnitt 5.5.1 und dem Abschnitt 5.5.2 werden in der Ontologie repräsentiert, ohne einen Verweis auf die Anforderung, aus welcher sie extrahiert wurden, zu besitzen. Um die Zugehörigkeit von Konzepten und Relationen zu Anforderungen herzustellen, werden die externen und internen Anforderungen als Instanzen modelliert. Die zugehörigen Konzepte und Relationen werden auf eine Instanz verwiesen.

5.5.3.1 Externe Anforderungen

Für jede externe Anforderung wird eine Instanz erzeugt. Diese Instanz bekommt die Identifikationsnummer aus der URL der Anforderung als Namen. Der Text der Anforderung wird als eine Annotation in der Instanz gespeichert. Alle Konzepte und Relationen, welche aus der Anforderung extrahiert wurden, werden auf die Instanz verwiesen. Die Instanz der Anforderung kann als ein Behälter für die darauf zutreffenden Konzepte und Relationen betrachtet werden. Die Instanz enthält somit alle darauf zutreffenden PDU, Signale, Akronyme, Terme, Überschriften, Module, Adjektive und Verben. Da die internen Anforderungen ebenfalls als Instanzen modelliert werden, wird als Typ der Instanz das Konzept „Externe Anforderung“ angeben, um zwischen den internen und externen Instanzen unterscheiden zu können. Des Weiteren wird eine Relation „Abbildet auf“ zwischen der Instanz der externen Anforderung und der Instanz der internen Anforderung erstellt, falls die externe Anforderung auf diese interne Anforderung abbildet.

Ein Beispiel der Instanz einer externen Anforderung befindet sich in der Abbildung 5.6. Die extrahierten Konzepte und Relationen befinden sich in dem Ausschnitt der Ontologie. Die extrahierten Konzepte verweisen auf die Instanz der externen Anforderung. Die Beziehung ”Abgebildet auf” verweist auf die Instanz einer internen Anforderung, auf welche die externe Anforderung abgebildet wurde.

Abbildung 5.7: Beispiel der Konzeptpräfixe der Ontologie-Konzepte

5.5.3.2 Interne Anforderungen

Für jede interne Anforderung wird ebenfalls eine Instanz erzeugt. Diese Instanz bekommt die Identifikationsnummer der Anforderung als Namen. Der Text der internen Anforderung wird als eine Annotation in der Instanz gespeichert. Als Typ der Instanz wird "Interne Anforderung" angegeben.

5.5.4 Gemeinsame Konzeptpräfixe

Nachdem alle Konzepte und Relationen sowie Instanzen erstellt wurden, werden die gemeinsamen Wortbestandteile der Konzepte identifiziert. Dies geschieht auf der Grundlage der gemeinsamen Wortpräfixe der Konzepte. Für die Unterkonzepze von PDU, Signal und Akronym werden die gemeinsamen Wortpräfixe identifiziert.

Beginnend ab einer festgelegten Präfixlänge P werden die Konzepte nach ihren Präfixen gruppiert. Danach wird P um eins vermindert und die Präfixe der Länge P nach ihren Präfixen der Länge $P-1$ gruppiert. Dieses Verfahren wird rekursiv bis zu einer festgelegten Präfixlänge P_{min} fortgesetzt. Danach wird geprüft, ob die Gruppen mit der Präfixlänge P mehr als einen Präfix enthalten. Falls dies der Fall ist, wird aus dem Präfix der Gruppe ein neues Konzept erzeugt, welches unter jeweiliges Konzept PDU, Signal oder Akronym angehängt wird. Die Präfixe in der Gruppe werden als Unterkonzepte daran angehängt. Danach wird mit der Präfixlänge von $P-1$ fortgefahren. Dieses Verfahren wird bis zu einer Präfixlänge P_{min} wiederholt.

Die Abbildung 5.7 zeigt ein Beispiel der Präfixfindung bei den Unterkonzepten von Signal. So wurden die Begriffe „zza1_b_fill“ und „zza1_ha_param“ nach dem Präfix „zza1_“ der Länge fünf gruppiert, da diese Gruppe als einzige mehr als einen Begriff enthielt. Die Begriffe „zza2_freigabe“ und „zza2_rückprofil“ wurden nach dem Präfix „zza2_“ der Länge fünf gruppiert. Danach wurden die Präfixe „zza1_“ und „zza2_“ nach ihrem gemeinsamen Präfix „zza“ der Länge drei gruppiert.

Die extrahierten Terme werden nach einem anderen Verfahren gruppiert. Bei jedem Begriff wird geprüft, ob es sich um ein Kompositum handelt. Ein neuronales Netzwerk von Tuggener [Tug16] wird verwendet, welches auf die Erkennung deutscher Komposita trainiert ist. Für jeden Term wird dabei die linke Konstituente ermittelt und die Begriffe werden nach diesen Konstituenten gruppiert. Die Konstituenten werden als Oberkonzepte von den Komposita modelliert. Es werden die linken Konstituenten verwendet, da die Komposita in der Anforderungsdomäne links oftmals die Zugehörigkeit zu bestimmten Fahrzeugkomponenten enthalten. Die Abbildung 5.8 zeigt zwei Gruppierungen anhand gemeinsamer Konstituenten. So wurden die Begriffe „Kodierdaten“, „Kodierschalter“, „Kodierungsplausibilisierung“ und „Kodierwerte“ unter die Konstituente „Kodier“, und die Begriffe „Kombiinstrument“ und „Kombimeldung“, unter die Konstituente „Kombi“ eingruppiert.

Abbildung 5.8: Beispiel der Konstituentenpräfixe der Ontologie-Konzepte

Tabelle 5.8: Bag-of-Words-Vektor für die Konzepte der Anforderung „Die präventive Schutzfunktion gibt ZZA_aktiv_1 aus“

Signale					Terme			
ZZA_ aktiv_1	ZZA_ aktiv_2	ZZA_ aktiv	ZZA	...	Schutz	Schutz funktion	Not bremse	...
1	0	1	1	...	1	1	0	...

5.6 Merkmalerzeugung aus der Ontologie

Das primäre Zweck der Ontologie in dieser Arbeit ist die Erzeugung von neuen Merkmalen. Für jede Instanz der externen Anforderungen wird ein Merkmalsvektor erstellt. Zuerst werden alle Konzepte der PDU, Signale, Akronyme und Terme jeder Instanz als ein Bag-of-Words-Vektor repräsentiert. Der Bag-of-Words-Ansatz wird verwendet, da es im Abschnitt 5.4.3.1 mit der Kodierung der Eigennamen eine deutliche Verbesserung der Klassifikation gezeigt hat. Dabei werden nicht nur die einzelnen Konzepte, sondern auch ihre Oberkonzepte verwendet, welche in dem Abschnitt 5.5.4 erstellt wurden. Dadurch soll die Unabhängigkeit des Vektors von der exakten Formulierung der Begriffe sichergestellt werden. Die Anforderung mit dem Begriff „ZZA_aktiv_1“ und eine andere Anforderung mit dem Begriff „ZZA_aktiv_2“ erhalten dadurch eine 1 in ihrem Vektor an der Stelle „ZZA_aktiv“. Dadurch erkennt man, dass beide Anforderungen das Signal „ZZA_aktiv“ behandeln. Die Abbildung 5.8 zeigt einen Ausschnitt des Bag-of-Words-Vektors für Signale und Terme einer Instanz einer externen Anforderung. Der Text der Anforderung lautet „Die präventive Schutzfunktion gibt ZZA_aktiv_1 aus“.

Danach werden alle Relationen, die in der Instanz vorkommen als Bag-of-Words-Vektor repräsentiert, indem die Konzepte aus den Relationen als Bezeichnungen verwendet werden. Die Tabelle 5.9 zeigt einen Ausschnitt des Bag-of-Words-Vektors für „Akteur Von“- und „Beschreibt“-Relationen einer Instanz einer externen Anforderung. Der Text der Anforderung lautet „Die präventive Schutzfunktion gibt ZZA_aktiv_1 aus“.

Anschließend werden die Überschriften und Module einer Instanz als ein Bag-of-Words-Vektor kodiert. Alle drei entstandenen Vektoren werden zu einem Vektor kombiniert und es wird ein Verweis zu den Identifikationsnummern der zutreffenden internen Anforderungen gespeichert. Es ergibt sich ein Vektor als eine Form der Repräsentation einer externen

Tabelle 5.9: Bag-of-Words-Vektor für die Relationen der Anforderung „Die präventive Schutzfunktion gibt ZZA_aktiv_1 aus“

„Akteur von“			„Beschreibt“			
Schutzfunktion_ ausgeben	Notfunktion_ ausgeben	...	Präventiv_ Schutzfunktion	Präventiv_ Maßnahme		...
1	0	...	1	0		...

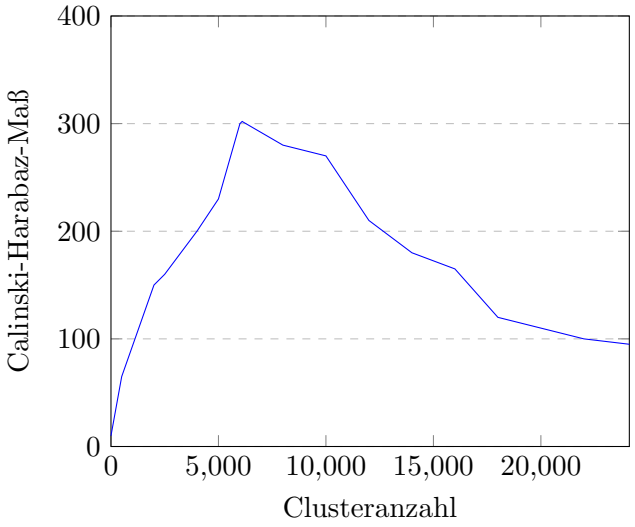


Abbildung 5.9: Calinski-Harabaz-Maße für die Clusteranzahl 0- 24 114

Anforderung. Diese Repräsentation ist unabhängig von dem Rauschen in den Texten, unter der Bedingung, dass die Modelle der Extraktion das Rauschen nicht als ein Konzept oder eine Relation erkennen.

5.7 Clustering der Vektoren

In diesem Abschnitt wird ein Clustering-Verfahren verwendet, um mithilfe der erzeugten Vektoren aus dem Abschnitt 5.6 ein Vorschlagssystem zu entwickeln. Durch Clustering können ähnliche Vektoren gruppiert werden. Dadurch, dass jeder Vektor einen Verweis zu den internen Anforderungen besitzt, enthalten die entstandenen Cluster eine Menge an Verweisen zu den internen Anforderungen. Wenn für eine neue externe Anforderung ebenfalls ein Vektor erstellt wird, kann dieser Vektor einem bestehenden Cluster zugeordnet werden und die Menge an Verweisen als Vorschläge für eine Abbildung ausgegeben werden. Um das Clustering umzusetzen, wird ein hierarchisches Clusteringverfahren verwendet. Dafür wird die Implementierung des Von-Unten-Nach-Oben-Verfahrens aus der Bibliothek **scikit-learn** [PVG+11] verwendet. Als Lerndaten ergeben sich 24 114 Vektoren, die nach den Regeln im Abschnitt 5.6 erzeugt wurden. Um die beste Anzahl der Cluster auszuwählen, wird das Calinski-Harabaz-Maß mit der Obergrenze 24 114 verwendet. Bei der Obergrenze von 24 114 kann kein Clustering stattfinden, da es der Anzahl der Vektoren entspricht. Es werden alle möglichen Clusterzahlen darunter ausprobiert. Als Distanz für das Clustering wird die euklidische Distanz verwendet. In der Abbildung 5.9 werden die resultierenden Calinski-Harabaz-Maße dargestellt. Der höchste Wert von 302 befindet sich bei einer Clusteranzahl von 6 103 auf der untersten Hierarchieebene. Diese Clusteranzahl wird verwendet.

5.8 Abbildungsvorschläge für eine neue Anforderung

Wenn eine neue externe Anforderung eintrifft, müssen daraus alle Begriffe extrahiert werden, sodass sich PDU, Signale, Akronyme und Terme der Anforderung, und die darauf zutreffenden Relationen „Akteuer Von“, „Agiert Auf“ und „Beschreibt“ ergeben. Zusätzlich müssen die Überschrift und das Modul der Anforderung aus den Metainformationen extrahiert werden. Als nächstes müssen die Oberklassen von den Entitäten aus der Ontologie gefunden werden. Aus allen Informationen kann ein Vektor erzeugt werden, welcher

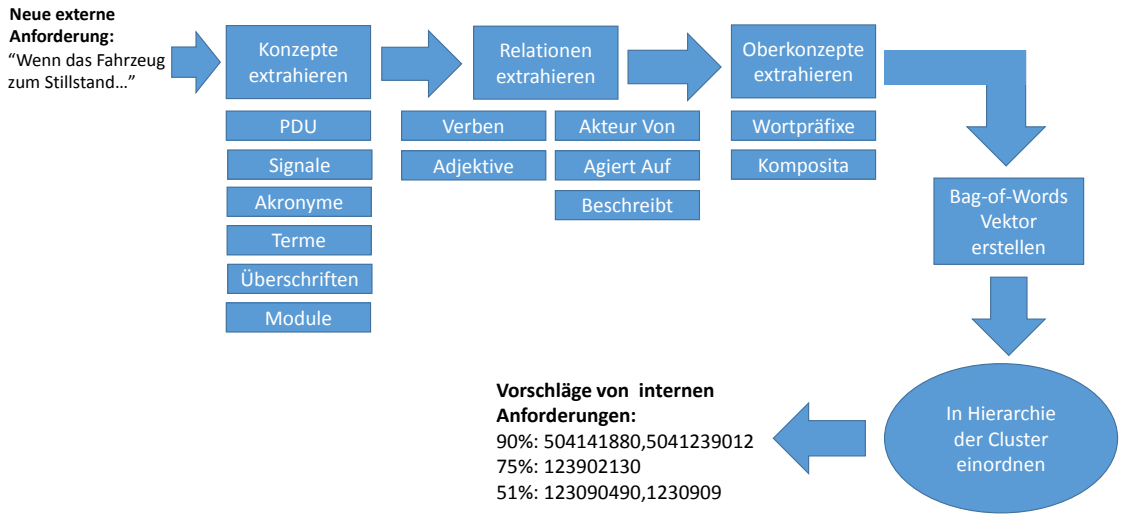


Abbildung 5.10: Erzeugung der Vorschläge für eine neue Anforderung mithilfe von Clustering und Ontologie

dieselbe Struktur hat, wie der Vektor aus dem Abschnitt 5.6. Der neue Vektor wird in die Clustering-Hierarchie eingeordnet und der zutreffende Cluster wird ausgegeben. Danach werden die Vektoren innerhalb des Clusters nach ihrer Ähnlichkeit zu dem Vektor der neuen Anforderung sortiert. Als Distanzmaß für die Ähnlichkeit wird die euklidische Distanz verwendet. Nach der Sortierung werden die Verweise auf die internen Anforderungen von jedem Vektor ausgegeben. Dadurch ergeben sich die gewerteten Vorschläge. Der Ablauf ist in der Abbildung 5.10 dargestellt. Dort sind die einzelnen Schritte durch Pfeilübergänge repräsentiert.

6. Evaluation

In diesem Abschnitt werden die entwickelten Verfahren evaluiert. Zuerst wird die entstandene Ontologie auf Fehler überprüft. Das Ziel dabei ist, den Anteil der Konzepte und Relationen zu bestimmen, welche aus den Anforderungen falsch extrahiert wurden. Danach wird das Ontologie-basierende Vorschlagssystem evaluiert, indem die Maße Präzision, Ausbeute und F-Maß bei der Erzeugung der Vorschläge aus den Testdaten berechnet werden. Diese Maße werden anschließend mit den zuvor entwickelten Richtlinien in Vergleich gesetzt.

6.1 Evaluation der Ontologie

Im diesem Abschnitt der Evaluation wird die entstandene Ontologie auf fehlerhafte Konzepte und Relationen überprüft. Dabei werden die falsch positiven Konzepte und Relationen gesucht, also die Konzepte und Relationen, welche durch eine fehlerhafte Extraktion in die Ontologie eingeordnet wurden. Die falsch negativen Konzepte und Relationen, also solche, die in die Ontologie eingeordnet werden sollen, aber durch die Extraktion ignoriert wurden, werden nicht betrachtet. Dafür müssen alle Texte der Anforderungen manuell betrachtet werden, was den zu Verfügung stehenden Zeitraum dieser Arbeit übersteigt.

6.1.1 Evaluation der Konzepte

Es werden die Oberkonzepte PDU, Signal, Akronym, Term, Adjektiv und Verb der Ontologie betrachtet. Für jedes ihrer Unterkonzepte wird überprüft, ob das Unterkonzept fehlerhaft eingeordnet wurde. Des Weiteren wird die Anzahl der Konzepte ausdiskutiert. Die Konzepte Modul, Überschrift, „externe Anforderung“ und „interne Anforderung“ werden nicht betrachtet, da diese direkt aus den Metadaten der Anforderungen erzeugt werden und nicht durch die Fehler der Extraktion betroffen sind.

6.1.1.1 Fehlerhafte PDU-Konzepte

Für jedes Unterkonzept des PDU-Konzepts wird geprüft, ob sich dieses Konzept in dem Lexikon der PDU wiederfinden lässt. Wenn das Unterkonzept im Lexikon vorkommt, ist es nicht fehlerhaft, sonst wird es als fehlerhaft eingestuft. Die Abbildung 6.1 zeigt die Anzahl der PDU-Unterkonzepte und wie viel davon fehlerhaft sind. So wurden insgesamt aus den 24 115 Anforderungen 37 PDU-Bezeichnungen extrahiert. Alle Unterkonzepte der PDU ließen sich im Lexika wiederfinden, außer dem Unterkonzept „bremse_xa_03“, welches nicht im Lexikon vorkommt, aber von der Eigennamenerkennung als PDU erkannt wurde.

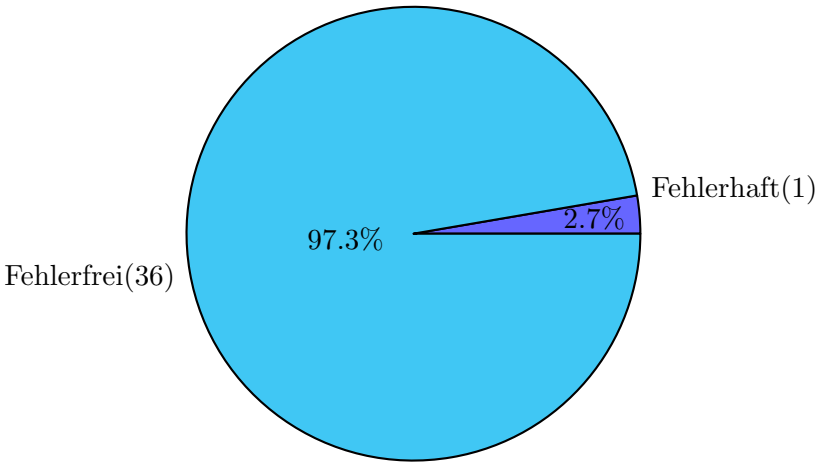


Abbildung 6.1: Fehlerhafte PDU-Konzepte

Tabelle 6.1: Fehlerhafte Signale

Nr.	Konzeptbezeichnung
1	src
2	pextenratecorrection
3	aag_bz
4	tqt
5	mrepetitioncycle

Dies ist auf die Erkennungsrate des Modells der Eigennamenerkennung aus dem Abschnitt 5.4.3.1 zurückzuführen. So liegt dort die Präzision bei 91%, was die Fehler wie „bremse_xa_03“ zulässt. Bei „bremse_xa_03“ handelt es sich um einen Signal. Als mögliche Lösung könnte man die extrahierten PDU-Unterkonzepte automatisch mit dem PDU-Lexikon abgleichen. Dies setzt voraus, dass das PDU-Lexikon vollständig ist.

6.1.1.2 Fehlerhafte Signal-Konzepte

Für jedes Unterkonzept des Signal-Konzepts wird geprüft, ob sich dieses Konzept im Lexikon der Signale wiederfinden lässt. Ähnlich wie bei den PDU-Konzepten aus dem Abschnitt 6.1.1.1 wird es als fehlerhaft gewertet, wenn es nicht im Lexikon ist. Die Abbildung 6.2 zeigt die Anzahl der fehlerfreien und fehlerhaften Signal-Konzepte. Es wurden 476 Signale extrahiert und 9 davon wurden nicht im Lexika gefunden. Die Tabelle 6.1 zeigt fünf Beispiele aus den fehlerhaften Signalkonzepten. Bei einigen handelt es sich um Akronyme, wie zum Beispiel „src“ oder „tqt“. Auch ganze Wörter wie „pextenratecorrection“ wurden als ein Signal erkannt.

Diese Fehler weisen auf die Fehlerrate der Eigennamenerkennung hin. Die extrahierten Signale sollten mit dem Signal-Lexikon abgeglichen werden. Wie auch bei PDU, setzt es die Vollständigkeit des Signal-Lexikons voraus.

6.1.1.3 Fehlerhafte Akronym-Konzepte

Bei den Akronymen stehen keine Lexika zu Verfügung. Um die Fehler in den Akronymen zu finden, wird jedes einzelne Unterkonzept des Akronyms manuell angeschaut. Die Abbildung 6.3 zeigt den Anteil der fehlerhaften Akronyme. Die Tabelle 6.2 zeigt fünf fehlerhafte

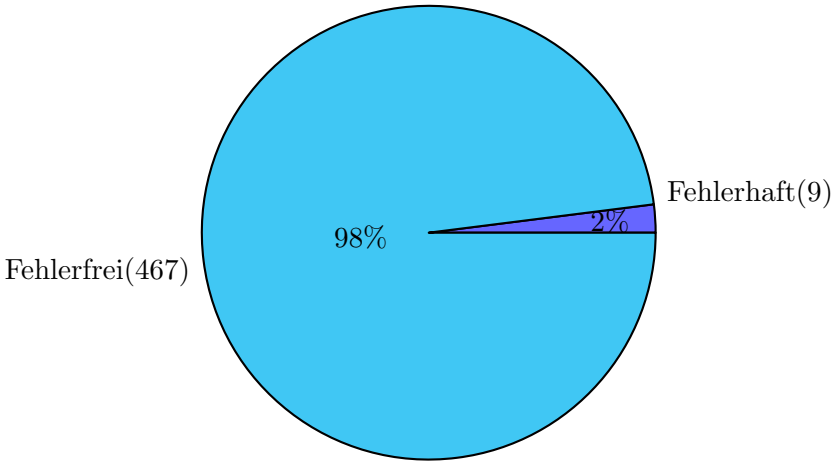


Abbildung 6.2: Fehlerhafte Signal-Konzepte

Tabelle 6.2: Fehlerhafte Akronyme

Nr.	Konzeptbezeichnung
1	AIX.5301388.b_signalbeschreibung_ASW__2.docx
2	n]-[q]+1)-zyklen
3	wenn:((CDE_naesselevel
4	-3,5M/S^2
5	T=

Akronyme. So wurden Verweise auf Dateien wie „AIX.5301388.b_signalbeschreibung_ASW__2.docx“ als Akronyme erkannt. Des Weiteren hat der Portionierer des Computerlinguistik-Fließbands offenbar Probleme bei der Trennung der Wörter, sodass Formeln und Bedingungen wie „n]-[q]+1)-zyklen“ und „wenn:((CDE_naesselevel“ als Akronyme erkannt werden. Aber auch Begriffe, die nicht den Akronymregeln entsprechen, wurden von der Eigennamenerkennung als Akronym erkannt, wie zum Beispiel „T=“.

Der Anteil der fehlerhaften Akronyme liegt 19% und übersteigt somit die Fehlerrate des Modells der Eigennamenerkennung aus dem Abschnitt 5.4.3.1 um das Doppelte. Möglicherweise waren in den Testdaten der Eigennamenerkennung wenig Akronyme enthalten und deshalb fiel die Fehlerrate des Modells deutlich geringer aus. Die Regeln für die Akronyme sind zu schwach definiert. Derzeit gilt, dass ein Akronym mehrere hintereinander folgende Großbuchstaben enthalten muss und durch einen Unterstrich oder Trennstrich getrennt sein. Allerdings werden durch diese Regeln auch solche Wörter wie mit Großbuchstaben definierte Geschwindigkeitsangaben „50_KMH“ als Akronyme erkannt. Das sorgt für Fehler während der Trainingsphase der Eigennamenerkennung. Um das Problem zu lösen, sollten die Regeln für Akronyme stärker definiert werden. Beispielsweise dürfen die Zahlen nicht am Anfang des Wortes stehen oder ein Maximallimit an Unterstrichen kann als eine weitere Regel definiert werden.

6.1.1.4 Fehlerhafte Term-Konzepte

Ein Unterkonzept von Term wird als fehlerhaft bezeichnet, falls es sich dabei nicht um ein deutsches Nomen handelt. Die deutschen Nomen mit Rechtschreibfehlern gelten ebenfalls als fehlerhaft. Um die fehlerhaften Konzepte zu ermitteln, wird jedes Unterkonzept manuell überprüft. Die Abbildung 6.4 zeigt die Anzahl der fehlerhaften Terme. 11% der Terme

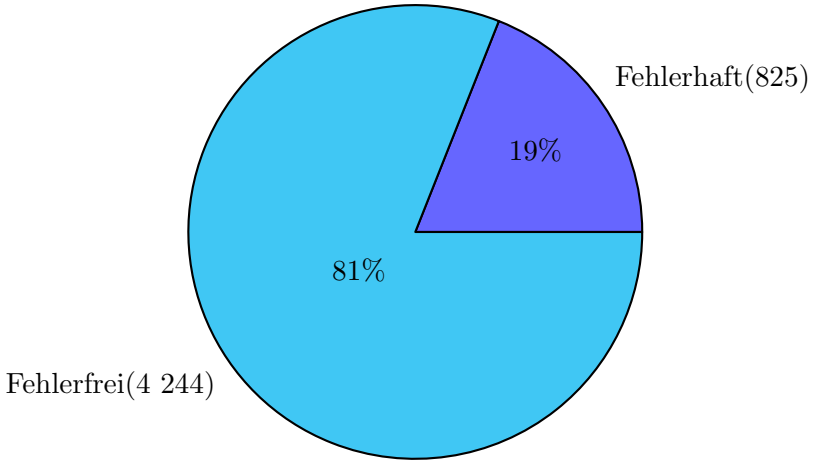


Abbildung 6.3: Fehlerhafte Akronym-Konzepte

Tabelle 6.3: Fehlerhafte Terme

Nr.	Konzeptbezeichnung
1	controlsetting==off
2	max.mustermann@a1.de
3	signal(e
4	t>10s
5	aabruch

sind fehlerhaft. Die Tabelle 6.3 zeigt fünf Beispiele der fehlerhaften Terme. So werden englische Begriffe als Nomen erkannt, wie zum Beispiel „controlsetting==off“ und „that“. Eine Email-Adresse wurde ebenfalls als ein Nomen erkannt. Durch die Fehler im Portionierer werden Formeln und Klammerabschnitte fehlerhaft behandelt und dadurch als Nomen erkannt, beispielsweise „signal(e“ und „t>10s“. Auch Nomen mit Rechtschreibfehlern wie „aabruch“ werden als Nomen erkannt.

Der fehlerhafte Anteil von 11% zeigt, dass die Zuverlässigkeit des Etikettierers von **spacy** für die Anforderungsdomäne eingeschränkt ist. Das Modell von **spacy**, das für die Nomenerkennung benutzt wird, wurde auf den deutschen Nachrichtentexten trainiert, welche typischerweise keine mathematischen Angaben enthalten. Außerdem handelt es sich bei den Nachrichtentexten um grammatikalisch korrekte Sätze. Die Anforderungstexte dagegen können nur aus Aufzählungen und einzelnen Wörtern bestehen. Das macht die Nomenerkenung besonders schwierig, da keine Umgebung der Wörter betrachtet werden kann. Um das Problem zu lösen, könnte das Modell von **spacy** mit ausgewählten Texten der Anforderungen nachtrainiert werden. Diese Texte müssen allerdings manuell etikettiert werden.

6.1.1.5 Fehlerhafte Verb-Konzepte

Alle Unterkonzepte von Verb werden betrachtet. Ein Unterkonzept gilt als fehlerhaft, falls es sich nicht um ein deutsches Verb handelt. Dafür muss jedes Unterkonzept manuell betrachtet werden. Die Abbildung 6.5 zeigt den Anteil der fehlerhaften Verben. 29% der Verben sind fehlerhaft. Die Tabelle 6.4 zeigt fünf repräsentative Beispiele der fehlerhaften Verben. So werden beispielsweise Adjektive als Verben erkannt wie „bedarfsgerechte“. Auch

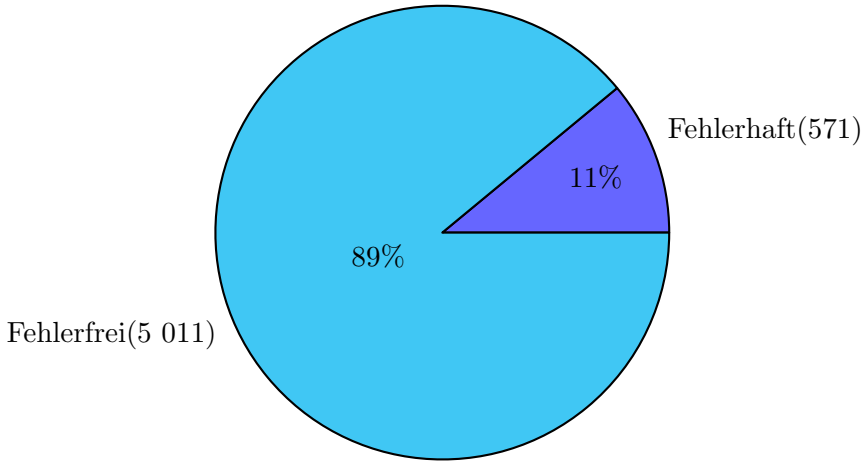


Abbildung 6.4: Fehlerhafte Term-Konzepte

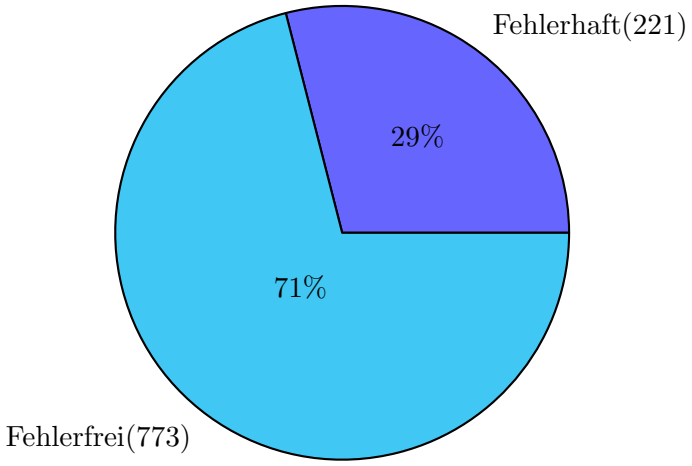


Abbildung 6.5: Fehlerhaft Verb Konzepte

Teile eines Signals wie „anhänger_erkannt“ oder falsche Wortrennungen wie „beschreiben-sein“ und „druckverlauf_sein“ werden als Verb gekennzeichnet. Eine Zeitangabe „50ms“ wurde als Verb erkannt.

Der Anteil von 29% zeigt, dass der Etikettierer von **spacy** nicht für die Verberkennung in der Anforderungsdomäne geeignet ist. Es ist auf die gleiche Problematik wie bei der Erkennung der Nomen im Abschnitt 6.1.1.4 zurückzuführen. Die Texte der Anforderungen sind oft nicht grammatikalisch korrekt, sodass die Verben nicht durch die Umgebungsinformationen der Wörter erkannt werden können. Hier wäre eine mögliche Lösung das Nachtrainieren des Modells mithilfe von Anforderungstexten, bei welchen die Verben zuvor manuell etikettiert wurden.

6.1.1.6 Fehlerhafte Adjektiv-Konzepte

Für jedes Unterkonzept von Adjektiv wird manuell geprüft, ob es sich um ein deutsches Adjektiv handelt. Der Anteil der fehlerhaften Adjektive wird in der Abbildung 6.6 gezeigt. Dieser beträgt 26%. Es wurden beispielsweise Verben als Adjektive erkannt wie „öffnen“.

Tabelle 6.4: Fehlerhafte Verben

Nr.	Konzeptbezeichnung
1	bedarfsgerechte
2	beschreiben-sein
3	anhänger_erkannt
4	druckverlauf_sein
5	50ms

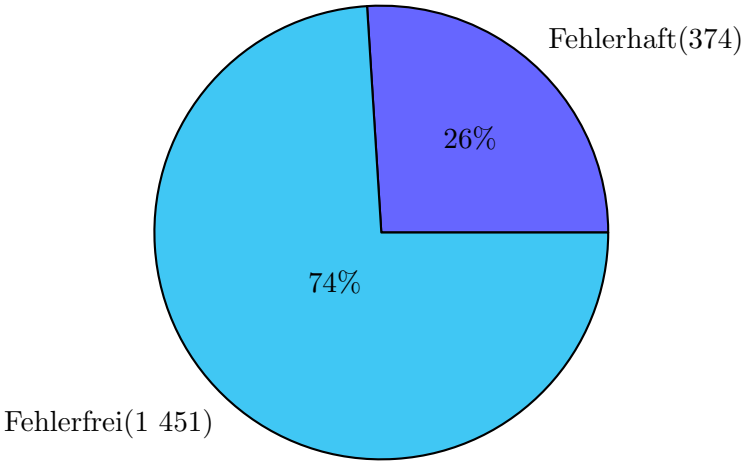


Abbildung 6.6: Fehlerhafte Adjektiv Konzepte

Auch Abkürzungen wie „w.“ oder „v-vzg“ wurden als Adjektive erkannt. Des Weiteren wurden englische Begriffe als Adjektive erkannt, wie „other“, und Teile von PDU, wie „diagnose_10“, was in der Tabelle 6.5 gezeigt wird.

Wegen dem hohen Fehleranteil ist der Etikettierer von **spacy** ebenfalls nicht für die Adjektiverkennung in der Fahrzeugdomäne geeignet. Ähnlich wie in Abschnitt 6.1.1.4, könnte das Nachtrainieren des Modells mit Anforderungstexten für eine Verbesserung der Erkennung sorgen. Auch hier müssen die Adjektive manuell etikettiert werden.

6.1.2 Evaluation der Relationen

In diesem Abschnitt werden die resultierenden Relationen der Ontologie betrachtet. Es wird der Anteil der fehlerhaften Relationen ausgewertet. Folgende Relationen werden betrachtet: „Akteur Von“, „Agiert Auf“ und „Beschreibt“. Die Relation „Bildet Auf“ wird nicht betrachtet, da diese aus den Metadaten einer Abbildung erzeugt wird und nicht durch die Textextraktion betroffen ist.

Tabelle 6.5: Fehlerhafte Adjektive

Nr.	Konzeptbezeichnung
1	öffnen
2	w.
3	v-vzg
4	other
5	diagnose_10

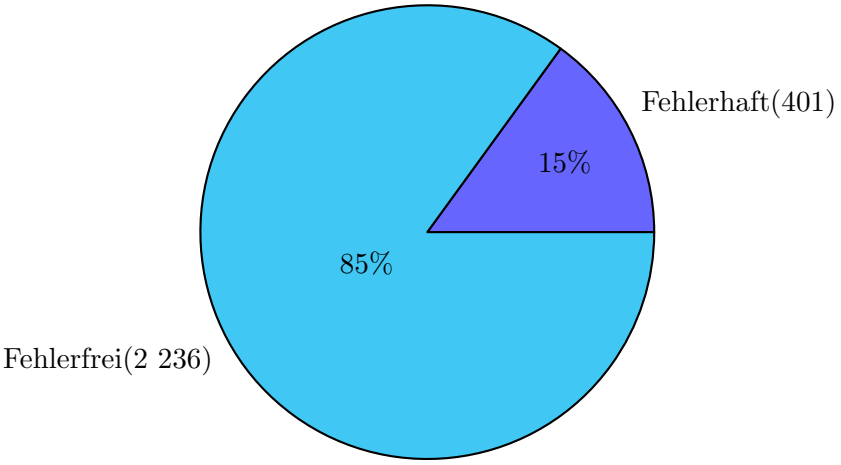


Abbildung 6.7: Fehlerhafte „Akteur Von“-Relationen

Tabelle 6.6: Fehlerhafte „Akteur Von“-Begriffe

Nr.	Konzeptbezeichnung
1	zielbremsung_fahrergetriggert_verz_info==0x1
2	a_change
3	that_soll
4	t=_bedarfsgerechte
5	aabruch_beschreiben-sein

6.1.2.1 Fehlerhafte „Akteur Von“-Relationen

Die „Akteur Von“-Relation setzt sich aus einem Unterkonzept von PDU, Signal, Term oder Akronym als Subjekt und einem Verb als Prädikat zusammen. Dementsprechend kann die Anzahl der fehlerhaften Relationen im schlechtesten Fall gleich der Summe ihrer fehlerhaften Komponenten sein. Um die Anzahl der fehlerhaften „Akteuer Von“-Relationen zu ermitteln, wurde bei der vorherigen Betrachtung der fehlerhaften Konzepte im Abschnitt 6.1.1 die Anzahl ihrer „Akteuer Von“-Relationen notiert. Dies ist möglich, da im Abschnitt 5.5.2 die Relationen zwischen zwei Konzepten erstellt werden und man somit Verweise auf Relationen seitens der Konzepte besitzt. In der Abbildung 6.7 wird die Anzahl der fehlerhaften „Akteur Von“-Relationen gezeigt. Der fehlerhafte Anteil beträgt 15%. Die Tabelle 6.6 zeigt fünf repräsentative Beispiele. So werden die fehlerhaften Verben gemischt wie in „zielbremsung_fahrergetriggert_verz_info==0x1“ oder englische Begriffe werden mit Abkürzungen gemischt wie in „a_change“ und „that_soll“. Eine Mischung aus den fehlerhaften Nomen und Verben wie in „t=_bedarfsgerechte“ und „aabruch_beschreiben-sein“ ist ebenfalls präsent.

Obwohl 15% der Relationen fehlerhaft sind, ist die Tatsache, dass sich der Anteil der fehlerhaften „Akteur Von“-Relationen nicht aus der Summe ihrer fehlerhaften Komponenten zusammensetzt, dennoch vorteilhaft. Es ist ein Indiz dafür, dass die große Menge an fehlerhaften Verben aus dem Abschnitt 6.1.1.5 keinem Subjekt zugeordnet werden kann. Da sich die Syntaxanalyse von **spacy** auf die Ergebnisse der Etikettierer aufbaut, könnte die Lösung mit dem Nachtrainieren des Modells für Nomen- und Verberkennung, auch bessere „Akteuer Von“-Relationerkennung ermöglichen.

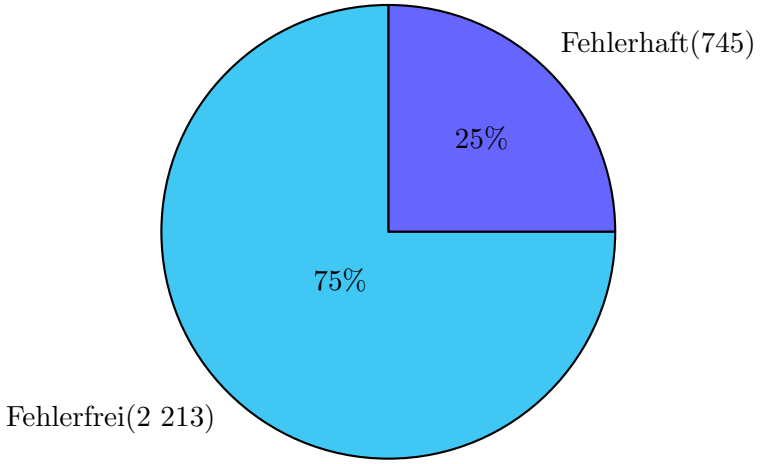


Abbildung 6.8: Fehlerhafte „Agiert auf“-Relationen

Tabelle 6.7: Fehlerhafte „Agiert Auf“-Begriffe

Nr.	Konzeptbezeichnung
1	gehen_d.h
2	gelten_e10)und
3	können_o.
4	sein_oder
5	ref_motordrehzahl

6.1.2.2 Fehlerhafte „Agiert Auf“-Relationen

Die „Agiert Auf“-Relation setzt sich aus einem Verb als Prädikat und einem Objekt aus einem Unterkonzept von PDU, Signal, Term oder Akronym zusammen. Um die Anzahl der fehlerhaften „Agiert Auf“-Relationen zu ermitteln, wurde bei der vorherigen Betrachtung der fehlerhaften Konzepte in dem Abschnitt 6.1.1 die Anzahl ihrer „Agiert Auf“-Relationen notiert. In der Abbildung 6.8 wird die Anzahl der fehlerhaften „Agiert Auf“-Relationen gezeigt. Hierbei ist der Anteil von 25% problematisch. Beispielsweise tauchen als Relationen Kombinationen von Verben und Konjunktionen auf, wie „gehen_d.h“ und „sein_oder“. Auch Fehler durch den Portionierer finden sich wieder, wie „gelten_e10)und“ und „können_o.“. Außerdem tauchen fehlerhaft erkannte Verben in der Relation auf, beispielsweise „ref_motordrehzahl“.

Aus dem hohen Fehleranteil folgt, dass die Syntaxanalyse das Objekt in den Anforderungstexten unzuverlässig erkennt und es auch einem fehlerhaften Verb zuordnen kann. Da in einem Satz mehrere Objekte auftauchen können, ist die Erkennung des Objekts schwieriger als die Erkennung des Subjekts. Da die Sätze der Anforderungstexte grammatikalisch oft nicht korrekt sind, wird die Objekterkennung dadurch noch weiter erschwert.

6.1.2.3 Fehlerhafte „Beschreibt“-Relationen

Die „Beschreibt“-Relation setzt sich aus einem Adjektiv und einem Unterkonzept von PDU, Signal, Term oder Akronym zusammen. In der Abbildung 6.9 wird die Anzahl der fehlerhaften „Beschreibt“-Relationen gezeigt. 28% der Relationen sind fehlerhaft. Die „Beschreibt“-Relationen stellen mit einer Gesamtzahl von 5 171 die größte Relationenart dar. Das bedeutet, dass viele Adjektive mehrfach verwendet werden, da die Gesamtzahl der Adjektive in der Ontologie 1 451 beträgt. Die Tabelle 6.8 zeigt repräsentative Beispiele der

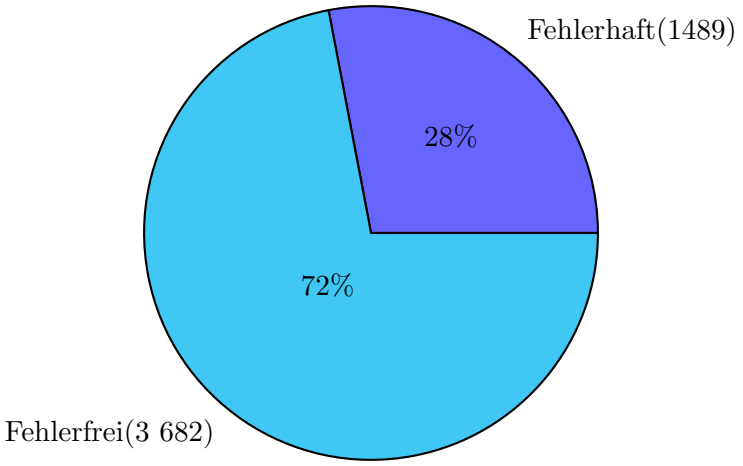


Abbildung 6.9: Fehlerhafte „Beschreibt“-Relationen

Tabelle 6.8: Fehlerhafte „Beschreibt“-Begriffe

Nr.	Konzeptbezeichnung
1	30sec_default
2	o.a_algorithmus
3	while_freeze
4	b._kapitel
5	1_august

fehlerhaften Relationen. So werden Zahlenangaben als zutreffende Adjektive erkannt, wie in „30sec_default“ und „1_august“. Auch englische Begriffe werden fälschlich zugeordnet wie „while_freeze“ und „30sec_default“. Des Weiteren werden beispielsweise Abkürzungen „o.a_algorithmus“ und „b._kapitel“ als Adjektive erkannt.

Es gibt dabei mehr fehlerhafte Relationen als es insgesamt Adjektive gibt. Die fehlerhaften Adjektive werden also mehrmals einem Konzept zugeordnet. Das sorgt für eine Ausbreitung der Fehler und könnte bei größeren Datenmengen für einen weitaus höheren Anteil als 28% sorgen. Die Lösung aus dem Abschnitt 6.1.1.6 könnte in diesem Fall helfen, wenn dadurch die fehlerhaften Adjektive eliminiert werden.

6.1.3 Alle Konzepte und Relationen

Nach den Überprüfungen der einzelnen Konzepte und Relationen lassen sich folgende Kennzahlen bestimmen. In der Ontologie existieren 11 992 aus den Textdaten extrahierte Konzepte. 2 001 davon sind fehlerhaft. Das entspricht 16%, dieser Sachverhalt ist in der Abbildung 6.10 zu sehen. Den größten Anteil an den fehlerhaften Konzepten haben die Akronym Konzepte. Dieser beträgt 41%.

In der Ontologie existieren 10 793 Relationen und 2 635 sind dabei fehlerhaft. Im diesem Fall entspricht es 24%, was in der Abbildung 6.11 gezeigt wird. Den größten Anteil an den fehlerhaften Relationen von 56% haben die ”Beschreibt”-Relationen.

Die Fehleranteile von 16% und 24% können die Merkmalserzeugung aus der Ontologie im Abschnitt 5.6 beeinträchtigen. Wenn beispielsweise eine Geschwindigkeitsangabe „50_KMH“ als ein fehlerhaftes Akronym erkannt wird und darauf zusätzlich in mehreren „Agiert Auf“-Relationen auftaucht, wird diese Information im Merkmalsvektor kodiert.

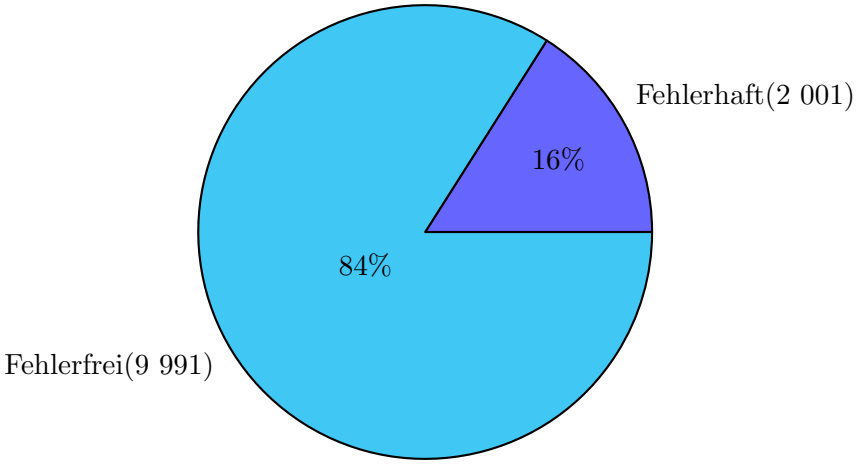


Abbildung 6.10: Fehlerhafte Konzepte insgesamt

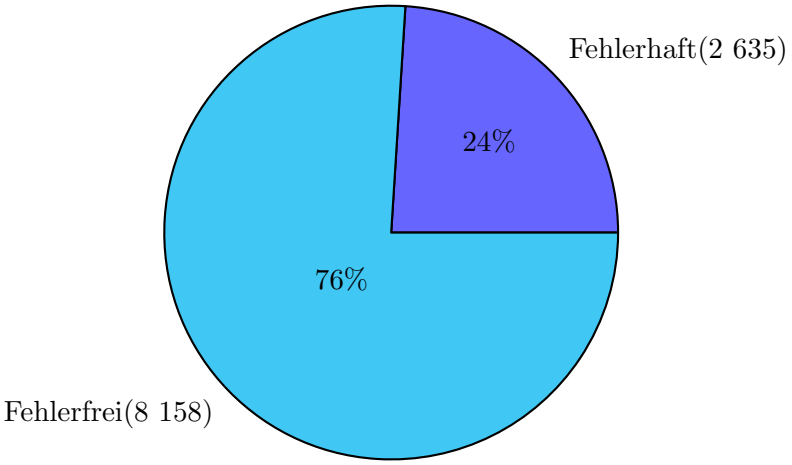


Abbildung 6.11: Fehlerhafte Relationen insgesamt

Diese Geschwindigkeitsangabe hat wenig Bedeutung für die Entscheidung einer Abbildung, könnte aber dafür sorgen, dass die Vektoren nach dieser Geschwindigkeitsangabe während des Clusterings gruppiert werden.

6.1.4 Evaluation der Konzeptpräfixe

Um die Gruppierung der Konzepte PDU, Signal, Akronym und Term nach Wortpräfixen sowie Komposita aus dem Abschnitt 5.5.4 zu evaluieren, werden die entstandenen Gruppen betrachtet. Es wird manuell geprüft ob sich die Konzepte, welche in dem Abschnitt 6.1.1 als fehlerhaft bezeichnet wurden, in den Gruppen befinden. Da die Gruppen erst ab zwei Konzepten gebildet werden, ist es möglich, dass fehlerhafte Konzepte, welche nur einmal extrahiert wurden, in den Gruppen nicht auftauchen.

6.1.4.1 Fehlerhafte PDU-Gruppen

Es wird manuell geprüft ob eine Gruppe, die durch die Gruppierung der PDU-Begriffspräfixe entstanden ist, ein fehlerhaftes PDU-Unterkonzept enthält. Falls dies der Fall ist, wird die Gruppe als fehlerhaft bezeichnet. Es wurden 6 Gruppen aus den PDU-Unterkonzepten erstellt, wobei keine der Gruppen das fehlerhafte Konzept aus dem Abschnitt 6.1.1.1 enthält. Insgesamt wurden 19 aus 37 PDU-Unterkonzepten in die Gruppen eingeordnet.

6.1.4.2 Fehlerhafte Signal-Gruppen

Es wird manuell geprüft ob eine Gruppe, die durch die Gruppierung der Signal-Begriffspräfixe entstanden ist, ein fehlerhaftes Signal-Unterkonzept enthält. Aus den Signal-Unterkonzepten wurden 46 Gruppen erstellt, wobei 451 sich in eine Gruppe einordnen ließen. Die 9 fehlerhaften Konzepte aus dem Abschnitt 6.1.1.2 wurden keiner Gruppe zugeordnet. Dies ist vorteilhaft, da dadurch die Prüfung der fehlerhaften Konzepte eingeschränkt werden kann. Nur die Konzepte, welche keiner Gruppe zugeordnet wurden, müssen als fehlerhaft überprüft werden.

6.1.4.3 Fehlerhafte Akronym-Gruppen

Die Anzahl der fehlerhaften Gruppen der Akronym-Begriffspräfixe wird in der Abbildung 6.12 gezeigt. Bei den Akronym-Unterkonzepten wurden 275 Gruppen erstellt, wobei 33 davon fehlerhafte Konzepte enthalten. In der Abbildung 6.13 ist eine fehlerhafte Gruppe dargestellt. So wurden zwei durch den Portionierer falsch getrennte Wörter „ABS-(toleranzschwelle“ und „ABS-(ackermann“ zu einer Gruppe zusammengefasst. In der Abbildung 6.14 wird ein weiteres Beispiel gezeigt. Drei Internetadressen „https:doors-bosch.com:123“, „https:doors-bosch.com:456“ und „https:req-bosch.com“ wurden falsch als Akronyme erkannt und zu einer Gruppe zusammengefasst. Zuerst wurden diese nach dem Präfix „doors-bosch.com“ gruppiert und daraufhin nach dem Präfix „https:“.

Hier kann durch die Gruppierung nichts gewonnen werden, da 12% der Gruppen fehlerhafte Akronyme enthalten. Das bedeutet, dass die Merkmale, welche aus den Gruppenpräfixen erzeugt werden, ebenfalls fehlerhaft sein können. Eine Lösung liefert die Verbesserung der Eigennamenerkennung, wie im Abschnitt 6.1.1.3 beschrieben wird.

6.1.4.4 Fehlerhafte Term-Gruppen

Jede entstandene Gruppe wird manuell überprüft, ob diese fehlerhafte Term-Unterkonzepte enthält. Es ergeben sich 414 Gruppen, von welchen keine fehlerhaft ist. Dies ist dadurch zu erklären, dass das Modell der Kompositazerlegung für die fehlerhaft erkannten Nomen aus dem Abschnitt 6.1.1.4 keine Konstituenten generieren kann. Die Gruppierung nach den Term-Gruppen eignet sich daher als ein Filter, um die fehlerhaften Term-Unterkonzepte zu entfernen.

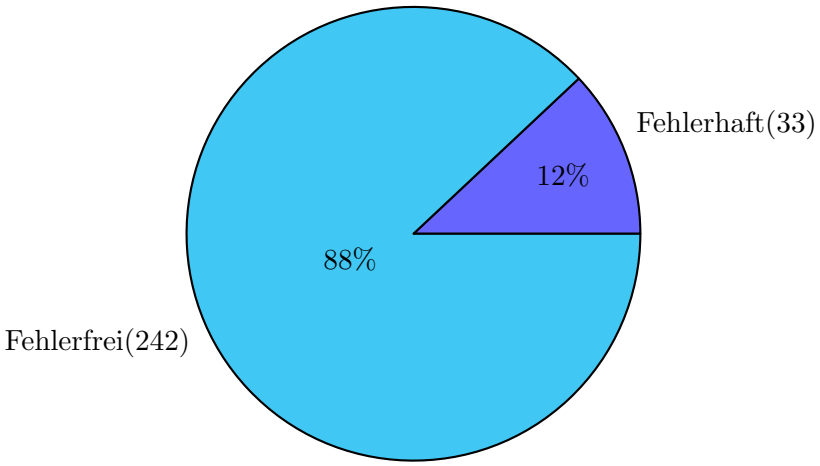


Abbildung 6.12: Fehlerhafte Akronym-Gruppen

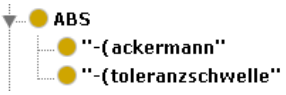


Abbildung 6.13: Fehlerhafte Akronym-Gruppe

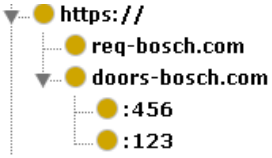


Abbildung 6.14: Fehlerhafte Akronym-Gruppe

6.2 Evaluation des Ontologie-basierenden Vorschlagssystems

In diesem Abschnitt wird das Ontologie-basierende Vorschlagssystem aus dem Abschnitt 5.8 mithilfe der vordefinierten Maße Präzision, Ausbeute und F-Maß evaluiert. Des Weiteren werden die Ergebnisse der Klassifikation mit den erzeugten Richtlinien aus dem Abschnitt 5.4 verglichen.

6.2.1 Evaluation der Vorschläge

Um die Maße Präzision, Ausbeute und F-Maß anwenden zu können, werden zuerst die Trainingsdaten erzeugt. Aus den züruckgehaltenen 6 478 Anforderungen der drei aktuellsten Projekte werden nach dem Verfahren aus dem Abschnitt 5.8 Merkmalsvektoren generiert. Die Merkmalsvektoren werden einem Cluster zugeordnet und die zutreffenden Verweise auf die internen Anforderungen werden ausgegeben. Um die Klassifikation zu evaluieren, werden die Maße Präzision@k und Ausbeute@k verwendet. Das Maß Präzision@k bedeutet, dass die Präzision innerhalb der ersten k Ergebnisse berechnet wird (Gleichung 6.1).

$$\text{Präzision@k} = \frac{\text{wahr korrekt innerhalb k}}{\text{k}} \tag{6.1}$$

Das Maß Ausbeute@k bedeutet, dass die Ausbeute innerhalb der ersten k Ergebnisse berechnet wird (Gleichung 6.2).

$$\text{Ausbeute@k} = \frac{\text{wahr korrekt innerhalb k}}{\text{wahr korrekt}} \tag{6.2}$$

Diese Maße sind sinnvoll, da die Ausgabe der Verweise in Form einer geordneten Liste stattfindet und diese Liste als Vorschläge der Abbildung für den Mitarbeiter dient. Um den Mitarbeiter mit einer großen Zahl an Vorschlägen nicht zu überfordern, werden nur die k ersten Ergebnisse betrachtet. Die Klassifikation wird für k im Intervall von $\{1, 10\}$ durchgeführt. Die Ergebnisse der Präzision befinden sich in der Abbildung 6.15. So bleibt die Präzision bis einem k von 4 bei über 60%. Die Ergebnisse der Ausbeute befinden sich in der Abbildung 6.16. Ab $k = 5$ bleibt die Ausbeute konstant bei 68%. Um die beiden Werte in Verhältnis zu setzen, wird das F-Maß daraus errechnet. Die Ergebnisse der F-Maße befinden sich in der Abbildung 6.17. Die höchsten F-Maße befinden sich bei $k = 1$ und $k = 5$.

Die Klassifikation mithilfe des Clustering-Ansatzes kann gut mit den Anforderungen umgehen, welche eine n-zu-1-Abbildungen besitzen. Dies wird durch eine gute Präzision bei $k = 1$ deutlich. Der k-Wert von 1 wird als Vergleichswert für die Gegenüberstellung mit den Richtlinien verwendet, da die Richtlinien ebenfalls nur eine mögliche Klasse ausgeben. Der k-Wert von 5 zeigt ein vergleichsweise hohes F-Maß und wird ebenfalls für die Gegenüberstellung mit den Richtlinien verwendet.

Für ein Produktivsystem sind diese Werte allerdings zu gering. Bei einer hohen Anzahl an falschen Vorschlägen, was im derzeitigen Zustand möglich ist, würde ein solches System den Mitarbeiter möglicherweise sogar stören, anstatt ihm zu helfen. Ein Produktivsystem müsste sehr gute Ergebnisse liefern können, was einem F-Maß über 90% und auch einer ebenso hohen Ausbeute entspricht.

6.2.2 Gegenüberstellung mit den Richtlinien

Die Richtlinien und die beiden besten Ergebnisse des Vorschlagssystems werden in diesem Abschnitt verglichen.

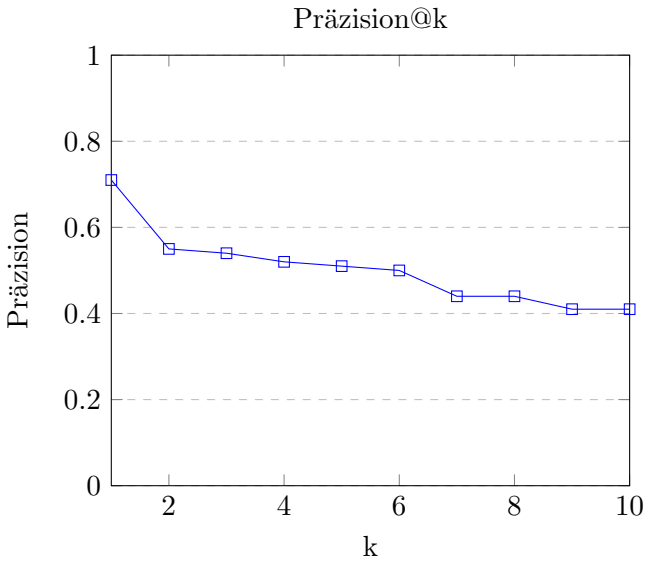


Abbildung 6.15: Präzision@k für das Ontologie-basierende Vorschlagssystem

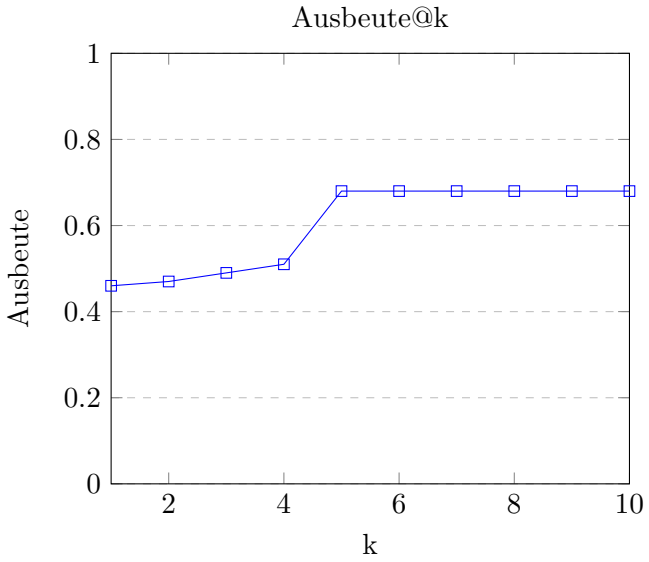


Abbildung 6.16: Ausbeute@k für das Ontologie-basierende Vorschlagssystem

Tabelle 6.9: Gegenüberstellung der Richtlinien und des Ontologie-basierenden Vorschlagssystems

Verfahren	Präzision	Ausbeute	F-Maß
Alle Felder, URL-Klassen, Naïve-Bayes	0.07	0.08	0.07
Alle Felder, URL-Klassen, Random-Forest	0.15	0.18	0.16
Alle Felder, Modul-Klassen, Naïve-Bayes	0.61	0.59	0.60
Alle Felder, Modul-Klassen, Random-Forest	0.69	0.68	0.68
Eigennamen, URL-Klassen, Naïve-Bayes	0.41	0.49	0.44
Eigennamen, URL-Klassen, Random-Forest	0.53	0.55	0.54
Ontologie-basierendes Vorschlagssystem@k1	0.71	0.46	0.56
Ontologie-basierendes Vorschlagssystem@5	0.51	0.68	0.58

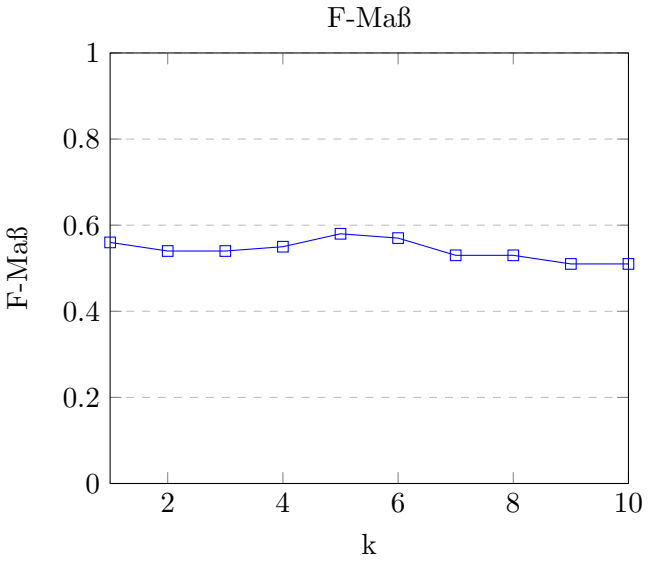


Abbildung 6.17: F-Maß für das Ontologie-basierende Vorschlagssystem

Das beste F-Maß liefert der „Alle Felder, Modul-Klassen, Random-Forest“-Ansatz. Allerdings werden bei diesem Verfahren Module als Klassen ausgegeben. Ein Modul kann mehrere Tausend Anforderungen enthalten, was für den Mitarbeiter in so einem Fall keine Hilfe wäre.

Das zweitbeste F-Maß von 0.58 kann mit dem Verfahren „Ontologie-basierendes Vorschlagssystem@5“ erreicht werden. Als Ergebnisse werden dabei fünf Verweise auf die internen Anforderungen ausgegeben. Das drittbeste F-Maß von 0.56 erreicht der Ansatz „Ontologie-basierendes Vorschlagssystem@1“, welches als Ergebnis einen Verweis auf eine interne Anforderung ausgibt.

Verglichen mit den Ansätzen „Alle Felder, URL-Klassen, Random-Forest“ und „Alle Felder, URL-Klassen, Naïve-Bayes“, die alle Felder der Anforderungen als Merkmale verwenden und als Ergebnis ebenfalls einen Verweis auf eine interne Anforderung ausgeben, erreichen die beiden Ontologie-basierenden Ansätze deutlich bessere Werte.

Wenn man als Merkmale nur die Fachbegriffe in den Anforderungen verwendet, was bei den Ansätzen „Eigennamen, URL-Klassen, Random-Forest“ und „Eigennamen, URL-Klassen, Naïve-Bayes“ der Fall ist, dann kommt man auf die F-Maße 0.44 und 0.54. Insbesondere der letzte Wert ist nur um 0.04 geringer als bei dem „Ontologie-basierendes Vorschlagssystem@5“-Ansatz. Das lässt darauf schließen, dass die Fachbegriffe in der Anforderungsdomäne eine starke Aussagekraft besitzen und die zusätzlich extrahierten Verb- und Adjektivbeziehungen nur eine geringe Verbesserung mit sich bringen.

7. Zusammenfassung und Ausblick

7.1 Zusammenfassung

In dieser Arbeit wurde ein Verfahren des maschinellen Lernens entwickelt, das einen Mitarbeiter der Robert Bosch GmbH bei dem Abbildungsprozess von Kundenanforderungen auf eine interne Anforderungsdatenbank in Form eines Vorschlagssystems unterstützt. Es wurde zuerst eine Datenbereinigung des vorliegenden Anforderungsdatensatzes durchgeführt, bei welcher irrelevante Datenfelder entfernt wurden, Duplikate zusammengefasst und Abbildungsinformationen präzisiert wurden. Danach wurden aus den Daten Informationen identifiziert, welche als Merkmale für maschinelle Lernverfahren verwendet werden können.

Mit den erzeugten Merkmalen wurden Verfahren des maschinellen Lernens trainiert. Das Ziel war, mit den Merkmalen für eine bestimmte externe Anforderung die zutreffenden internen Anforderungen voraussagen zu können. Mehrere Merkmalskombinationen wurden ausprobiert. Die Ergebnisse dieser Verfahren wurden als Richtlinien festgelegt. Danach wurde durch die Verfahren der Computerlinguistik aus den Textinformationen der externen Anforderungen eine Ontologie automatisch generiert. Die Ontologie enthielt Fachbegriffe sowie ihre Beziehungen und diente dazu neue Merkmale für die Klassifikation zu schaffen.

Mithilfe von hiearchischem Clustering-Verfahren wurden ähnliche externe Anforderungen mit den Merkmalen aus der Ontologie zu Clustern zusammengefasst. Dabei wurden die Abbildungen auf interne Anforderungen gespeichert, auf welche die externen Anforderungen eines Clusters abbilden. Um eine Abbildung für eine neue externe Anforderung vorzuschlagen, wurde die neue Anforderung in Merkmale transformiert und einem Cluster zugeordnet. Danach wurde innerhalb des Clusters die Ähnlichkeit der neuen Anforderung zu den anderen Anforderungen berechnet. Für eine festgelegte Anzahl an ähnlichsten Anforderungen wurden ihre Abbildungen auf die internen Anforderungen als Vorschläge ausgegeben.

Um das Ontologie-basierende Vorschlagssystem zu evaluieren, wurde zuerst die Ontologie auf Fehler untersucht. Der Anteil der ermittelten fehlerhaften Konzepte in der Ontologie betrug 16%. Der Anteil der fehlerhaften Relationen betrug 24%. Anschließend wurde die Ontologie-basierende Vorschlagserzeugung evaluiert und mit den bestehenden Richtlinien in Vergleich gesetzt. Die Voraussage der Vorschläge mithilfe der Ontologie erreichte ein F-Maß von 58%.

7.2 Ausblick

Die Ontologie-basierende Erzeugung der Vorschläge erreicht zurzeit einen F-Maß von 58%, was zu gering ist, um als ein Produktivsystem für Abbildungsunterstützung betrachtet zu werden. Ein solches System müsste sehr gute Ergebnisse liefern können, was einem F-Maß über 90% und auch einer ebenso hoher Ausbeute entspricht. Um dies zu erreichen, sollen mehr Daten des Kunden A als die 30 593 abgebildete externe Anforderungen zu Verfügung gestellt werden. Aus den Daten kann dann ein Trainingskorpus erzeugt werden.

Im ersten Schritt müssen die Textdaten der Anforderungen umfangreich bereinigt werden. Alle Verweise auf weitere Dateien, Tabellen und Formeln müssen innerhalb des Textes erkannt und entfernt werden. Des Weiteren müssen die Textdaten auf Fehler überprüft werden, sodass alle fehlerhaften Begriffe entfernt werden. Mit den bereinigten Daten können die Modelle nachtrainiert werden, welche für die Ontologie die Extraktion der Konzepte und Relationen aus den Texten der Anforderungen durchführen. Dadurch kann die Qualität der Ontologie verbessert werden.

Um die Domäne genauer darzustellen, können für die Ontologie mehr Konzepte und Relationen definiert werden. In dieser Arbeit wurden beispielsweise die Werte von einzelnen Signalen und Akronymen nicht behandelt. So wäre es vorteilhaft zu wissen, ob ein bestimmter Signal in einer Anforderung auf aktiv oder inaktiv gesetzt wird. Außerdem können weitere Relationen wie Präpositionen und Adverbien extrahiert werden. Durch eine solche Erweiterung der Ontologie können neue Merkmale erzeugt werden, welche womöglich eine Verbesserung der Vorschläge bedeuten. Das nächste Ziel wäre demnach, das Vorschlagssystem mit den neu erzeugten Merkmalen zu implementieren. Auch kann die Ontologie als eine Wissensbasis für einen Mitarbeiter verwendet werden, beispielsweise falls mögliche Werte für einen Signal nachgeschaut werden sollen. Die Eignung einer solchen automatisch generierten Ontologie als Wissensbasis zu evaluieren, wäre ein ein mögliches nächstes Ziel.

Schließlich kann die Eignung des Ontologie-basierenden Vorschlagssystems auf die Daten von anderen Kunden getestet werden. Neben dem Kunden A existieren bei Robert Bosch GmbH noch einige andere Kunden, welche jeweils ihre eigenen Lastenheftformate mitbringen. Der Aufbau der Lastenhefte kann sich von den des Kunden A grundlegend unterscheiden. Es wäre wissenswert zu evaluieren, ob das entwickelte Vorschlagssystem auf die Anforderungen anderer Kunden anwendbar ist, und falls dies der Fall ist, ob das F-Maß von 58% gehalten oder möglicherweise sogar übertroffen werden kann.

Literaturverzeichnis

[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 Seite 10).

[ER03] Endres, A. ; Rombach, H.D.: *A Handbook of Software and Systems Engineering: Empirical Observations, Laws, and Theories*. Pearson/Addison Wesley, 2003 (Fraunhofer IESE series on software engineering). `https://books.google.de/books?id=QrsBoLfyD1IC`. – ISBN 9780321154200 (zitiert auf den Seiten 4 und 5).

[HM17] Honnibal, Matthew ; Montani, Ines: spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. In: *To appear* (2017) (zitiert auf Seite 37).

[MCCD13] Mikolov, Tomas ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: Efficient Estimation of Word Representations in Vector Space. In: *CoRR* abs/1301.3781 (2013). `http://arxiv.org/abs/1301.3781` (zitiert auf Seite 12).

[MJDS07] Manku, Gurmeet S. ; Jain, Arvind ; Das Sarma, Anish: Detecting Near-duplicates for Web Crawling. In: *Proceedings of the 16th International Conference on World Wide Web*. New York, NY, USA : ACM, 2007 (WWW '07). – ISBN 978–1–59593–654–7, 141–150 (zitiert auf den Seiten 17 und 31).

[Mur12] Murphy, K.P.: *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012 (Adaptive computation and machine learning). `https://books.google.co.in/books?id=NZP6AQAAQBAJ`. – ISBN 9780262018029 (zitiert auf den Seiten 5, 6, 7, 8, 9, 10, 11 und 13).

[PVG+11] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830 (zitiert auf den Seiten 36 und 46).

[RK15] Roth, Michael ; Klein, Ewan: Parsing Software Requirements with an Ontology-based Semantic Role Labeler. In: *Proceedings of the IWCS Workshop Language and Ontologies 2015*, 2015 (zitiert auf den Seiten 16, 40 und 41).

[ROW13] Rashwan, A. ; Ormandjieva, O. ; Witte, R.: Ontology-Based Classification of Non-functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier. In: *2013 IEEE 37th Annual Computer Software and Applications Conference*, 2013, S. 381–386 (zitiert auf den Seiten 15 und 28).

[SSH10] Saake, Gunter ; Sattler, Kai-Uwe ; Heuer, Andreas: *Datenbanken - Konzepte und Sprachen, 4. Auflage.* MITP, 2010. – I–XVII, 1–783 S. – ISBN 978–3–8266–9057–0 (zitiert auf Seite 29).

[STHH17a] Smith, Samuel L. ; Turban, David H. P. ; Hamblin, Steven ; Hammerla, Nils Y.: Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In: *CoRR* abs/1702.03859 (2017). http://arxiv.org/abs/1702.03859 (zitiert auf Seite 17).

[STHH17b] Smith, Samuel L. ; Turban, David H. P. ; Hamblin, Steven ; Hammerla, Nils Y.: Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In: *CoRR* abs/1702.03859 (2017). http://arxiv.org/abs/1702.03859 (zitiert auf den Seiten 28 und 30).

[TMKW] Taylor, Matthew E. ; Matuszek, Cynthia ; Klimt, Bryan ; Witbrock, and M.: Autonomous Classification of Knowledge into an Ontology. http://www.cyc.com/doc/white_papers/FLAIRS07-AutoClassificationIntoAnOntology.pdf. – bibtex: Taylor bibtex[owner=SV_KO] (zitiert auf Seite 16).

[Tug16] Tuggener, Don: *Incremental Coreference Resolution for German*, University of Zurich, Diss., 2016. https://doi.org/10.5167/uzh-124915 (zitiert auf Seite 44).

[VB13] Verma, R.P. ; Beg, M.R.: Representation of Knowledge from Software Requirements Expressed in Natural Language. In: *Emerging Trends in Engineering and Technology (ICETET), 2013 6th International Conference on*, 2013, S. 154–158. – bibtex: Verma2013 (zitiert auf Seite 16).

[WG13] WIJEWICKREMA, CHAAMINDA M. ; Gamage, Ruwan: An Ontology Based Fully Automatic Document Classification System Using an Existing Semi-Automatic System, 2013 (zitiert auf Seite 16).

[WV16] Winkler, Jonas ; Vogelsang, Andreas: Automatic Classification of Requirements Based on Convolutional Neural Networks. (2016), 09 (zitiert auf den Seiten 15 und 29).