Karlsruhe Institute of Technology

Konzept eines Dokumentationsassistenten zur Erzeugung strukturierter Anforderungen basierend auf Satzschablonen

Bachelorarbeit
von

Ryan Christopher Arbai

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:	M.Sc. Tobias Hey
Zweiter betr. Mitarbeiter:	M.Sc. Simon Fritz

Bearbeitungszeit: 05. Juni 2019 – 31. Okt. 2019

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

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

Karlsruhe, 30. Okt. 2019

(**Ryan Christopher Arbai**)

Publikationsgenehmigung

Melder der Publikation

Hildegard Sauer

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

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

Erklärung des Verfassers

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

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

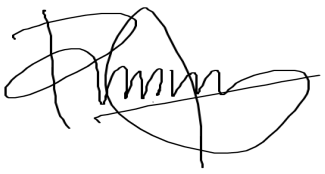
Bei etwaigen Forderungen Dritter stelle ich das KIT frei.

Eventuelle Mitautoren sind mit diesen Regelungen einverstanden.

Der Betreuer der Arbeit ist mit der Veröffentlichung einverstanden.

Art der Abschlussarbeit:	Bachelorarbeit
Titel:	Konzept eines Dokumentationsassistenten zur Erzeugung strukturierter Anforderungen basierend auf Satzschablonen
Datum:	30. Okt. 2019
Name:	Ryan Christopher Arbai

Karlsruhe, 30. Okt. 2019



(**Ryan Christopher Arbai**)

Kurzfassung

Um die Qualität und Glaubwürdigkeit eines Produktes zu erhalten, ist ein systematisches Anforderungsmanagement erforderlich, wobei die Merkmale eines Produkts durch Anforderungen beschrieben werden. Deswegen wurde im Rahmen dieser Arbeit ein Konzept für einen Dokumentationsassistenten entwickelt, mit dem Benutzer strukturierte Anforderungen basierend auf den Satzschablonen nach SOPHIST erstellen können. Dies beinhaltet einen linguistischen Aufbereitungsansatz, der semantischen Rollen aus freiem Text extrahiert. Während des Dokumentationsprozesses wurden die semantischen Rollen benutzt, um passendste Satzschablone zu identifizieren und diese als Hilfestellung dem Benutzer aufzuzeigen. Zudem wurde eine weitere Hilfestellung angeboten, nämlich die Autovervollständigung, die mithilfe von Markovketten das nächste Wort vorhersagen kann. Insgesamt wurden rund 500 Anforderungen aus verschiedenen Quellen herangezogen, um die Integrität des Konzepts zu bewerten. Die Klassifizierung der Texteingabe in eine Satzschablone erreicht ein F1-Maß von 0,559. Dabei wurde die funktionale Anforderung Satzschablone F1-Maß von 0,908 am besten identifiziert. Außerdem wurde der Zusammenhang zwischen den Hilfestellungen mit Hilfe eines Workshops bewertet. Hierbei konnte gezeigt werden, dass die Anwendung des vorliegenden Konzepts, die Vollständigkeit von Anforderungen verbessert und somit die Qualität der zu dokumentierenden Anforderungen steigert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Das Anforderungsmanagement (AM) ist eine in der Praxis meist unterschätzte Disziplin, welche jedoch maßgeblich über den Erfolg oder Misserfolg eines Entwicklungsprojektes entscheiden kann. Vor allem für kleine und mittelständische Unternehmen (KMU) ist es aufgrund ihrer begrenzten personellen und finanziellen Ressourcen schwierig AM-Prozesse in vollem Umfang in die Unternehmensprozesse zu integrieren. Hauptgrund sind hierfür vor allem die meist an Großunternehmen gerichteten Softwarelösungen, welche zwar die notwendigen Funktionen für ein adäquates Anforderungsmanagement bieten, jedoch wegen ihrer Komplexität für KMU nicht sinnvoll nutzbar sind. Dies hat zur Folge, dass überwiegend Office-Dokumente für das Management von Anforderungen genutzt werden. Diese Dokumente bergen jedoch die Gefahr, dass diese nicht aktuell oder die darin enthaltenen Informationen nicht semantischen miteinander verknüpft sind, wodurch wichtige Informationen nicht abgebildet werden können.

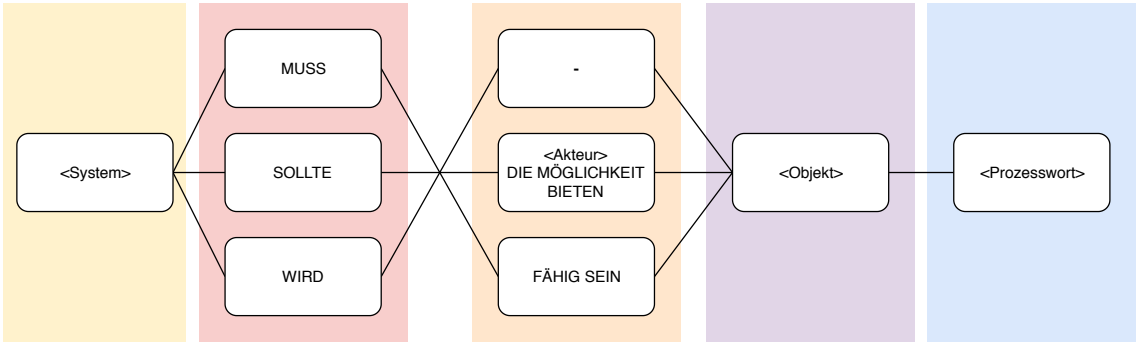


Abbildung 1.1: Funktionale Anforderung Schablone nach SOPHIST [Rol13]

Die Anforderungen werden hierbei meist in Freitext formuliert, was zur Folge hat, dass sie häufig unvollständig und nicht standardisiert sind. Um dieses Problem zu eliminieren bietet sich die Anwendung von Anforderungsschablonen (siehe Abbildung 1.1) an, welche die standardisierte Dokumentation sicherstellen und die weitere Verarbeitung in datenbankgetriebenen Systemen erleichtern. Aber Satzschablonen bergen auch Nachteile. Einerseits ist die Bedienung der Schablonen, aufgrund ihrer Vielzahl, vor allem für Nicht-Anforderungsmanagement-Experten wenig intuitiv. Andererseits ist ihre Anwendung zeitaufwändiger als die freie Formulierung von Anforderungen, weshalb auch geschultes Personal sie nur selten einsetzen.

Um dennoch die Vorteile von Satzschablonen sinnvoll nutzbar zu machen, ist das Ziel dieser Arbeit ein Konzept zu entwickeln, um Autoren von Anforderungen aktiv im Dokumentationsprozess zu unterstützen. Hierzu soll der vom Anwender eingetragene Text kontinuierlich mit den in der Literatur empfohlenen Satzschablonen abgeglichen werden. Sobald das System erahnen kann, welche Art von Anforderung der Anwender eingeben möchte, sollen die am ehesten zutreffenden Satzschablonen dem Benutzer angeboten werden. Dieser hat anschließend die Möglichkeit, eine der Schablonen auszuwählen, woraufhin der bereits eingegebene Text, automatisch in die ausgewählte Satzschablone überführt wird.

1.1 Rahmen der Arbeit

Die zuvor beschriebene Aufgabenstellung entstammt dem Forschungsprojekt „Digitaler Assistent für das Anforderungsmanagement für KMU“ (DAM4KMU) [Gia19]. Dieses wittmet sich vorrangig der Entwicklung eines Konzepts, um das Anforderungsmanagement KMU-gerechter zu gestalten.

Ein wesentlicher Punkt hierbei ist die ganzheitliche Integration des Anforderungsmanagements in die Produktentwicklung. Hierzu sollen Anforderungen mit allen Projektrelevanten Elemente (engl. Assets) verknüpft werden. Hierzu zählen unter anderem die im Rahmen eines Entwicklungsprojektes anfallenden Aufgaben, Lösungsideen zur Realisierung von Anforderungen sowie die tatsächliche Realisierung und deren Teilkomponenten.

Derzeitige AM-Softwarelösungen weisen häufig Schwächen in der Funktionalität und Ausstattung auf. So bieten diese bspw. keine aktive Unterstützung bei der Überführung von Freitextanforderungen in die entsprechenden Satzschablonen, sind aus Sicht der Anwender nur schwer bedienbar und haben eine schlechte Handhabung bei Anforderungsänderungen. Auch ist die Akzeptanz dieser Softwarelösungen bei Nicht-AM-Experten meist gering und die Lizenzgebühren sind sehr hoch. Dies macht deutlich, dass aktuelle Softwarelösungen den durch die zunehmende Produktkomplexität steigenden Anforderungen kleiner und mittelständischer Unternehmen (KMU) nicht gerecht werden.

Aus diesem Grund setzt das Projekt DAM4KMU an. Die Ziele sind Methoden und Konzepte zu entwickeln, um das Anforderungsmanagement mit Hilfe eines webbasierten digitalen Assistenzsystems auch für KMU effizienter und prozessintegrierter gestalten zu können.

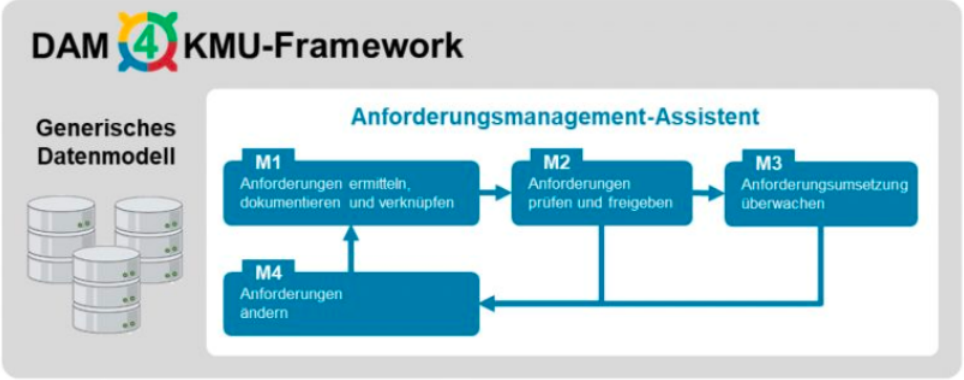


Abbildung 1.2: DAM4KMU Framework.

1.2 Zielsetzung

Die vorliegende Arbeit hat zum Ziel den Dokumentationsprozess (M1 siehe Abbildung 1.2) von Anforderungen mit Hilfe von digitalen Assistenz-Methoden wie z.B. einer intelligenten Autovervollständigung oder Hilfestellungen über Einblendungen zu unterstützen.

Hierdurch soll der Anwender befähigt werden, auch ohne die Kenntnis der in der Literatur [Rol13] empfohlenen Anforderungssatzschablonen, vollständige und standardisierte Anforderungen zu formulieren.

2 Grundlagen

In diesem Kapitel werden die Grundlagen zur Erzeugung einer strukturierten Anforderung mittels Satzschablonen beschrieben. Als erstes wird der Begriff Anforderungsmanagement und seine Methoden (Abschnitt 2.1) eingeführt. Dieser Abschnitt erläutert das Management und Arten von Anforderungen und bzw. deren entsprechenden Anforderungsmuster/-Satzschablonen. Danach wird ein Überblick über die Verarbeitung der natürlicher Sprache (engl. Natural Language Processing, im folgenden kurz NLP) und deutsche Grammatik gegeben (Abschnitt 2.2), was für die spätere Klassifikation der Texteingabe in eine der passenden Satzschablonen benötigt wird. Bei dem Ableitungsprozess einer Freitext-Anforderung in ihre standardisierte Form können mehrere Hilfestellungen angewendet werden, und eine davon ist die Autovervollständigung Methode, die in Abschnitt (Abschnitt 2.3) eingeführt wird. Der letzte Abschnitt gibt einen Überblick darüber, welche maschinellen Lernmethoden zum Klassifizieren einer sequentiellen Texteingabe geeignet sind.

2.1 Anforderungsmanagement

Für den Begriff Anforderung gibt es verschiedene Definitionen, die sich nach den Kriterien der Anwendung unterscheiden (z.B. Linguistik, Prozess, Software-Anforderung usw.).

Nach Institute of Electrical and Electronics Engineers (IEEE) „Standard Glossary of Software Engineering Terminology“ [Wal01] wird den Begriff „Anforderung“ wie folgt definiert:

Definition 2.1. Eine Anforderung ist:

1. Eine Bedingung oder Fähigkeit, die ein Benutzer benötigt, um ein Problem zu lösen oder ein Ziel zu erreichen
2. Eine Bedingung oder Fähigkeit, die ein System oder Teil eines Systems erfüllen oder besitzen muss, um einen Vertrag, einen Norm, eine Spezifikation oder andere, formell vorgegebene Dokument zu erfüllen.
3. Eine dokumentierte Repräsentation einer Bedingung oder Fähigkeit wie in (1.) oder (2.) referenziert.

Übersetzt aus [Wal01]

Anforderungen können in unterschiedlichen Formen dokumentiert werden, Beispiele hierfür sind als ausformulierte Texte, Stichwortsammlungen, Zeichnungen, Tabellen, Diagramme (z.B. Microsoft Visio, UML-Diagramme, ER-Diagramme, PPP), HTML- oder XML-Dokumente [Sri19].

In der Literatur wird der deutsche Begriff Anforderungsmanagement (AM) häufig mit dem englischen Begriff „Requirements Engineering (RE)“ gleichgesetzt und das englische Requirements Management als Teilaktivität des RE aufgefasst. Aufgrund dessen werden die beiden Begriffe Anforderungsmanagement und Requirements Engineering in dieser Arbeit gleichbedeutend verwendet [Sch09, S. 5].

Das AM ist ein kooperativer, iterativer, inkrementeller Prozess, dessen Ziel es ist zu gewährleisten, dass [Kla09, S. 12]:

1. alle relevanten Anforderungen bekannt und in dem erforderlichen Detaillierungsgrad verstanden sind,

2. die involvierten Stakeholder (Definition 2.1) eine ausreichende Übereinstimmung über die bekannten Anforderungen erzielen

3. alle Anforderungen konform zu den Dokumentationsvorschriften dokumentiert bzw. konform zu den Spezifikationsvorschriften spezifiziert sind.

Definition 2.2. Stakeholder: Ein Stakeholder eines Systems ist eine Person oder Organisation, die (direkt oder indirekt) Einfluss auf die Anforderungen des betrachteten Systems hat. [Kla09]

Das Anforderungsmanagement hat im Wesentlichen zur Aufgabe Anforderungen zu Ermitteln, Dokumentieren, Abzustimmen und zu Validieren [Poh08, S. 44-46]. Diese Kernaktivitäten lassen sich wie folgt definieren:

- *Ermittlung*: Das Ermitteln von Anforderungen wird verwendet, um Anforderungen von Stakeholdern oder anderer Quellen mithilfe von unterschiedlichen Techniken (z.B. durch Umfrage, Workshops usw.) zu gewinnen, zu detaillieren, und zu verfeinern

- *Dokumentation*: Bei der Dokumentation der Anforderungen werden verschiedene Techniken genutzt, um entstehende Anforderungen hinreichend zu beschreiben und in Prosa oder in Modellen zu dokumentieren.

- *Abstimmung und Validierung*: Die dokumentierte Anforderungen sollten frühzeitig geprüft und abgestimmt werden, um zu garantieren, dass sie alle geforderten Qualitätskriterien oder Eigenschaften erfüllen.

- *Verwaltung*: Die Anforderungsverwaltung (engl. Requirements Management) ergänzt alle anderen Aktivitäten und umfasst alle erforderlichen Maßnahmen, die notwendig sind (z.B. die Abhängigkeit zwischen Anforderungen kann die Rückverfolgbarkeit der Anforderung gewonnen werden), um Anforderungen zu strukturieren, konsistent zu halten und umzusetzen.

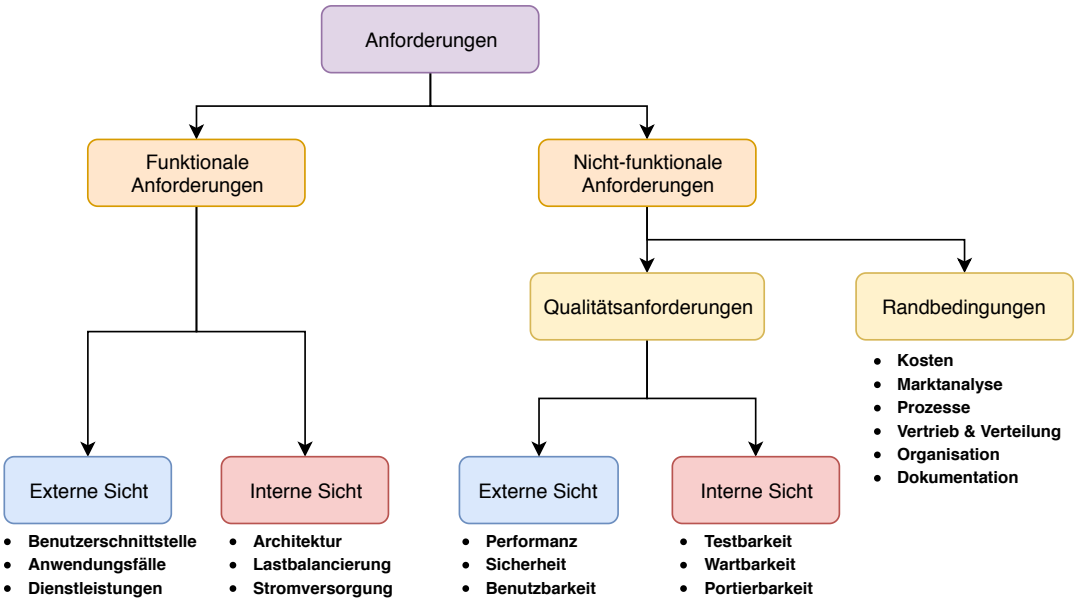


Abbildung 2.1: Anforderungsarten, eigene Darstellung aus [Ebe14]

2.1.1 Arten von Anforderungen

Nach Pohl [Poh08] können drei verschiedene Arten von Anforderungen unterschieden werden: Funktionale Anforderungen, Qualitätsanforderungen und Rahmenbedingungen beziehungsweise Randbedingungen. Durch die Definition von weiteren Subtypen der drei Klassen können die Anforderungstypen konkretisiert und an den jeweiligen Anwendungskontext angepasst werden. Somit lässt sich dieses Klassifizierungsschema beliebig weit verfeinern und stellt ein Rahmen für die Kategorisierung von Anforderungen dar [Poh08, S. 14-15].

Definitionen von Anforderungsarten nach Pohl: [Poh08]

- **Funktionale Anforderung**: Eine funktionale Anforderung definiert eine vom System bzw. von einer Systemkomponente bereitzustellende Funktion oder einen bereitzustellenden Service. Als Benutzeranforderung kann eine funktionale Anforderung sehr allgemein beschrieben sein. Als Bestandteil einer Spezifikation beschreibt eine funktionale Anforderung detailliert die Eingaben und Ausgaben sowie bekannte Ausnahmen.
- **Qualitätsanforderungen**: Eine Qualitätsanforderung definiert eine qualitative Eigenschaft des gesamten Systems, einer Systemkomponente oder eine Funktion.
- **Randbedingungen oder Rahmenbedingungen**: Eine Rahmenbedingung ist eine organisatorische oder technologische Anforderung, die die Art und Weise einschränkt, wie ein Produkt entwickelt wird.

Ebert unterscheidet beispielsweise in externe und interne Sicht [Ebe14], der in Abbildung 2.1 dargestellt ist. Die funktionale Anforderungen und Qualitätsanforderungen werden als Subtypen definiert, die durch weitere Subtypen noch detaillierter unterschieden werden können. Durch die interne Sichtweise werden Anforderungen beschrieben, die Einfluss auf die Entwicklung haben. Den zweiten Subtyp stellt die externe Sicht aus Kunden- oder Benutzerperspektive dar. Hierbei geht es um den Zusatznutzen durch das jeweilige Produkt aus der Sicht dessen, der das notwendige Kapital zur Verfügung stellt. Dieser Stakeholder muss aber nicht der tatsächliche Benutzer des Produkts sein [Ebe14, S. 31].

Die Qualitätsanforderungen und Rahmenbedingungen können wir als *Nicht-funktionale Anforderungen* zusammengefasst werden, da beide keine direkten Funktionen des Systems

sondern nur die Merkmale des Systems beschreiben. Um die Arten von Anforderungen besser zu verstehen, betrachten wir die folgenden Beispiele:

Funktionale Anforderung

Die Flasche muss fähig sein, Flüssigkeiten aufzunehmen, ohne dass es zu Undichtigkeit kommt.

Nicht-funktionale Anforderung (Zeitverhalten)

Die Antwortzeit des Systems sollte innerhalb von 50ms berechnet werden.

2.1.2 Anforderungsmuster/Satzschablone

Um Anforderungen auf eine sorgfältige Art und Weise natürlichsprachlich zu dokumentieren, können in der Praxis Satzschablonen genutzt werden. Pohl und Rupp definieren eine Satzschablone als einen „Bauplan für die syntaktische Struktur einer einzelnen Anforderung“ [Kla09, S. 61]. Die Vorteile von Anforderungsschablonen sind, einerseits gestaltet sich die Anwendung einfach bzw. leicht verständlich. Andererseits ist es durch den vordefinierten Bauplan zur Beschreibung von Anforderungen möglich, Redundanzen in der Formulierung zu minimieren und somit eine syntaktisch eindeutige Anforderung zu dokumentieren. Weiterhin optimieren Satzschablonen den Dokumentationsprozess im Hinblick auf die Zeitdauer, die Kosten und die Qualität der Anforderungsbeschreibung [Kla09, S. 61].

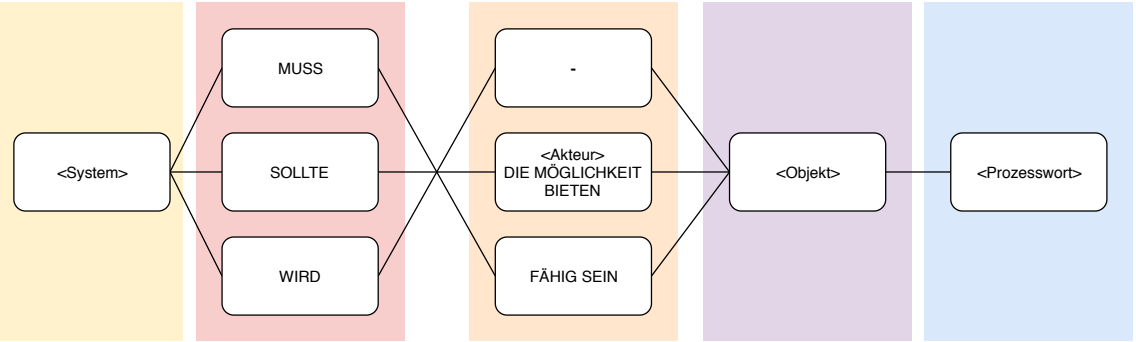


Abbildung 2.2: Funktionale Anforderung ohne Bedingung [Rol13]

Abbildung 2.2 zeigt ein Beispiel für eine Satzschablone zur Beschreibung einer funktionalen Anforderung ohne Bedingung. In einem ersten Schritt sieht die Schablone vor, die rechtliche Verbindung einer Anforderung festzulegen. Hierbei stehen dem Autor die Modalverben muss (rechtlich bindende Anforderung), sollte (dringend empfohlene Anforderung) oder wird (zukünftige Anforderung) zur Verfügung [Kla09, S. 62]. Anschließend ist es notwendig, die Funktionalität der Anforderung zu identifizieren bzw. festzulegen. Hierzu ist der Baustein „Prozesswort“ definiert. Dieser Schritt wird aus dem Grund an zweiter Stelle vorgenommen, „da sich der gesamte Satz [...] an das Prozesswort bindet“ [Rup14, S. 220]. Nachdem das Prozesswort (z. B. drucken, berechnen) festgelegt wurde, ist es in einem dritten Schritt wichtig, die Art der Funktionalität zu bestimmen. Hierzu stehen dem Autor die folgenden drei Möglichkeiten zur Verfügung [Kla09, S. 62]:

- Selbsttätige Systemaktivität: Der Prozess wird selbstständig durch das System durchgeführt.

- Benutzerinteraktion: Die Prozessfunktionalität wird dem Benutzer vom System zur Verfügung gestellt.
- Schnittstellenanforderung: Das System führt eine Aktion nur in Abhängigkeit von einem Dritten (z. B. Fremdsystem) aus.

In einem letzten Schritt ist es erforderlich, das Objekt zu identifizieren, für das die Funktionalität gefordert wird [Rup14, S. 223]. Ein Beispiel für die Beschreibung einer funktionalen Anforderung ohne Bedingung ist der folgende Satz: „Der Computer muss dem Anwender die Möglichkeit bieten, mehrere Prozesse gleichzeitig auszuführen."

Neben der vorgestellten Satzschablone existiert eine weitere Schablone zur Beschreibung einer funktionalen Anforderung mit Bedingung (siehe Abbildung 2.3).

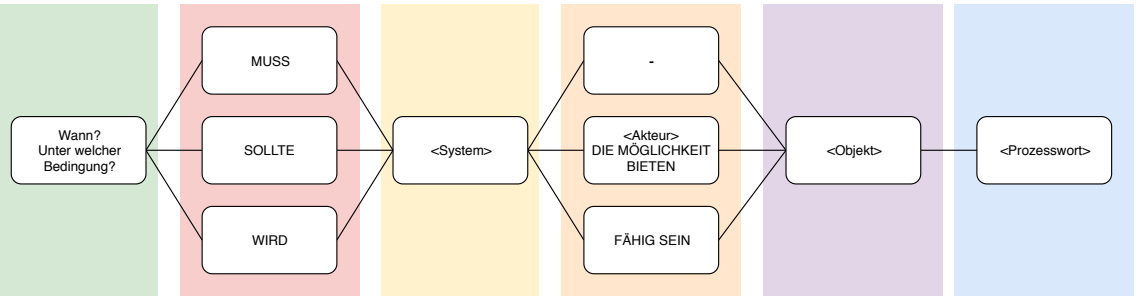


Abbildung 2.3: Funktionale Anforderung mit Bedingung [Rol13]

Die Funktionsweise dieser Schablone gestaltet sich ähnlich zu der ohne Bedingung. Es wurde lediglich ein Baustein hinzugefügt („Wann? Unter welcher Bedingung?"), der die Formulierung einer logischen oder zeitlichen Bedingung vornimmt, an die die Anforderung geknüpft ist. Durch das Voranstellen dieses Bausteins ändert sich zudem die Satzstellung [Kla09, S.64]. Der folgende Satz zeigt ein Beispiel für die Beschreibung einer funktionalen Anforderung mit Bedingung: „Falls der Anwender ein Dokument bearbeitet, muss der Computer dem Anwender die Möglichkeit bieten, das Dokument zu speichern."

Weitere Satzschablonen stellt Rupp in ihrem Werk „Requirements-Engineering und -Management" vor [Rup14, S. 215-246]. Hierbei stellt sie sowohl die bereits dargestellten Satzschablonen als auch mehrere Varianten für die Spezifikation einer nicht-funktionalen Anforderung (mit und ohne Bedingung) vor.

2.2 Verarbeitung von natürlicher Sprache

In dieser Arbeit sollen Anforderungen als Freitexte in einer der Anforderungsmuster/-Satzschablonen überführt werden. Das zu entwickelnde System muss den textuellen Input (in dieser Arbeit in deutscher Sprache) verarbeiten können. Hierfür werden Erkenntnisse der Linguistik und Methoden der Computerwissenschaft benötigt. *Die Computerlinguistik (CL)* vereint diese beiden Bereiche und ermöglicht beispielsweise die direkte Analyse von gesprochener oder geschriebener Sprache oder Text und Die „Verarbeitung von natürlicher Sprache (engl. Natural Language Processing (NLP)" bezieht sich auf Systeme, die versuchen oder analysieren, eine oder mehrere menschlichen Sprachen wie Englisch, Deutsch, Japanisch usw. zu verstehen oder zu produzieren [All03].

2.2.1 Deutsche Grammatik

In diesem Abschnitt werden Teile der deutschen Grammatik, die notwendig für die linguistische Aufbereitung von Freitext und die Ableitung möglicher Hilfestellungen sind, genannt. Der Begriff Syntax und Semantik wird als erstes erklärt. Danach werden Wortarten und semantischen Rollen definiert.

2.2.1.1 Syntax und Semantik

Beim Sprechen und Schreiben drücken wir nicht nur einen einzelnen Ausdruck oder nicht zusammenhängende Wörter aus, sondern eine Kombination zwischen den Wörtern, das heißt Sätze. Offensichtlich können wir Wörter nicht wahllos auswählen oder kombinieren, um einen Satz zu bilden.

Definition 2.3. Ein **grammatischer Satz** ist ein Satz, der den Regeln der deutsche Grammatik entspricht [Jö07].

Als Beispiel ist der Satz „*Ich höre morgen auf zu rauchen*“ ein grammatischer Satz und der Satz „*Ich zu rauchen morgen aufhöre*“ **ungrammatisch**.

Definition 2.4. Syntax beschreibt die Regeln, nach denen Wörter zu grammatischen Sätzen kombiniert werden. [Jö07].

Die Bedeutung von dem Satz ist unwichtig beim Syntax, man braucht nur die Regeln der Grammatik beachten. Um das zu erreichen, muss man untersuchen, wie Sätze aufgebaut sind, was für eine Struktur sie haben.

Definition 2.5. Die **Semantik** ist das Teilgebiet der Linguistik, das sich mit der Bedeutung von sprachlichen Ausdrucken (Wörtern, Phrasen und Sätzen) beschäftigt. [Jö07]

Die Bedeutungen sind feste Verbindung zwischen einem Ausdruck und einem Inhalt [Rol10]. Wir können die Bedeutung der Semantik in verschiedenen Aspekten betrachten wie z.B. die *lexikalische Semantik*, die mit der Bedeutung von Wörtern sich kümmern, die *Satzsemantik*, die mit der Bedeutung von größeren Syntaktische Einheiten (Phrasen, Teilsatz), und so weiter.

2.2.1.2 Semantische Rolle und Wortarten

In dieser Arbeit sind Kenntnisse über Wortarten und semantische Rollen wichtig, da es sich tatsächlich um die Bestandteile eines Satzes handelt. Diese Bestandteile werden später in dieser Arbeit analysiert werden, wie ein freier Text extrahiert werden kann, damit er in eine der Satzschablone eingeordnet werden kann.

Definition 2.6. Unter dem Begriff Wortart definiert Ludger wie folgt „Wenn in einer Sprache Wörter nach ihrem funktionalen Beitrag zu einer Äußerung, den sie in iheren Formen erbringen, zu klassifizieren sind, können für sie Wortartkategorien angenommen werden [Hof09].“

Wortarten in der Linguistik, die in dieser Arbeit gebraucht wird, sind z.B. Nomen, Pronomen, Verben, Adjektiv und Adverb. Diese Wortarten wird jetzt kurz beschrieben.

- Im Deutschen kann man das Nomen oder Substantiv durch folgende strukturelle Kriterien unterscheiden, nämlich nach Genus, Numerus, und Kasus [Kat10] [FH05]. Die Kasus unterscheidet Nomen in der Abhängigkeit ihrer Position in dem Satz und die deutschen Kasus sind Nominativ, Akkusativ, Dativ, und Genitiv. Für jedes Nomen liegt das Genus fest und wird durch den deutschen Artikel repräsentiert. Die Numerus unterscheidet Nomen im Singular und Plural.
- Das Pronomen ist eine Schreibweise, die an der Stelle eines Nomens oder eines Artikels gebraucht [Hil12]. Das Pronomen kann weiter klassifiziert werden, zum Beispiel als Personalpronomen, Demonstrativpronomen, oder Possessivpronomen.

Semantische Rolle	Bedeutung
System / Bezugobjekt	Der Begriff System meint den Liefergegenstand, an den Anforderungen gestellt werden, sodass er realisiert werden kann.
Priorität	Priorität liegt die Wichtigkeit oder die rechtliche Verbindlichkeit der Anforderung fest und wird mit folgende drei Schlüsselwörter muss, sollte und wird repräsentiert.
Prozesswort	Prozesswort ist der semantische Kern der Anforderung, denn es nennt die Funktion eines Systems.
Objekt	Objekte sind fachlich wertvolle Begriffe im Kontext des betrachteten Gegenstandes
Funktionswort	Funktionswort beschreibt, ob die Art der Funktionalität der Anforderung um selbstätige Systemaktivität, Benutzerinteraktion, oder Schittstellenanforderung geht.
Akteur	Akteur steht als Überbegriff für die potentiellen Benutzer eines Systems, die in der Interaktion mit dem System verschiedene Rollen einnehmen können.
Attribut	Attribut beschreiben die Eigenschaft oder welcher Qualität, die das System der Anforderung hat.
Attributswert	Attributswert kann ein Zahlwert und die semantisch passende Einheit sein z.B. Zeit, Gewicht, Längenmaß usw.
Konditionswort / Bedingungswort	Konditionswort kann ein Logik (falls), Ereignis (sobald) und Zeitraum (solange) sein und wird als das Schlüsselwort der Bedingungsschablone.
Vergleichsoperator	Vergleichsoperator ist die Angabe des Attributswerts, wie z.B. mindestens, maximal, oder größer (als).

Tabelle 2.1: Liste der semantische Rollen der Satzschablonen nach SOPHIST eigene Darstellung aus [Rol13]

- Das Verb ist eine Wortart, die eine Tätigkeit, ein Ereignis oder einen Zustand ausdrückt [Hil12]. Das Verb kann weiter klassifiziert werden, zum Beispiel als transitive und intransitive Verben, reflexive Verben, oder Modalverben.
- Das Adjektiv ist ein Eigenschaftswort [Hil12], das die Eigenschaft eines Nomens beschreibt.
- Das Adverb ist ein Umstandswort [Hil12], das ein anderes Wort beschreiben oder modifizieren kann.

In 2.6 sieht man die Definition von der Bergriff *Wortart*, aber was versteht man unter dem Begriff *semantische Rolle*? In dieser Arbeit werden Bestandteile oder Komponente der Satzschablonen nach SOPHIST als die semantische Rolle bezeichnet. Hierbei werden die Bestandteile der funktionalen, nicht-funktionalen und Bedingung Satzschablonen in Tabelle 2.1 kurz beschrieben [Rol13].

Bemerkung: Die Bestandteile, die in der funktionalen Anforderung Satzschablone stattfinden, wurden bereits im Abschnitt 2.1.2 beschrieben.

2.2.2 Generelle Methoden der Verarbeitung von natürlicher Sprache

Es existieren bereits verschiedene Werkzeugenketten, wie etwa das Natural Language Toolkit (NLTK), Apache OpenNLP, StanfordCoreNLP oder spaCy, für die Verwendung der Verarbeitung von natürlicher Sprache. In der englischen Sprache sind diese relativ mächtig - weitere Sprachen, unter anderem Deutsch, befinden sich noch in der Entwicklung.

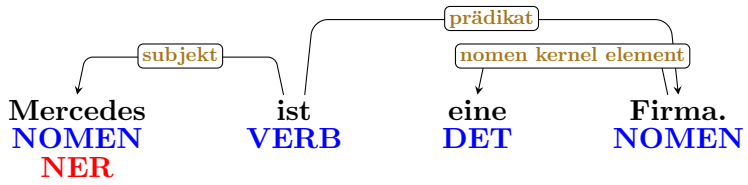


Abbildung 2.4: Beispielsatz für die Einsatz der generelle Methoden der Verarbeitung von natürlicher Sprache (aus spaCy)

Diese Werkzeugen können die als Freitext vorliegende Eingabe Schritt für Schritt in eine Form umwandeln, mit der wir Anwendungen oder Funktionen erstellen können, die große Textmengen verarbeiten und verstehen. Hierfür werden verschiedene Bearbeitungsschritte bezüglich der Syntax 2.2.1.1 angewendet.

- **Tokenisierung**: Token sind zusammenhängende Einheiten. Bei der Tokenisierung wird ein Satz in verschiedene Token zerlegt. In Falle eines Satzes als in Wörter und Satzzeichen.
- **Lemmatisierung**: Nomen und Verben können je nach Anwendungsfall unterschiedliche Formen annehmen. Dieser Prozess wird bei Nomen Deklination und bei Verben Konjugation genannt. Durch die Lemmatisierung können diese deklinierten und konjugieren Wörter auf ihre Grundform zurückgeführt werden.
- **Wortart-Markierung** (engl. Part-of-speech-Tagging (POS-Tagging)): Ein Wort oder ein Satzzeichen gehört immer zu einer Wortart. Es gibt Fälle, bei welchen die Position im Satz eine Rolle für die Zugehörigkeit zu einer Wortart spielt. Beim POS-Tagging wird die Wortart eines Wortes im Zusammenhang mit einem Satz wiederhergestellt.
- **Eigennamenserkennung** (engl. Named Entitiy Recognition (NER)): Beim NER werden Eigennamen erkannt und zu einer vorbestimmten Entität (z.B. Organisation, Ort, Sprache usw.) zugeordnet.
- **Abhängigkeitsanalyse** (engl. Dependency parsing) : Bei der Abhängigkeitsanalyse werden Zusammenhänge zwischen Wörtern eines Satzes dargestellt. Diese Zusammenhänge werden meist mithilfe eines Abhängigkeitsbaum abgebildet.

Für den einfachen Beispielsatz „Mercedes ist eine Firma.“ sieht die Tokenisierung, Lemmatiesierung, Abhängigkeitsanalyse und Wortart-Markierung wie in Abbildung 2.4.

Der Beispielsatz werden als erstes tokenisiert. Danach der blau markierte Text in der Abbildung 2.4 unter jedes tokensierte Wort beschreibt die Wortart-Markierung und der rot markierte Text ist die Eigennamenserkennung. Der Pfeil zeigt die Abhängigkeit zwischen den Wörtern an.

2.3 Autovervollständigung

Bar-Yossef und Kraus [Ziv11] ist verstehen unter der Autovervollständigung von Anfragen (engl. Query Autocompletion [QAC]) die Funktionalität, mit der dem Benutzer unter dem Suchfeld Anfragenvervollständigungen oder die ganze Anfragen während der Eingabe in Echtzeit, z.B. als Dropdown-Menü, angezeigt wird. Also man kann auch sagen, eine Methode, mit der die Anzahl der Tastenanschläge verringert werden kann, die zum Vervollständigen eines Wortes erforderlich sind [H. 68].

2.4 Maschinelles Lernen

In diesem Abschnitt werden bestimmte Voraussetzungen beschrieben, die ein maschinelles erfüllen sollte, um bei der Entwicklung von Textklassifizierungs- und Vorhersageanwendungen verwendet zu werden. Danach werden einige Metriken diskutiert, die später zur Auswertung eines maschinelles Lernalgorithmus verwendet werden. Zum Schluss werden ein paar Algorithmen für maschinelles Lernen beschreiben, die zur Textklassifizierung und sequentiellen Vorhersage geeignet sind.

Voraussetzungen an maschinelle Lernsysteme

Damit ein ML-System (Maschinelles Lernen System) bei der Lösung von Textklassifizierungs- und Vorhersagen-Aufgaben nützlich ist, sind folgende Merkmale erwünscht: gute Leistung, die Fähigkeit, mit fehlenden Daten und mit verrauschten Daten (Datenfehlern) angemessen umzugehen, Transparenz der diagnostischen Klassifikation, die Fähigkeit, Entscheidungen zu erklären, und die Fähigkeit des Algorithmus, die Anzahl der Tests zu verringern, die erforderlich sind, um zuverlässige Prognosen zu erhalten. [Kon01]

- **Gute Leistung:** Der Algorithmus muss in der Lage sein, wichtige Informationen aus den verfügbaren Daten zu extrahieren. Typischerweise sollte die Algorithmen mindestens so gut wie der Experten arbeiten. Wenn daher die Möglichkeit besteht, die Genauigkeit des Ergebnisses von einem Fachmann zu messen, kann deren Leistung als Untergrenze für die erforderliche Genauigkeit eines ML-Systems bei dem gegebenen Problem verwendet werden.
- **Umgang mit unvollständigen oder fehlenden Daten (Wörtern):** Bei der Verarbeitung von freiem Text in natürlicher Sprache kann die Eingabe so zufällig sein, dass der Text semantisch falsch ist oder ein Teil der Sprache fehlt. ML-Algorithmen müssen in der Lage sein, mit einer solchen unvollständigen oder falschen Texteingabe angemessen umzugehen.
- **Reduzierung der Anzahl der Texteingaben:** Es ist wünschenswert, einen Klassifikator zu haben, der mit einer geringen Anzahl von Texteingaben zuverlässig vorhersagen kann. Dies kann überprüft werden, indem alle Kandidatenalgorithmen mit einer begrenzten Menge derselben Daten versehen werden. Das Bestimmen der richtigen Teilmenge von Daten kann jedoch zeitaufwendig sein, da es sich im Wesentlichen um ein kombinatorisches Problem handelt. Einige ML-Systeme sind selbst in der Lage, eine geeignete Teilmenge von Attributen auszuwählen. Die Auswahl erfolgt beispielsweise während des Lernprozesses und ist möglicherweise geeigneter als andere, denen diese Möglichkeit fehlt.

2.4.1 Metriken zur Auswertung eines maschinelles Lernalgorithmus

Zur Auswertung der Leistung von einem maschinelles Lernalgorithmus werden Metriken wie Präzision, Ausbeute, und das F1-Maß gebraucht. Um diese Metriken besser zu verstehen, wird eine Kontigenztabelle mit einigen Notationen dargestellt (siehe Abbildung 2.5).

Die Ausbeute (engl. Recall) ist der Anteil der *aktuell* positiven Fälle, die korrekt als positiv *vorhergesagt* werden [Pow07].

$$Ausbeute = \frac{tp}{tp + fn} \tag{2.1}$$

Hier, das tp (true positive) sind die vorhergesagte Fälle, die mit dem aktuellen Modell übereinstimmen und das fn (false negative) sind die nicht erwartete Ergebnisse. Hingegen

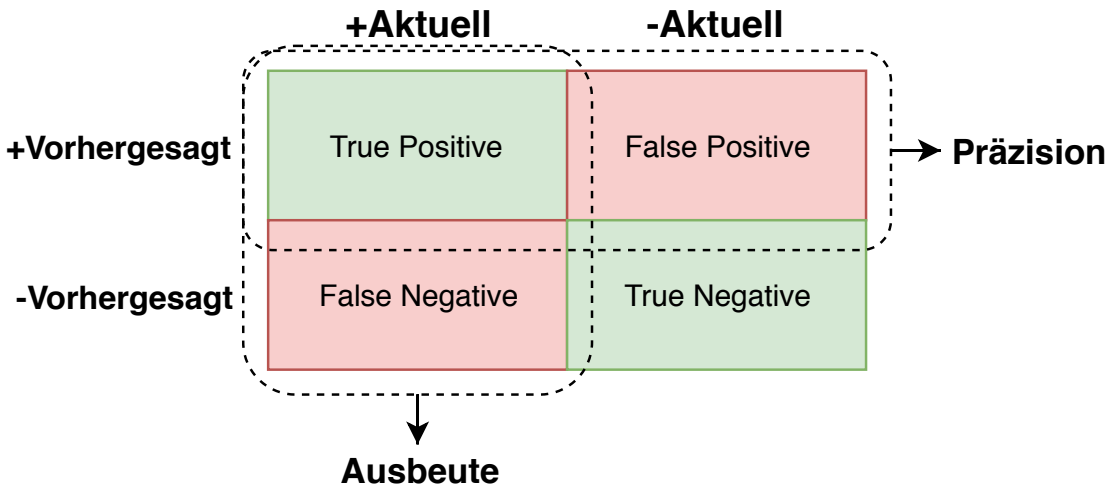


Abbildung 2.5: Systematische und traditionelle Notationen in einer binären Kontingenztabelle eigene Darstellung [Pow07]

gibt die Präzision (engl. Precision) den Anteil der *vorhergesagten* positiven Fälle an, bei denen es sich korrekt um *aktuell* Positiven handelt [Pow07].

$$Präzision = \frac{tp}{tp + fn} \tag{2.2}$$

Das fp (false positive) beschreibt den Anteil der aktuell Negatives, die als vorhergesagt Positive auftreten. Das F1-Maß (engl. F1-Score) bewertet das Zusammenspiel von Präzision und Ausbeute mittels des gewichteten harmonischen Mittels.

$$F1 = 2 \cdot \frac{Präzision \cdot Ausbeute}{Präzision + Ausbeute} \tag{2.3}$$

2.4.2 Algorithmen für maschinelles Lernen

In diesem Abschnitt werden einige maschinellen Lernalgorithmen kurz beschrieben, die geeignet für Klassifikation einer Texteingabe sind.

Rekurrentes Neuronales Netz (engl. Recurrent Neural Network, RNN)

Der Begriff Neuronale Netze sind Datenverarbeitungssysteme, die aus einer Vielzahl einfacher, stark miteinander verbundener Verarbeitungselemente in einer Architektur bestehen, die von der Struktur des Gehirnrindenabschnitts des Gehirns inspiriert ist [Uhr95]. Die neuronale Netze interpretieren sensorische Daten durch maschinelle Wahrnehmung, Kennzeichnung oder Bündelung von Rohdaten. Sie können in Vektoren enthaltene numerische Muster erkennen und in reale Daten wie Bilder, Ton, Text oder Zeitreihen umwandeln.

Im Gegensatz zu anderen neuronalen Netzen Arten können RNNs ihren internen Zustand (Speicher) verwenden, um Sequenzen von Eingaben zu verarbeiten. Dies macht sie für Aufgaben wie unsegmentierte, verbundene Handschrifterkennung anwendbar [Ale09]. Long Short Term Memory (LSTM) [HS97] ist eine spezielle Art von RNN, die in der Lage ist, langfristige Abhängigkeiten zu lernen. Eine Einheit der LSTM besteht aus einem Eingangsgatter, einem Ausgangsgatter, einem Speicher und einem Vergessungsgatter.

LSTM funktioniert im Allgemeinen wie folgt: Es werden zeitlich aufeinanderfolgende Eingaben gegeben, und der Speicher der LSTM-Einheit ist dafür verantwortlich, die Eingabesequenz zu verfolgen. Das Eingangsgatter steuert, welcher neue Wert in den Speicher

fließen kann, und das Vergessungsgatter entscheidet, welche Werte im Speicher verbleiben. Das Ausgangsgatter wird dann verwendet, um zu steuern, welcher Wert im Speicher verwendet wird, um die Ausgabe des LSTM zu bestimmen.

Stützvektormaschine (engl. Support Vector Machine, SVM)

Stützvektormaschinen (SVM) sind eine Reihe von überwachten Lernmethoden, die zur Klassifizierung, Regression und Erkennung von Ausreißern verwendet werden [SVM17]. Wenn es eine bestimmte Anzahl von Kategorien oder Klassen gibt, erstellt das SVM-Modell einen klaren Abstand zwischen die Klassen, sodass neue Daten einfach kategorisiert werden können, indem geprüft wird, auf welche Seite sie fallen.

Hidden Markov Model (HMM)

Hidden Markov Model (HMM) ist ein doppelt stochastischer Prozess mit einem zugrunde liegenden stochastischen Prozess, der nicht beobachtbar ist (er ist verborgen), sondern nur durch einen anderen Satz von stochastischen Prozessen beobachtet werden kann, die die Folge der beobachteten Symbole erzeugen [RJ86]. Ein HMM besteht aus einer Anzahl von Zuständen, denen jeweils eine Übergangswahrscheinlichkeit von einem Zustand in einen anderen Zustand zugeordnet ist. Die Zustände hängen zu jedem Zeitpunkt nur vom Zustand zum vorhergehenden Zeitpunkt ab. Ein Symbol ergibt sich aus einem der HMM-Zustände gemäß den den Zuständen zugeordneten Wahrscheinlichkeiten. HMM-Zustände sind nicht direkt beobachtbar und können nur durch eine Folge von beobachteten Symbolen beobachtet werden [YOI92].

3 Verwandte Arbeiten

In diesem Kapitel werden verwandte Arbeiten zu dem Anforderungsmanagement, Satzschablonen und die Verarbeitung von natürlicher Sprache (engl. NLP) bzw. deren Probleme zusammengefasst.

3.1 Satzschablone und Mustererkennung

Der Begriff Anforderungsmuster oder Satzschablone werden von Hull, Jackson, und Dick auf eine textuelle Schablone einer Anforderung verweisen [El05]. Eine Satzschablone besteht aus einer Folge von Attributen und festen Syntaxelementen (engl. Fixed Syntax Elements (FSEs)). Eine allgemeine Beispiel für eine Satzschablone ist „<System> sollte <Prozesswort>“. In diesem Beispiel <System> und <Prozesswort> sind Attributen und „sollte“ ist ein festes Syntaxelement. Es ist möglich mehrere Satzschablonen durch einfache Verkettung zu kombinieren, mit dieser Funktion kann man die Anzahl der benötigten Satzschablonen gering halten und gleichzeitig eine hohe Flexibilität ermöglichen. Während der Anforderungsdokumentation werden textuelle Eingabe zu den geeigneten Attributen der Satzschablonen zugeordnet.

Stalhane, Omoronyia und Reichenbach [Sta10] erweiterten die Satzschablonen mit Domain-Ontologie, indem sie die Werte der Attribute der Schablone mit Ontologiekonzepten verknüpfen. Basierend auf dem vorherigen Ontologiekonzept präsentieren Farfelder und Kollegen [FMK$^+$11] einen Ansatz, um Ontologien zur Analyse von Anforderungen zu verwenden. Der Ansatz versucht Qualitätsaspekte wie Vollständigkeit, Korrektheit und Konsistenz der Anforderungen zu untersuchen. Die Ideen von ihnen sind wie folgt :

- „Wir wollen ein System, das zumindest Teile der Anforderungen automatisch unter Verwendung von Informationen aus einer Domain-Ontologie vorschlägt.“
- „Wir wollen ein System, das Beziehungen und Axiome der Domain-Ontologie ausnutzt, d.h. ein System, das leistungsfähiger ist als nur ein einfaches Wörterbuch. Die Beziehungen und Axiome der Domain-Ontologie repräsentieren einvernehmliches Wissen der Stakeholder 2.2. Durch ihre Verwendung hoffen wir, die Präzision der Anforderungen zu verbessern.“

Farfelder und Kollegen fügte ein semantisches Leitsystem, das diese beiden Ideen implementiert, zu ihrem Werkzeug zum Ermmitlung von Anforderungmusters hinzu. Zum Testen haben sie das Werkzeug aus dem Bereich des Tür-Verwaltung-Sytem (engl. DOORS Management Systems (DMS)) angewendet.

Das Werkzeug von ihnen bietet den Benutzer eine Liste von Vorschlägen beim Ausfüllen einer Anforderungsschablone an. Die bereitgestellte Vorschläge hängen von dem Attribut der Schablone ab, das der Benutzer derzeit ausfüllt, z.B. sind die Vorschläge für das „System“ völlig anders als für die „Aktion bzw. Prozesswort“. Mit dieser Vorschlagsfilterfunktion des Werkzeug kann der Benutzer eine überwältigende vollständige Liste von Ontologie-Entitäten vermeiden. Durch die Eingabe von Zeichen wird diese Liste der Vorschläge weiter gefiltert, sodass nur die Einträge angezeigt werden, die der eingegebenen Zeichenfolge entsprechen. Die Benutzer werden nicht gedrängt, aus der Vorschlagsliste zu wählen, d.h. sie können etwas völlig anders eingeben und bei diesem Fall wird eine Aktualisierung der Ontologie durchgeführt werden, damit die Anleitung des Werkzeug für ähnlichen Anforderungen auf dem neuen Stand halten kann.

3.1.1 Satzschablone Erkennung

Zum Erkennung einer Satzschablone können wir die gleichen Methoden wie „Satzschablonen - Programmierung“ (engl. Boilerplate-Code) Erkennungsverfahren verwenden. In der Textverarbeitung sind Satzschablonen ein standardisiertes, zuvor gespeichertes Textschema, das zum Erstellen eines neuen Dokuments verwendet werden kann [IEE91].

Für die automatische Erkennung von Satzschablonen werden in der Literatur viele unterschiedliche Arten von Klassifikationsverfahren angewandt wie z.B. Stützvektormaschine (engl. Support Vector Machines (SVM)) [BDD+07], Conditional Random Fields (CRF) [SMP08], Naive Bayes [PR09], Mehrlagiges Perzeptron (engl. Multilayer Perceptrons (MLP)) [SB13]. Um jedes einzelnen Verfahren bewerten zu können, kann das F1-Maß (engl. F-score) verwendet werden (siehe Abschnitt 2.4.1). Das resultierende F-Maß der obigen Verfahren wird in Schäfer ermittelt (siehe Abbildung 3.1)

Quelle	Verfahren	F-Maß
[BDD+07, S. 120]	SVM	0,652
[SMP08, S. 16]	CRF	ca. 0,8
[PR09, S. 977-979]	Naive Bayes	über 0,9
[SB13]	MLP	0,75

Abbildung 3.1: F-Maß der Boilerplates-Erkennungsverfahren [Sch]

Der Vorteil des MLP (insbesondere bei Verwendung der FANN-Bibliothek (Fast Artrificial Neural Network)) liegt in der rechnerischen Effizienz des Verfahrens und an der Tatsache, dass mit diesem Verfahren leicht extrahierbare Daten verwendet werden können, um diesen zu verbessern. Schäfer optimiert sein MLP-Verfahren und erzielt durch die Extraktion geeigneter textblockinterner Daten eine Genauigkeit von bis zu 95% (Genauigkeit, Trefferquote und F-Maß über 0,95).

Weiterhin definiert Kohlschütter, Fankhauser, und Nejdl [KFN10] einen Ansatz zur Erkennung einer Satzschablone mittels „flacher Text-Verfahren (engl. shallow text features)“,der theoretisch auf stochastischen Texterzeugungsprozessen aus der quantitativen Linguistik beruht. Der Begriff „Shallow Learning“ ist eine Art von Algorithmus für maschinelles Lernen, der mit nur wenigen Kompositionsebenen ein gutes verallgemeinertes Vorhersagemodell erzeugen kann [PS16]. Diese Art von Verfahren kann eine gute Leistung erbringen, obwohl nur eine begrenzte Anzahl von Proben verfügbar ist. Beispielverfahren für das Shallow Learning sind artifiziell neuronale Netze mit einer verdeckte Schicht und SVM.

Semantische Rolle

Semantische Rolle Markierung (engl. Semantic Role Labeling, kurz SRL) ist eine Form der flachen semantischen Syntaxanalyse, deren Ziel es ist, die Prädikat-Argument-Struktur

jedes Prädikats in einem bestimmten Eingabesatz zu ermitteln [ZX15]. Es gibt schon verschiedenen Ansätze oder Lernverfahren, die die SRL erkennen können wie z.B. durch Abhängigkeitsbaum (dependency tree) [Hac04], Stützvektormaschine (SVM) [PWH+05], und LSTM [ZX15] [HLLZ17].

Zhou und Xu schlugen ein End-to-End-System vor, das ein bidirektionales LSTM verwendet, um die semantische Rolle zu bestimmen. Sie verwenden nur Originaltext als Eingabe-Features, *ohne zwischengeschaltete Tags* wie syntaktische Informationen (z.B. Wortartmarkierung, NER) [ZX15]. Im Gegensatz zu Zhou und Xu Ansätze behauptet Punyakanok, dass die syntaktische Informationen eine wichtige Rolle zur Erkennung einer semantischen Rolle spielen [PRY08].

3.2 Nominalisierungen

Bei der Dokumentation der Anforderungen können verschiedene Probleme auftreten, wie z.B. unvollständige und fehlerhafte Beschreibungen der Anforderungen. Zu diesem Problem haben Korner und Brumm das „Requirements Engineer Specification Improver (RESI)“ entwickelt. RESI überprüft die Anforderungsbeschreibung nach sprachlichen Mängeln [KB09]. Es ist dabei im Stande Mängel wie unvollständige Prozesswörter oder Konditionen sowie Mehrdeutigkeiten und Nominalisierungen zu erkennen.

Weitere Präzisierung zu dem Nominalisierung-Problem wird Körner et al. mit dem Werkzeug DeNom klassifizieren [LKT+15]. Die Idee des Werkzeugs ist, dass nicht jede Nominalisierung ein Problem darstellt. Aus diesem Grund hat DeNom das Ziel zwischen akzeptablen und problematischen Nominalisierung zu unterscheiden. Hierzu verwendet DeNom zunächst die Funktionen von RESI, um Nominalisierungen zu erkennen. Hierbei werden diese in vier Kategorien unterteilt (siehe Abbildung 3.2).

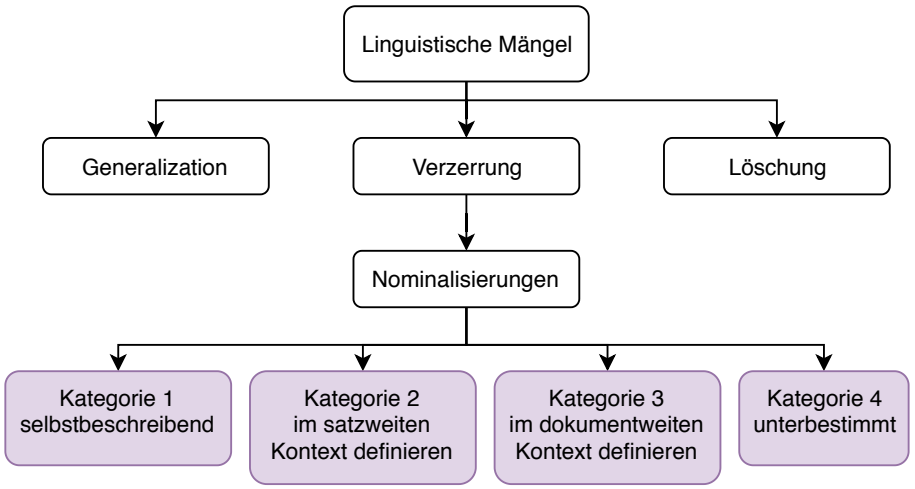


Abbildung 3.2: Kategorisierung der linguistische Mängel und der Nominalisierung aus [KB09]

Danach werden die erhaltene Ergebnisse verarbeitet und die problematischen Nominalisierungen (4. Kategorie) dem Benutzer dargestellt.

3.3 Schwachen Worten (engl. Weak Words)

Pohl und Rupp empfehlen Anforderungsautoren, die Verwendung von Passivsätzen und schwachen Worten (engl. Weak Words) zu vermeiden, da dies zu unvollständigen Anforderungen führen kann [Kla09]. Zu diesem Thema haben Krisch und Houdek die Auswirkungen von Passiv und schwachen Worten diskutiert [KH15]. Das Ergebnis der Diskussion

ergab, dass es von dem Kontext abhängt, ob die Verwendung von Passivsätzen oder schwachen Worten akzeptabel ist oder nicht. Passivsätze sind fast nie problematisch und nur 12% der schwachen Worte führen zu problematischen Anforderungen.

Es existieren Werkzeuge, welche automatisch schwache Worte erkennen wie z.B. ReQualize [Hei10] und DESIRe [Fra09]. Die beiden benutzen einen Ansatz, der auf Wortlisten basierend. Diese Wortlisten werden mit den Worten der Anforderungsbeschreibung abgeglichen. Enthält eine Anforderung eines der vordefinierten schwachen Worte, wird eine entsprechende Warnung ausgegeben. Der sprachliche Kontext der Anforderung wird in diesen Fällen jedoch nicht berücksichtigt [KH15].

3.4 Transformation komplexer Sätze in eine semantische Hierarchie

Neben der Erkennungsfunktion einer problematischen Anforderungen wäre es auch wichtig, die Struktur der textuelle Eingabe einer Anforderung zu vereinfachen. Niklaus und Kollege schlugen [Nik19] einen Ansatz zur rekursive Aufteilung und Umformulierung komplexer **englischer** Sätze in eine neueartige semantische Hierarchie vor.

Unter Verwendung von selbst definierten Transformationsregeln werden Eingabesätze rekursiv in zweischitige hierarchische Darstellungen in Form von Kernsätzen und zugehörigen Kontextangaben umgewandelt. Um dies besser zu verstehen, betrachtet wir das Beispiel von [Nik19]: „Although the Treasury will announce details of the November refunding on Monday, the funding will be delayed if Congress and President Bush fail to increase the Treasury's borrowing capacity“. Diese Eingabe wird in ein paar einfache Sätze umgewandelt, die wie in diese Abbildung 3.3 dargestellt sind.

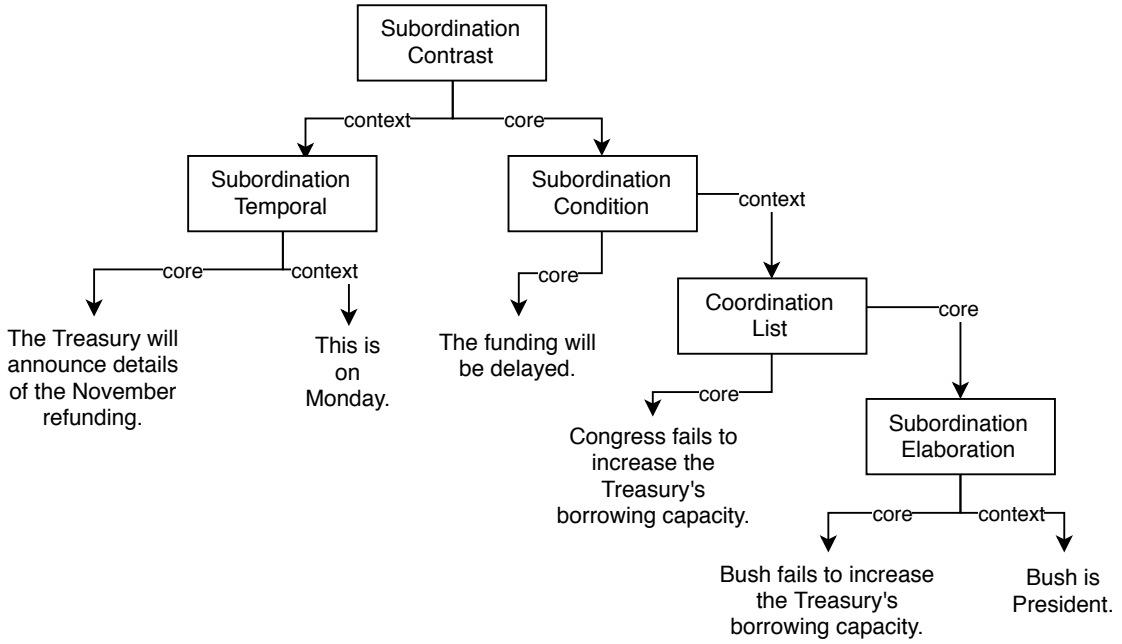


Abbildung 3.3: Finale Zerlegung des Beispiel-Eingabe aus [Nik19]

Diese Darstellung veranschaulicht den Vorteil dieses Verfahrens. Durch die hierarchische Strukturierung des komplexen Satzes, wird der Inhalt verständlicher und übersichtlicher. Zudem können die so extrahierten Informationen einfacher weiterverarbeitet werden.

3.5 Autovervollständigung

Autovervollständigung ist eine inzwischen sehr etablierte Hilfsfunktion, welche unter anderem bei namhaften Suchmaschinen zum Einsatz kommt, um den Anwender bei der Eingabe zu unterstützen. Bar-Yossef und Kraus [Ziv11] beschreiben hierzu einen Ansatz, um die Vervollständigung der Texteingabe zu unterstützen. Es basiert auf einer etablierten Vervollständigungsmethode der „Most Popular Completion“ (MPC). Die hierbei vorgeschlagenen Wortvervollständigungen basieren auf der Suchpopularität von Abfragen, welche mit dem vom Benutzer eingegebenen Präfix übereinstimmen.

Die dabei ermittelten Vervollständigungsmöglichkeiten werden anschließend regelbasiert und nach vordefinierten Kriterien gefiltert und das Ergebnis dem Anwender dargestellt (siehe Abbildung 3.4).

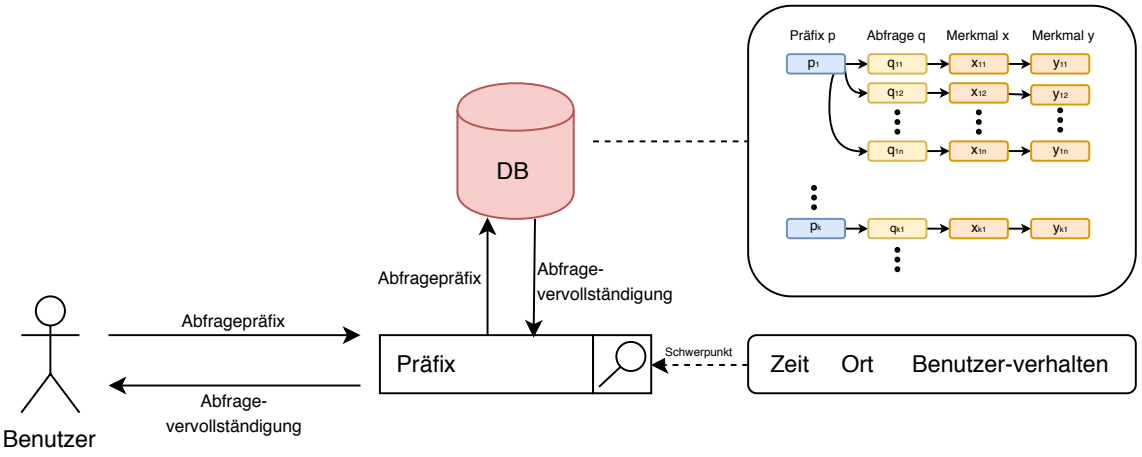


Abbildung 3.4: QAC Framework [Fei16]

Wie aus der obigen Abbildung hervorgeht, werden die ermittelten Möglichkeiten nach möglichen Kriterien wie z.B. Zeit, Ort oder dem Benutzerverhalten sortiert.

Die Ziele der Autovervollständigung von Abfragen besteht darin:

- Zeit für die Eingabe einer Abfrage sparen
- Rechtschreibfehler vermeiden
- Relevante Suchbegriffe ermitteln, indem man die vom Benutzer vorgesehene Abfrage an die oberste Position einer Liste von Abfragevervollständigungen zurückgibt [Fei16].

Die Vorhersage des nächstes Wortes

Der entscheidende Vorteil der Wortvorhersage ist die Einsparung von Aufwand, da für eine bestimmte Textlänge weniger Tastenanschläge erforderlich sind und die Textproduktion fehlerfreier ist [MH02]. Nakamura et al. entwickelt einen Ansatz, der mit Hilfe von neuronalen Netzen die Wortkategorien zu vorhersagen [NMKS90]. Dies gibt man die Idee, dass die Vervollständigungsvorschläge der Autovervollständigung möglicherweise nicht direkt auf dem Wort, sondern auf der Wortkategorien basiert. Dadurch erhaltet man mehr Vervollständigungsvorschläge. Andere Arten von Darstellung der Vorhersage eines Wortes haben Linmei Hu et al. vorgestellt, nämlich die Vorhersage eines Unterereignis (engl. Subevent) durch kontextuelle hierarchische LSTM [hLN+17]. Ein Modell wird entwickelt, das die Texte, die frühere Unterereignisse beschreiben, direkt als Eingabe verwendet und automatisch einen kurzen Text generiert, der ein mögliches zukünftiges Unterereignis beschreibt.

Sonstiges

Morillot et al. schlagen einen Vorverarbeitungsansatz vor, um Grundlinienversatz und Schwankungen mittels BLSTM (engl. Bidirectional Long Short-Term Memory) effizient zu korrigieren [MLSG13]. Dieser Ansatz zeigen, dass RNN zur Korrektur von Wörtern geeignet ist. Eine Hilfestellung wie Rechtschreibfehlerprüfer kann man z.B basierend auf diesem Ansatz entwickelt.

3.6 Werkzeuge für das Anforderungsmanagement

Auf dem heutigen Markt gibt es eine Vielzahl an Softwarelösungen, welche sich auf das Management von Anforderungen spezialisiert haben wie z.B. IBM Doors, PTC Integrity oder die System Engineering Suite der REUSE Company.

Insbesondere die System Engineering Suite der REUSE Company stellt durch ihre umfangreichen Assistenz-Module die für diese Arbeit am interessanteste Softwarelösung dar, welche im Folgenden genauer untersucht wird.

3.6.1 RAT und KM

Die Systems engineering Suite besteht aus mehreren Teilmodulen: „Requirement Authoring Tools (RAT)“ [Comb] und „Knowledge Manager (KM)“ [Coma].

Das Modul RAT unterstützt den Anwender bei der Dokumentation von Anforderungen. Hierzu werden Satzschablonen verwendet, welche zuvor in dem Modul KM definiert werden müssen, um den Anwender bei der Erstellung von Anforderungen zu unterstützen. Hierzu werden in KM Satzschablonen bestehend aus aneinander gereihten Wortarten definiert (siehe Abbildung 3.5).

The	aircraft	shall	fly
ART	NOUN	MODAL VERB	VERB

Abbildung 3.5: KM Mustererkennung Beispiel [Coma]

Zusätzlich zu den Wortarten ist es möglich, semantische Rollen einzuführen, welche anschließend mit Wortlisten verknüpft werden, in denen der Anwender entsprechende Begriffe definieren muss, welche das System für die spätere Erkennung der semantischen Rollen verwendet (siehe Abbildung 3.6).

The	pilot	shall	fly	the	aircraft
ART	NOUN	MODAL VERB	VERB	ART	NOUN
	Stakeholder		Action		System

Abbildung 3.6: KM Schablone mit hinzugefügten semantischen Rollen [Coma]

Basierend auf diesen Schablonen und den vordefinierten Wortlisten, kann das RAT Modul die Eingabe mit den vordefinierten Schablonen abgleichen und entsprechende Hilfestellungen geben. Außerdem bietet KM eine hierarchische Struktur zwischen Schablonen, mit der eine Schablone mit einer anderen Schablone als Komponente erstellt werden kann (siehe Abbildung 3.7.

When	**the**	**light**	**is**	**green,**	**the car shall run**
Time ADV	**ART**	**NOUN**	**VERB To-Be**	**NOUN**	**Schablone aus 3.5**

Abbildung 3.7: KM wieder Verwendung von exsitierende Muster [Coma]

4 Analyse

Wie in Abschnitt 1.2 beschrieben wird, ist das Ziel dieser Arbeit die Entwicklung eines Konzepts zur Unterstützung des Dokumentationsprozesses von Anforderungen. Als Eingabedaten werden hierzu Anforderungen verwendet, welche in Form von Freitext vorliegen. Des weiteren werden die nach SOPHIST beschriebenen Satzschablonen als Grundlage verwendet, um die zu dokumentierenden Anforderungen zu standardisieren und ihre Vollständigkeit sicher zu stellen (siehe auch Kapitel 2).

Ziel dieses Kapitels ist die Analyse des Stands der Technik und Forschung, um Anforderungen an das zu entwickelnde Konzept abzuleiten. Hierzu werden in einem ersten Schritt die in Kapitel 3 beschriebene Software hinsichtlich der Probleme im Dokumentationsprozess von Anforderungen analysiert und Minimalanforderungen der Arbeit an das vorliegende Konzept abgeleitet. Im zweiten Schritt wird der linguistische Aufbau von Anforderungen analysiert, um mögliche Herausforderungen bei der computerlinguistischen Auswertung der Freitexteingabe abzuleiten. Anschließend werden mögliche Hilfestellungen diskutiert und entsprechende Anforderungen an das Konzept abgeleitet.

Bemerkung: Die *Anforderung der Arbeit* wird ab jetzt als die *Voraussetzung der Arbeit* genannt, um den Begriff „Anforderung“ als *Daten* und als *Bedarf* zu unterscheiden.

4.1 Analyse aktueller Softwarelösungen

In diesem Abschnitt werden die Werkzeuge RAT und KM aus dem Kapitel 3 diskutiert. Der Dokumentationsprozess des RAT und KM wird als der Vergleichspunkt verwendet, um einen Überblick über die Funktionsweise des Dokumentationsprozesses einer Anforderung zu erhalten. Dieser Abschnitt wird in zwei Teile unterteilt. Im ersten Teil werden die minimale Voraussetzungen der Arbeit besprochen. Dies sind die Funktionen, die RAT und KM bereits haben, und die Kriterien, die in dieser Arbeit erfüllt werden müssen. In dem zweiten Teil werden dann die weiterentwickelte oder neue Vorbrautsetzungen der Arbeit aus RAT und KM untersucht, welche die Mängel von RAT und KM untersucht werden.

4.1.1 Minimale Voraussetzungen

Automatische Satzschablone Erkennung aus Freitext

Wie in Kapitel 3 erwähnt, RAT und KM bieten die Möglichkeit, eine textuelle Eingabe zu einer Satzschablone zuzuordnen. Betrachtet man jetzt die Abbildung 3.5 in der Kapitel 3.

Der Beispielsatz von der Abbildung 3.5 ist ein Beispiel von einer Satzschablone des KM. Und gleich wie diese Arbeit, die Texteingaben des RAT und KM sind in Form von freiem Text. Die Eingaben werden als nächstes extrahiert und dazu bekommt man gereihten Wortarten als Ergebnisse. Diese gereihten Wortarten werden mit dem existierenden Schablone verglichen. Falls es zu einer der Schablone übereinstimmt, wird der gesamte freie Text als eine Anforderung mit der entsprechenden Satzschablone bezeichnet. Sonst wird die Eingabe nicht akzeptiert und man muss dazu eine neue Satzschablone erzeugen.

Voraussetzung 4.1. Das System muss in der Lage sein, eine passende Satzschablone aus einem grammatikalisch gleich strukturierten Freitext automatisch zu erkennen.

Voraussetzung 4.2. Das System muss in der Lage sein, neue Satzschablone bestehend aus mehreren aneinander gereihten Wortarten zu definieren.

Wiederverwendung von Satzschablone

Die oben geschriebene Erkennungsalgorithmus wird auch bei der Erzeugung einer neuen Satzschablone verwendet, da in KM beim Erstellen einer neuen Schablone ein Beispielsatz angegeben werden muss, der auch in Form von freiem Text vorliegt. Die Idee der Nutzung des Algorithmus dahinter sind, damit man nicht die gleiche strukturierte existierende Schablone als eine neue Schablone erzeugt (Redundanz vermeiden) und in dem Fall, dass eine Schablone nur ein Teil des Satzes ist, kann die Schablone durch das Algorithmus erkannt und wiederverwendet werden. Die wiederverwendete Schablone wird dann als ein Teil der neuen Schablone verwendet (siehe Abbildung 3.7.

Betrachtet man jetzt z.B. „die funktionale Anforderung mit Bedingung“ Schablone. Angenommen es existiert schon *die funktionale Anforderung Schablone* und *die Bedingung Schablone*. Mit dem Wiederverwendungskonzept des KM kann man die funktionale Anforderung mit Bedingung Schablone aus der zwei vorhandenen Schablonen kombinieren. Diese ermöglicht man, dass die komplexere Zusammenhänge aus bereits bekannten Schablonen ableiten kann.

Voraussetzung 4.3. Das System muss in der Lage sein, beim Erzeugung einer neuen Schablone auch andere Schablonen zu integrieren.

Semantische Rolle

Das letzte Konzept von RAT und KM, das als minimale Voraussetzung der Arbeit angenommen wird, ist das Konzept von semantischen Rollen. Bei RAT und KM hat jedes Wort eine bestimmte Wortart (wie z.B. Nomen, Verb, Modalverb) und kann auch einer semantische Rolle (wie z.B. Stakeholder, Aktion, System) zugeordnet sein. Bei der Abbildung 3.6 schaut man, dass die Wortart „Nomen“ unterschiedliche semantische Rolle haben. Die semantische Rollen bieten mehr Möglichkeiten, eine Schablone auf vielen verschieden Formen zu erstellen als nur mit Wortarten aus der Linguistik. Anderseits kann man die semantische Rolle als der Kontext für die Vervollständigungsvorschläge des nächstes Wort benutzt.

Voraussetzung 4.4. Das System muss in der Lage sein, semantische Rollen zu erkennen.

4.1.2 Weiterentwickelte Voraussetzungen

Ein Problem, welches sich für Nicht-Experten bei der Anwendung von RAT und KM stellt, ist die fehlende Unterstützung bei Texteingaben, welche weder zur vordefinierten Schablone noch der vordefinierten Wortlisten entspricht. In diesem Fall, muss der Anwender die angezeigten Fehler selbst interpretieren können, was dazu führt, dass das Werkzeug nur von Experten sinnvoll genutzt werden kann. Eine mögliche Lösung hierbei wäre eine Funktion, welche dem Anwender mit Hilfe von Beispielen darstellt, wie er die aktuelle Eingabe anpassen sollte, um die vordefinierten Satzschablonen einzuhalten.

Voraussetzung 4.5. Das System sollte in der Lage sein, eine passende Satzschablone vorzuschlagen, auch wenn die angegebene Eingabe nicht den vorgegebenen Satzschablone entspricht.

Ein weiteres Verbesserungspotential in der Vereinfachung der Expertenwerkzeuge RAT und KM, stellt die Autonomisierung der NLP-Algorithmen von Wortlisten dar. RAT verwendet für die Erkennung von semantischen Rollen eine Kombination aus Wortarten und Wortlisten. Wörter welche nicht in der Wortliste enthalten sind, werden nicht korrekt erkannt, was zu einem hohen initialen Definitionsaufwand führt, welcher vor allem für kleine und mittelständische Unternehmen nicht realisierbar ist. Zudem erfordert dieses Verfahren entsprechendes Expertenwissen, was ebenfalls zu aufwändigen Schulungen und Mehrkosten führt. Um dies zu vermeiden, soll mit Hilfe aktueller computerlinguistischer Analyseverfahren ein Konzept entwickelt werden, welches den flächendeckenden Einsatz von Wortlisten vermeidet. Welche folgt die Verbesserungsmöglichkeit von der Voraussetzung 4.4.

Voraussetzung 4.6. Das System muss in der Lage sein, semantische Rollen auch ohne umfangreiche Wortlisten zu erkennen.

Andere Verbesserungspotential stellt die Benutzerfreundlichkeit der Anwendung dar. Da RAT und KM Werkzeuge sind, welche für Anforderungsmanagementexperten ausgelegt sind, sind die Kontextmenüs komplex und wenig intuitiv (siehe Abbildung 4.1.

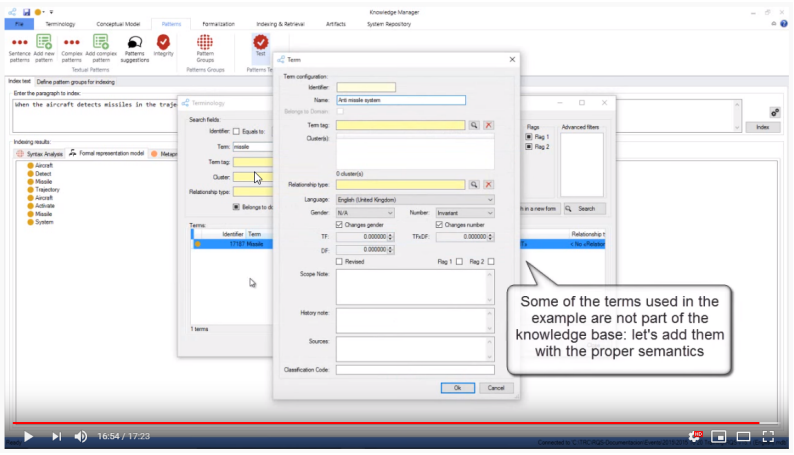


Abbildung 4.1: Ein Screenshot von einem Menu des KM

Eine Lösung hierfür wäre eine vereinfachte Oberfläche, welche sich auf die wesentlichen Merkmale beschränkt und mögliche Hilfestellungen nur dann einblendet, wenn diese benötigt werden z.B. durch Popup-Menüs oder in einer Seitenleiste der Hauptanwendung.

Zusammenfassend sind die Funktionen, die RAT und KM für das Anforderungsmanagement bereitstellen (Wiederverwendung von Satzschablonen, semantische Rolle, usw.), eine gute Basis für das Konzept zur Dokumentationsprozess der Anforderung. Außerdem muss man die Mängel von RAT und KM verbessern, die unter bestimmten Umständen den neuen Anforderungstext nicht erkennen oder mit seinem Muster abgleichen kann, so dass in dieser Situation der Aufwand für den Benutzer nicht verschwendet wird.

4.2 Aufbau von Anforderungen

Für die vorliegende Arbeit dienen die Satzschablonen nach SOPHIST als Basis für die Dokumentation eine gute Anforderung (siehe auch Kapitel 2). Da Satzschablonen jedoch

die Freiheiten des Anwenders einschränken, sollte das Konzept in der Lage sein, Freitexteingaben, automatisch in die vordefinierten Satzschablonen zu überführen.

Voraussetzung 4.7. Das System sollte fähig sein, einen Freitext zu der am ehesten zutreffenden Satzschablone umformen zu können.

Beispielsatz: Umformung eines unvollständigen Text zu einer nicht funktionale Anforderung

Der rote Apfel. –> Die <Eigenschaft?> des Apfels muss gleich rot sein.

Weil eine Satzschablone aus gereihten Wortarten oder semantische Rolle bestehend, muss vor allem eine linguistische Analyse für die freie Texteingabe durchgeführt werden. Hierbei werden die innerhalb der Eingabe befindlichen semantischen Rollen erkannt und markiert. Die semantische Rollen in dieser Arbeit werden nach Satzschablonen von SOPHIST abgeleitet. Daraus folgt folgende Voraussetzung für das vorliegende Konzept (Zusatzbedingung zur Voraussetzung 4.4).

Voraussetzung 4.8. Das System muss in der Lage sein, semantische Rollen nach SOPHIST automatisch zu erkennen.

4.2.1 Gute Anforderungen

In diesem Abschnitt werden mehr auf die funktionale Anforderung und nicht-funktionale Anforderung Satzschablone als die Bedingung Schablone betrachtet. Aus dem Kapitel 2, eine funktionale Anforderung kann in drei Arten von Funktionalität unterscheiden und zwar die:

- selbsttätige Systemaktivität,
- Benutzerinteraktion,
- und Schnittstellenanforderung.

Diese drei Arten werden jetzt diskutiert, denn eine gute bzw. standardisierte Anforderung in dieser Arbeit ist eine Anforderung, die in einer der Satzschablone von SOPHIST übereinstimmt.

Das Auto des Jahres	sollte	dem Fahrer	die Möglichkeit bieten	4 Personen	zu transportieren
SYSTEM	**PRIORITÄT**	**AKTEUR**	**FUNKTION**	**OBJEKT**	**PROZESSWORD**

Abbildung 4.2: Beispielsatz für eine funktionale Anforderung mit Benutzerinteraktion Funktionalität

Abbildung 4.2 zeigt ein Beispiel wie die Benutzerinteraktion-Funktionalität aussieht. Die semantische Rollen bei diesem Satz, die der Unterschied zwischen den anderen machen, sind die „Akteur“ und „Funktion“. Ohne die beiden semantischen Rollen wird dem Beispielsatz als eine selbsttätige Systemaktivität Anforderung erkannt „Das Auto des Jahres sollte 4 Personen transportieren“. Für die Schnittstellenanforderung, die wichtigste semantische Rolle ist die semantische Rolle „Funktion“, welche mit dem Wort „fähig sein“ gefüllt ist. Dies beschreibt die funktionale Vereinbarungen und Einschränkungen des System.

Außerdem überlegt man andere Möglichkeiten von einer guten funktionalen Anforderung, indem man einen Teil der semantischen Rollen entfernt oder einfügt. Der Satz der Anforderung sollte in diesem Fall noch sinnvoll sein, damit der als eine gute Anforderung

Das Auto sollte transportieren.(1) Das Auto sollte fahren.(2)

bezeichnet werden kann. Betrachtet man nun die beiden folgenden Sätze, welche die semantische Rolle „Objekt“ entfernt werden 4.2.1.

Der (1) und (2) Satz haben die gleiche semantische Rolle, nur die Inhalte des Prozesswortes ist unterschied. Der (2) Satz ist ein sinnvoller Satz, welche man als eine funktionale Anforderung bezeichnet kann. Andernfalls der (1) Satz ist grammatikalisch richtig aber semantisch nicht. Da das Verb „transportieren“ ein Objekt braucht, um der Satz sinn zu machen. In deutscher Sprache gibt es Verben, die ohne Objekt sinnvoll (wie z.B. fahren, laufen, fliegen) sind, die mindestens eine andere Objekt haben (wie z.B. transportieren, essen, finden) und die mindestens zwei anderen Objekte (akkusativ und dativ) haben, wie z.B. (bringen, empfehlen, geben). Falls das Prozesswort einer funktionalen Anforderung ist das Verb mit mindestens zwei anderen Objekte sollte die Benutzerinteraktion-Funktionalität Art verwendet werden, da die „Akteur“ als das Dativ-Objekt des Satzes angewendet werden kann. Daraus folgt, dass das Prozesswort eine große Rolle spielt, ob der Satz ein Objekt hat oder nicht und welche Art von Funktionalität der funktionalen Anforderung verwendet wird.

Voraussetzung 4.9. Das System sollte in der Lage sein, Arten von Verben (Wortart) als eine der Arten von Prozesswörtern (semantische Rolle) zu erkennen.

Eine weitere Möglichkeit ist durch eine zusätzliche semantische Rolle heißt *Präzisierung*. Zu diesem semantische Rolle hat SOPHIST bei der detaillierter FunktionMASTer bzw. funktionale Anforderung erläutert[Rol13]. Wobei die Detaillierungsmöglichkeiten beziehen sich auf die letzten beiden semantischen Rollen der Schablone: Präzisierung des „Objekts“ und Konkretisierung des „Prozessworts“. Die Präzisierung des Objekts wird mit Hilfe von Fragewörtern wie „welche?“ und „wessen?“. Auf der anderen Seite die Konkretisierung des Prozesswortes bezieht sich auf die Fragewörtern wie „wann“, „wo“, „wohin“, und „woher“. Aber in dieser Arbeit wird eine Präzisierung oder Konkretisierung wie Definition 4.10 definiert.

Definition 4.10. Eine Präzisierung ist eine Angabe (wie Zeitangabe, Lokalangabe, usw.), ein Relativsatz oder ein Wortart, die eine semantische Rolle weiter erklärt oder ergänzt.

Ab dem 15. März sollte das Auto fahren können.(3) Das Auto sollte ab dem 15. März fahren können. (4) Das Auto, dessen Farbe gelb ist, sollte fahren können.(5)

Der (3) und (4) Satz sind Beispiele von Zeitangaben als Präzisierung des System in zwei verschiedene Reihenfolge oder Position und der (5) Satz repräsentiert einen Relativsatz. Diese drei Beispiele erklärt, warum die Präzisierung des SOPHIST wird nicht eins zu eins benutzt, sondern weiter entwickelt. Erster Grund ist, die Zeitangabe oder andere Arten von Angabe eines Satzes muss nicht unbedingt vor dem Prozesswort stehen. Nächster Grund ist, eine Präzisierung kann auch in ein ein Bezugsobjekt oder einen Akteur stattfindet.

Zur Erkennung einer Präzisierung sollte man als erstes bestimmen, nach welcher semantischen Rolle passt eine Präzisierung an. Die Präzisierung kann man z.B. mit der Hilfe von Kombination zwischen NLP-Methoden und eigene selbst definierten Regeln erkennen (siehe Abschnitt 5.1 für die ausführliche Schritte).

Voraussetzung 4.11. Das System sollte in der Lage sein, die Präzisierungen der Anforderung zu erkennen.

Aber was passiert, wenn der Präzisierungen komplex oder zu viel sind?

> Die Fräsmaschine, die vier Fräsern mit Typ XZ1 hat, muss fähig sein, Metalle mit hohem Widerstand mit Bezug auf das Referenzmodell (z.B. MX23) inklusive Meta-Informationen aus der Firma „Metallica“ abzufräsen(6)

Der (6) Satz hat eine Präzisierung für die Fräsmaschine, eine innere Präzisierung für den Fräser und viele Präzisierungen für Metalle. Es wird überlegt, ob es sinnvoll ist, alle diese Angaben zu speichern. Hierbei betrachtet man den (5) und (6) Satz. Die Präzisierung dieser Sätze beschreibt die Eigenschaft von einem Nomen (Bezugsobjekt oder Objekt), welche ein Eigenschaft oder Attribut ein offensichtliche Kennzeichnung ist um eine nicht-funktionale Anforderung (sieheKapitel 2) zu spezifizieren. Auf diesem Grund kann z.B. diese Präzisierung als eine andere Anforderung speichern, zudem betrachtet man wieder der (5) Satz. Dieser Satz kann dann in zwei Teilen zerteilt und zwar zu einer funktionalen Anforderung und einer nicht-funktionalen Anforderung.

> Das Auto sollte fahren können.(7)
> Die Farbe des Autos sollte gleich gelb sein.(8)

In dem Fall von (3) und (4) Satz, die Präzisierung „Ab dem 15. März“ beschreibt die Bedingung oder Einschränkung von der gesamte Anforderung. Hierbei kann die Bedingung Satzschablone (siehe Abbildungen .3 und .4) angewendet werden. Da die Präzisierung als ein Ereignis interpretiert werden kann, benutzt man die „Solange“ Bedingungsschablone. Sodass die Anforderung kann wie folgt formuliert, „Solange die Datum 15 März ist, sollte das Auto fahren können“. Falls die 4.11 erfüllt ist, dann sollte man z.B einen Auslöser definieren um den Unterschied zwischen Präzisierung eines Objekt und Präzisierung der gesamten Anforderung zu kennzeichnen (siehe Abbildung 4.3).

Voraussetzung 4.12. Das System sollte in der Lage sein, die Präzisierung von einem Objekt und gesamten Anforderung zu unterscheiden.

Bemerkung: Der Grund, warum eine Anforderung mit komplexe Sätze wird zu mehreren einfacher Anforderung zerteilt, ist, basierend auf die Auswertung von der Christina Niklaus kann die Präzision und Rückverfolgbarkeit einer Information-Extraktion System verbessert wird [Nik19]. Was mit Rückverfolgbarkeit in dieser Arbeit gemeint ist, die Fähigkeit, vorhandene oder existierende Objekte der Anwendung wiederzuverwenden und die Beziehungen zwischen ihnen und entsprechende Anforderung zu erzeugen (siehe Abbildung 5.2).

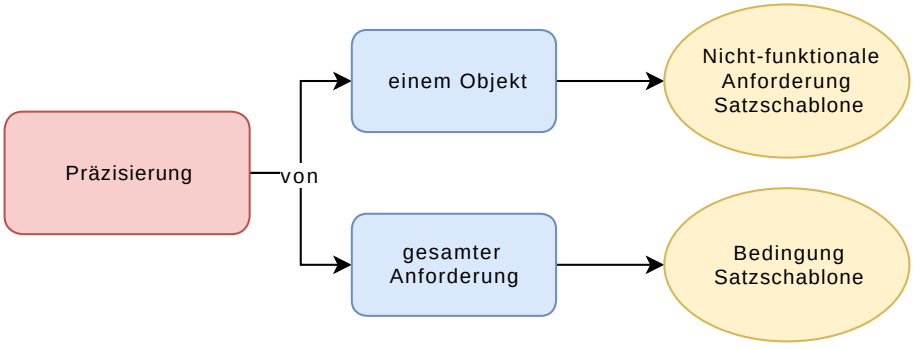


Abbildung 4.3: Mögliche Kategorien der Präzisierung

4.2.2 Problematische Anforderungen

Einige Arten von einer guten Anforderung wurde gerade besprochen. Nun bleibt die Frage, ob im Allgemein die Anforderungen, die in der Realität gefunden wird, vollständig oder recht strukturiert sind. Die Antwort ist offensichtlich nein. Da manche Anforderung **unvollständig, unklar, nicht eindeutig** ist. Um das Problem genauer zu wissen, betrachtet man jetzt dieses Beispiel:

Beispielsatz: Eine unklare Anforderung

Das System muss robust sein.

Das Problem in diesem Beispielanforderung ist das Wort „robust“. Robustes System ist in der Tat eine sehr wünschenswerte Sache, aber es gibt kein quantitatives Element in dieser Aussage, weil man die Robustheit so nicht messen kann.

Den Ausdruck von Robustheit durch Metriken, die im Allgemeinen als Indikator für diese Qualität verwendet werden, ist eine schnelle und einfache Möglichkeit, die vom Kunden bereitgestellten Information zu verbessern. In diesem Fall kann zum Beispiel die Abfrage der Zielzeit für einen Neustart nach einem Fehler, als eine Indikator der Robustheit des System verwendet werden.

Ein weiteres Problem ist die **Unvollständigkeit** einer Anforderung. Schaut man jetzt den (8) Satz einer nicht-funktionalen Anforderung. Aus dem Beispiel wird untersucht, die Möglichkeit einer unvollständigen Anforderung zu permutieren, indem bestimmte semantische Rolle oder einen Teil der Schablone entfernt wird. Die Ziele von diesen Permutationen ist wie bei der funktionalen Anforderung Permutationen 4.2.1 aber fokussiert mehr auf die Richtung schlechte Anforderung zu finden.

Die Farbe sollte gleich gelb sein.(9)
Das Auto sollte gleich gelb sein.(10)
Die Farbe des Autos sollte gelb sein.(11)
Das Auto sollte lackiert werden.(12)

Dem Satz (11) wurde der Vergleichsoperator entfernt. Trotzdem hat er dieselbe Bedeutung wie die ursprüngliche Anforderung. Dies ist jedoch nicht immer der Fall, wenn das Vergleichsoperator nicht „gleich“ ist. In dem Fall die Vergleichsoperatoren „ungleich, größer als, oder kleiner als“ verwendet werden, unterscheiden sich die Bedeutungen zwischen der ursprünglichen und der modifizierten Anforderung. Der (9) Satz ist unvollständig, da der fehlt „auf welche System wird die Farbe gleich gelb sein“. Der (12) Satz ist eigentlich eine vollständige funktionale Anforderung Schablone aber zudem kann man z.B. der fehlende Attributswert ausfüllen.

Das Beispiel (10) ist etwas anders, obwohl die semantische Rolle „Attribut“ fehlt ist, kann man möglicherweise das fehlende Attribut zum Beispiel mit Hilfe einer Ontologie-basierte Datenbank erfüllen (siehe Kapitel 2 noch nicht geschrieben). Im Vergleich zu einem normalen Datenbank, Ontologie-basierte Datenbank kann Beziehungen zwischen den Daten von unterschiedlichen Klassen abbilden. Klassen hierbei sind die semantischen Rollen. Dies bedeutet, dass der Wert oder das Schlüsselwort „gelb“ (Attributswert) als „Farbe“(Attribut) abgezogen werden kann.

Eine Ontologie-basierte Datenbank oder eine andere ähnliche Methode kann die Eindeutigkeit einer Anforderung nicht garantieren, da nicht alle Attributswerte eindeutig sind. Betrachtet man z.B. den Satz „Das System muss schnell sein“ an. Der Attributswert „schnell“

ist nicht eindeutig, da es verschiedene Attribute wie Geschwindigkeit oder Laufzeit gibt, die durch das Wort „schnell“ beschrieben werden können. Die nicht eindeutigen Attributswerte können mit dem gleichen Verfahren behoben werden, wie bei Satz (9) oder (12) Satz. Die Eindeutigkeit und Vollständigkeit einer Anforderung ist wichtig für die Performanz des Systems.

Außerdem kann aus dem obigen Beispiel ein anderes Problem identifiziert werden, nämlich wie zwischen die semantische Rolle „Attribut“ und die semantische Rolle „System“ zu unterscheiden ist.

Voraussetzung 4.13. Das System sollte fähig sein, zwischen einem *System* und einem *Attribut* zu unterscheiden. Die beide sind eine semantische Rolle mit gleichem Wortart-Markierung und Abhängigkeit-Markierung (NOUN sb).

Eine mögliche Lösung zu der Voraussetzung 4.13 ist durch der Nutzung von einer Wortliste oder ein Datenbank. Die Liste sollte die Wörter enthalten, die die Kandidaten für die semantischen Rolle „Attribut“ wie das Wort „Farbe“, „Geschwindigkeit“, und „Beschleunigung“ sind. In der Tabelle 2.1 aus Grundlagen Kapitel, werden Informationen zu den semantischen Rollen gegeben.

4.2.3 Andere Problemen einer Extraktion eines Textes

Das Lernspektrum im Kontext der natürlichen Sprache ist zweifellos riesig, deshalb gibt es auch viele Probleme, die beim Extrahieren eines Freitextes mit Hilfe einem NLP-Werkzeug entstehen können. Genauso wie im vorherigen Abschnitt werden weitere Beispielsätze untersucht, die zu verschiedenen Problemen oder Aspekte des Extraktionsprozesses führen.

Aufzählung

Beispielsatz: Aufzählung Problem
Die Tür muss aus zwei Scharniere [,] sechs Schrauben [,] und einen Griff bestehen

Das Beispiel ist eine selbsttätige Systemaktivität funktionale Anforderung mit 3 Objekten. Es geht hier um eine Aufzählungsproblem, wobei dieser Satz mehrere Objekte hat.

Voraussetzung 4.14. Das System sollte in der Lage sein, einen komplexen Satz in einen relativ einfacheren Satz umzuwandeln.

Es gibt zwei Lösung zur 4.14: Entweder mache die Struktur des Satzschablone flexibler oder zerteilen die Aufzählung zu mehreren Anforderung. Was damit gemeint mit flexibler zu dem obigen Beispielsatz ist wie folgt:

Oberklasse: Funktionale Anforderung Schablone Name : Selbsttätige Systemaktivität Funktionale Anforderung mit mehreren Objekte <SYSTEM> <PRIO> LIST<OBJEKTE> <PROZESS WORD>

Als Beispiel kann das System dem Benutzer die Möglichkeit bieten, eine neue Arten von Satzschablone zu definieren, bevor die Anforderung gespeichert werden. Die Oberklasse der neuen Satzschablone sollte automatisch definiert, basierend auf eine Satzschablone-Erkennungsalgorithmen, aber falls es noch nicht bekannt ist, fragt das System den Benutzer nach dem Name der Oberklasse. Anderer Fall ist, wenn der Struktur der Satzschablone gleich bleibt. Dann wird die semantische Rolle „<OBJEKT>“ alle Objekte der Anforderung beinhalten, wobei die Regel des Extraktionsprozess wie benötigt geändert werden

sollte. Außerdem kann das gleiche Verfahren von den [Nik19] verwendet werden. Dieses Verfahren wird den Beispielsatz zu 3 funktionale Anforderungen mit jeweils eine Objekte zerteilt.

Die Tür muss aus zwei Schaniere bestehen. Die Tür muss aus sechs Schrauben bestehen. Die Tür muss aus einen Griff bestehen.

Die Zusammenhang zwischen die zerteilte Anforderungen können danach zum Beispiel sortiert nach der Name des Systems oder des Prozesswortes (siehe Bemerkung 4.2.1).

Adjektivdeklination

Beispielsatz: Adjektivdeklination Problem

Das gelbe Auto sollte fahren können.

Dieses Problem ist ein bisschen knifflig. In natürliche Sprache es ist üblich eine Adjektivdeklination zu verwenden um die Eingenschaft des Objekts zu definieren, welche wichtig ist für eine nicht-funktionale Anforderung.

Voraussetzung 4.15. Das System sollte fähig sein, Adjektivdeklination Problem zu behandeln

Als eine der Lösung zu diesem Problem 4.15: Von der Beispielanforderung wird eine neue nicht-funktionale Anforderung erzeugt. Die Adjektivdeklination der ersten Anforderung wird dann als Inhalt des Attributswerts in der neuen erzeugten nicht-funktionale Anforderung betrachtet.

Falls eine Ontologie-basierte Datenbank schon existiert, kann eine Funktion entworfen werden, die die semantische Rolle „Attribut“ automatisch von dem Inhalt des Attirbutswerts generiert. Der Beispielsatz wird anhand der Adjektivdeklination „gelbe“ und die Datenbank das Wort „Farbe“ automatisch erzeugt.

Die <Farbe ?> des Autos sollte gleich gelb sein

Schwache Worte

Beispielsatz: Schwache Worte

Es sollte fähig sein, eine E-mail zu schreiben. Das System muss robust sein.

Schwache Worte (siehe Abschnitt sec:weakWord) sind im Grunde ein Pronomen oder ein Adjektiv, die unklar oder nicht eindeutig sind. Der Erkennungsprozess von schwachen Wörtern kann z.B. mithilfe von einer vordefinierten Wortliste und von Regeln, die Pronomen extrahieren, erreicht werden. Nach der Erkennung kann Hilfestellungen angeboten werden, z.B. eine Pop-Up Frage, um die Unklarheit zu beheben.

Voraussetzung 4.16. Das System sollte fähig sein, Schwache Worte (engl. Weak Words) anhand einer vordefinierten Wortliste zu detektieren.

Sonderzeichnen und Abkürzung

Beispielsatz: Sonderzeichnen und Abkürzung Problem

Das Produkt bzw. das Essen von A&W sollte 30% billiger als KFC.

Voraussetzung 4.17. Das System sollte fähig sein, Sonderzeichen und Abkürzungen zu benutzen oder zu einem anderen Form umzuformen.

Das Problem 4.17 kann man lösen, falls es eine Regel existiert, die das Sonderzeichen und die Abhängigkeit zwischen dem Sonderzeichen und anderen Wörtern erkennt, die nicht so einfach sind. Als Beispiel betrachtet man das Wort „A& W“ und die „zwei Schaniere & sechs Schrauben“. Die beiden Worte haben das Sonderzeichen „&“ aber eine ist nur ein Objekt und die andere sind zwei Objekte.

Das spaCy gibt uns die Möglichkeit, ein solches Ereignis zu trainieren. Zuerst wird ein Modell von spaCy geladen. Dies hat schon Wortart-Markierungen und andere Markierungen, die bereits auf Basis des TIGER-Korpus trainiert sind. Dieses Modell kann mit neuen Trainingsdaten trainiert werden. Die Struktur der Beispiel-Trainingsdaten besteht aus einem Text und der Position des zu trainierenden Wortes.

```
trainData = [
(„Das A& W ist eine Firma“, [(4, 6, ’ORG’)]),
(„Die C& A wird eine neue Kleidungen morgen verkaufen“, [(4, 6, ’ORG’)]),
(„Die Tür bestehen aus zwei Schaniere & sechs Schrauben“, [(21, 34, ’BESTANDTEIL’),
(38, 51, ’BESTANDTEIL’) ])],
```

In spaCy, „ORG“ ist eine Wortart-Markierung für eine Organisation. Aber man kann auch eigene Wortart-Markierung definieren, wie z.B. „BESTANDTEIL“, die aus einem Anzahl und Objekt (Wortart) besteht. Die Zusammenhang zwischen der Anzahl und ihrem Objekt kann dann als eine semantische Rolle „OBJEKT“ oder „Bezugsobjekt“ extrahiert wird.

5 Entwurf

Das Ziel dieser Arbeit ist es, eine einfache Prototyp-Anwendung zu erstellen, die einen Freitext analysiert oder extrahiert. Die extrahierten Ergebnisse werden dann in einer der Schablonen klassifiziert. Im Rahmen dieser Arbeit (siehe Abbildung 5.1 für das Konzeptschema) sollen im ersten Schritt Anforderungsdokumente aus der Praxis (unter anderem aus dem Projekt-Konsortium DAM4KMU) gesichtet und analysiert werden. Anhand dieser Daten, sollen generische Anwendungsfälle abgeleitet werden, welche als Grundlage der Entwicklung der digitalen Assistenzmethoden dienen.

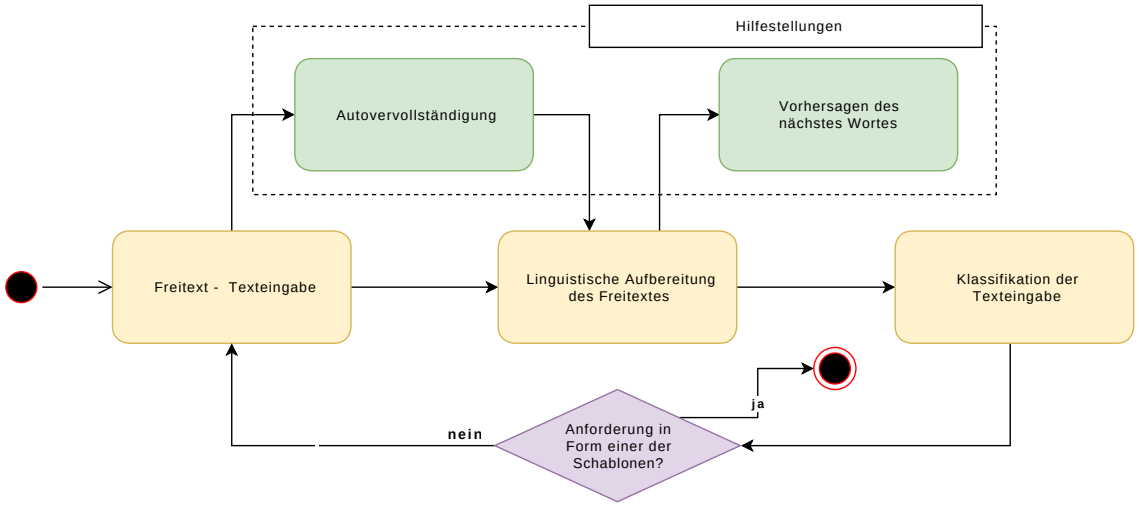


Abbildung 5.1: Das Konzeptschema dieser Arbeit [Fri19]

Generell wird hierbei das Konzept verfolgt, Anforderungsautoren mit Hilfe von Algorithmen zur Autovervollständigung mit möglichen Vervollständigungs-Möglichkeiten zu unterstützen. Dies soll es ermöglichen, schon während der Eingabe einer Anforderung, die damit in Verbindung stehenden Objekte zu verknüpfen (siehe Abbildung 5.2).

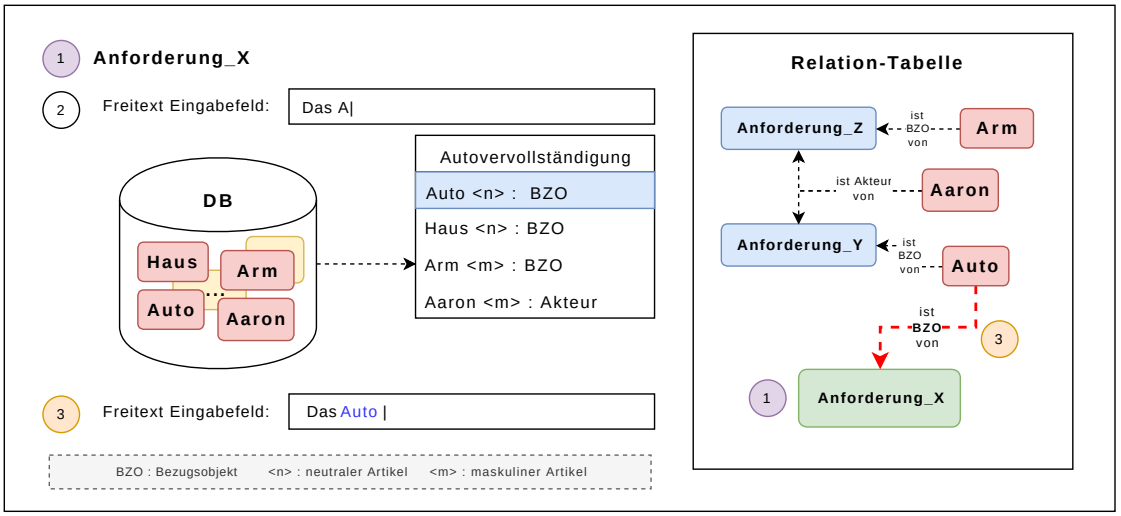


Abbildung 5.2: Beispiel Verknüpfung, die nach der Eingabe eines Freitext erzeugt wird.

In einem zweiten Schritt soll der bereits erzeugte Text mit den in der Literatur definierten Anforderungsschablonen abgeglichen werden und dem Anwender die am besten zutreffendste Schablone angeboten werden. Sollte der Anwender eine der angezeigten Schablonen wählen, wird der bisherige Text, automatisch in die gewählte Schablone überführt. Um dies zu gewährleisten, sollen die bereits eingegebenen Worte zur Laufzeit hinsichtlich ihrer Wortart und ihrer Funktion innerhalb des Satzes analysiert werden. Basierend auf den zum Einsatz kommenden Satzschablonen, soll hierbei z.B. unter folgenden semantischen Rollen unterschieden werden:

- System
- Priorität
- Objekt
- Prozesswort
- Akteur
- Attribut
- Attributswert
- Konditionswort
- Vergleichsoperator
- Funktionswort

Bei der Erkennung von System oder Bezugsobjekt, Priorität und Prozesswort, kann im Rahmen dieser Arbeit auf die Masterarbeit von “Automatisierte Extraktion sicherheitsrelevanter Anforderungen aus juristischen und technischen Dokumenten” [Sri19] angeknüpft werden. Frau Srikanthan hat in ihrer Arbeit ein Verfahren entwickelt, welches es ermöglicht Anforderungen mit und ohne Bedingung aus Freitext zu extrahieren. Anschließend werden mit Hilfe von Verarbeitung von natürlicher Sprache (engl. NLP) Methoden das System, die Priorität und das Prozesswort extrahiert. Dieses Verfahren soll im Rahmen dieser Arbeit um die oben genannten Begriffe erweitert werden.

Basierend auf der Extraktion dieser semantischen Rollen, soll zum einen die Datenbank hinsichtlich bereits bekannter Objekte durchsucht werden. Dies soll verhindern, dass Objekte mehrmals instanziiert, oder jeweils in unterschiedlicher Form genannt werden. Hierbei

Name des Nomen	Abhängigkeitsmarkierung
Schuhe	sb (Subjekt)
Benutzer	da (Dativ Objekt)
Möglichkeit	oa (Akkusativ Objekt)
Berge	oa (Akkusativ Objekt)

Tabelle 5.1: Liste von Nomen aus der Abbildung 5.3

soll im Rahmen dieser Arbeit die Anwendung heuristischer als auch lernbasierter Verfahren untersucht werden. Diese Schritte aus dem Konzeptschema werden dann in diesem Kapitel in einem separaten Abschnitt näher erläutert.

5.1 Linguistische Aufbereitung von Freitext

Wie in Kapitel 4 und Kapitel 2 erklärt, Werkzeuge der Verarbeitung von natürlicher Sprache (NLP) können Texte oder Sätze in Tokens extrahieren. Hierbei werden auch Markierungen wie Wortart-Markierung und Abhängigkeitsmarkierung zu jedem tokenisierten Wort gegeben. Die Frage ist, ob es bereits ausreicht, nur die tokenisierte Wörter zu verwenden, um die Freitext Eingabe in einer standardisierte Anforderungsschablone umzuwandeln. Um die zu antworten, betrachtet man jetzt diese Abbildung 5.3.

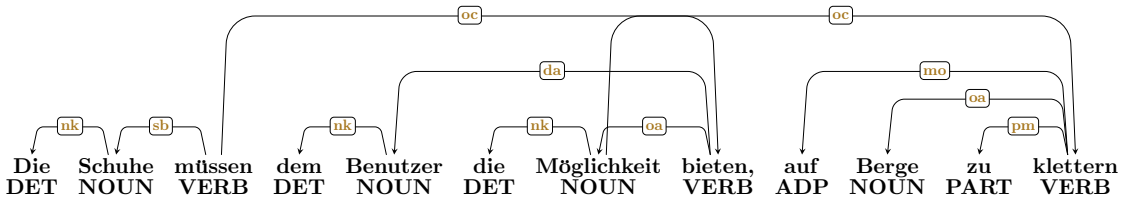


Abbildung 5.3: Beispiel für Tokenisierung eines Textes in Wortart-Markierung und Abhängigkeitsanalyse aus spaCy.

Das obige Beispiel ist eigentlich ein Standardbeispiel für funktionale Anforderung nach Chris Rupp [Rup14] oder nach SOPHIST. Aber nur mit dem Wortart-Markierung von dem spaCy selbst, können die Wörter aus dem Text nicht als die semantische Rolle von der Schablone (siehe Tabelle 2.1) umgewandelt werden. Der Grund dafür ist, dass Wortart-Markierung wie zum Beispiel die **DET + NOUN** in der Abbildung 5.3 die Unterschiede zwischen einem System, einem Objekt oder einem Akteur nicht erkennen werden können.

Semantische-Rolle-Erkenner

Zu diesem Problem werden Regeln zur Angabe der Unterschiede zwischen dem erstellt. Die Idee ist, die Ergebnisse der Abhängigkeitsanalyse und Wortart-Markierung zu kombinieren. Zurück zu der Abbildung 5.3, betrachtet man die Wörter „Schuhe“, „Benutzer“, „Möglichkeit“ und „Berge“, die jeweils das **„NOUN“** als Wortart-Markierung haben.

In Tabelle 5.1 sieht man, dass jedes Wort unterschiedliche Abhängigkeitsmarkierung hat. Diese Markierung wird als die nächste Filterschritt verwendet. Es wird wie folgt definiert, dass die Wörter, die „NOUN“ als Wortart-Markierung und „sb“ (Subjekt) als Abhängigkeitsmarkierung haben, die semantische Rolle „System“ sind. In dem Fall der Abhängigkeitsmarkierung „da“ (Dativ Objekt) ist, dann wird das Wort als der „Akteur“ von der Schablone betrachtet und wenn es um ein „oa“ (Akkusativ Objekt) handelt, ist das Wort die semantische Rolle „Objekt“.

Das Wort „Möglichkeit“ ist besonders, es wird nicht als ein Objekt der Schablone betrachtet, da es ein wichtiges Stichwort von der Schablone-arten nach Chris Rupp. Der gesamte Ausdruck „die Möglichkeit bieten“ ist eine „funktionales Wort“, die der Benutzerinteraktion des Systems repräsentiert. Um den Ausdruck zu extrahieren, sollte der Semantische-Rollen-Erkenner beispielsweise eine Regel haben, wenn er das Wort „Möglichkeit“ erkennt, sucht der Erkenner, ob das Wort mit dem Verb „bieten“ abhängig ist. Wenn dies der Fall ist, sollten die beide Worte zusammen mit dem Substantivkernelement (nk) „die“ als semantische Rolle „Funktionswort“ gespeichert werden.

Nicht alle semantische Rolle aus der Schablone können mit der Kombinationen von Wortart-Markierung und Abhängigkeitsmarkierung abgedeckt werden. Zu dem betrachten wir jetzt die nicht-funktionale Anforderung Schablone und die Abbildung 5.4.

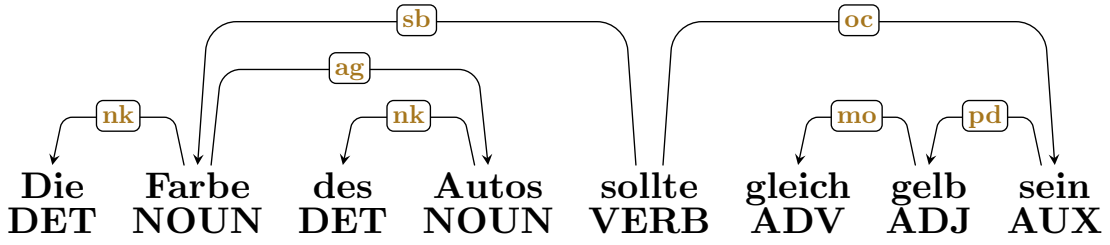


Abbildung 5.4: Beispielsatz für Nicht-funktionale Anforderung mit ihrer Tokenisierung

Das Wort „Farbe“ im Beispiel 5.4 hat die gleiche Wortart-Markierung und Abhängigkeitsmarkierung wie das Wort „Schuhe“ im Beispiel 5.3, jedoch aus der Perspektive der Satzschablone die semantischen rollen von beider Beispiele sind unterschiedlich. Die semantische Rolle des Wortes „Farbe“ ist ein *Attribut* und das anderes Wort „Schuhe“ ist ein *Bezugsobjekt* (siehe Voraussetzung 4.13). Deswegen sollte man eine anderen Weg finden, um die semantische Rolle „Attribute“ und „System“ zu unterscheiden. Wie bei der Kapitel 4 kann zum Beispiel Ontologie-basierte Datenbank verwendet werden, um die Beziehung zwischen geeigneten Adjektive und die semantische Rolle „Attribute“ zu bestimmen.

Andere Weg kann man eine bestimmte Wortliste verwenden, wobei die Wörter der Liste als ein Attribut angesehen werden können. Die Kandidaten für diese Wortliste ist ein Substantiv oder Nomen, das ein anderes Nomen beschreiben oder modifizieren kann. Zudem sind die *nominalisierte Worte* ein guter Kandidat. Ein nominalisiertes Wort ist ein Wort, das durch Nominalisierung eines Verbs oder eines Adjektivs erhalten wird. Wie in Abschnitt 2.2.1.2 bereits erklärt, Verben beschreiben eine Handlung oder ein Ereignis in syntaktische Aspekt eines Texts. Es folgt dann, dass ein nominalisierte Wort von einem Verb kann auch ein Substantiv modifizieren, wegen seiner Herkunft als ein Handlung. Neben der nominalisierten Worte sind Worte mit Endungen *-heit, -keit, -tät,* und *-zahl* als ein Attribut auch geeignet (siehe Tabelle 5.5).

Wortendung	**Beispielsatz**
-heit	Die **Gesundheit** des Patient
-keit	Die **Geschwindigkeit** des Autos
-tät	Die **Qualität** des Systems
-zahl	Die **Postleitzahl** der Stadt
Nominalisiertes Wort	Der **Druck** der Maschine

Abbildung 5.5: Beispiel Regel für Bestimmung eines Attributs 5.3

Nachdem eine Wortliste definiert wurde, wenn der Semantische-Rollen-Erkenner ein Ob-

jekt erkannt, wird ein Abgleich zwischen die lemmatisierte Eingabe und die Wortliste ausgeführt. Falls die Eingabe in die Liste vorkommt, dann die semantische Rolle ist ein Attribut, sonst wird es als entweder ein System, Akteur oder Objekt betrachtet. Die gesamte Liste von semantischen Rollen mit ihrem Beispiel und entsprechenden Wortart werden in die Tabelle 5.2 dargestellt.

Semantische Rolle	**Wortart aus der Linguistik**	**Beispielwort**
System	Subjekt	Auto, Tür
Attribute	Subjekt	Geschwindigkeit, Farbe
Objekt	Akkusative Objekt	Auto, Tür
Akteur	Dativ Objekt	Admin, Benutzer
Priorität	Modalverb	muss, sollte, wird
Funktion Wörter	festes Wort	fähig sein, die Möglichkeit bieten
Vergleichsoperator	Adverb	gleich, ungleich, mindestens
Attributswert	Adjektiv	gelb, blau, stark, schnell
Prozessword	Verb	fahren, geben, bekommen
Konditionswort	Konjunktiv, Adverb	Falls, Sobald, Solange

Tabelle 5.2: Semantische Rolle mit ihrem Vergleich zu dem Wortart aus der Linguistik

5.2 Klassifikation der Texteingabe

In diesem Abschnitt wird ein Konzept zur Klassifizierung eines freien Texts in eine Satzschablone erläutert. In Kapitel 2 wurde es definiert, dass es verschiedene Methoden des maschinellen Lernens gibt, um einen Text oder eine Eingabe zu klassifizieren, wie die Rekurrentes neuronale Netze (RNN), Long Short Term Memory (LSTM), Hidden Markov Model (HMM) und Stützvektormaschine (SVM). Wegen wenigen Trainingsdaten werden LSTM bzw. RNN in dieser Arbeit nicht benutzt.

Das entschiedene Verfahren, das für die Klassifikation verwendet wird, ist das SVM-Verfahren. Gleich wie bei der anderen Klassifikation Verfahren braucht SVM Trainingsdaten und Kategorien. In diesem Fall werden Satzschablone als Kategorien der SVM und die extrahierte semantische Rollen als die eingegebene Trainingsdaten. Warum nicht die Texte selbst als die Eingabe? Es gibt zwei Begründung dafür. Erster Grund, eine Satzschablone der SOPHIST besteht aus ihren einzigartigen Wortarten, welche die in dieser Arbeit als semantische Rollen genannt werden. Der andere Grund ist, durch die Verwendung der semantischen Rollen als Eingabe sind die Anzahl der Kategorien der SVM niedriger als durch einzelne eindeutige Wörter. Je niedriger die Kategorien sind, desto weniger Trainingsdaten werden benötigt, was bei dieser Arbeit der Fall ist.

Die semantische Rollen werden als nächstes in Form von TF-IDF (term frequency-inverse document frequency) umgewandelt. Die TF-IDF wird benutzt, um die Wichtigkeit einer semantische Rolle des aktuellen Text im Vergleich zu anderen semantischen Rollen zu bestimmen. Der TF-IDF-Wert einer semantischen Rolle erhöht sich proportional zu der Häufigkeit, mit der die semantische Rolle im Text vorkommt. Die Werte werden in Form eines Vektors gespeichert und für die SVM trainiert. Da die Trainingsdaten aus Vektoren von Häufigkeiten der semantischen Rollen sind, ermöglicht diese eine Klassifikation der Texteingabe in eine Satzschablone, ohne die Reihenfolge von den extrahierten semantischen Rollen zu berücksichtigen.

Ein weiterer Algorithmus für maschinelles Lernen bei geringen Datenmengen ist der HMM (Hidden Markov Model). Im Vergleich zu SVM werden die initialen Trainingsdaten als Wahrscheinlichkeit Matrizen ersetzen (siehe Kapitel 2. Die semantischen Rollen sind die

Zustände des HMM und die Satzschablonen sind die beobachteten Symbolen. Es gibt also insgesamt drei Matrizen, die wichtig für HMM sind, und zwar: die Matrize der Anfangswahrscheinlichkeit, die Matrize der Übergangswahrscheinlichkeit, und die Matrize der Ausgabewahrscheinlichkeit. Die Anfangswahrscheinlichkeit bestimmt die semantischen Rollen initialer Wahrscheinlichkeit und die Übergange zwischen den semantischen Rollen wird von die Übergangswahrscheinlichkeit gesteuert. Die Ausgabewahrscheinlichkeit kontrolliert dann welche Schablone als Ausgabe kommt. Die Position zwischen der semantischen Rolle und der Schablone kann ausgetauscht werden, sodass z.B. die nächste semantischen Rolle vorhergesagt werden können.

Schablone-Ranking

Nur eine einzelne Satzschablone zu repräsentieren ist anscheinend nicht so innovativ, aus diesem Grund kann ein *Satzschablone-Ranking* dargestellt werden, indem der prozentuale Ansicht jeder Schablone berechnet wird. Die Idee ist, dass der Benutzer eine andere mögliche ähnliche Schablone neben der besten überlegen kann.

5.3 Unterstützung durch Autovervollständigung und andere Hilfestellungen

In diesem Abschnitt werden Hilfestellungen der Dokumentationsassistenten beschrieben. Als erstes wird ein Entwurf für eine ideale oder gute Autovervollständigung überlegt. Danach wird eine Hilfestellung zur Vorhersage des nächstes Wortes besprochen, die von der Autovervollständigung verwendet wird. Zum Schluss wird das Konzept zur automatischen Umformung in eine Satzschablone entwickelt.

Autovervollständigung

Stellen Sie sich vor, dass man während der Anforderungsdokumentation Ablauf die komplette Sätze selbst schreiben muss. Dies wird zu einem Problem führt, wenn dem Schreiber nicht weißt, was er alles nennen muss um einen standardisierten und vollständigen Anforderung zu formulieren. Daneben besteht auch die Möglichkeit eines Tippfehlers während des Schreibens. Um diese Problem zu lösen oder zu vermeiden gibt es verschiedene Lösungen wie zum Beispiel durch eine Autovervollständigungsfunktion oder durch die Verwendung einer Anwendung zur Erkennung (bzw. Autokorrektur) von Rechtschreibfehlern (z.B. für Grammarly, LanguageTool, Reverso usw.).

Autovervollständigung ist eine Hilfestellung, die viele Vorteile bietet. Bei ordnungsgemäßer Implementierung können Suchvorschläge beispielsweise die Produktentdeckung anregen, den Umfang Ihrer Benutzer nach Kategorien eingrenzen oder Rechtschreibfehler korrigieren. Was brauchen wir im Allgemeinen, um eine grundlegende Autovervollständigung durchzuführen? Normalweise eine Autovervollständigung besteht es aus (Präfix-) Eingabe und eine Liste möglicher Ausgaben. Die Präfix-Eingabe wird als Einschränkung verwendet, die die Ausgabedaten aus der Liste der möglichen Ausgaben filtert. Andere Einschränkungen wie z.B. die Zeit oder die Häufigkeit der Verwendung des Vorschlags können hinzugefügt werden, um die automatische Vervollständigung präziser und interaktiver zu gestalten.

Von letztem Abschnitt können semantischen Rollen von einem Text abgeleitet werden. Die Idee für die Autovervollständigung dieser Arbeit ist die Verwendung von den semantischen Rollen als die andere Einschränkung. Was aber für die Vervollständigungsvorschläge benötigt wird, ergibt sich nicht aus den aktuellen semantischen Rollen, sondern aus den semantischen Rollen des nächsten Wortes. Deswegen wird ein Verfahren benötigt, um das nächste Wort oder nächste semantische Rolle unter Verwendung des vorherigen Wort zu vorhersagen (siehe nächster Abschnitt).

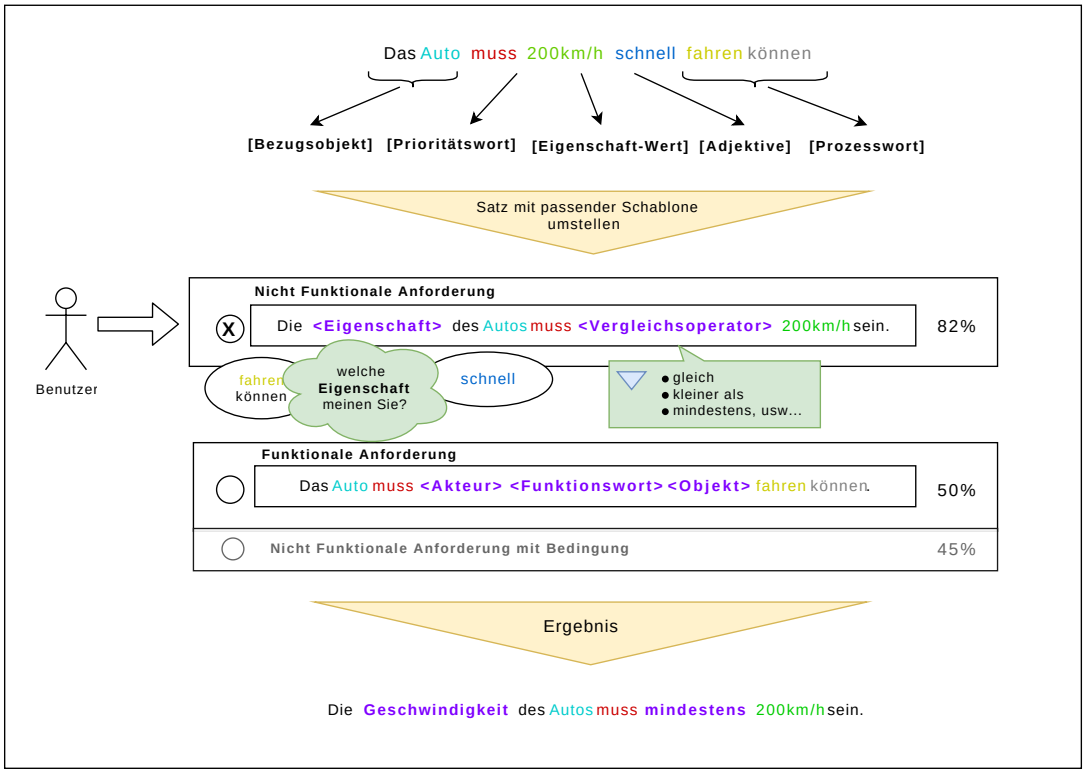


Abbildung 5.6: Beispiel Ablauf von einer Satzumstellung durch die Anpassung einer passenden Satzschablone.

Vorhersage des nächstes Wortes

Nach der Eingabe eines Wortes, sollte das nächste Wort berechnet werden. Dieses Problem 4.16 ist auch ein Art von Textklassifikation oder Mustererkennung Problem. Wie bereits erwähnt bei der letzten Abschnitt 5.2, es gibt mehrere Arten von Verfahren, die für die Textmusterkennung geeignet sind. Aber für das nächstes Wort wird in dieser Arbeit ein einfache Markovketten verwendet. Die extrahierte semantischen Rollen aus der Semantische-Rollen-Erkenner werden als Eingabe benutzt.

In der Graphentheorie der Markovketten sind die semantischen Rollen hier die Knoten und für die Kanten wird sie durch die Übergangswahrscheinlichkeit von einer semantischen Rolle zur anderen semantischen Rollen dargestellt. Die Wahrscheinlichkeit eines Übergangs zwischen der anfänglichen und der nächsten semantischen Rolle kann bestimmt werden, indem man zuerst die Anzahl des Vorkommens des Übergangs zwischen der ersten und der nächsten Rolle berechnet und dann dividiert man die durch alle möglichen Übergänge der Startknoten.

Automatische Umformung in eine Satzschablone

Nach der vollständigen Eingabe eines Wortes, soll anschließend der bisher eingegebene Text, dessen Worte und bzw. deren semantischen Rollen mit den in der Literatur beschriebenen Anforderungsschablonen abgeglichen werden. Je nach Häufigkeit zutreffender semantischen Rollen innerhalb einer Satzschablone, wird die jeweilige Anforderungsschablone im Ranking höher oder niedriger bewertet (siehe Abbildung 5.6).

Die vorhandene semantischen Rollen werden in den Satzschablonen angepasst. Fehlende semantische Rolle wird z.B. mit Pop-up Frage nachgefragt. In der Abbildung fehlt der Beispielsatz noch die semantische Rolle „Attribut oder Eigenschaft“. Eine Benachrichtigung wird dann gezeigt, z.B. „Welche / was für eine Eigenschaft meinen Sie?“.

6 Implementierung

In diesem Kapitel wird die Umsetzung der in Kapitel 5 beschriebene Analysen und ihrer Teilschritte basierend auf die Abbildung 5.1 erläutert. Zur Unterstützung der Benutzer bei der Anforderungsdokumentation wird in dieser Arbeit Semantische-Rollen-Erkenner im Bezug der linguistischen Aufbereitung, SVM-Modell zum Klassifikation der Texteingabe, und Markovketten-Modell für die Vorhersage des nächstes Wortes entwickelt und umgesetzt. Es wird eine einfache prototypische Anwendung erstellt, um diese Unterstützungen zu demonstrieren. Als Framework für die Anwendung wird das Django-Webframework [1] verwendet. Mit Django kann eine Python-Webanwendung mit integriertem und einfach zu modifizierendem Datenbankmodell erstellt werden. Das von Django bereitgestellte Datenbankmodell kann zum Speichern neuer Anforderungen oder anderer Modelle verwendet werden. Diese gespeicherten Daten werden später in dem Kapitel 7 dieser Arbeit ausgewertet und verglichen. Nachfolgend wird der funktionale Aufbau des vorliegenden Konzepts in Abbildung 6.1 dargestellt.

In 6.1 sieht man, dass der Ablauf des Konzepts in sechs Schritte unterteilt. Als erstes werden das SVM-Modell und Markovketten-Modell mit Trainingsdaten in Form von CSV-Daten trainiert. Danach kommt die Benutzereingabe von der Benutzeroberfläche. Die Benutzeroberfläche und die Funktionen dazwischen werden mit HTML, Javascript, und CSS implementiert, die von Django als statische Elemente des Projekts verwaltet werden können. Die Benutzereingabe muss an den Server weitergegeben werden, daher wird AJAX (Asynchronous Javascript and XML) [2] verwendet, um die Dateieinaustausch zwischen der Webseite und dem Server (Python-Dateien) zu realisieren. Die eingehenden Daten werden vom Semantische-Rollen-Erkenner mit Hilfe von spaCy und selbst definierten Regeln in gereihten semantische Rollen extrahiert. Diese semantische Rolle werden in das SVM-Modell und das Markovketten-Modell verwendet, um die eingegebene Texte zu klassifizieren und ein nächstes Wort vorherzusagen. Letzter Schritt werden die Ergebnisse von den beiden Modelle zurück an das Benutzeroberfläche gegeben. Falls eine nächste Eingabe vom Benutzer kommt, beginnt sie erneut mit dem zweiten Schritt wieder. Ein weiterer persönlicher Grund, warum Django verwendet wird, liegt in meinen Hiwi-Tätigkeiten im FZI. Ich habe ein bisschen Erfahrung mit dem Django-Framework. Die Gliederung dieses Kapitels werden wie das Kapitel Kapitel 5 unterteilt, wobei wird die einzelne Umsetzung der Unterstützung in Details erklärt.

[1] Quelle: `https://www.djangoproject.com/`, zuletzt besucht am 13.10.2019.
[2] Quelle: `https://www.w3schools.com/js/js_ajax_intro.asp`, zuletzt besucht am 13.10.2019.

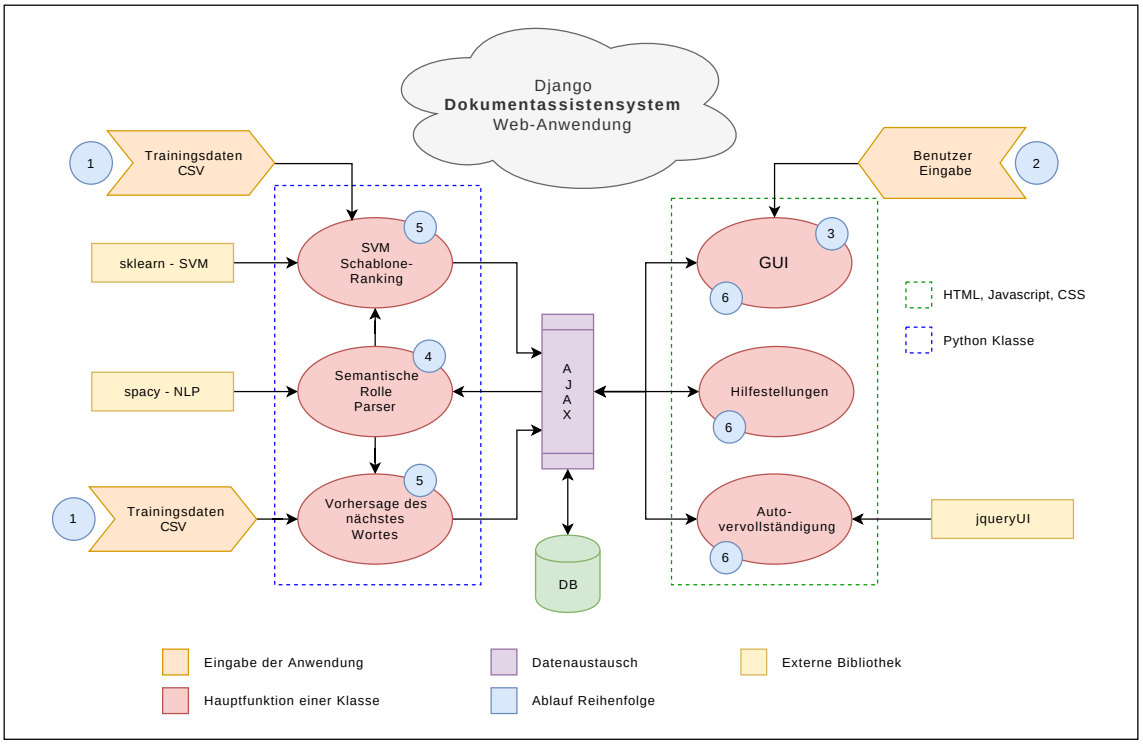


Abbildung 6.1: Grobes Diagramm des gesamten Dokumentationsassistent-Frameworks [Fri19]

6.1 Linguistische Aufbereitung von Freitext

Wie bereits erwähnt, werden freie Texte als die Eingabe der Anwendung benutzt. Diese Eingabe erfolgt durch Ausfüllen des Textfelds in der Benutzeroberfläche. Als die Verarbeitung von natürlicher Sprache wird spaCy version 2.1.4 verwendet. Mit spaCy wird die erhaltene Texte tokenisiert und mit Wortart- bzw. Abhängigkeitsmarkierungen markiert. Die Markierungen von spaCy verwendet das „de_core_news_md“ spaCy-Modell, das trainiert auf dem TIGER [3] und WikiNER Corpus. Danach wird die tokenisierte Daten in semantischer Rolle basierend auf die Satzschablone von SOPHIST extrahiert. Wie in Kapitel 4 und Kapitel 5 beschrieben, werden Regeln zur Erkennung der semantischen Rollen definiert. Diese Regeln sind die Kombinationen zwischen der Wortart-Markierung und Abhängigkeit-Markierung von spaCy (siehe Tabelle 6.1 und Abbildung 6.2).

Der besondere herausfordernde Fall besteht darin, die Regeln zur Bestimmung des Prozesswortes des Textes zu definieren. Zum Beispiel kann das Verb aufgrund der Struktur der deutschen Sprache in zwei Abschnitte unterteilt werden, nämlich das trennbare Verb ist. Andere Arten wie passive Verben, und der Unterschied zwischen Modalverben als Priorität und Modalverben als Prozesswort sollten auch berücksichtigt werden. Aufgrund des Komplexität der Arten der Prozessworte wird rekursive Abhängigkeitsanalyse verwendet. Dies ermöglicht z.B. im Fall das Prozesswort ein trennbares Verb ist, dann folgt, wenn einer der Teile des Verbs erkannt wird, werden die vorherige oder nächste Abhängigkeiten des Teils durchgesucht, bis der anderen Teil des Verbs gefunden ist.

Bemerkung: Der Semantische-Rollen-Erkenner wird nach jedem Wort mittels „Leerzeichnen“ ausgelöst.

[3] Quelle: `https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/TIGERCorpus/annotation/index.html`, zuletzt besucht am 13.10.2019.

Semantische Rolle	Abhängigkeitsmarkierung	Wortart-Markierung
System	sb, ROOT	NOUN, PROPN
Attribut	Wortliste und -heit, -tät, -keit, -zahl Wortendungen	NOUN, PROPN
Objekt	oa	NOUN, PROPN
Akteur	da	NOUN, PROPN
Priorität	ROOT	VMFIN, VMINF, VERB, ADJ
Funktionswort	Wortliste	AUX, ADJ, NOUN
Vergleichsoperator	mo	ADV
Attributswert	–	ADJ
Prozessword	–	alle VERB außer VMFIN und VMINF
Kondition Wörter	–	KOUS

Tabelle 6.1: Liste von semantischen Rollen mit ihren verwendeten Wortart- und Abhängigkeitsmarkierung

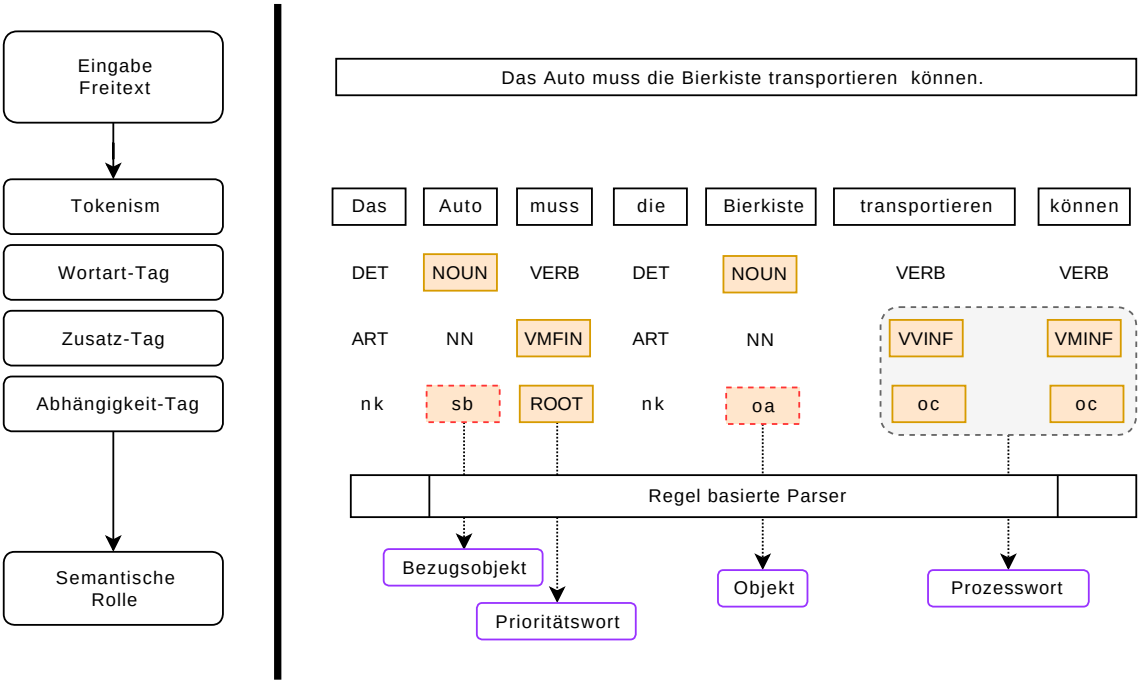


Abbildung 6.2: Beispielablauf von Semantische-Rollen-Erkenner

6.2 Klassifikation der Texteingabe

Einer der Hauptfunktion der Arbeit ist die Schablone-Klassifikation Algorithmen. Diese Klassifikation Algorithmen wird durch Verwendung der SVM von „sklearn“ realisiert. Das SVM Modell von sklearn, das benutzt wird, ist die *LinearSVC*. Weil es mehrere Klassen (mehr als zwei) klassifizieren kann, welche für diese Arbeit benötigt werden. Außerdem kann die LinearSVC im Vergleich zu anderen SVM Klassen in der „sklearn“ eine prozentuale Ansicht anzeigen, welcher die für die Darstellung der Satzschablone-Ranking benutzt wird. Für die Trainingsdaten des SVM Modells wird eine CSV (Comma-seperated values) Datei benutzt. Diese CSV Datei besteht aus zwei Spalten, gereihten semantischen Rollen und einer vordefinierten Satzschablone. Die Daten sind eine Kombination der möglichen Sequenzen von semantischen Rollen, die noch standardisierte Anforderungen nach SOPHIST sind (siehe Tabelle 6.2).

Satzschablone Art	Anzahl der Daten
Funktionale Anforderung	18
Nicht-funktionale Anforderung	4
Funktionale Anforderung mit Bedingung	180
Nicht-funktionale Anforderung mit Bedingung	40

Tabelle 6.2: Anzahl der Daten pro Satzschablone Klasse

Als nächstes werden Die Daten zu 80% als Trainingsdaten und zu 20% als Testdaten aufgeteilt. Die Treffgenauigkeit der Testdaten ist 0,746. Wegen der Ungleichmäßigkeit Anzahl werden die Klassen mit weniger Daten mit redundanten Daten ergänzt, bis sie ungefähr die maximale Anzahl (180 Daten) erreicht. Nachdem die Anzahl der Daten gleichmäßig ist, hat die Treffgenauigkeit der Testdaten von 0,965.

Wie bei der Kapitel 5 beschrieben, die semantische Rollen werden zuerst in der Form von TF-IDF (*Term frequency-Inverse document frequency*) umgewandelt. Die Häufigkeit der auftretenden semantischen Rollen innerhalb eines Satzes wird in Form von einem Vektor gezählt. Danach wird dieser Vektor in das SVM-Modell trainiert. Falls es sich um eine Benutzereingabe oder um Testdaten handelt, wird am Ende des Vorgangs die Eingabe in eine Satzschablone klassifiziert und eine Ausgabe aller Satzschablonen mit prozentualem Ansicht gegeben (siehe Abbildung 6.3).

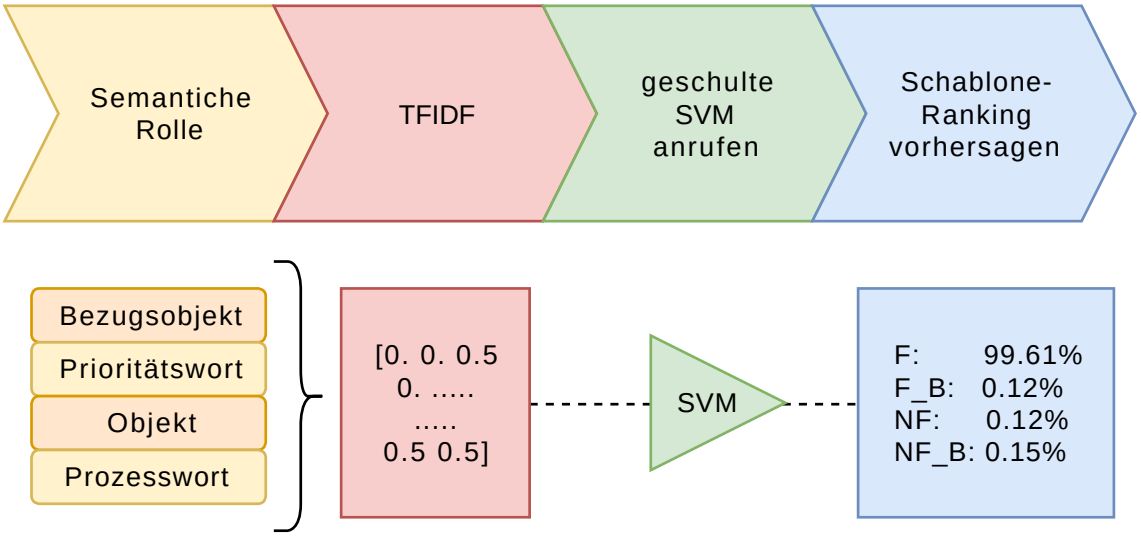


Abbildung 6.3: Ablaufschema und Beispiel der Klassifikation der Texteingabe mittels SVM

6.3 Implementierung möglicher Hilfestellungen

In diesem Abschnitt werden die Implementierung von möglichen Hilfestellungen beschrieben. Zuerst wird die Autovervollständigung zusammen mit der Vorhersage des nächstes Wortes dieser Arbeit besprochen. Als nächstes wird die automatische Umformung Hilfestellung diskutiert. Es wird erklärt, wie man von einer gereihten semantische Rollen zu einer Schablone durchgeführt.

6.3.1 Autovervollständigung und die Vorhersage des nächstes Wortes

Für Autovervollständigung Hilfestellung dieser Arbeit wird der Bibliothek von „Jquery-UI Autocomplete“ als Basismethode verwendet. Diese Funktion wird in Javascript-Datei implementiert. Die implementierte Funktion wird mit Hilfe von Jquery in einer HTML-Elemente (z.B. Inputsfelder) angehängt. Die Ansicht, Anzahl, und andere Aspekte der Vervollständigungsvorschläge kann man modifizieren, indem man die Parameter der vorhandenen Methoden des „Jquery-UI Autocomplete“ ändert. Für die Vervollständigungsdaten wird eine Wortliste verwendet, die aus Wörtern mit ihren semantischen Rolle besteht.

Der Kontext der Vervollständigung wird durch die Vorhersage des nächstes Wortes filtert. Um das nächste Wort zu bestimmen, wird mittels einfache Markovketten umgesetzt [Fri19] (siehe Abbildung 6.4 und 6.5).

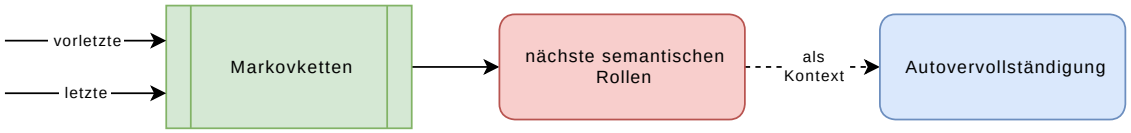


Abbildung 6.4: Ablaufschema von der Vorhersage des nächstes Wortes

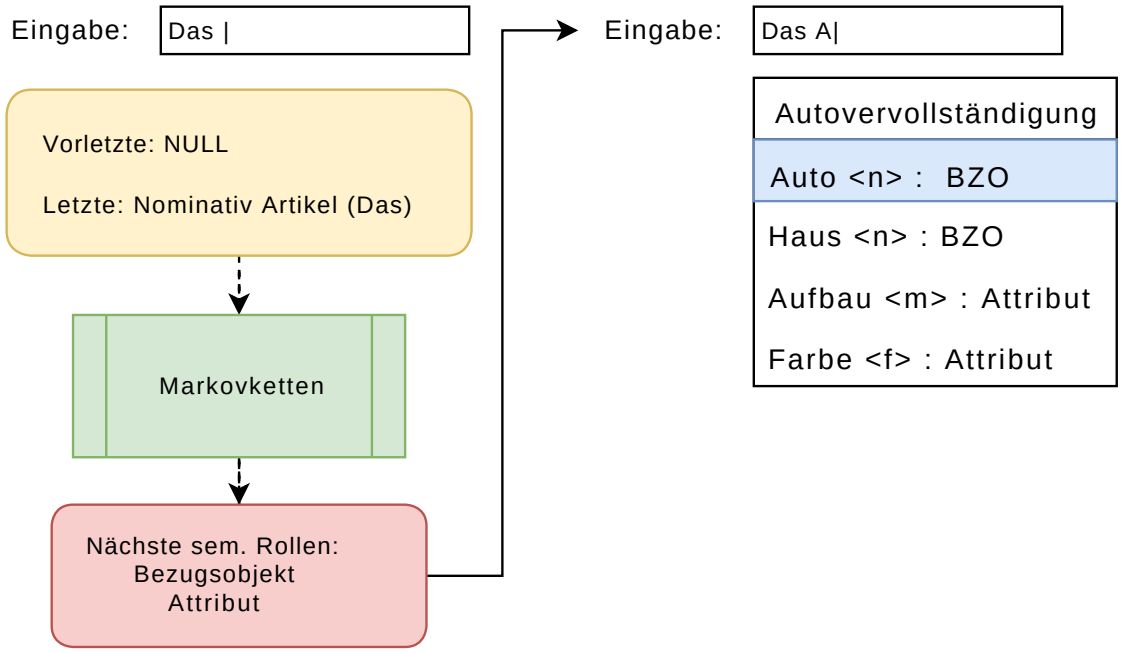


Abbildung 6.5: Beispiel von Zusammenhang zwischen die Vorhersage des nächstes Wortes und Autovervollständigung Funktion

Die Transition der Markovketten ist basiert nur auf das letzte und vorletzte Wort des eingegebenen Textes. Anhand der beiden Wörtern wird die nächste semantischen Rollen

Semantische Rolle	Abhängigkeitsmarkierung	Wortart-Markierung
Übergeordnetes System	(ag, pnc, nk) & sb	NOUN, PROPN
Übergeordnetes Objekt	(ag, pnc, nk) & oa	NOUN, PROPN
Übergeordnetes Akteur	(ag, pnc, nk) & da	NOUN, PROPN
Artikel Nominativ	ancestor = sb	DET
Artikel Akkusativ	ancestor = ak	DET
Artikel Dativ	ancestor = da	DET
Artikel Genitiv	ancestor = ag	DET
Präposition	–	ADP
Schwache Wort	–	PRON

Tabelle 6.3: Zusätzliche semantische Rolle für die Prognose des nächsten Wort

gesucht. Die Trainingsdaten für die Markov-Kette ist gleich wie bei SVM, aber es gibt zusätzlichen semantischen Rollen wie Artikel und übergeordnete Elemente (siehe Tabelle 6.3).

Bemerkung: Bei Bedarf ist es auch möglich, die Transition der Markov-Kette mit mehreren vorherigen Parametern zu implementieren. Falls zahlreiche Trainingsdaten verfügbar sind, kann z.B. LSTM verwendet werden.

Der Grund warum diese semantische Rollen werden zusätzlich hinzugefügt ist, um die Präzision des nächstes Wortes zu verbessern. Als Beispiel betrachtet man wieder die Abbildung 6.5. Der „Nominativ Artikel“ begrenzt die Ausgabe der nächsten semantischen Rolle in zwei Möglichkeit, und zwar das Bezugsobjekt und das Attribut. Der Artikel bestimmt was für ein Nomen-Typ als nächstes kommt.

6.3.2 Automatische Umformung eines Textes in eine Satzschablone

Abschließend wird die Hilfestellung für die automatische Umformung eines Textes in eine Satzschablone implementiert. In Kapitel 5 wird ein Ansatz der automatische Umformung definiert, indem die semantische Rollen oder Bestandteilen der Satzschablonen mit den angegebenen extrahierten semantischen Rollen gefüllt werden. Dieser Ansatz wird gleich wie beschrieben implementiert, aber anstatt der fehlenden semantischen Rollen zu zeigen, wird ein Beispielsatz für jede Schablone dargestellt (siehe Abbildung 6.6). Die Idee ist, dass der Benutzer ohne Vorkenntnisse von Wortarten ein Vergleichsobjekt oder Beispiel hat, um eine standardisierte Anforderung zu formulieren.

Am Anfang wird die Beispielsätze in grau gefärbt und durch jede erfolgreichen extrahierten semantische Rollen, wird an der passende Stelle die Texte der Beispielsätze ersetzt. Der Text wird dann mit anderer Farbe gefärbt und fett geschrieben. Um die Vollständigkeit einer Anforderung zu überprüfen, kann man z.B. bestimmen, wenn die Texte eines Beispielsatzes alles fett geschrieben sind.

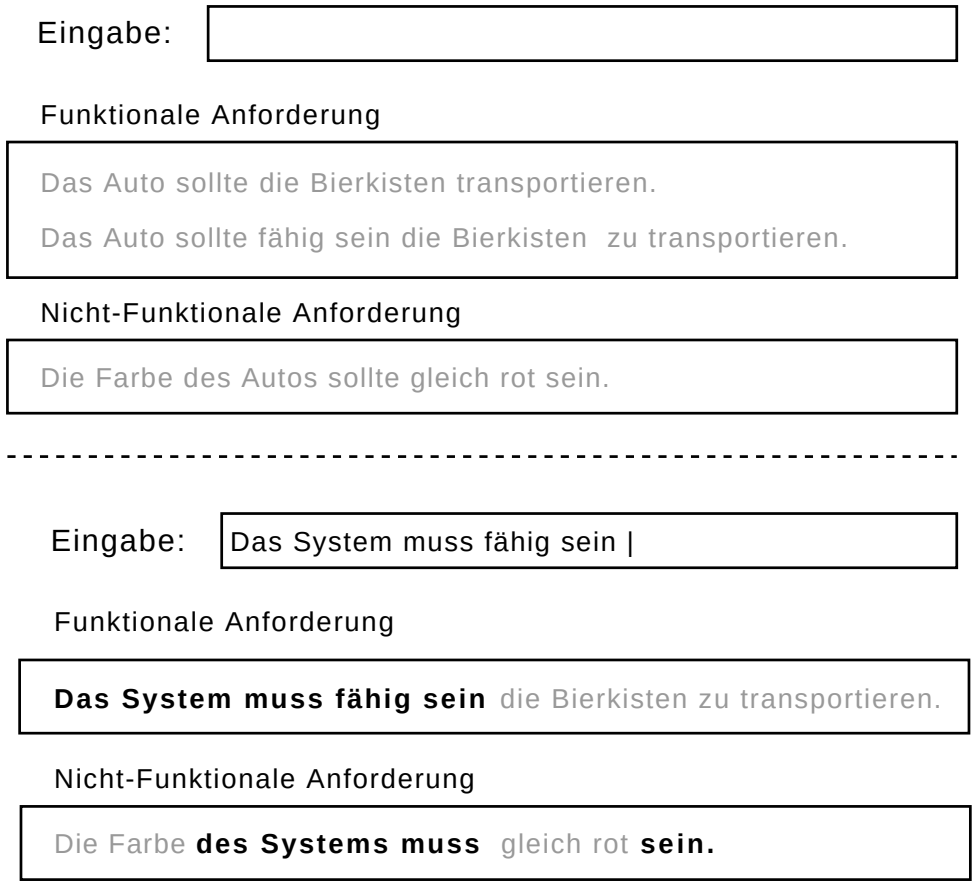


Abbildung 6.6: Umformungsprozess von freiem Text zu einer Schablone mittels Anforderungsbeispiele [Fri19].

7 Evaluation

Nachfolgend werden anhand des funktionalen Aufbaues der Konzeptes, die in Kapitel 5 abgeleiteten Voraussetzungen an das Konzept hinsichtlich ihrer Umsetzung evaluiert. Hierzu werden im ersten Schritt die linguistische Aufbereitung, Klassifikation der Texteingabe, und die Autovervollständigung untersucht. Die Ausbeute (engl. Recall) und Präzision (engl. Precision) der semantischen Rollen und das erstellende SVM-Modell zur Klassifikation der Satzschablone aus der Implementierung werden evaluiert, indem reale Anforderungen eines Partners des DAM4KMU Projekts verwendete werden. Im zweiten Schritt werden die Ergebnisse eines Evaluations-Workshops besprochen. Hier wird getestet und verglichen, wie sich die Ergebnisse zwischen Anforderungsschreibern ohne und mit Verwendung des Konzepts unterscheiden.

7.1 Linguistische Aufbereitung

Ziel dieser Arbeit für die linguistische Aufbereitung ist es, die semantischen Rollen aus einem Freitext zu bestimmen. In Kapitel 6 wurde die Umsetzung eines Semantische-Rollen-Erkenner beschrieben, der die Voraussetzung 4.4, 4.6, und 4.8 (siehe Tabelle .1 im Anhang) erfüllt. Um den Semantische-Rollen-Erkenner zu evaluieren, werden Anforderungen aus eines Partners des DAM4KMU Projekts und Quelldokumenten aus der Masterarbeit von Frau Srikathan [Sri19] benutzt. Der Grund, warum die beiden Quellen verwendet werden, besteht darin, die Robustheit oder die Funktionalität des Erkenners im Allgemeinen zu bewerten. Außerdem sind die Anforderungen des Partners strukturierte Anforderungen, die manuell basierend auf den SOPHIST Satzschablonen geschrieben wurden. Die Anzahl der Anforderungen des Partners sind 382 Anforderungen, die von denen 353 funktionalen, 19 funktionalen mit Bedingung, und 10 nicht-funktionalen sind. Zum anderen bieten die Quelldokumente aus der Masterarbeit von Frau Srikathan unstruktrurierte Anforderungen aus juristischen und technischen Dokumenten. Dies sind 306 gültige Anforderungen mit unterschiedlicher Positionierung der semantischen Rollen, die noch nicht in einer der Satzschablonen angeordnet sind. Weitere Anmerkung zu diesen beiden Quellen, dass die Daten aufgrund der Geheimhaltung nicht frei zugänglich sind.

Als nächster Schritt werden 100 Daten aus beiden Quellen ausgewählt. Die Daten bestehen aus 35 funktionalen, 22 nicht-funktionalen, 20 funktionalen mit Bedingung, und 19 nicht-funktionalen mit Bedingung Anforderungen. Für die Verteilung zwischen den Quellen sind 66 Daten die Anforderungen des Partners und die restlichen 34 Daten stammen

aus den Quelldokumenten der Masterarbeit. Danach werden diese Daten an den Dokumentationsassistenten (DA) übergeben und die semantischen Rollen automatisch extrahiert. Die Ergebnisse werden anschließend manuell mit den Musterlösungen verglichen und die Präzision und Ausbeute berechnet (siehe Abschnitt 2.4.1).

Semantische Rolle	**Präzision**	**Ausbeute**	**F1-Maß**
System	0,887	0,859	0,873
Priorität	0,935	1,00	0,966
Prozesswort	0,888	0,851	0,869
Objekt	0,971	0,705	0,817
Funktionswort	1,000	0,968	0,984
Akteur	0,928	1,000	0,962
Attribut	1,000	0,451	0,622
Attributswert	0,964	0,936	0,949
Konditionswort	1,000	0,842	0,914
Vergleichsoperator	0,391	0,409	0,400
Gesamtwert	0,896	0,802	0,835

Tabelle 7.1: Ergebnisse der Präzision und Ausbeute des semantischen Rolle Erkenner

In Tabelle 7.1 sieht man, dass die Präzision des Erkenners als gut angesehen werden kann, wobei fast alle semantischen Rollen über 0,89 liegen. Das „Funktionswort“ und „Attribut“ haben beide eine Präzision von 1, da ein vordefiniertes Wort oder eine Wortliste verwendet wird. Auf der anderen Seite wird der „Vergleichsoperator“ mit einer Präzision von 0,39 erkannt. Die aktuell definierten Regeln erkennen fälschlicher Weise auch Worte wie „wo“, „nur“ oder „noch“ als Vergleichsoperatoren, was zu vielen „false positive“ Ergebnissen führt.

Im Falle der Ausbeute, hat die semantische Rolle „Attribut“ im Vergleich zu seiner Präzision einen sehr niedrigen Wert von 0,45. Dies ist darauf zurück zu führen, dass für die Erkennung des „Attributs“ eine Wortliste und die Wortendungen „-heit“, „-keit“, „-tät“ und „-zahl“ verwendet werden, welche jedoch nicht alle Arten von „Attributen“ abdecken. Ein Beispiel hierfür stellt das Wort „Beschleunigung“ dar, welche nicht in der Wortliste enthalten war. Die Endung „-ung“ ist jedoch nicht als Regel definierbar, da es Worte gibt, welche trotz dieser Endung kein Attribut sind z.B. „Kleidung“. Dieses Problem kann jedoch durch eine Vergrößerung der Wortliste behoben werden. Insgesamt hat die Erkennung der semantischen Rollen ein F1-Maß von 0,835, was ausreichend genau ist, um die Klassifikation der Texteingabe und weitere Maßnahmen in dieser Arbeit durchzuführen.

7.2 Klassifikation der Texteingabe

Basierend auf den erkannten semantischen Rollen, wird der eingegebene Text hinsichtlich der vordefinierten Satzschablonen klassifiziert. Hierzu wird eine SVM-Modell aus Kapitel 6 verwendet, welche anhand der Häufigkeit der innerhalb eines Satzes auftretenden semantischen Rollen trainiert wurde. Für das Training wurden alle 385 Anforderungen des Partners und 104 Anforderungen aus der Quelldokumenten der Masterarbeit verwendet. Daraus ergeben sich 399 funktionale (F), 27 nicht funktionale (NF), 44 funktionale mit Bedingung, und 19 nicht-funktionale mit Bedingung Anforderungen. Nicht berücksichtigt wurden hierbei komplexe juristische Anforderungen aus der Masterarbeit, da diese zu stark von den Satzschablonen nach SOPHIST abweichen und daher nicht sinnvoll durch das Konzept identifiziert werden können. Nachfolgend werden in Tabelle 7.2 die Ergebnisse der Klassifikation dargestellt. Die Präzision wird hierbei mit „Präz.“ und die Ausbeute mit „Ausb.“ abgekürzt.

Satzschablone Art	Vorhergesagt F	NF	F_B	NF_B	Präz.	Ausb.	F1-Maß
F (n = 399)	344	10	19	26	0,960	0,862	0,908
NF (n = 27)	7	16	1	3	0,484	0,592	0,532
F_B (n = 44)	7	2	22	13	0,488	0,500	0,493
NF_B (n =19)	0	5	3	11	0,207	0,578	0,304
Gesamwert	358	33	45	53	0,534	0,633	0,559

Tabelle 7.2: Ergebnisse der Präzision und Ausbeute des SVM-Modell

Wie in Tabelle 7.2 zu sehen ist, können funktionale Anforderungen mit einer hohen Präzision und Ausbeute erkannt werden. Das schlechte Ergebnis der Klassifikation nicht-funktionaler Anforderungen, ist auf das schlechte F1-Maß bei der Erkennung der semantischen Rollen „Attribut“ und „Vergleichsoperator“ zurückzuführen, welche maßgebliche Merkmale einer nicht-funktionalen Anforderung oder einer Bedingung sind. Ein weiteres Problem, stellen Sonderzeichen dar, welche allgemein zu Fehlern in der Erkennung semantischer Rollen führen.

7.3 Autovervollständigung

Außer für die Klassifikation der Texteingabe werden die semantischen Rollen im Kontext der Autovervollständigung benutzt. Diese Unterstützung basiert auf der Vorhersage des nächstes Wortes. Im Kapitel 6 wurde die Unterstützung mittels Markovketten realisiert, indem die vorletzte und letzte semantischen Rolle verwendet werden, um eine semantische Rolle, die schon in Trainingsdaten vorhanden ist, vorherzusagen. Diese Trainingsdaten sind semantische Rollen in richtigen Reihenfolgen basierend auf den Satzschablonen nach SOPHIST. Um die Güte des Markovketten-Modells zu evaluieren, werden strukturierte Anforderungen benutzt. Dies sind die gleichen Testdaten, die für die Semantische-Rollen-Erkenner verwendet werden, aber nur die 66 Anforderungen des Partners. Jedes Wort der Beispieleingabe wird einzeln eingegeben. Nach der Eingabe eines Wortes vergleichen wir, ob die vorgeschlagene semantische Rolle mit der semantischen Rolle des nächsten Wortes aus der Beispieleingabe übereinstimmt. Das Ergebnis für das F1-Maß des Markovketten-Modells ist nicht so gut und es ist nur 0,59.

Eines der Probleme, die bei dieser Bewertung aufgetreten sind, besteht darin, dass während des Dokumentationsprozesses Wörter in einer bestimmten Position mit unterschiedlichen Markierungen von spaCy erkannt wurden. Zum Beispiel das Wort „Satzschablone“ in einem unvollständigen Satz „Das System sollte fähig sein, die Satzschablone“ hat keine semantische Rolle, denn in diesem Fall ist die Kombination der Markierungen (Wortart-Markierung und Abhängigkeit-Markierung) des spaCy noch nicht in der Regel des Semantischen-Rollen-Erkenner enthalten. Und nachdem der Satz mit einem Prozesswort abgeschlossen ist, wird das Wort „Satzschablone“ als die semantische Rolle „Objekt“ erkannt. Wegen dieser fehlenden semantischen Rolle führt es zu einer falschen Vorhersage des nächstes Wortes. Außerdem sind auch Aufzählungen oder Präzisierungen eines Objekts ein Problem, da dies dazu führen kann, dass das Ergebnis eine redundante semantische Rolle ist. Wenn dies der Fall ist, hat das Markovketten-Modell das letzte und vorletzte semantische Rollen derselben semantischen Rolle, die nicht in das Modell trainiert sind und deswegen wird es keine Ausgabevorhersagen vorliegen.

7.4 Ergebnisse des Evaluationsworkshops

Um das Konzept des Assistenzsystem zur aktiven Unterstützung der Anforderungsdokumentation zu evaluieren, wurde im Rahmen dieser Arbeit ein Evaluationsworkshop am FZI Forschungzentrum Informatik gemacht. Ziel dieses Workshops ist es, der Ablauf der Anforderungsdokumentation zu untersuchen. Hierzu sollen neben den subjektiven Merkmalen wie z.B. die Benutzerfreundlichkeit, auch messbare Merkmale wie die Dauer und Vollständigkeit der mithilfe des Dokumentationsassistenten (DA) erzeugten Anforderungen mit Anforderungen verglichen werden, welche ohne den DA erstellt wurden.

Es wurden Szenarien erstellt, die für den Vergleich des Dokumentationsassistenten verwendet werden. Es gibt zwei Szenarien. Das erste Szenario wird ohne die Hilfe des DA bearbeitet, und das zweite Szenario mit ihm. Beide Szenarien werden so gestaltet, als ob ein Kunde ein Produkt wünscht. Die Tester des Workshops fungieren als Mitarbeiter, die Anforderungen schreiben, die aus dem Szenario abgeleitet werden können. Die Szenarien sieht wie folgendermaßen aus:

Szenario 1:

> Guten Tag, ich wünsche mir ein Auto, welches mindestens 100km/h schnell fahren kann, mindestens 2 Bierkisten transportieren kann, gelb ist und es sollte einen maximalen Verbrauch von 6 Litern pro 100 Kilometern haben.

Szenario 2:

> Guten Tag, ich hätte gerne eine Fräsmaschine, die bis zu 100 Werkzeuge bevorraten kann. Der Vorschub sollte mindestens 60m/Minute betragen. Die Fräse soll zudem auch Carbon bearbeiten können und schwarz angemalt sein, wie unser Firmenlogo.

Als nächstes werden Schritte des Anforderungsdokumentationsprozesses des ersten Szenarios erläutert. Für das erste Szenario wird ein Textverarbeitungsprogramm (Mircrosoft Word) verwendet, um die Zuweisung vorzunehmen und die neuen Anforderungen zu dokumentieren. Die Tester wurden angewiesen, Anforderungen aus dem Szenario herauszuholen. Die Bearbeitungszeit der gesamten Dokumentation und die erstellten Anforderungen werden aufgezeichnet.

Beim zweiten Szenario wird den Testern kurz erklärt, wie man mit dem Dokumentationsassistenten arbeitet und welche Hilfestellungen wie die gegebenen Beispielsätze zur Verfügung stehen. Danach wird nach jedem Speicherbefehl die benutzte Satzschablone und die Eingabetexte gespeichert. Ein stiller Beobachter protokolliert die gesamte Bearbeitungszeit und schaut, ob der Tester die Hilfestellungen benutzt oder nicht. Am Ende des Workshops wird allgemeines Feedback des Testers erfragt und gesammelt. Als eine mögliche Lösung der beiden Szenarien werden folgende standardisierte und vollständige Anforderungen nach SOPHIST dargestellt (siehe Tabelle 7.3).

Insgesamt nahmen 8 Probanden an dem Versuch teil. Die Tester sind 5 Studenten und 3 Forscher, von denen fünf Leute bereits im Studium oder in der Arbeit einige Anforderungen z.B für Softwareentwicklung geschrieben haben. Aber es gibt ein Kriterium, das die Tester erfüllen müssen, nämlich dass sie nicht wissen, wie die Satzschablonen von SOPHIST aussehen. Vorkenntnisse darüber, wie eine gute Anforderung oder ähnliches zu stellen ist, sind in diesem Workshop nicht erforderlich. Hierbei wurde die Vollständigkeit und Dokumentationsdauer untersucht. Die Probanden wird nummeriert, und nach Erfahrung (in Tabelle 7.4 als Erf. abgekürzt) mit dem Schreiben von Anforderungen zugeordnet. Tabelle 7.4 zeigt, dass sich die Vollständigkeit der Anforderungen durch den Einsatz des Dokumentationsassistenten stark verbessert.

Satzschablone Art	**Text**
Szenario 1	
Nicht-funktionale	Die Geschwindigkeit des Autos muss mindestens 100 km/h schnell sein.
Funktionale	Das Auto sollte mindestens 2 Bierkisten transportieren können.
Nicht-funktionale	Die Farbe des Autos muss gleich gelb sein.
Nicht-funktionale	Der maximale Verbrauch des Autos muss gleich 6l/100km sein.
Szenario 2	
Funktionale	Die Fräsemaschine sollte bis zu 100 Werkzeuge bevorraten können.
Nicht-funktionale	Der Vorschub der Fräsemaschine sollte mindestens 60m/Minute sein.
Funktionale	Die Fräsemaschine sollte Carbon bearbeiten können.
Nicht-funktionale	Die Farbe der Fräsemaschine sollte gleich schwarz sein.

Tabelle 7.3: Mögliche Anforderungsbeispiele für die Szenarien

		Szenario 1		**Szenario 2**	
	Proband_ID	**Vollständigkeit**	**Dauer**	**Vollständigkeit**	**Dauer**
ohne Erf.	Proband 1	31,25%	1m:38s	87,5%	2m:10s
	Proband 4	85%	2m:56s	93,75%	3m:15s
	Proband 7	43,75%	1m:42s	87,5%	2m:12s
mit Erf.	Proband 2	43,75%	1m:11s	87,5%	2m:15s
	Proband 3	100%	2m:10s	100%	1m:56s
	Proband 5	56,25%	2m:23s	87,5%	2m:18s
	Proband 6	45%	1m:18s	93,75%	2m:56s
	Proband 8	56,25%	3m:11s	100%	2m:41s

Tabelle 7.4: Ergebnisse des Evaluationsworkshops

Die Bearbeitungsdauer hingegen nimmt um ca. 20% zu. Schaut man sich die Bearbeitungszeiten der einzelnen Probanden genauer an, ist jedoch zu sehen, dass die Bearbeitungszeiten der Probanden, welche auch ohne Assistent gute Anforderungen geschrieben haben, nur geringfügig ändern. Probanden welche ohne Dokumentationsassistenten unvollständige Anforderungen geschrieben haben, brauchen im Vergleich wesentlich länger. Dies lässt darauf schließen, dass diese Probanden sich zunächst mit den Hilfestellungen auseinandersetzen mussten, um ihre initiale Denkweise hinsichtlich der Dokumentation von Anforderungen zu überarbeiten. Im Gegenzug, wurden ihre Anforderungen jedoch deutlich besser, was auf einen merklichen Lerneffekt schließen lässt. Ein Beispiel hierfür ist Proband 2 welcher für die erste Szenario Anforderungen in einer stark abstrahierten Form formuliert hat z.B. „Gepäckraum für 2 Kisten“. Durch die Hilfestellungen im Dokumentationsassistenten, konnte erreicht werden, dass er vollständigere Anforderungen definiert z.B. „Der Vorschub sollte mindestens 60m/Minute betragen“ oder „Die Fräsmaschine soll Carbon bearbeiten können“. Dies zeigt eindeutig, dass die Hilfestellungen dem Probanden die Struktur der Satzschablonen nach SOPHIST nahegebracht haben.

Es sollte auch berücksichtigt werden, dass die Reihenfolge der Auswertung, in der welches Szenario zuerst geschrieben werden muss, das Ergebnis beeinflusst. In diesem Fall besteht die Möglichkeit, dass die Tester aufgrund des Szenario 1 daran gewöhnt sind, eine Anforderungen zu schreiben. Solches Problem kann vermieden werden, indem der Bewertungstest in zwei Gruppen unterteilt wird. Eine Gruppe muss nur Anforderungen ohne den Doku-

mentationsassistenten schreiben und die andere Gruppe testet nur mit dem DA. Danach werden die Ergebnisse von den beiden verglichen werden.

8 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Konzept zur Unterstützung der Benutzer bei der Anforderungsdokumentation erstellt. Diese Unterstützungen sind in drei Hauptbereiche unterteilt: den Semantische-Rollen-Erkenner, die Klassifizierung der Texteingabe in eine Satzschablone nach SOPHIST mit dem SVM-Modell und die Autovervollständigung mit der Verwendung von Markovketten.

Die Eingaben des Dokumentationsassistenten erfolgen in Form von Freitext in deutscher Sprache. Diese Eingaben werden als nächstes mit Hilfe von Methoden zur Verarbeitung von natürlicher Sprache verarbeitet, die von spaCy bereitgestellt werden. Durch die Verwendung verschiedener Markierungen, z. B. Wortart-Markierungen und Abhängigkeitsmarkierungen, aus spaCy, extrahierte der Semantische-Rollen-Erkenner dieser Arbeit die semantische Rollen. Diese semantischen Rollen sind die Bestandteile der Satzschablonen nach SOPHIST, die in dieser Arbeit als Basis einer standardisierten und vollständigen Anforderung deklariert wurden. Während des Eingabevorgangs liefert der Dokumentationsassistent nach jedem vollständigen Wort einen Vorschlag zur automatischen Vervollständigung, der auf der Satzschablone bzw. vorherigen semantischen Rollen basiert. Diese Hilfestellung ist so konstruiert, dass der Benutzer angeleitet wird, eine standardisierte Anforderung zu schreiben. Als nächstes wurden die semantischen Rollen aus der Semantische-Rollen-Erkenner in einer der Satzschablone mittels das SVM-Modell klassifizieren. Das SVM-Modell wurde mit den Häufigkeiten der semantischen Rollen trainiert, die in den Satzschablonen vorkommen, so dass auch eine unstrukturierte Texteingabe klassifiziert werden kann.

Der Dokumentationsassistent wurde in zwei Arten ausgewertet. Für den ersten Bewertungsansatz wurden reelle Anforderungen eines Partners des DAM4KMU-Projekts (382 Anforderungen) und aus juristischen und technischen Quelldokumenten (306 Anforderungen) verwendet, um die Leistung jeder Unterstützung zu bewerten. Der Semantische-Rollen-Erkenner wurde mit 100 Anforderungen aus einer Mischung zwischen den beiden Quellen ausgewertet, und hat ein F1-Maß von 0,835. Dabei wurden die semantischen Rollen „Attribut“ (F1-Maß von 0,622) und „Vergleichsoperator“ (F1-Maß von 0,400) durch den Erkenner am schlechtesten identifiziert. Außerdem kann das F1-Maß verringert werden, weil der Erkenner bestimmte Wörter wie „Sonderzeichen, Aufzählungen, und Präzisierungen“ noch nicht extrahieren kann. Das Ergebnis des Semantische-Rolle-Erkenners beeinflusst die Auswertung der beiden anderen Unterstützungen. Anhand von 500 Anforderungen wurde das SVM-Modell bewertet und erzielt ein F1-Maß von 0,559. Von den vier Satzschablonen wurde nur die funktionale Anforderung Satzschablone (F1-Maß von

0,908) gut extrahiert. Die anderen drei Schablonen haben ein niedrigeres F1-Maß, weil die semantische Rolle „Attribut“ und „Vergleichsoperator“, deren Erkennung bereits ein niedriges F1-Maß aufweist, in ihnen enthalten sind. Die letzte Unterstützung ist die Vorhersage des nächsten Wortes, die mit Markovketten realisiert wurde, wobei das Modell die letzte und vorletzte semantische Rolle als Eingabe verwendet, um die Vorhersage zu definieren. Wegen ihres einfachen Aufbaus und falscher Erkennung einiger Wörter beträgt das F1-Maß des Markovketten-Modells nur 0,592. Für den zweiten Bewertungsansatz wurde ein Evaluationsworkshop gemacht. Der Workshop diente dazu, die Vollständigkeit und Dokumentationsdauer von Anforderungen zu messen. Es wurden zwei Szenarien im Workshop vorgestellt. Die Tester wurden angewiesen, ein Szenario ohne und das andere mit dem Dokumentationsassistenten zu schreiben. Das Ergebnis des Workshops zeigt, dass sich die Vollständigkeit der Anforderungen durch den Einsatz des Dokumentationsassistenten stark verbessert hat. Die Bearbeitungsdauer nimmt allerdings um ca. 20% zu.

In Kapitel 4 wurden Voraussetzungen der Arbeit beschrieben. Es gibt einige Voraussetzungen, die in dieser Arbeit noch nicht umgesetzt sind (siehe Tabelle .1). Die fehlenden Voraussetzungen sind beispielsweise die Fähigkeit, einige semantische Rollen zu extrahieren, wie z.B. Sonderzeichen, Aufzählungen, und Präzisierungen. Wenn diese erfüllt sind, kann der Dokumentationsassistenten komplexere Sätze extrahieren und einige falsche Vorhersagen vermeiden, wodurch die F1-Maß des Semantische-Rollen-Erkenners verbessert wird. Wenn es in Zukunft genügend Trainingsdaten gibt, kann man die Vorhersage des nächsten Wortes aus Markov-Ketten durch LSTM oder andere RNN-Methoden ersetzen.

Als weitere Verbesserungsmöglichkeit kann man die Voraussetzung 4.2 und 4.3 erfüllen. Dies gibt dem Dokumentationsassistenten die Möglichkeit, neue Arten von Satzschablonen zu erstellen und eine Schablone in einer anderen Schablone zu verwenden, wodurch eine hierarchische Struktur zwischen Schablonen und semantischen Rollen erstellt wird. Die hierarchische Struktur ermöglicht es, nicht nur die Regeln zwischen semantischen Rollen und einer Schablone zu definieren, sondern auch zwischen Vorlagen, wodurch die Möglichkeit einer besseren Klassifizierung noch größer wird. Außerdem wenn ein unklares oder schwaches Wort erkannt wird, kann eine Pop-up Hilfestellung entwickelt und angeboten werden. Dies soll den Benutzer anleiten, indem eine weitere Aktion zur Klärung des Problems angefordert wird.

Literaturverzeichnis

[El05] Elizabeth Hull, Ken Jackson, Jeremy Dick: *Requirements Engineering.* Springer London, 2005. – ISBN 9781849964043 (zitiert auf Seite 17).

[Ale09] Alex Graves, Marcus Liwicki, Santiago Fernandez, Roman Bertolami, Horst Bunke, Jurgen Schmidhuber: *A Novel Connectionist System for Unconstrained Handwriting Recognition.* 2009 (zitiert auf Seite 14).

[All03] Allen, James F.: Natural Language Processing. In: *Encyclopedia of Computer Science, 4th* (2003), S. 1–145 (zitiert auf Seite 9).

[BDD+07] Bauer, Daniel ; Degen, Judith ; Deng, Xiaoye ; Herger, Priska ; Gasthaus, Jan ; Giesbrecht, Eugenie ; Jansen, Lina ; Kalina, Christin ; Kr, Thorben ; Schmidt, Martin ; Scholler, Simon ; Steger, Johannes: FIASCO: Filtering the Internet by Automatic Subtree Classification, Osnabrück. (2007), 07 (zitiert auf Seite 18).

[Coma] Company, Reuse: *Knowledge Manager - KM.* `https://www.reusecompany.com/km-knowledge-manager` (zitiert auf den Seiten xi, 22 und 23).

[Comb] Company, Reuse: *Requirement Authoring Tools - RAT.* `https://www.reusecompany.com/rat-authoring-tools` (zitiert auf Seite 22).

[Ebe14] Ebert, Christof: *Systematisches Requirement Engineering. Anforderungen ermitteln, spezifizieren, analysieren, und verwalten.* Dpunkt, 2014. – ISBN 978–3–86490–139–3 (zitiert auf den Seiten xi und 7).

[Fei16] Fei Cai, Maarten de Rijke: *A Survey of Query Auto Completion in Information Retrieval.* `https://staff.fnwi.uva.nl/m.derijke/wp-content/papercite-data/pdf/cai-survey-2016.pdf`. Version: 2016 (zitiert auf den Seiten xi und 21).

[FH05] Fabricus-Hansen, Cathrine: *Die Grammatik.* DUDEN, 2005 (zitiert auf Seite 10).

[FMK+11] Farfeleder, Stefan ; Moser, Thomas ; Krall, Andreas ; Stålhane, Tor ; Omoronyia, Inah ; Zojer, Herbert: Ontology-Driven Guidance for Requirements Elicitation, 2011, S. 212–226 (zitiert auf Seite 17).

[Fra09] Frank Stöckel, Philip Stolz, Ifthaker Uddin, Larissa Endriss: *Dynamic Expert System for Improving Requirements.* 2009 (zitiert auf Seite 20).

[Fri19] Fritz, Simon: *Wissenschaftliches Gespräch mit Herrn Fritz im August 2019.* 8 2019 (zitiert auf den Seiten xi, 35, 44, 47 und 49).

[Gia19] Gianna Reich, Dominik Stober: *DAM4KMU – Digitaler Assistent für das Anforderungsmanagement für KMU – Projektstart.* `https://blog.netsyno.com/2019/netsyno-ist-koordinator-beim-`

dam4kmu-projekt-gefordert-vom-bundesministerium-fur-bildung-und-forschung/. Version: 5 2019 (zitiert auf Seite 2).

[H. 68] H. Christopher Longuet-Higgins, Andrew Ortony: The adaptive memorization of sequences. In: *Proceedings of the 3rd Annual Machine Intelligence Workshop* (1968), S. 311–322 (zitiert auf Seite 12).

[Hac04] Hacioglu, Kadri: Semantic Role Labeling Using Dependency Trees. In: *Proceedings of the 20th International Conference on Computational Linguistics*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2004 (COLING '04) (zitiert auf Seite 19).

[Hei10] Heidenreich, Martin: *Metrics and tool support for inspection of requirements*. http://www.martin-heidenreich.com/download/scripte/Artikel-ObjektSpektrum-RE.pdf. Version: 2010 (zitiert auf Seite 20).

[Hil12] Hilke Dreyer, Richard Schmitt: *Lehr- und Übungsbuch der deutschen Grammatik*. Hueber, 2012. – ISBN 978–3–19–307255–9 (zitiert auf den Seiten 10 und 11).

[HLLZ17] He, Luheng ; Lee, Kenton ; Lewis, Mike ; Zettlemoyer, Luke: Deep Semantic Role Labeling: What Works and What's Next. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada : Association for Computational Linguistics, Juli 2017, 473–483 (zitiert auf Seite 19).

[hLN+17] hu, Linmei ; Li, Juanzi ; Nie, Liqiang ; li, Xiaoli ; Shao, Chao: What Happens Next? Future Subevent Prediction Using Contextual Hierarchical LSTMt, 2017 (zitiert auf Seite 21).

[Hof09] Hoffmann, Ludger: *Handbuch der deutschen Wortarten*. Walter de Gruyter, 2009 (zitiert auf Seite 10).

[HS97] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long Short-Term Memory. In: *Neural Computation* 9 (1997), Nr. 8, 1735-1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735. – DOI 10.1162/neco.1997.9.8.1735 (zitiert auf Seite 14).

[IEE91] IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. In: *IEEE Std 610* (1991), Jan, S. 1–217. http://dx.doi.org/10.1109/IEEESTD.1991.106963. – DOI 10.1109/IEEESTD.1991.106963 (zitiert auf Seite 18).

[Jö07] Jörg Meibauer, Ulrike Demske, Jochen Geilfuss-Wolfgang, Jürgen Pafel, Karl Heinz Ramers, Monika Rothweiler, Markus Steinbach: *Einführung in dei germanistische Linguistik*. J. B. Metzler Stuttgart . Weimar, 2007 (zitiert auf Seite 10).

[Kat10] Katja Kessel, Sandra Reimann: *Basiswissen Deutsche Gegenwartssprache*. UTB, Stuttgart, 2010 (zitiert auf Seite 10).

[KB09] Korner, S. J. ; Brumm, T.: RESI - A Natural Language Specification Improver. In: *2009 IEEE International Conference on Semantic Computing*, 2009, S. 1–8 (zitiert auf den Seiten xi und 19).

[KFN10] Kohlschütter, Christian ; Fankhauser, Peter ; Nejdl, Wolfgang: Boilerplate Detection Using Shallow Text Features, 2010, S. 441–450 (zitiert auf Seite 18).

[KH15] Krisch, J. ; Houdek, F.: The myth of bad passive voice and weak words an empirical investigation in the automotive industry. In: *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, 2015. – ISSN 1090–705X, S. 344–351 (zitiert auf den Seiten 19 und 20).

[Kla09] Klaus Pohl, Chris Rupp: *Basiswissen Requirements Engineering*. Dpunkt, 2009. – ISBN 978–3–89864–613–0 (zitiert auf den Seiten 6, 8, 9 und 19).

[Kon01] Kononenko, Igor: Machine Learning for Medical Diagnosis: History, State of the Art and Perspective. (2001) (zitiert auf Seite 13).

[LKT+15] Landhausser, M. ; Korner, S. J. ; Tichy, W. F. ; Keim, J. ; Krisch, J.: DeNom: a tool to find problematic nominalizations using NLP. In: *2015 IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, 2015, S. 1–8 (zitiert auf Seite 19).

[MH02] Magnuson, Tina ; Hunnicutt, Sheri: Measuring the effectiveness of word prediction: The advantage of long-term use. In: *TMH-QPSR* 43 (2002), 01 (zitiert auf Seite 21).

[MLSG13] Morillot, Olivier ; Likforman-Sulem, Laurence ; Grosicki, Emmanuèle: New baseline correction algorithm for text-line recognition with bidirectional recurrent neural networks. In: *Journal of Electronic Imaging* 22 (2013), Nr. 2, 1 – 12. http://dx.doi.org/10.1117/1.JEI.22.2.023028. – DOI 10.1117/1.JEI.22.2.023028 (zitiert auf Seite 22).

[Nik19] Niklaus, Christina and Cetto, Matthias and Freitas, Andre and Handschuh, Siegfried: *Transforming Complex Sentences into a Semantic Hierarchy*. 06 2019 (zitiert auf den Seiten xi, 20, 30 und 33).

[NMKS90] Nakamura, Masami ; Maruyama, Katsuteru ; Kawabata, Takeshi ; Shikano, Kiyohiro: Neural Network Approach to Word Category Prediction for English Texts. In: *Proceedings of the 13th Conference on Computational Linguistics - Volume 3*. Stroudsburg, PA, USA : Association for Computational Linguistics, 1990 (COLING '90). – ISBN 952–90–2028–7, 213–218 (zitiert auf Seite 21).

[Poh08] Pohl, Klaus: *Requirements Engineering*. Dpunkt, 2008. – ISBN 978–3–89864–550–8 (zitiert auf den Seiten 6 und 7).

[Pow07] Powers, David M W.: *Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness, and Correlation*. 2007 (zitiert auf den Seiten xi, 13 und 14).

[PR09] Pasternack, Jeff ; Roth, Dan: Extracting article text from the Web with maximum subsequence segmentation, 2009, S. 971–980 (zitiert auf Seite 18).

[PRY08] Punyakanok, Vasin ; Roth, Dan ; Yih, Wen-tau: The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. In: *Computational Linguistics* 34 (2008), Nr. 2, 257-287. http://dx.doi.org/10.1162/coli.2008.34.2.257. – DOI 10.1162/coli.2008.34.2.257 (zitiert auf Seite 19).

[PS16] Pasupa, K. ; Sunhem, W.: A comparison between shallow and deep architecture classifiers on small dataset. In: *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2016, S. 1–6 (zitiert auf Seite 18).

[PWH+05] Pradhan, Sameer ; Ward, Wayne ; Hacioglu, Kadri ; Martin, James H. ; Jurafsky, Daniel: Semantic Role Labeling Using Different Syntactic Views.

In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2005 (ACL '05), 581–588 (zitiert auf Seite 19).

[RJ86] Rabiner, L. R. ; Juang, B. H.: An introduction to hidden Markov models. In: *IEEE ASSp Magazine* (1986) (zitiert auf Seite 15).

[Rol10] Rolf Bergmann, Peter Pauly, Stefanie Stricker: *Einführung in die deutsche Sprachwissenschaft*. Winter Heidelberg, 2010 (zitiert auf Seite 10).

[Rol13] Roland Kluge, SOPHIST GmbH: *Schablonen für alle Fälle*. 2013 (zitiert auf den Seiten xi, xiii, 1, 3, 8, 9, 11 und 29).

[Rup14] Rupp, Chris: *Requirements-Engineering und -Management*. Hanser, 2014. – ISBN 978–3–446–43893–4 (zitiert auf den Seiten 8, 9 und 37).

[SB13] Schäfer, Roland ; Bildhauer, Felix: Web Corpus Construction. In: *Synthesis Lectures on Human Language Technologies* 6 (2013), 07, S. 1–145. http://dx.doi.org/10.2200/S00508ED1V01Y201305HLT022. – DOI 10.2200/S00508ED1V01Y201305HLT022 (zitiert auf Seite 18).

[Sch] Schäfer, Roland: *Efficient High-precision Boilerplate Detection Using Multilayer Perceptrons (Extended abstract of unpublished paper)*. https://pdfs.semanticscholar.org/9eab/eb3eb48db20c1943430da986318895b5e7db.pdf (zitiert auf den Seiten xi und 18).

[Sch09] Schedl, Sonja: *Integration von Anforderungsmanagement in den mechatronischen Entwicklungsprozess*, Technischen Universität München, Diss., 2009 (zitiert auf Seite 6).

[SMP08] Spousta, Miroslav ; Marek, Michal ; Pecina, Pavel: Victor: the Web-Page Cleaning Tool, 2008. – ISBN 2951740840, S. 12–17 (zitiert auf Seite 18).

[Sri19] Srikanthan, Vethiga: *Automatisierte Extraktion sicherheitsrelevanter Anfoderungen aus juristischen und technischen Dokumenten - Master Arbeit*, Karlsruhe Institute of Technology, Diplomarbeit, 5 2019 (zitiert auf den Seiten 6, 36 und 51).

[Sta10] Stalhane T., Omoronyia I., Reicenbach F.: Ontology-guided requirements and safety analysis. In: *6th International Conference on Safety of Industrial Automated Systems* (2010) (zitiert auf Seite 17).

[SVM17] *Support Vector Machine*. https://scikit-learn.org/stable/modules/svm.html. Version: 2017 (zitiert auf Seite 15).

[Uhr95] Uhrig, R. E.: Introduction to artificial neural networks. In: *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics* Bd. 1, 1995, S. 33–37 vol.1 (zitiert auf Seite 14).

[Wal01] Wallmüller, Ernest: *Software-Qualitätsmanagement in der Praxis*. Carl Hanser Verlag, 2001. – ISBN 3446213678 (zitiert auf Seite 5).

[YOI92] Yamato, J. ; Ohya, J. ; Ishii, K.: Recognizing human action in time-sequential images using hidden Markov model. In: *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1992, S. 379–385 (zitiert auf Seite 15).

[Ziv11] Ziv Bar-Yossef, Naama Kraus: Context-sensitive query auto-completion. In: *Proceedings of the 20th International World Wide Web Conference* (2011), S. 107–116 (zitiert auf den Seiten 12 und 21).

[ZX15] Zhou, Jie ; Xu, Wei: End-to-end learning of semantic role labeling using recurrent neural networks. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China : Association for Computational Linguistics, Juli 2015, 1127–1137 (zitiert auf Seite 19).

Anhang

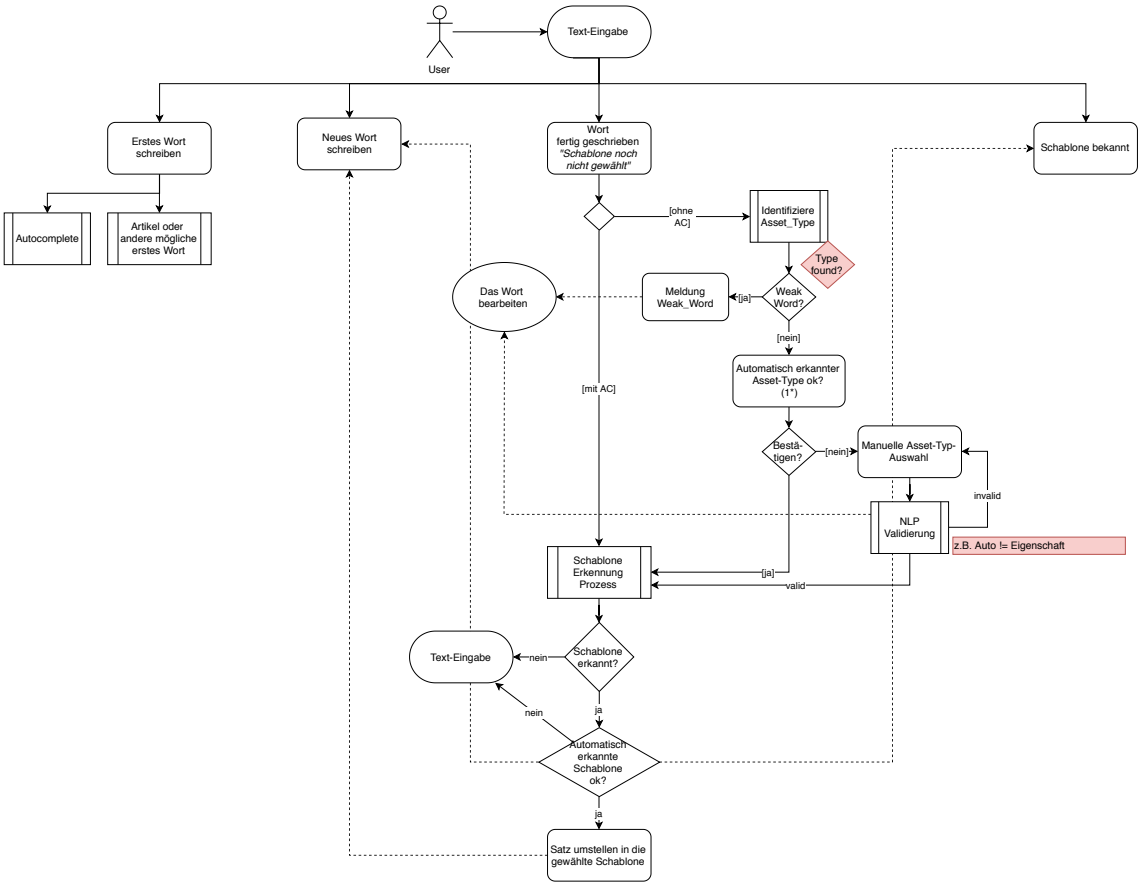


Abbildung .1: Beispiel Ablauf des gesamtes Prozess

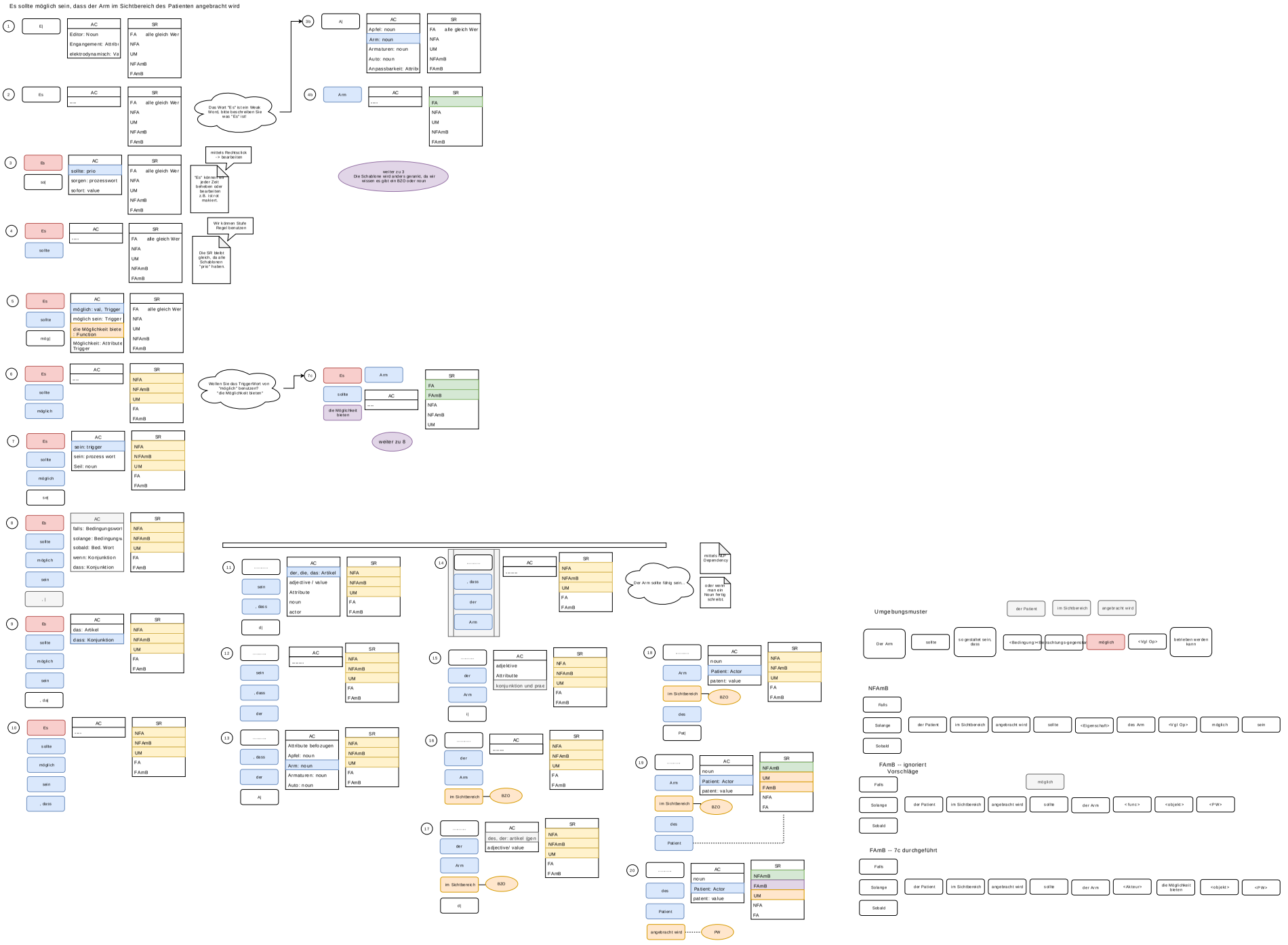


Abbildung .2: Beispiel Autovervollständigung Ablauf ins Detail

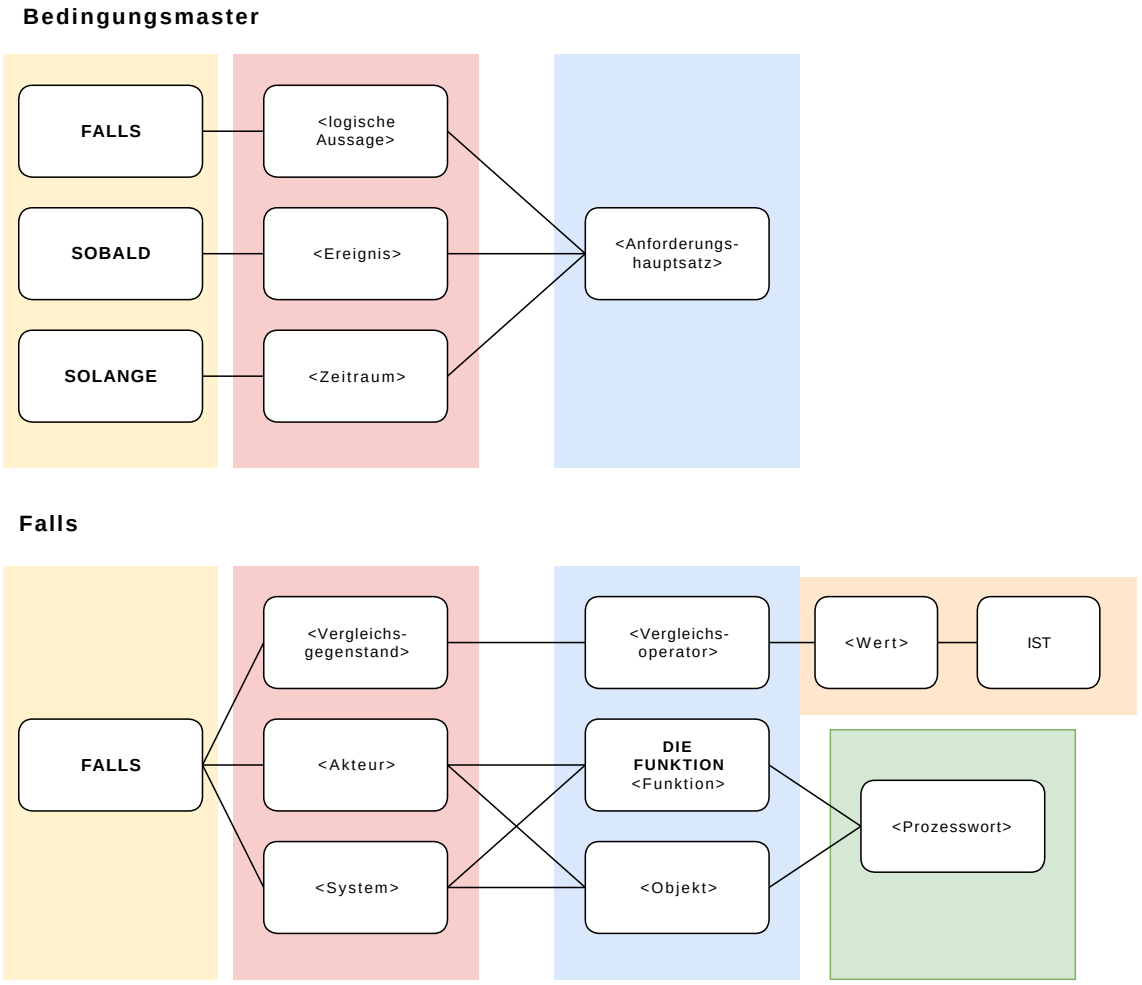


Abbildung .3: Bedingung Schablonen-1

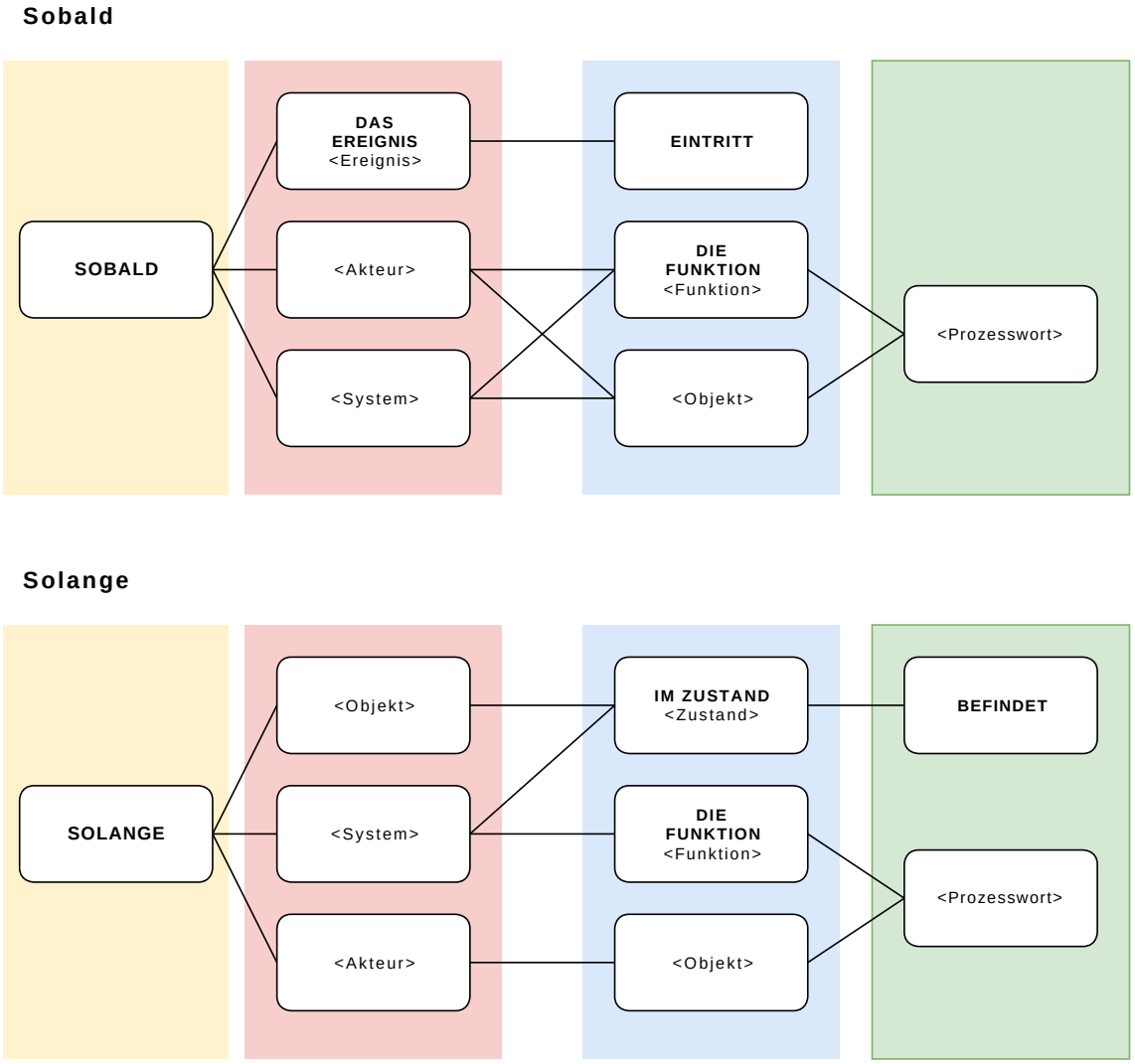


Abbildung .4: Bedingung Schablonen-2

Vorrausetzung Nummer	Inhalt	Umgesetzt
4.1	Das System muss in der Lage sein, eine passende Satzschablone aus einem grammatikalisch gleich strukturierten Freitext automatisch zu erkennen.	ja
4.2	Das System muss in der Lage sein, neue Satzschablone bestehend aus mehreren aneinander gereihten Wortarten zu definieren.	nein
4.3	Das System muss in der Lage sein, beim Erzeugung einer neuen Schablone auch andere Schablonen zu integrieren.	nein
4.4	Das System muss in der Lage sein, semantische Rollen zu erkennen.	ja
4.5	Das System sollte in der Lage sein, eine passende Satzschablone vorzuschlagen, auch wenn die angegebene Eingabe falsch den vorgegebenen Satzschablone entspricht.	ja
4.6	Das System muss in der Lage sein, semantische Rollen auch ohne umfangreiche Wortlisten zu erkennen.	ja
4.8	Das System muss in der Lage sein, semantische Rollen nach SOPHIST automatisch zu erkennen.	ja
4.9	Das System sollte in der Lage sein, Arten von Verben (Wortart) als eine der Arten von Prozesswörtern (semantische Rolle) zu erkennen.	nein
4.11	Das System sollte in der Lage sein, die Präzisierungen der Anforderung zu erkennen.	nein
4.12	Das System sollte in der Lage sein, die Präzisierungen von einem Objekt und gesamten Anforderung zu unterscheiden.	nein
4.13	Das System sollte fähig sein, zwischen einem System und einem Attribut oder Eigenschaft zu unterscheiden. Die beide sind eine semantische Rolle mit gleichem Wortart-Markierung und Abhängigkeit-Markierung (NOUN sb).	ja
4.14	Das System sollte in der Lage sein, einen komplexen Satz in einen relativ einfacheren Satz umzuwandeln.	nein
4.15	Das System sollte fähig sein, Adjektivdeklination Problem zu behandeln.	nein
4.16	Das System sollte fähig sein, schwache Worte (engl. Weak Words) anhand einer vordefinierten Wortliste zu detektieren.	ja
4.17	Das System sollte fähig sein, Sonderzeichen und Abkurzungen zu benutzen oder zu einem anderen Form umzuformen.	nein

Tabelle .1: Tabelle von Voraussetzungen der Arbeit

POS	Name	Beispiele
ADJ	Adjektiv	hässlich, grau, schnell, schön
ADP	Präposition	aus, bei, mit, um, für, durch
ADV	Adverb	hier, da, dort, unten, gestern, morgen
AUX	Hilfsverb, Auxiliarverb	sein, haben, werden(passiv)
CONJ	Konjunktion	und, oder, aber, denn
DET	Determinator	ein, eine, der, die, das
INTJ	Interjektion	ach, ähh, aua, ey, hatschi
NOUN	Nomen	Apfel, Burg, Haus
NUM	Zahlen	1, 12, 23, 14
PART	Partikel	nicht
PRON	Pronomen	ich, du, sie, Sie, er, es, wir
PROPN	Eigenname	Simon, Ryan, FZI, BBC
PUNCT	Zeichensetzung/Interpunktion	. ( ) ?
SCONJ	Nebensatz Konjunktion	wenn, während, falls, solange, als (temporale)
SYM	Symbole	$, %, &
VERB	Verb	kaufen, machen, lachen, programmieren
X	Sonstige	axaxafff, abdecfeg, 13daax

Tabelle .2: Liste von spaCy Wortart-Markierungen.

TAG	POS	DESCRIPTION
$(	PUNCT	other sentence-internal punctuation mark
$,	PUNCT	comma
$.	PUNCT	sentence-final punctuation mark
ADJA	ADJ	adjective, attributive
ADJD	ADJ	adjective, adverbial or predicative
ADV	ADV	adverb
APPO	ADP	postposition
APPR	ADP	preposition; circumposition left
APPRART	ADP	preposition with article
APZR	ADP	circumposition right
ART	DET	definite or indefinite article
CARD	NUM	cardinal number
FM	X	foreign language material
ITJ	INTJ	interjection
KOKOM	CONJ	comparative conjunction
KON	CONJ	coordinate conjunction
KOUI	SCONJ	subordinate conjunction with “zu” and infinitive
KOUS	SCONJ	subordinate conjunction with sentence
NE	PROPN	proper noun
NNE	PROPN	proper noun
NN	NOUN	noun, singular or mass
PAV	ADV	pronominal adverb
PROAV	ADV	pronominal adverb
PDAT	DET	attributive demonstrative pronoun
PDS	PRON	substituting demonstrative pronoun
PIAT	DET	attributive indefinite pronoun without determiner
PIDAT	DET	attributive indefinite pronoun with determiner
PIS	PRON	substituting indefinite pronoun

Tabelle .3: Liste von spaCy-TAG-1

TAG	POS	DESCRIPTION
PPER	PRON	non-reflexive personal pronoun
PPOSAT	DET	attributive possessive pronoun
PPOSS	PRON	substituting possessive pronoun
PRELAT	DET	attributive relative pronoun
PRELS	PRON	substituting relative pronoun
PRF	PRON	reflexive personal pronoun
PTKA	PART	particle with adjective or adverb
PTKANT	PART	answer particle
PTKNEG	PART	negative particle
PTKVZ	PART	separable verbal particle
PTKZU	PART	
PWAT	DET	attributive interrogative pronoun
PWAV	ADV	adverbial interrogative or relative pronoun
PWS	PRON	substituting interrogative pronoun
TRUNC	X	word remnant
VAFIN	AUX	finite verb, auxiliary
VAIMP	AUX	imperative, auxiliary
VAINF	AUX	infinitive, auxiliary
VAPP	AUX	perfect participle, auxiliary
VMFIN	VERB	finite verb, modal
VMINF	VERB	infinitive, modal
VMPP	VERB	perfect participle, modal
VVFIN	VERB	finite verb, full
VVIMP	VERB	imperative, full
VVINF	VERB	infinitive, full
VVIZU	VERB	
VVPP	VERB	perfect participle, full
XY	X	non-word containing non-letter
SP	SPACE	space

Tabelle .4: Liste von spaCy-TAG-2

LABEL	DESCRIPTION
ac	adpositional case marker
adc	adjective component
ag	genitive attribute
ams	measure argument of adjective
app	apposition
avc	adverbial phrase component
cc	comparative complement
cd	coordinating conjunction
cj	conjunct
cm	comparative conjunction
cp	complementizer
cvc	collocational verb construction
da	dative
dh	discourse-level head
dm	discourse marker
ep	expletive es
hd	head
ju	junctor
mnr	postnominal modifier
mo	modifier
ng	negation
nk	noun kernel element
nmc	numerical component
oa	accusative object
oa	second accusative object
oc	clausal object
og	genitive object
op	prepositional object
par	parenthetical element
pd	predicate
pg	phrasal genitive
ph	placeholder
pm	morphological particle
pnc	proper noun component
rc	relative clause
re	repeated element
rs	reported speech
sb	subject
sp	“subject or predicate
svp	separable verb prefix
uc	unit component
vo	vocative
ROOT	root

Tabelle .5: Liste von spaCy Abhängigkeitsmarkierungen

Evaluationsworkshop Ergebnisse

Musterlösung:

Die Vollständigkeit der Anforderung wird von 25% abgezogen, wenn eine bestimmte semantische Rolle fehlt.

Für diese Musterlösung werden die Satzschablonen

- F (Funktionale) : System, Priorität, Objekt, Prozesswort
- NF (Nicht-funktionale): Attribut, System, Priorität, Attributswert

verwendet. Falls alle 4 semantischen Rollen da sind, wird die Satzschablone die Vollständigkeit von 100% bewertet.

<table>
<tr><th>Proband</th><th>Szenario 1</th><th>Gesamte Vollst.</th></tr>
<tr><td>Proband 1</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>NF</td><td>Mindestens 100 km/h fahren kann.</td><td>System, Attribut, Priorität</td><td>25%</td></tr>
<tr><td>F</td><td>Mindestens 2 Bierkisten transportieren kann.</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Farbe gelb ist.</td><td>System, Attribut, Priorität</td><td>25%</td></tr>
<tr><td>NF</td><td>Maximal 6 L/km Verbrauch haben.</td><td>System, Attribut, Priorität</td><td>25%</td></tr>
</table>
</td><td>31,25%</td></tr>
<tr><td>Proband 2</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>NF</td><td>Verbrauch: 6l/100km</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Farbe: gelb</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Mindestgeschwidigkeit: 100 km/h</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>F</td><td>Gepäckraum für 2 Kisten</td><td>System, Priorität, Prozesswort</td><td>25%</td></tr>
</table>
</td><td>43,75%</td></tr>
<tr><td>Proband 3</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>NF</td><td>Das Auto soll eine VMax von mindestens 100km/h aufweisen.</td><td>-</td><td>100%</td></tr>
<tr><td>F</td><td>Das Auto soll einen Laderaum bereitstellen, der mindestens zwei Bierkisten fasst. (2x50x30x20cm)</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Die Außenfarbe des Autos soll Gelb sein. RGB(x,y,z)</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Das Auto soll maximal 6 Liter Diesel pro 100km kombinierten Verbrauch aufweisen.</td><td>-</td><td>100%</td></tr>
</table>
</td><td>100%</td></tr>
<tr><td>Proband 4</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>NF</td><td>Die Reifen müssen mindestens 100 km/h unterstützen.</td><td>Attribut</td><td>75%</td></tr>
<tr><td>F</td><td>Der Motor muss das Auto auf mindestens 100 km/h beschleunigen können.</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Das Auto muss gelb lackiert sein.</td><td>Attribut</td><td>75%</td></tr>
<tr><td>NF</td><td>Der Motor soll maximal 6 Liter auf 100 km verbrauchen.</td><td>Attribut</td><td>75%</td></tr>
<tr><td>NF</td><td>Das Kofferraumvolumen soll mindestens 80 Liter betragen.</td><td>-</td><td>100%</td></tr>
</table>
</td><td>85%</td></tr>
<tr><td>Proband 5</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>NF</td><td>Ein Auto mit mindestens 100km/h Höchstgeschwindigkeit</td><td>Priorität</td><td>75%</td></tr>
<tr><td>NF</td><td>Transportfähigkeit für mindestens 2 Bierkisten</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Farbe Gelb</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Maximaler Verbrauch von 6 Litern / 100km</td><td>System, Priorität</td><td>50%</td></tr>
</table>
</td><td>56,25%</td></tr>
</table>

<table>
<tr><td>Proband 6</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>F</td><td>Auto</td><td>Priorität, Objekt, Prozesswort</td><td>25%</td></tr>
<tr><td>NF</td><td>Maximalgeschwindigkeit >= 100 km/h</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Farbe: gelb</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Ladekapazität: >= zwei Bierkästen</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Verbrauch <= 6 L / 100 km</td><td>System,, Priorität</td><td>50%</td></tr>
</table>
</td><td>45%</td></tr>
<tr><td>Proband 7</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>F</td><td>Produkttyp: Auto</td><td>Priorität, Objekt, Prozesswort</td><td>25%</td></tr>
<tr><td>NF</td><td>Minimal Geschwindigkeit: 100 km/h</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Minimal Kapazität: 2 Bierkisten</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Farbe: gelb</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Maximal Verbrauch : 6 Liter pro 100 Kilometern</td><td>System,, Priorität</td><td>50%</td></tr>
</table>
</td><td>43,75%</td></tr>
<tr><td>Proband 8</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>NF</td><td>100km/h soll einfach zu erreichen sein</td><td>System, Attribut</td><td>50%</td></tr>
<tr><td>F</td><td>Genug Platz im Kofferaum</td><td>Priorität, Prozesswort</td><td>50%</td></tr>
<tr><td>NF</td><td>Farbe = gelb</td><td>System, Priorität</td><td>50%</td></tr>
<tr><td>NF</td><td>Gasverbrauch soll max 0.06/km sein</td><td>System</td><td>75%</td></tr>
</table>
</td><td>56.25%</td></tr>
</table>

<table>
<tr><th>Proband</th><th>Szenario 2</th><th>Gesamte Voll.</th></tr>
<tr><td>Proband 1</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>F</td><td>fräsmaschine soll bis 100 Werkzeuge bevorraten</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>fräsmaschine soll mindestens 60m/minute betragen</td><td>Attribut</td><td>75%</td></tr>
<tr><td>F</td><td>fräsmaschine soll Carbon bearbeiten können</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>fräsmaschine hat schwarze Farbe</td><td>Priorität</td><td>75%</td></tr>
</table>
</td><td>87,5%</td></tr>
<tr><td>Proband 2</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>F</td><td>Die Fräsmaschine soll Carbon bearbeiten können.</td><td>-</td><td>100%</td></tr>
<tr><td>F</td><td>Die Fräsmaschine soll bis zu 100 Werkzeuge bevorraten.</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Der Vorschub sollte mindestens 60m/Minute betragen</td><td>System</td><td>75%</td></tr>
<tr><td>NF</td><td>Die Fräsmaschine soll schwarz angemalt sein</td><td>Attribut</td><td>75%</td></tr>
</table>
</td><td>87,5%</td></tr>
<tr><td>Proband 3</td><td>
<table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>F</td><td>Die Fräsmaschine soll ein Lager für 100 Werkzeuge haben.</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Die Fräsmaschine soll einen Vorschub von mindestens 60m pro Minute bereitstellen.</td><td>-</td><td>100%</td></tr>
<tr><td>F</td><td>Die Fräsmaschine soll Carbon bearbeiten können.</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Die Fräsmaschine soll in dem XYZ-Schwarz unseres Firmenlogos markiert sein.</td><td>-</td><td>100%</td></tr>
</table>
</td><td>100%</td></tr>
</table>

<table>
<tr><td>Proband 4</td><td><table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>F</td><td>Der Werkzeugspeicher soll 100 Werkzeuge fassen können.</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Der Vorschub der Fräsmaschine sollte größer als 60 m/Minute sein.</td><td>-</td><td>100%</td></tr>
<tr><td>F</td><td>Carbon soll bearbeitet werden können.</td><td>System</td><td>75%</td></tr>
<tr><td>NF</td><td>Die Farbe der Fräsmaschine soll schwarz sein.</td><td>-</td><td>100%</td></tr>
</table></td><td>93,75%</td></tr>
<tr><td>Proband 5</td><td><table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>F</td><td>Die Fräsmaschine sollte 100 Werkzeuge bevorraten</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Der Vorschub sollte mindestens 60m/min betragen</td><td>System</td><td>75%</td></tr>
<tr><td>F</td><td>Die Fräse soll Carbon bearbeiten können</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Die Fräse soll schwarz sein</td><td>Attribut</td><td>75%</td></tr>
</table></td><td>87,5%</td></tr>
<tr><td>Proband 6</td><td><table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>NF</td><td>Die Fräsmaschine sollte schwarz sein.</td><td>Attribut</td><td>75%</td></tr>
<tr><td>F</td><td>Die Fräsmaschine soll Carbon bearbeiten können.</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Die Fräsmaschine sollte einen Vorschub von mindestens 60m/Minute haben.</td><td>-</td><td>100%</td></tr>
<tr><td>F</td><td>Die Fräsmaschine sollte bis zu 100 Werkzeuge bevorraten können.</td><td>-</td><td>100%</td></tr>
</table></td><td>93,75%</td></tr>
<tr><td>Proband 7</td><td><table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>NF</td><td>Die Fräse soll schwarz angemalt sein.</td><td>Attribut</td><td>75%</td></tr>
<tr><td>F</td><td>Die Fräse soll Carbon bearbeiten können</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Die Fräsmaschine sollte mindestens 60m/Minute transportieren.</td><td>Attribut</td><td>75%</td></tr>
<tr><td>F</td><td>Die Fräsmaschine sollte bis zu 100 Werkzeuge bevorraten kann.</td><td>-</td><td>100%</td></tr>
</table></td><td>87,5%</td></tr>
<tr><td>Proband 8</td><td><table>
<tr><th>Art</th><th>Text</th><th>Fehlende semantische Rolle</th><th>Vollst.</th></tr>
<tr><td>NF</td><td>Die Fräsmaschine sollte mindestens 60m/min Vorschub haben</td><td>-</td><td>100%</td></tr>
<tr><td>NF</td><td>Die Fräsmaschine sollte die Farbe Schwarz haben</td><td>-</td><td>100%</td></tr>
<tr><td>F</td><td>Die Fräsmaschine sollte Carbon bearbeiten können</td><td>-</td><td>100%</td></tr>
<tr><td>F</td><td>Die Fräsmaschine sollte bis zu 100 Werkzeuge bevorraten</td><td>-</td><td>100%</td></tr>
</table></td><td>100%</td></tr>
</table>