Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Automatisierte Erkennung von destruktiven Äußerungen in Meetings von Softwareprojekten

Automated Detection of Destructive Statements in Meetings of Software Projects

Masterarbeit

im Studiengang Informatik

von

Zena Obeidi

Prüfer: Prof. Dr. rer. nat. Kurt Schneider
Zweitprüfer: Dr. rer. nat. Jil Ann-Christin Klünder
Betreuer: Dr. rer. nat. Jil Ann-Christin Klünder

Hannover, 18.11.2021

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 18.11.2021

Zena Obeidi

Zusammenfassung

Meetings gehören in Softwareprojekten zu den effektivsten Möglichkeiten der Kommunikation. In Echtzeit können Informationen ausgetauscht und Aufgaben koordiniert werden. Trotz der Vorteile, die Meetings mit sich bringen, kann sich eine dysfunktionale Kommunikation in Meetings negativ auswirken. Ein übermäßiges destruktives Verhalten kann Unzufriedenheit bei Entwicklern hervorrufen und sogar ein Gesundheitsrisiko darstellen. Die negative Haltung der Entwickler kann den Nutzen eines Meetings mindern und darüber hinaus den Erfolg des Projekts gefährden. Mit Methoden der Interaktionsanalyse wie act4teams®-SHORT kann der Anteil an destruktiven Äußerungen in einem Meeting gemessen werden. Einem menschlichen Beobachter ist es jedoch nicht möglich, alle Aussagen in Echtzeit zu bewerten. Daher wurde im Rahmen dieser Arbeit ein Konzept entwickelt, um destruktive Äußerungen in Softwareprojekt-Meetings automatisiert zu erkennen. Das Konzept verwendet zwei Methoden zur Klassifikation. Nach einer Vorverarbeitung der Texteinheiten mit *Natural Language Processing* werden mit Hilfe von *rule-based Matching* destruktive Aussagen anhand ihrer syntaktischen Satzstrukturen erkannt. Auf diese Weise wird nur ein Teil der vorhandenen destruktiven Äußerungen identifizert. Daher werden Aussagen, die als *nicht destruktiv* klassifiziert wurden, für die Klassifikation mit maschinellen Lernverfahren vorbereitet. Die drei Klassifikatoren Support Vector Machine, Gradient Boosting und Naive Bayes treffen Vorhersagen mit Hilfe eines extrahierten Merkmalsvektors, ob Äußerungen *destruktiv* oder *nicht destruktiv* sind. Anhand der Ergebnisse können Vorschläge für das Entwicklerteam zur Verbesserung ausgearbeitet werden, um destruktive Äußerungen in Meetings zu reduzieren. Das Konzept wurde in einem Meeting evaluiert. Die dabei gemessenen Genauigkeitswerte der Klassifikatoren zeigten nur minimale Abweichungen. Der SVM- und Naive-Bayes-Klassifikator erreichten jeweils eine Genauigkeit von ca. 74%. Dies konnte vom Gradient-Boosting-Klassifikator mit einer Genauigkeit von etwa 77% knapp übertroffen werden.

Abstract

Automated Detection of Destructive Statements in Meetings of Software Projects

Meetings are one of the most effective ways of communicating in software projects, as information can be shared and tasks can be coordinated in real time. Despite benefits of meetings, dysfunctional communication can still have a negative impact. Excessive destructive behavior can cause dissatisfaction among developers and even pose a health risk. The negative attitude of developers can diminish the outcome of a meeting and, moreover, threaten the success of the project. Methods of interaction analysis such as act4teams®-SHORT can be used to measure the amount of destructive comments in a meeting. However, it is not possible for a human observer to evaluate all statements in real time. Therefore, this thesis introduces a concept that automatically detects destructive statements in software project meetings. The concept uses two methods for classification. After preprocessing the text units with *Natural Language Processing*, rule-based matching is used to detect destructive statements based on their syntactic sentence structures. In this way, a subset of existing destructive statements can be partially identified. Statements that are not classified as destructive after this step are prepared for classification using machine learning techniques. The three classifiers Support Vector Machine, Gradient Boosting, and Naive Bayes classify the statements as *destructive* or *non-destructive* using an extracted feature vector. Based on the results, suggestions for improvement can be prepared for the development team to reduce destructive statements in meetings. The concept was evaluated in one meeting: the accuracy values of the classifiers measured in this process showed minimal deviations. The SVM and Naive Bayes classifier each achieved an accuracy of approx. 74%. This was slightly surpassed by the gradient boosting classifier with an accuracy of approx. 77%.

Inhaltsverzeichnis

Kapitel 1

Einleitung

In Softwareprojekten ist Kommunikation notwendig, um einen Informationsaustausch zu ermöglichen [30, 38]. Dazu gehört auch die Koordination von Aufgaben, Erlangen neuer Fähigkeiten und die Vermeidung von Unsicherheiten [30, 38]. Unzureichende Kommunikation kann daher zu redundanter Arbeit und Unvorhersehbarkeiten bezüglich der Software oder der Aufgaben der Entwickler führen [38]. Da Software meist aus mehreren Modulen besteht, ist es ebenfalls wichtig, dass die verschiedenen Entwicklerteams durch Kommunikation zu einem gemeinsamen Verständnis für die zu entstehende Software kommen [30, 38], anderenfalls könnte eine Integration der Module fehlschlagen. Eine der effektivsten Arten der Kommunikation ist die Besprechung von Angesicht zu Angesicht, da Informationen schnell und ohne Verzögerung weitergegeben werden können [3, 50]. Aus diesem Grund findet in Softwareprojekten die Kommunikation meist in Meetings statt [61].

Gopal et al. [18] beschreiben im Jahr 2001, dass Kommunikation, sowohl unter den Projektteilnehmern als auch mit den Auftraggebern die Performanz in einem Projekt positiv beeinflusst. Aus diesem Grund wurden in der Forschung Konzepte entwickelt, um die Qualität der Kommunikation in Meetings zu messen und zu optimieren [32, 34]. Dafür werden Äußerungen in Kategorien eingeordnet, die grob in funktionale und dysfunktionale Kategorien aufgeteilt werden können [32, 34]. Kauffeld und Lehmann-Willenbrock [32] kamen im Jahr 2012 zu dem Schluss, dass der negative Effekt durch dysfunktionale Kommunikation stärker ist als der positive Effekt durch funktionale Kommunikation. Die Konsequenz sind Unzufriedenheit bezüglich der Meetings und eine negative und pessimistische Haltung der Entwickler [32]. Schulte et al. [62] zeigten sogar, dass dysfunktionale Kommunikation in Meetings zu einem Gesundheitsrisiko führen kann. Ferner ist dadurch der Erfolg eines Softwareprojekts gefährdet [5].

Destruktives Verhalten in Meetings, das der dysfunktionalen Kommunikation zugeordnet wird, kann durch das Kompetenzmessverfahren act4teams® [32] identifiziert werden. Für Meetings im Bereich des Software Engineering ist jedoch die vereinfachte und speziell für Softwareprojekte konzipierte Weiterentwicklung act4teams®-SHORT [34] besser geeignet. Für eine Meetinganalyse werden Aussagen von Entwicklern in Kategorien eingeteilt und auf diese Weise bewertet. Auf Grundlage der Ergebnisse, die in einer Analyse gewonnen werden, kann die Kommunikation in Entwicklerteams optimiert werden, um die Qualität der Meetings und somit auch den Einfluss auf den Projekterfolg zu verbessern.

1.1 Problemstellung

Wie zuvor erwähnt kann destruktive Kommunikation in Softwareprojekten zu Unzufriedenheit und Pessimismus unter Entwicklern führen [32]. Übermäßiges destruktives Verhalten in Meetings hat zusätzlich einen negativen Einfluss auf die Gesundheit der Entwickler [62]. Schlimmstenfalls kann die negative Stimmung im Team den Erfolg des Projekts gefährden [5]. Es gibt bereits Ansätze, die den Anteil an destruktivem Verhalten in einem Meeting messen, damit Vorschläge daraus für das Team erarbeitet werden können. Kompetenzmessverfahren wie act4teams® und act4teams®-SHORT sind Beispiele dafür. Für die Anwendung von act4teams® wird eine audiovisuelle Aufnahme benötigt. Geschulte Analysten führen eine zeitintensive Evaluation des Meeting mit Hilfe der Aufnahmen durch [32]. Mit der Vereinfachung act4teams®-SHORT, das speziell für Meetings in Softwareprojekten konzipiert wurde, wurde eine Evaluierung in Echtzeit von ungeschulten Beobachtern durch eine Reduktion des Kodierschemas ermöglicht [34]. Bei einer Echtzeitanalyse kann nicht jede Aussage bewertet werden, da ein Mensch diese kognitive Leistung nicht erbringen kann [34]. Für diese Probleme soll ein Konzept zur automatisierten Erkennung aller destruktiven Aussagen aus einem Meeting entwickelt werden, ohne den Arbeits- und Zeitaufwand zu erhöhen. Die Untersuchung fokussiert sich zunächst auf destruktives Verhalten, da der Effekt dieses Verhaltens schwerwiegendere Konsequenzen hat.

1.2 Lösungsansatz

Um Aussagen in einem Meeting auf destruktives Verhalten zu analysieren, müsste ein Beobachter dem Meeting beiwohnen und in Echtzeit eine Bewertung durchführen. Nun soll eine Lösung entwickelt werden, die die Aussagen anhand von Texteigenschaften automatisch als *destruktiv* oder *nicht destruktiv* klassifiziert. Eine Software würde alle Aussagen erkennen und automatisch klassifizieren. Durch eine einfache und unkomplizierte Anwendung könnten Mitglieder des Teams das Softwaretool selbst bedienen und bräuchten somit keine zusätzliche Person für die Analyse. Nach Beendigung des Analysevorgangs wird der Anteil an destruktiven Aussagen im Meeting ausgegeben.

1.3 Struktur der Arbeit

In dieser Arbeit folgen noch sieben weitere Kapitel. Die Grundlagen zu dem Thema Kommunikation in Meetings und ihre Analyse werden in Kapitel 2 beschrieben. In Kapitel 3 werden verwandte Arbeiten vorgestellt und inhaltlich von dieser Arbeit abgegrenzt. Die Vorstellung des Konzepts erfolgt in Kapitel 4. Kapitel 5 behandelt die Umsetzung des Konzepts und die Implementierung des Prototyps. Die Ergebnisse der Evaluierung des Prototyps werden in Kapitel 6 vorgestellt. In Kapitel 7 werden die Ergebnisse zusammengefasst und diskutiert in Hinsicht auf Allgemeingültigkeit. Der Prozess zur Entwicklung des Prototyps und die Resultate werden in Kapitel 8 zusammengefasst. Zusätzlich folgt ein Ausblick über zukünftige Arbeiten.

Kapitel 2

Grundlagen

In dieser Arbeit wird ein Konzept entwickelt und untersucht, um destruktive Äußerungen in Softwareprojekt-Meetings automatisiert zu erkennen. In diesem Kapitel werden daher zunächst Grundlagen, die für das Verständnis der Thematik des Forschungsziels notwendig sind, behandelt. Zu Beginn soll in Abschnitt 2.1 die Kommunikation im Bereich des Software Engineering betrachtet werden. Abschnitt 2.2 gibt eine Übersicht über die Möglichkeiten Kommunikation in Meetings zu analysieren. Die Definition einer destruktiven Äußerung erfolgt in Abschnitt 2.3. Für diese Arbeit ist die Unterscheidung zwischen gesprochener und geschriebener Sprache wichtig, daher gibt Abschnitt 2.4 einen Einblick dazu. Die automatisierte Erkennung erfolgt über Text Mining, dessen Prozess in Abschnitt 2.5beschrieben wird.

2.1 Kommunikation in Software Projekten

Im Bereich der Softwareentwicklung ist Kommunikation notwendig für den Informationsaustausch in Projekten [38]. Dabei müssen verschiedene Rollen in einem Projekt miteinander kommunizieren, um ihre Aufgaben effizient und erfolgreich zu erfüllen oder Unsicherheiten zu vermeiden [18, 38]. Ein Austausch von Information kann auf verschiedenem Wege stattfinden. Dazu gehören beispielsweise Meetings unter Entwicklern oder Workshops mit Kunden, der dazu dient, Anforderungen an eine Software-Lösung zu ermitteln [18]. Sowohl Abschnitt 2.1 als auch die Untersuchungen dieser Arbeit konzentrieren sich auf die Kommunikation in Entwicklerteams.

2.1.1 Kommunikationsmittel

Entwicklerteams können beliebig groß sein, da es sowohl Teams mit drei Mitgliedern gibt [63] als auch welche mit einer Größe von mehreren hundert [46]. Zumindest lässt sich im Bereich der agilen Softwareentwicklung die Zahl etwas eingrenzen, denn Entwicklerteams bestehen dort meist aus

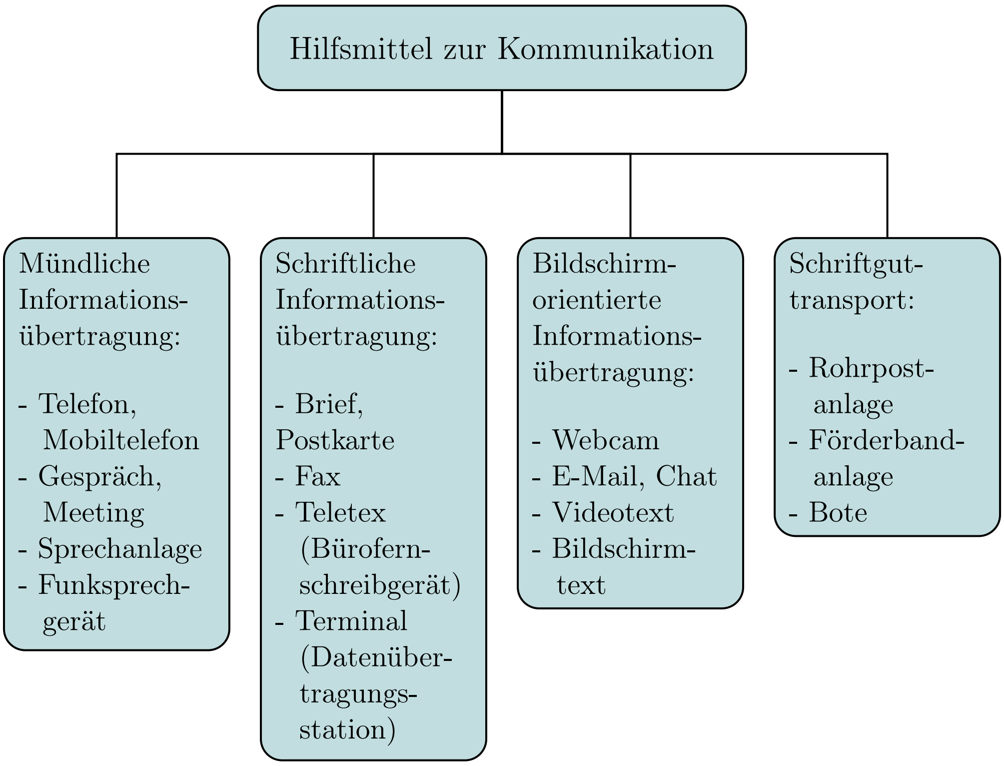


Abbildung 2.1: Aufteilung der Informationsübertragungsmedien [4]

drei bis neun Mitgliedern [63]. Wie in Abbildung 2.1 [4] dargestellt, kann Kommunikation auf unterschiedlichem Wege stattfinden. Appel et al. [4] teilen die Hilfsmittel zur Kommunikation in mündlich, schriftlich, bildschirmorientiert und Schriftguttransport auf. Zur mündlichen Informationsübermittlung gehören unter anderem Gespräche in Person, Telefonate oder auch Meetings. Briefe und mit dem Fax übertragene Dokumente sind der schriftlichen Kommunikation zuzuordnen [4]. Videochats, E-Mails und Kurznachrichten über Messenger-Tools gehören zur bildschrim-orientierten Informationsübertragung. Dokumente, die über Transportanlagen wie etwa Förderbänder oder Rohrpostsysteme versendet werden, fallen unter die Kategorie der Schriftguttransporte [4].

In Softwareprojekten findet ein Großteil der Kommunikation in Meetings statt [61]. Untersuchungen haben gezeigt, dass destruktive Äußerungen sich negativ auf den Erfolg eines Meetings auswirken [32]. Doch die Konsequenzen sind noch weitreichender, denn der Ausgang eines Meetings hat auch Einfluss auf den Erfolg des gesamten Projekts [36, 57]. Daher liegt Fokus dieser Arbeit auf die in Meetings stattfindende Kommunikation unter Entwicklern. Für die Untersuchungen werden Transkripte, aufgezeichnete Tonspuren oder Live-Gespräche benötigt, um einzelne Aussagen auszuwerten.

2.1.2 Kommunikation für den Erfolg eines Projekts

Laut Baccarini [5] ist der Erfolg eines Projekts abhängig vom Erfolg des Projektmanagements (engl.*project management success*) und des Produkts (engl. *product success*). *Project management success* ist dann gegeben, wenn drei Komponenten erfüllt sind:

Das magische Dreieck (Kosten-Zeit-Qualität): Es wird stets ein Optimum aus niedrigen Kosten, geringem Zeitaufwand und hoher Qualität im Sinne von Funktionalität und nicht-funktionalen Eigenschaften angestrebt [5].

Qualität des Projektmanagementprozesses: Hier ist die Qualität des Managements eines Projekts in Hinsicht auf Effizienz entscheidend. Zu diesem Prozess gehören Initialisierung, Planung, Ausführung, Controlling und Abschluss eines Projekts [40].

Zufriedenheit der Projekt-Stakeholder: Die Bedürfnisse der Stakeholder, die am Projektmanagementprozess beteiligt sind, zu erfüllen, ist essentiell für den Erfolg des Projektmanagements [5].

Product success ist abhängig von drei Faktoren:

Ziel des Projekts: Ein Unternehmen verfolgt mit der Durchführung eines Projekts ein strategisches Ziel, das dem Unternehmen zugute kommt. Das Erreichen des Ziels, das vor der Durchführung eines Projekt vom Unternehmen klar definiert wird, trägt zum Erfolg des Produkts bei [5].

Zweck des Projekts: Der Zweck eines Projekts ist es, ein Produkt zu entwickeln, das reale Bedürfnisse erfüllen kann. Ein enger Kontakt mit den Nutzern ist daher entscheidend für die Gewinnung und Erfüllung der Nutzer-Anforderungen und folglich auch für die Erfüllung des Zweck des Projekts [5].

Zufriedenheit der Stakeholder: Das Produkt kann nur erfolgreich sein, wenn die Bedürfnisse der Produkt-Stakeholder, unter anderem jene, die das Produkt nutzen, erfüllt werden. Bei Konflikten sollte man sich für die Zufriedenheit der Nutzer entscheiden, um den Erfolg des Produkts zu steigern [5].

Da die Entwickler zu den Projekt-Stakeholdern gehören, hat die Kommunikation in Entwicklerteams Einfluss auf den Erfolg des Projektmanagements. Somit ist auch der Erfolg des gesamten Projekts abhängig davon, in welcher Art und Weise sich die Entwickler austauschen. Aus diesem Grund ist es von Bedeutung, die Kommunikation in Entwicklerteams, insbesondere in Meetings, zu analysieren und zu verbessern. Der folgende Abschnitt 2.2 beschäftigt sich mit der Kommunikationsanalyse in Meetings.

2.2 Analyse der Kommunikation in Meetings

Kommunikation kann in drei Dimensionen aufgeteilt werden [35]. Die *Struktur* betrachtet die Zusammenstellung des Teams und wer mit wem spricht. In der Dimension *Mimik und Gestik* geht es um die Art und Weise der Kommunikation und um den Umgang der Teammitglieder miteinander [35]. Die Dimension *Inhalt* beschäftigt sich damit, was kommuniziert wird und welche Emotionen durch bestimmte Worte dabei übermittelt werden. Kommunikation kann auf allen drei Dimensionen untersucht werden [35]. Bei der sozialen Netzwerkanalyse wird die *Struktur* untersucht, indem das Entwicklerteam als soziales Netzwerk betrachtet wird. In der Dimension *Mimik und Gestik* sind Interaktionen in Meetings Gegenstand der Untersuchung bei der Methode act4teams®, kurz für *The Advanced Interaction Analysis for Teams*, welches im Folgenden vorgestellt wird. Mit der Methode Sentiment Analysis werden Nachrichten Emotionen zugeordnet, um den *Inhalt* zu untersuchen [35].

2.2.1 act4teams®

Die Methode act4teams®, die im Jahre 2000 zunächst unter dem Namen „Kasseler Kompetenz Raster“ veröffentlicht wurde [33], dient zur Messung beruflicher Handelskompetenz. Kauffeld [31] bestimmte den Begriff der beruflichen Handelskompetenz, indem mehrere Definitionen aufgegriffen wurden, ohne eine von ihnen hervorzuheben. Eine der Definitionen verstand Kompetenz als eine „Kombination von Fähigkeiten, Fertigkeiten und Wissensbeständen, die bei der Bewältigung konkreter sowohl vertrauter als auch neuartiger Arbeitsaufgaben selbstorganisiert, aufgabengemäß, zielgerichtet, situationsbedingt und verantwortungsbewusst - oft in Kooperation mit anderen - handlungs- und reaktionsfähig machen und sich in der erfolgreichen Bewältigung konkreter Arbeitsanforderungen zeigen“ [31, S. 19–20]. Der Grund für eine genauere Betrachtung der Thematik um Kompetenzen ist die Kenntnis, dass eine Feststellung und Förderung der Kompetenzen der Mitarbeiter eine entscheidende Rolle für den Erfolg eines Unternehmens einnimmt [31].

Für die Anwendung von act4teams® wird eine Videoaufzeichnung einer Gruppendiskussion benötigt. Bei diesem Verfahren werden sogenannte Sinneinheiten nach dessen Kodierschema kategorisiert [32].

Kodierschema

Das Kodierschema des act4teams® umfasst 44 Kategorien, die in 4 Kompetenzbereiche aufgeteilt werden [32]. Jede Aussage aus einem Meeting wird einer der 44 Kategorien zugeordnet.

Die **professionelle Kompetenz** beinhaltet problemfokussierte Aussagen, die Probleme ansprechen, Problemlösungen diskutieren, Entscheidungen treffen und somit Übereinstimmungen herbeizuführen [32]. Außerdem können problemfokussierte Aussagen zu Wissensaustausch führen, wenn Probleme und mögliche Lösungsvorschläge behandelt werden.

Der Bereich der **Methodenkompetenz** umfasst prozedurale Aussagen [32]. Diese Art der Aussagen dienen dem Zweck eine Diskussion zu strukturieren und zu organisieren. Zudem führen positive prozedurale Aussagen zu produktiveren Diskussionen, indem sie destruktives Verhalten reduzieren [32].

Sozio-emotionale Aussagen sind dem **Sozialkompetenzbereich** zuzuordnen und betreffen die zwischenmenschlichen Beziehungen in einer Diskussion. Während aktives Zuhören und Unterstützung zu den positiven sozio-emotionalen Aussagen gehören, zählen Tadel und Unterbrechung zu den negativen sozio-emotionalen Aussagen [32].

Zum Bereich der **Selbstkompetenz** gehören aktionsorientierte Aussagen, die zeigen, wie groß die Motivation ist, Maßnahmen zu ergreifen, um die Arbeit des Teams zu verbessern [32]. Konkretes Vorgehen planen oder Interesse für Veränderungen zeigen sind Beispiele für konstruktive Mitwirkung. Zur destruktiven Mitwirkung gehören beispielsweise Jammern oder das Zeigen von Desinteresse an Veränderungen [32, 34].

2.2.2 Weiterentwicklung zu act4teams®-SHORT

Klünder [34] stellte im Jahr 2019 im Rahmen ihrer Dissertation act4teams®-SHORT vor, das speziell für Meetings in Softwareprojekten konzipiert wurde. Das Tool zur Meetinganalyse verwendet dabei eine reduzierte Version des Kodierschemas des bereits bekannten act4teams®. Act4teams®-SHORT ermöglicht die Untersuchung von Aussagen aus Softwareprojekt-Meetings ohne Einarbeitungszeit und in Echtzeit, also während des Meetings. Dabei muss der Beobachter nicht jede Aussagen kodieren, sondern nur jene, die relevant erscheinen [34]. Eine lückenlose Kodierung wäre für einen Menschen, dessen kognitive Leistungen nicht grenzenlos sind, schwer durchführbar. In einem iterativen Vorgehen wurde das neue Kodierschema entwickelt. Nach der ersten Iteration entstand ein reduziertes Kodierschema mit neun Kategorien, das auf Ergebnissen einer act4teams-Analyse auf Meetings in der Softwareentwicklung basierte [34]. Zudem konnten Psychologinnen mit ihren Erfahrungen aus vergangenen act4teams-Analysen dienen. In der zweiten Iteration wurde nach der ersten Evaluation der Implementierung festgestellt, dass das reduzierte Kodierschema zu grob sei und wurde daher in der dritten Iteration um zwei Kategorien erweitert [34, 56]. In einer vierten Iteration entstand die aktuelle Version des Kodierschemas, nachdem die Kategorien von elf auf neun Kategorien reduziert wurden [20, 35]. Eine Reduktion des Kodierschemas war notwendig, da die zum Teil sehr feinen Unterschiede zwischen einigen Kategorien nur von geschulten Beobachtern schnell erkannt werden können. Darüber hinaus kann bei einer Echtzeitanalyse nicht jede

Aussage nach 44 Kategorien unterschieden werden, da ein Mensch diese kognitive Leistung nicht erbringen kann [34]. Mit neun Kategorien sollte es ungeschulten Beobachtern möglich sein, Aussagen während des Meetings zu kodieren. Act4teams®-SHORT führt zwar zu gröberen Resultaten, jedoch sind die Ergebnisse beider Kodierschemata vergleichbar [34].

Die Maßnahmen für die Entwicklung von act4teams®-SHORT führten zu Verbesserungen, die im Folgenden genannt werden. Eine Einarbeitung in das Kodierschema ist nicht mehr notwendig und eine stundenlange Analyse des Video-Materials des Meetings entfällt [34]. Somit sinkt der Aufwand im Vergleich zum act4teams® erheblich. Zudem sind keine Probleme zu befürchten, die ein versehentlicher Verstoß gegen die Datenschutzrichtlinien bei audiovisuellen Aufnahmen mit sich bringen könnte. Zusammenfassend lässt sich sagen, dass act4teams®-SHORT die Untersuchung auf eine Weise erleichtert, sodass eine Echtzeit-Analyse in einem Softwareprojekt-Meeting möglich ist [34].

Probleme

Beschreibung

In diese Kategorie gehören alle Aussagen, in denen Probleme diskutiert werden. Dazu gehören das Benennen und Erklären von Probleme, sowie das Vergleichen von Problemen und auch die Angabe von Ursachen und Folgen von Problemen [34, 35]. Das Teilen von Erfahrungen, die sich auf die genannten Probleme beziehen, zählt ebenfalls dazu.

Beispiele

- Wenn wir die Buttons nicht farblich hervorheben, könnten sie übersehen werden.
- Man sieht die Hälfte des Textes nicht mehr.

Lösungen

Beschreibung

Alle Aussagen, die sich mit Lösungen beschäftigen, fallen in die Kategorie *Lösungen*. Dazu gehören unter anderem das Nennen, Sammeln und Ausarbeiten von Lösungsvorschlägen sowie das Vergleichen von Lösungen und Benennen von Vorteilen, Folgen oder Anforderungen an Lösungen [34, 35].

Beispiele

- Wir könnten zuerst Feature 2 und dann Feature 1 implementieren.
- Die Schrift können wir in einer anderen Farbe wählen.

Verknüpfungen und Vernetzungen

Beschreibung

Dazu zählen Aussagen, die Einwände und Bedenken gegen Lösungen ausdrücken oder Probleme mit ihren Lösungsvorschlägen verknüpfen [34, 35].

Beispiele

- Damit das Design ansprechender aussieht, probieren wir diese Farbkombination.
- Aber wenn wir das so machen, könnte Feature 1 kaputt gehen.

Destruktives Verhalten

Beschreibung

Hierzu gehören alle Aussagen, die die Stimmung in einem Meeting negativ beeinflussen können. Neben Tadeln bzw. Lästern fallen Jammern, das Suchen von Schuldigen eines Problems, das Führen von Seitengesprächen und das Ablehnen von Veränderungen [34, 35] in diese Kategorie.

Beispiele

- Das soll alles so bleiben, wie es ist.
- Du kommst immer zu spät.

Proaktives Verhalten

Beschreibung

Aussagen, die Engagement und Interesse an Veränderungen signalisieren, werden mit der Kategorie *Proaktives Verhalten* gekennzeichnet [34, 35]. Dazu zählen ebenfalls Aussagen, die den Unterkategorien *Übernehmen von Verantwortung* und *konkretes Planen von Maßnahmen* zugeordnet werden können.

Beispiele

- Ich wäre dafür, dass wir das so machen.
- Dann setzen wir für kommende Woche ein Interview mit den Kunden an.

Kollegiales Verhalten

Beschreibung

Humorvolle Aussagen sowie Äußerungen, die Wertschätzung ausdrücken, zählen zu dieser Kategorie. Aussagen, die sich positiv auf die Beiträge anderer beziehen, sowie das Ansprechen anderer eventuell auch stillerer Teilnehmer wird ebenfalls als kollegiales Verhalten gewertet [34, 35].

Beispiele

- Das hast du wirklich gut zusammengefasst.
- Tom, was meinst du dazu?

Methodisch-strukturiertes Verhalten

Beschreibung

Mit den Aussagen dieser Kategorie wird die Struktur und Ordnung in einem Meeting aufrechterhalten, um das Ziel des Meetings nicht zu verfehlen. Hierzu gehören Aussagen, die auf Ziele verweisen, Prioritäten setzen oder Aufgaben verteilen [34, 35].

Beispiele

- Diese Aufgabe ist am wichtigsten, deshalb sollte das zuerst erledigt werden.
- Beginnen wir jetzt mit dem nächsten Punkt auf der Agenda.

Informationsweitergabe & Wissenstransfer

Beschreibung

Diese Aussagen dienen dem Austausch von Wissen und Information innerhalb eines Meetings, beispielsweise über Fakten oder Ereignisse [34, 35]. Die Anwendung und der Transfer von Wissen ist auch Teil dieser Kategorie.

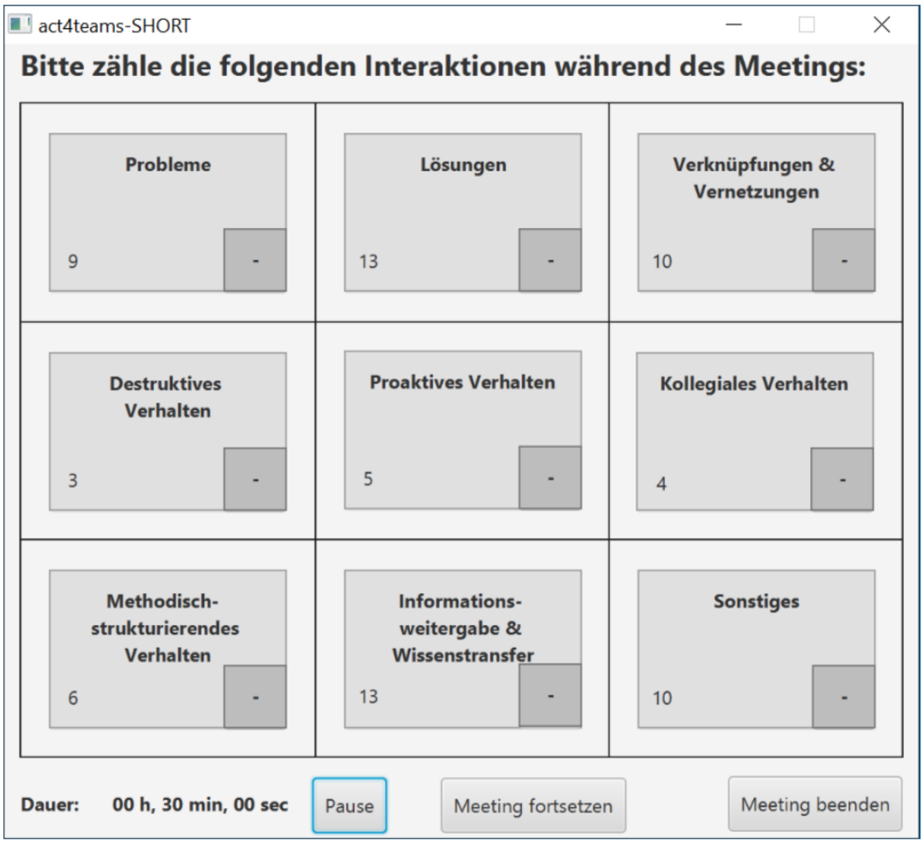
Beispiele

- Die Abnahme wird in zwei Wochen sein.
- Python hat eine Bibliothek, mit der wir das umsetzen können.

Sonstiges
Beschreibung
Hierzu gehören alle Aussagen, die der Beobachter als relevant empfindet, aber nicht sicher zuordnen kann [34, 35]. Wenn möglich, sollte diese Kategorie nur sehr selten eingesetzt werden, da aus den Aussagen dieser Kategorie keine Deutungen für die Analyse des Meetings gewonnen werden können.

2.2.3 Bisherige Vorgehensweise der Interaktionsanalyse

Für eine Analyse der Aussagen in einem Meeting wird ein Beobachter benötigt, der während des Meetings die für ihn relevante Aussagen in eine der neun Kategorien einordnet. Während der iterativen Entwicklung des act4teams®-SHORT entstand eine Java-Anwendung zum Kodieren von Aussagen, welches neun beschriftete Felder besitzt, mit denen die Aussagen gezählt werden können [20].



Bitte zähle die folgenden Interaktionen während des Meetings:

Probleme 9	Lösungen 13	Verknüpfungen & Vernetzungen 10
Destruktives Verhalten 3	Proaktives Verhalten 5	Kollegiales Verhalten 4
Methodisch-strukturierendes Verhalten 6	Informations-weitergabe & Wissenstransfer 13	Sonstiges 10

Dauer: 00 h, 30 min, 00 sec

Abbildung 2.2: Oberfläche des act4teams®-SHORT-Java-Anwendung [20]

Die Oberfläche des Tools wird in Abbildung 2.2 dargestellt. Jede Kategorie des act4teams®-SHORT Kodierschemas besitzt ein Feld mit einem Zähler. Sobald der Beobachter eine Interaktion kodieren möchte, drückt er den hellgrauen großen Button der entsprechenden Kategorie. Der Zähler wird inkrementiert und die Kategorie wird mitsamt Zeitstempel programmintern gespeichert [20]. Wenn fälschlicherweise einen Button gedrückt wird, kann dies wieder rückgängig gemacht werden, indem der kleine dunkelgraue Button des entsprechenden Feldes gedrückt wird. Der Zähler wird dekrementiert und der Zeitstempel mit der dazugehörigen Kategorie des letzten Klicks wird entfernt. Wenn das Meeting vorbei ist, kann der Nutzer mit dem *Meeting beenden*-Button die aktuelle Sitzung beenden. Es wird eine csv-Datei generiert, die eine Übersicht mit allen registrierten Kategorie-Zeitstempel-Paaren enthält, sowie eine Darstellung der neun Kategorien mitsamt der Gesamtanzahl der Vorkommnissen [20].

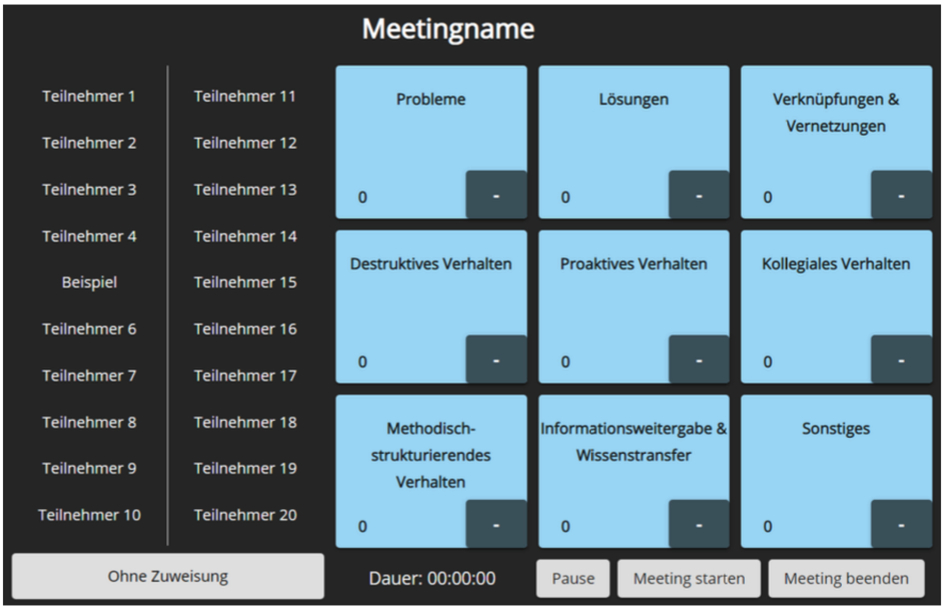


Abbildung 2.3: Oberfläche der act4teams®-SHORT-Webanwendung [13]

Für act4teams®-SHORT wurde kürzlich von Enns [13] eine Webanwendung entwickelt, in der auch die Teilnehmer eines Meetings angegeben werden können. Auf diese Weise können die Aussagen auch den Teilnehmern zugewiesen werden. Die Abbildung 2.3 zeigt ein Screenshot der Ansicht zum Zählen der Aussagen. Auf der linken Seite befindet sich eine Liste mit den Teilnehmern, die ausgewählt werden können, um ihnen eine Aktion zuzuweisen. Es kann auch eine Aktion gezählt werden, ohne sie einer Person zuzuordnen. Dafür wird der Button *Ohne Zuweisung* verwendet. Die Buttons auf der rechten Seite haben die gleiche Funktion wie die

Buttons aus der Java-Anwendung von Haak. In der Webanwendung ist es möglich ein Meetingtranskript als *csv*-Datei einzulesen, um eine Analyse durchzuführen. Nach dem Beenden des Meetings werden die Ergebnisse in mehreren Ansichten grafisch dargestellt. Beispielsweise können die pro Kategorie gezählten Aktionen, die pro Person gezählten Aussagen oder die Kombination aus beidem betrachtet werden.

2.3 Definition destruktiver Äußerungen

Das Ziel dieser Arbeit ist es, destruktive Äußerungen automatisch zu klassifizieren. Dies bedeutet, dass entweder eine Tonspur oder ein Transkript als Ausgangsmaterial verwendet werden muss. Somit kann Verhalten, welches nur visuell wahrgenommen kann, nicht berücksichtigt werden. Die Kategorie *Seitengespräche* ist ein Beispiel dafür, denn das Schreiben einer E-Mail oder einer SMS kann aus einer Tonspur nicht herausgelesen werden. Daher ist es unbedingt notwendig, destruktive Aussagen zunächst zu definieren und zu entscheiden, welche Art von destruktiven Aussagen aus einer Tonspur oder Transkript automatisch klassifiziert werden können. Laut Klünder [34] gehören zur Kategorie *Destruktives Verhalten* aus act4teams®-SHORT die Unterkategorien Tadel/Abwertung, Schuldigensuche, Jammern, Kein Interesse an Veränderungen, Seitengespräche aus act4teams®. Unterbrechungen gehören auch zu unkollegialen Interaktionen nach act4teams® und führen „oft zu unzufriedenen Entwicklern“ [34]. Jemandem das Wort abzuschneiden kann als despektierlich empfunden werden [68] und könnte zu Unmut zwischen den Teilnehmern führen.

Im Rahmen dieser Arbeit werden folgende Kategorien als destruktive Äußerungen gewertet:

Tadel/Abwertung

Beschreibung

Dazu gehören Aussagen, die getätigt werden, um andere zu beleidigen, abzuwerten oder zurechtzuweisen [34]. Diese Sätze können auch als grob oder schroff empfunden werden.

Beispiele

- Das Design ist echt grottenhässlich.
- Warum liest du nicht deine dummen E-Mails?
- Was du gemacht hast, ist einfach Mist.

Jammern

Beschreibung

Diese Art von Äußerungen betont den negativen Ist-Zustand oder heben die Unveränderlichkeit einer Situation hervor [34].

Beispiele

- Es ist doch immer das Gleiche.
- Jedes Mal gibt es Stress.
- Und schon wieder müssen wir warten.

Schuldigensuche

Beschreibung

Aussagen dieser Unterkategorie sollen jemandem die Schuld für Probleme geben und finden dabei auf persönlicher Ebene statt [34].

Beispiele

- Wir müssen immer später anfangen, weil du niemals pünktlich bist.
- Deinetwegen müssen wir Extra-Arbeit erledigen.
- Das ist alles deine Schuld.

Kein Interesse an Veränderungen

Beschreibung

Mit diesen Äußerungen drückt jemand aus, dass er keine Veränderung wünscht und versucht auch Gründe dafür zu finden [34].

Beispiele

- Wir machen das auf keinen Fall.
- Wollen wir uns wirklich diesen unnötigen Aufwand machen?
- Das ist doch viel zu schwierig.

Unterbrechungen

Beschreibung

Mit diesem Verhalten wird jemandem das Wort abgeschnitten [34]. Das bedeutet, dass der Vorredner seinen Satz nicht beenden kann. Die Äußerungen derer, die den Vorredner nicht aussprechen lassen, zeigen keine Besonderheit oder Gemeinsamkeit, denn sie können sowohl positiver als auch negativer Natur sein. Diese Kategorie kann also nur durch unvollständige Sätze des Vorredners identifiziert werden.

Beispiele

- Wenn er nicht ...
- Entweder schreibst ...
- Ich glaube, ich äh ...

2.4 Sprache

Bevor die Aussagen aus einem Meeting analysiert werden, werden zunächst Besonderheiten gesprochener Aussagen in Diskussionsrunden betrachtet. In der Linguistik wird Sprache in zwei Modalitäten aufgeteilt. Die gesprochene und geschriebene Sprache unterscheiden sich auf den Ebenen Grad der Planung, erwartete Maß an Formalität in der Situation, Art und Anzahl der Zuhörer und Thema [58].

2.4.1 Geschriebene Sprache

Geschriebene Texte sind eher durchdacht, da der Schreiber die Möglichkeit hat, seine Sätze solange zu verbessern, bis sie seinen Ansprüchen und formellen Standards genügen [58]. Der Schreiber sieht für gewöhnlich sein Publikum nicht, wodurch sie ihm kein sofortiges Feedback geben können. Die Leser geschriebener Werke sind zumeist von größerer Anzahl. Daher ist es möglich, dass Schreiber und Leser einander sogar völlig unbekannt sind [58]. Im Gegensatz zu gesprochenen Aussagen haben geschriebene Texte ein vielfältigeres Vokabular [8]. Die Konsequenz davon ist die Neigung, längere und schwierigerer Wörter zu verwenden. Häufig werden jedoch einfachere Satzstrukturen genutzt [8]. Geschriebene Texte haben den Vorteil, stets nach Sätzen gegliedert werden zu können [11]. Bestimmte Interpunktionen, wie Punkt, Ausrufezeichen oder Fragezeichen, geben das Ende eines Satzes an.

2.4.2 Gesprochene Sprache

Gesprochene Sprache findet vorwiegend mit einer überschaubaren Anzahl an Zuhörern statt, die dem Sprecher in den meisten Fällen bekannt sind [58]. Die Zuhörer haben sogar die Möglichkeit, mit dem Sprecher zu interagieren und ihm Feedback sowohl verbaler als auch non-verbaler Natur zu geben. Der Sprecher verwendet dabei häufig eine eher informelle Sprache und die Aussagen sind ungeplant [58]. Auch wenn vorher Notizen gemacht wurden, kommt es vor, dass der Sprecher sich für andere Formulierungen entscheidet. Gesprochene Sprache bedient sich eines vergleichsweise einfacheren Wortschatzes [8]. Es ist nicht ungewöhnlich, dass man sich wiederholt, um sich präziser auszudrücken. Bei gesprochenen Aussagen lassen sich häufiger komplizierte Satzstrukturen beobachten, wie beispielsweise mehrfach verschachtelte Sätze [8]. Dies könnte zur Folge haben, dass man den Faden verliert und seinen Gedankengang unterbricht, um erneut anzusetzen.

Ein Sprechstrom wird in Äußerungen aufgeteilt [9]. Solch eine Segmentierung ist nicht so trivial wie das Aufteilen eines Textes in Sätzen, da es keine Interpunktion zur Orientierung gibt. Das Konzept von Crookes [9] besagt, dass eine Äußerung von einer anderen getrennt wird, wenn mindestens eine der drei folgenden Merkmale zutreffen.

- Inhalt: Wenn der Sprechstrom einen veränderten Inhalt aufweist, stellt dies eine semantische Abgrenzung dar.
- Intonation: Eine fallende oder steigende Intonation signalisiert das Ende einer Einheit. Äußerungen werden mit einer fallenden Intonation und Fragen werden mit einer steigenden Intonation beendet.
- Pause: Eine Pause von ca. eine bis acht Sekunden zeigt das Ende einer Äußerung an.

Dürscheid und Schneider [11] ergänzen diese Liste um die Eceteraformel („...und so“).

Ein Teilgebiet der gesprochenen Sprache ist die von Murray [48] beschriebene *Meeting Speech*. Hierbei handelt es sich um eine rein spontane Rede, die häufig Diskursmarker aufweist, wie „Ähm“ oder „Äh“ [48]. Eine Ausnahme bilden Folienpräsentationen, doch auch in diesem Fall werden keine kompletten Sätze abgelesen. An der *Meeting Speech* sind mehrere Sprecher beteiligt und daher kann es zu Sprecherüberschneidungen kommen [48]. Dabei beschreibt ein *Turn* den Gesprächsbeitrag eines Teilnehmers, der an der Reihe ist.

Zudem wird festgestellt, dass die Sprecher dazu neigen, nicht in vollständigen oder grammatikalisch korrekten Sätzen zu sprechen [48]. Ellipsen und Parenthesen gehören zu solchen Phänomenen. Parenthesen sind meist mit Gedankenstrichen eingeschobenen Sätze, die auf Pronomen oder Ähnliches verzichten können (Ich habe - wenn ich mich recht erinnere - den Herd ausgemacht) [11]. In einem geschriebenen Satz würde der eingeschobene Teilsatz vor oder hinter dem Hauptsatz angehängt werden, denn dies wäre grammatikalisch korrekt. Bei der Verwendung von Ellipsen, können einige Satzteile ausgelassen werden, die sich aber vom Zuhörer wieder rekonstruieren lassen [11]. Im folgenden Unterabschnitt 2.4.3 werden Ellipsen näher betrachtet.

2.4.3 Besonderheiten

Mittlerweile können geschriebene Texte auch die gleichen Eigenschaften aufweisen wie gesprochene Sprache [11]. Dies ist häufig in Nachrichten über Chats oder Kommentaren in den sozialen Medien zu beobachten, in denen meist informelle Sprache bzw. Umgangssprache verwendet wird. Theoretisch besteht die Möglichkeit, sich Zeit zu nehmen und Sätze auszuformulieren, doch durch den Dialogcharakter eines Chats könnte der Druck entstehen, schnell antworten zu müssen, um seinen *Turn* (Gesprächsbeitrag) im Dialog nicht zu verlieren. Aus diesem Grund und auch aus Bequemlichkeitsgründen liegt die Vermutung liegt nahe, dass diese Sätze meist den Eindruck machen als wurden gesprochene Sätze unmittelbar aufgeschrieben. Trotz dessen gibt es eine Gegebenheit in der gesprochenen Sprache, die sich in der geschriebenen Sprache nicht beobachten lässt. Egal, ob Briefe, E-Mails, Protokolle oder Nachrichten aus Instant Messengern, es fällt auf, dass niemandem das Wort abgeschnitten werden kann. Selbst wenn zwei Chatteilnehmer jeweils eine Nachricht zur gleichen Zeit abschicken, können beide Nachrichten gelesen werden. Man beobachtet, dass in einem Meeting zwar zwei Teilnehmer gleichzeitig sprechen können, doch entweder könnten beide Gesprächsbeiträge unverständlich sein oder einer der beiden könnte darauf verzichten, seinen Beitrag zu vollenden. In Präsentationen haben zunächst nur die vortragenden Personen das Wort. Doch auch hier kann der Vortragende durch andere Teilnehmer unterbrochen werden. Daher liegt die Vermutung nahe, dass Unterbrechungen nur in gesprochener Sprache vorkommen können.

Für einen vollständigen deutschen Satz wird mindestens ein Subjekt und ein Prädikat benötigt [11]. Es gibt jedoch Ausnahmen, die von Hoffmann [23] als „Standard-Problemfälle“ bezeichnet werden. Dazu gehören beispielsweise Parenthesen, Ellipsen oder Imperative. Wie zuvor angegeben, können bei Ellipsen auf Satzteile verzichtet werden. Dabei könnte das Prädikat weggelassen, wie im altbekannten Spruch: „Erst [kommt] die Arbeit, dann [kommt] das Vergnügen.“ Es können aber auch größere Abschnitte ausgelassen werden. Da das Sprichwort „Wer anderen eine Grube gräbt[, fällt selbst hinein].“ fast jedem bekannt ist, wird häufig der zweite Teil des Satzes weggelassen. Der Sinn des Sprichwortes würde dadurch nicht verloren gehen, wenn der Adressat es bereits kennt. Ellipsen können auch kontextabhängig sein. Wenn man sich auf die Aussage des Vorredners bezieht, können ebenfalls Teile eines Satzes weggelassen werden, damit bereits Gesagtes nicht wiederholt wird.

Beispiel 2.1

A: Ich bleibe heute länger.

B: Ich [bleibe] auch.

Die Aussage von B aus Beispiel 2.1 ist kein vollständiger Satz im engen Sinne, da an dieser Stelle das Prädikat fehlt. Ein weiteres Beispiel wäre der Satz: „Du bearbeitest Feature 1 und ich Feature 2.“. Im zweiten Teil des Hauptsatzes wird das Prädikat weggelassen, um eine Wiederholung des Wortes „bearbeiten“ zu vermeiden, obwohl es in einer anderen Konjugation verwendet worden wäre.

Es ist auch möglich, in der Umgangssprache das Subjekt wegzulassen. Der Beispielsatz „[Das] passt schon.“ zeigt, dass hier auf das Subjekt „Das“ meist verzichtet wird. In der deutschen Sprache ist es auch möglich, bei der Verwendung eines Modalverbs das Hauptverb wegzulassen [71]. Normalerweise sind Modalverben auf ein Hauptverb angewiesen und bilden nicht alleine das Prädikat. Im Beispielsatz „Ich muss zur Schule [gehen].“ ist „muss “ das Modalverb und „gehen “ das Hauptverb, das ausgelassen werden kann.

All diese Besonderheiten zeigen, dass die Definition eines vollständigen Satzes nicht trivial ist und dass dies bei der Erkennung von unvollständigen Sätzen berücksichtigt werden muss.

2.5 Text Mining

Text Mining basiert auf den Methoden des Data Mining, jedoch unterscheidet sich die Beschaffenheit der Daten [22]. Während Data Mining auf strukturierte Daten angewendet wird, werden beim Text Mining Textdaten analysiert und benötigt daher andere Verarbeitungsmethoden, wie zum Beispiel *Natural Linguistic Processing* (kurz: NLP) [22]. Die folgenden Unterkapitel geben einen Überblick über den Prozess und angewendete Methoden des Text Mining.

2.5.1 Prozess

Text Mining durchläuft eine Reihe von Schritten, die in Abbildung 2.4 [22] abgebildet sind. Nachdem die Aufgabe definiert wurde, die mit Hilfe des Text Mining erfüllt werden soll, wird ein Datensatz zum Trainieren des Modells benötigt. Im Prozessschritt *Dokumentselektion* werden Dokumente ausgewählt, die für die Aufgabe relevant sind. Je nach definierter Aufgabe können statt Dokumente auch andere Einheiten, wie beispielsweise Absätze, Twitter-Threads oder Äußerungen von Personen, verwendet werden [43].

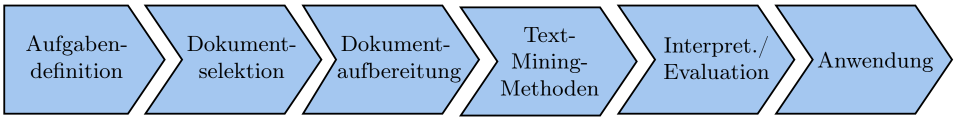


Abbildung 2.4: Schritte eines Text Mining Prozesses [22]

Zusammen mit der Vorverarbeitung der Texteinheiten gehört die Merkmalsextraktion zu den wichtigen Schritten im Text Mining, da sie einen Einfluss auf die Klassifikation hat [2]. Dafür werden in der *Dokumentaufbereitung* die Texte mit Hilfe von NLP verarbeitet [22]. Als Merkmale können beispielsweise repräsentative Terme oder statistische bzw. quantitative Werte eines Dokuments verwendet werden. Während statistische Metriken bereits eine numerische Darstellung besitzen, müssen extrahierte Terme auch in eine solche Form gebracht werden. Eine gängige Praxis ist das Erstellen eines Vektors, der auch als Term-Dokument-Matrix dargestellt werden kann [22]. Die Werte in diesem Vektor können die Häufigkeit des Vorkommens oder ein gewichtetes Maß (Tf-idf-Maß) eines Terms in einem Dokument ausdrücken [22]. Dieser Vektor repräsentiert die Struktur eines Dokuments, welches als Eingabe für die *Text Mining Methode* im nächsten Schritt zum Einsatz kommt. Klassifikation, Clustering, Informationsexktraktion und Visualisierung sind Methoden des Data Mining, die auf Texte angewendet werden können [25]. Während bei der Klassifikation die Dokumente bestimmten Kategorien zugeordnet werden, findet beim Clustering eine Gruppierung ähnlicher Dokumente statt. Die Informationsextraktion bezeichnet die Selektion relevanter Informationen aus Texteinheiten und die Zuordnung von Attributen [25, 27]. Es besteht auch die Möglichkeit Texteinheiten visuell darzustellen, um ein besseres Verständnis für eine große Datensammlung zu bekommen [25, 27]. Nach der Durchführung einer *Text Mining Methode* werden die Ergebnisse ausgewertet und interpretiert. Dieser Prozess kann auch iterativ durchlaufen werden [22]. Das bedeutet, wenn die *Interpretation und Evaluation* Fehler und Schwächen identifiziert, können die vorherigen Prozessschritte mit Modifikationen erneut durchgeführt werden. Erst wenn das Resultat der Evaluation den Erwartungen entspricht, können die Ergebnisse des Text Mining Prozesses in Anwendungen eingesetzt werden [22].

2.5.2 Natural Language Processing

Da Texte noch keine Form besitzen, die für die Text Mining Methoden verwendet werden können, müssen sie aufbereitet werden [22, 25]. Die Vorverarbeitung (engl. Pre-Processing) der Texteinheiten bedient sich verschiedener Techniken zur Strukturierung von Texten und Extraktion von Metriken.

Normalisierung: Der Text wird vollständig in Kleinbuchstaben formatiert [64]. Rechtschreibfehler werden ersetzt und Sonderzeichen, numerische Angaben oder Datumsangaben entfernt. Sogenannte Stoppwörter (engl. stopwords) sollten auch entfernt werden, da sie wenig semantische Bedeutung haben [43]. Zu den Stoppwörtern zählen beispielsweise Artikel, Pronomen, Präpositionen. Jedoch sollte vorher die Stoppwortliste überprüft werden, um eine Entfernung der Terme zu verhindern, die für die Text Mining Methoden relevant sind [64]. Andernfalls könnte dies zu verfälschten Ergebnisse führen.

Du hast das Design echt zerstört.
design echt zerstoert

Beispiel: Normalisierung

Tokenisierung: Texte werden in sogenannte *Tokens* aufgeteilt, wobei eine Worteinheit einen *Token* bildet [43]. Whitespaces oder Interpunktionen trennen Wörter voneinander und können somit als Trennzeichen fungieren.

Du hast das Design echt zerstört.
[‘Du’ , ‘hast’, ‘das’ , ‘Design’ , ‘echt’ , ‘zerstört’, ‘.’]

Beispiel: Tokenisierung

Stemming/Lemmatisierung: Tokens werden auf ihren Wortstamm zurückgeführt, um beispielsweise konjugierte Formen eines Verbs des gleichen Stamms nicht als verschiedene Wörter zu interpretieren [43]. Stemming und Lemmatisierung, auch lexikonbasiertes Stemming genannt, verfolgen unterschiedliche Ansätze, um den Wortstamm zu gewinnen. Es existieren mehrere Stemming-Algorithmen, von denen einer die Silben eines Wortes solange verkürzt bis eine minimale Anzahl an Silben übrig bleibt. Auf diese Weise wird eine Form ohne Präfixe und Suffixe gewonnen [52]. So würde beispielsweise aus *gegangen gegang* entstehen. Bei der Lemmatisierung wird die Grundform eines Wortes gesucht [25, 43]. Die Grundform eines Verbs entspricht dem Infinitiv Präsens Aktiv und die Grundform eines Nomens bildet der Nominativ Singular. Die Grundform des Verbs *gegangen* wäre also *gehen.* Es kann durchaus vorkommen, dass der Wortstamm, der beim Stemming gewonnen wird, nicht mit der Grundform des Wortes übereinstimmt [22, 43].

Du hast das Design echt zerstört.

Du haben das Design echt zerstören.

Beispiel: Lemmatisierung

Syntaktische Analyse: Texteinheiten können mit Hilfe von Part-of-Speech-Tagging (kurz POS-Tagging) untersucht werden [22, 25]. Dabei werden Tokens mit ihrer Wortart, wie Nomen oder Verb, markiert. Beim Parsing bekommen die Tokens Tags mit ihrer Stellung im Satz, wie zum Beispiel Subjekt oder Objekt [22, 25]. Für beide Vorgehensweisen ist eine korrekte Rechtschreibung sowie vollständige Interpunktion erforderlich. Andernfalls könnte aus dem Wort *fragen* nicht erkannt werden, ob es sich um das Substantiv plural des Nomens oder um den Infinitiv Präsens Aktiv des Verbs handelt. Falsch gesetzte Tags könnten die Folge sein.

POS-Tags	Personal-pronomen	Hilfverb	Artikel	Nomen
Text	*Du*	*hast*	*das*	*Design*
Parsing-Tags	Subjekt	Prädikat	Akkusativobjekt	

POS-Tags	Adverb	Verb	Interpunktion
Text	*echt*	*zerstört*	.
Parsing-Tags	Modal-adverbial	Prädikat	

Beispiel: Syntaktische Analyse

Semantische Analyse: Es wird untersucht, ob Terme inhaltlich zusammenhängen. Eine Möglichkeit der Analyse ist die Kookkurrenz von Termen zu betrachten [43]. Bei häufigem gemeinsamen Auftreten von Termen, wird angenommen, dass sie eine inhaltliche Einheit bilden.

Vektorisierung: In der linguistische Vorverarbeitung können extrahierte relevanten Terme vektorisiert werden, um Texteinheiten zu strukturieren und zu repräsentieren [22]. Bei der Vektorisierung kann entweder das Vorkommen dieser Terme gezählt werden oder ein gewichtetes Maß verwendet werden [22]. Vektoren, die lediglich das Auftreten von Wörtern betrachten, werden auch Bag-of-Words genannt. Alternativ kann ein gewichtetes Maß, auch als Tf-idf-Maß bekannt, berechnet werden, das die Relevanz eines Wortes in einer Texteinheit wiedergibt [67].

Die dargestellte Reihenfolge entspricht nicht der tatsächlichen Reihenfolge in der Anwendung. Sollte eine syntaktische Analyse angestrebt werden, sollten nur unbearbeitete Texte verwendet werden, um Fehler beim Tagging aufgrund einer inkorrekten Rechtschreibung zu vermeiden.

2.5.3 Methoden der Klassifikation

Neben Clustering, Informationsextraktion und Visualisierung ist Klassifikation eine Methode, die im Text Mining eingesetzt wird. Zwei unterschiedliche Ansätze, um Texteinheiten einer Kategorie zuzuordnen, sind maschinelle Lernverfahren und Pattern Matching [37].

Beim **maschinellen Lernen** werden auf Grundlage eines bekannten Datensatzes Muster und Zusammenhänge abgeleitet, sodass unbekannte Daten einer Klasse zugeordnet werden können [37, 65]. Man spricht von überwachtem Lernen, wenn der Ausgangsdatensatz bereits klassifizierte Texteinheiten besitzt [70]. Zur Auswahl stehen Klassifikatoren wie Support Vector Machine oder Naive Bayes, die verschiedene Algorithmen verwenden. Das bei einer Trainingsphase generierte Modell ist dann in der Lage eine Vorhersage für unbekannte Daten bezüglich ihrer Klassenzugehörigkeit zu treffen [70]. Die folgende Übersicht behandelt kurz einige zur Auswahl stehende Klassifikatoren. In Abschnitt 4.2 werden drei Klassifikatoren ausgewählt, die am besten geeignet sind.

Gaussian Naive Bayes (GNB): Der probabilistische Algorithmus berechnet mit Hilfe des Satzes von Bayes die Wahrscheinlichkeit einer Klassenzugehörigkeit für jedes Objekt. Außerdem geht man davon aus, dass eine Normalverteilung der Merkmale vorliegt [17]. Der Klassifikator entscheidet sich für die Klasse mit der höchsten Wahrscheinlichkeit.

Random-Forest-Classifier (RF): Bei dem RF-Algorithmus werden zunächst durch das Bootstrap-Verfahren mehrere Stichproben aus den Trainingsdaten erstellt [17]. Für jedes dieser Stichproben werden Entscheidungsbäume gebildet, die eine Klasse vorhersagen. Durch einen Mehrheitsentscheid bezüglich der vorhergesagten Klassen wird eine endgültige Entscheidung getroffen [17].

Support-Vector-Machine (SVM): Durch den SVM-Algorithmus wird jedes Objekt in einem n-dimensionalen Raum abgebildet, in dem die Datenpunkte verschiedener Klassen durch einen möglichst breiten Bereich getrennt werden sollen [17]. Auf diese Weise entstehen Grenzen zwischen den Klassen. Unbekannte Objekte werden ebenfalls in diesen Raum abgebildet, um die Klassenzugehörigkeit festzustellen [17].

AdaBoost-Classifier (AB): AdaBoost (adaptives Boosting) versucht mit einem Ensemble aus schwachen Klassifizierern einen starken Klassifizierer zu formen [17, 70]. In jeder Iteration wird ein schwacher Klassifizierer dem Ensemble hinzugefügt und die Gewichtung aller Klassifizierer angepasst, um die Klassifikation zu verbessern. Sobald keine Verbesserung mehr feststellbar ist oder die Höchstanzahl an Klassifizierern, die vom Benutzer festgelegt werden kann, erreicht ist, ist das Modell vollendet [17, 70]. Die endgültige Klassifikation wird durch einen gewichteten Mittelwert aus den Vorhersagen gebildet.

Gradient-Boosting-Classifier (GB): Der GB-Algorithmus hat eine ähnliche Funktionsweise wie AdaBoost [17, 19]. Der Unterschied liegt in der Anpassung, die in jeder Iteration vorgenommen wird. Beim Gradient-Boosting wird keine neue Gewichtung vorgenommen, sondern die Fehlvorhersagen des vorherigen Klassifikators reduziert. Dies wird erreicht, indem das Minimum der Verlustfunktion mit Hilfe des Gradientenabstiegs gesucht wird [17, 19].

Decision-Tree-Classifier (DT): Der DT-Algorithmus erstellt einen Entscheidungsbaum, der bei der Klassifikation von oben nach unten durchlaufen wird [17]. Die Wurzel und inneren Knoten stellen die Merkmale dar und die Blattknoten repräsentieren die Klassen. Nach der Auswahl einer Metrik können Entscheidungen getroffen werden beim Durchlaufen des Baums. Der Blattknoten am Ende des zurückgelegten Weges entscheidet über die Klassenzuordnung [17].

Pattern Matching ist eine weitere Methode zur Klassifikation, die mit vordefinierten Regeln bzw. Mustern (Pattern) arbeitet [70]. Zunächst werden manuell Muster erstellt und in einer Bibliothek gespeichert. Ein Matcher überprüft mit Hilfe dieser Bibliothek das Vorkommen von bekannten Mustern, beispielsweise einer Wortsequenz, in einem Datensatz [70]. In Abbildung 2.5 wird das beschriebene Vorgehen grafisch dargestellt. Anhand der Ergebnisse können Texteinheiten klassifiziert werden.

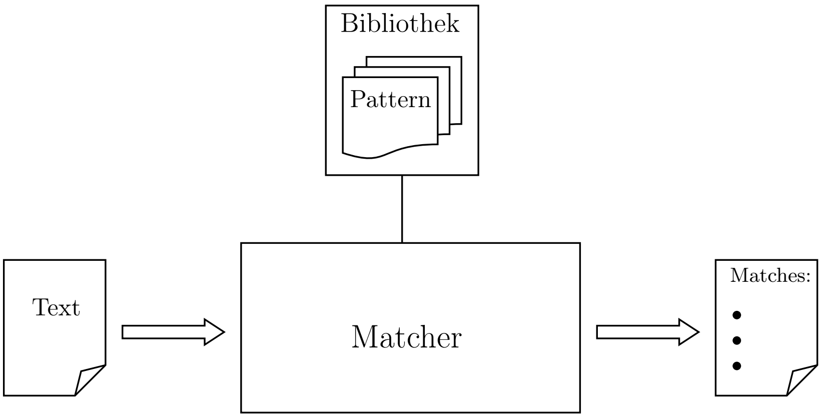


Abbildung 2.5: Funktionsweise des Pattern Matching

Es ist nicht zwingend notwendig sich für einen der beiden Vorgehensweisen zu entscheiden. Es existieren wissenschaftliche Publikationen über **hybride Methoden** in der Sentiment Analyse [45, 55], die beispielsweise maschinelle Lernmethoden mit Pattern Matching kombinieren. Hybride Methoden kombinieren so die Stärken beider Vorgehensweisen, um die Gesamtperformanz zu steigern.

Kapitel 3

Verwandte Arbeiten

In diesem Kapitel werden wissenschaftliche Arbeiten vorgestellt, die sich mit der Klassifizierung von geschriebenen oder gesprochenen Äußerungen beschäftigen. Um zu zeigen, dass die Aufgabenstellung dieser Arbeit bisher noch nicht untersucht wurde, wird diese Arbeit von den hier vorgestellten Arbeiten abgegrenzt. Dafür werden die Aspekte genannt, in denen sich die Aufgabenstellungen der Arbeiten unterscheiden.

Pang et al. [51] klassifizierten Filmbewertungen in ihrer wissenschaftlichen Arbeit mit Hilfe maschineller Lernverfahren. Drei Klassifikatoren sollten die Bewertungen als positiv oder negativ kennzeichnen. Dabei wurde untersucht, welche Merkmale am besten für die Sentiment Klassifikation geeignet sind. Als Merkmale wurden unter anderem Unigramme, Bigramme, nur Adjektive oder POS-Tags verwenden. Diese Merkmale wurden zum Teil auch miteinander kombiniert und die Performanz der Klassifikatoren verglichen.

Calefato et al. [7] haben festgestellt, dass die meisten Sentiment Analysis Tools aus nicht technischen Domänen stammen und dadurch Texte aus dem Bereich des Software Engineering schlechter klassifizieren können. Aus diesem Grund haben sie Senti4SD entwickelt, das mit einem Goldstandard aus Stack Overflow trainiert und validiert wurde. Für die Klassifikation werden lexikonbasierte, schlüsselwortbasierte und semantische Merkmale miteinander kombiniert. Ein SVM-Klassifikator teilt die Texteinheiten in die Polaritätsklassen *positiv*, *neutral* und *negativ* ein.

Lin et al. [42] wollten ein Empfehlungssystem für Softwarebibliotheken anhand von Meinungen von Entwicklern erstellen. Da im Bereich des Software Engineering die Erfahrung gemacht wurde, dass domänenspezifische Sentiment Analyse Tools eine bessere Leistung erzielen, sollte auch hier ein Datensatz aus den Meinungen der Entwickler auf Stack Overflow erstellt werden. Das Ergebnis des Trainings des Sentiment Analyse Tools war jedoch nicht zufriedenstellend. Man vermutet als Begründung für die Ergebnisse unter anderem, dass auf Stack Overflow technische Details diskutiert werden und nicht übermäßig Emotionen zum Ausdruck gebracht werden. Lin et al. sind der Meinung, dass das Gewinnen von Meinungen im Software Engineering noch nicht ausgereift genug ist. Sie hoffen weiterhin mit der Veröffentlichung ihrer wissenschaftlichen Arbeit, dass Forscher des Software Engineering von den Erkenntnissen profitieren können.

In der Dissertation von Klünder [34] wurde die Methode zur Interaktionsanalyse in Meetings act4teams ® von Kauffeld und Lehmann-Willenbrock [32] weiterentwickelt. Durch die Anwendung von act4teams® auf Meetings studentischer Softwareprojekte wurde eine Reduktion des Kodierschemas unter dem Namen act4teams®-SHORT erstellt, um eine Echtzeitanwendung der Analyse in Softwareprojekt-Meetings durch ungeschulte Beobachter zu ermöglichen. Haak [20] entwickelte zwei Tutorialkonzepte für das vorhandene act4teams®-SHORT-Tool. Diese Tutorials sollen unerfahrene Beobachter bei anfänglichen Schwierigkeiten in der Anwendung des Tools unterstützen.

Horstmann [24] führte in seiner Arbeit eine computer-gestützte Analyse des Kommunikationsverhaltens in Entwicklerteams durch. Mit Hilfe maschineller Lernverfahren und evolutionärer Algorithmen zur Optimierung des Lernverfahrens wurden textuelle Einheiten als positive, neutrale oder negative Nachrichten klassifiziert. Dieses Konzept wurde von Meyer [47] aufgegriffen und Möglichkeiten zur Verbesserungen von evolutionären Algorithmen gesucht. Basierend auf diesen Ergebnissen analysierte Herrmann [21] die mündliche Kommunikation in Meetings von Softwareprojekten. Das implementierte Tool ist in der Lage, Sprache über das Mikrofon oder eine Audiospur zu erkennen und ein Transkript zu erstellen. Es besteht zusätzlich die Möglichkeit, bereits transkribierte Dialoge aus Meetings zu verarbeiten. Je nach gemessener Stimmung der Aussagen erfolgt eine automatisierte Klassifikation in die Polaritätsklassen *positiv*, *neutral* und *negativ*. Zukünftige Arbeiten sollen statt der drei Polaritätsklassen die act4teams®-SHORT-Kategorien für die Klassifikation von Äußerungen verwenden. Diese Arbeit bildet den ersten Schritt in die Richtung dieses Ziels, indem Äußerungen aus Softwareprojekt-Meetings automatisiert als destruktiv oder nicht destruktiv klassifiziert werden. Dabei bilden die Kategorien des act4teams®-SHORT-Kodierschema die Basis für die Definition einer destruktiven Äußerung.

Die Arbeit von Kristo [39] beschäftigte sich mit der computergestützten Analyse von Videokommentaren für die Anforderungsanalyse. Dafür werden geschriebene Kommentare unter Vision Videos auf der Plattform YouTube extrahiert und zunächst grob in Spam und Ham klassifiziert, um relevante Kommentare für die Anforderungsanalyse zu identifizieren. Durch die Analyse der Texteigenschaften wurden eigene Klassen entwickelt, um Anforderungen aus Kommentaren zu extrahieren. Folglich fokussiert sich die Analyse der Kommentare auf den Bereich der Anforderungsanalyse und die Extraktion nützlicher Informationen. In dieser Arbeit hingegen werden Meetings auf destruktive Äußerungen, deren Definition sich vom act4teams- und act4teams®-SHORT-Kodierschema ableiten lässt, analysiert. Zudem werden Texteigenschaften geschriebener Kommentare, die sich von den Texteigenschaften gesprochener Äußerungen unterscheiden, untersucht.

Powelske [53] analysierte die Stimmung in Open-Source-Projekten anhand textbasierter Kommunikation. In Open-Source-Projekten kann Kommunikation beispielsweise im Kommentarbereich von Versionsverwaltungssystemen (engl. Repository) stattfinden. Nach einem Vergleich vorhandener Tools für die Sentiment Analyse größerer Datenmengen, wurde Senti4SD als das am besten geeignete Tool identifiziert. Das Konzept basiert auf einer Kombination aus Repository Mining und dem Tool Senti4SD. Der abgeleitete Emotionalitäts-Score soll Aufschluss über die Stimmung in Open-Source-Projekten geben.

Um Zusammenhänge zwischen Interaktionen und Stimmungen von Softwareprojekt-Meetings zu identifizieren, hat Schiller [60] im Rahmen seiner Arbeit Datensätze mit Hilfe der Sentiment Analyse in die Polaritätsklassen *positiv*, *neutral* und *negativ* eingeteilt. Jede Interaktion gemäß act4teams®-SHORT wurde auf das Verhältnis der drei Polaritätsklassen untersucht. Anhand dieser Information konnten festgestellt werden, ob und welche Zusammenhänge es zwischen Interaktionen und Stimmungen gibt.

Diese Arbeit unterscheidet sich von den zuvor genannten Arbeiten in mehreren Punkten. Die meisten Arbeiten beschäftigen sich mit textuellen Einheiten für die Klassifikation und betrachten daher andere Texteigenschaften als diese Arbeit. Automatisierte Klassifikationen und computergestützte Analysen, die in den bisherigen Arbeiten vorgenommen wurden, beschränkten sich auf die drei Polariätsklassen. Diese Arbeit hingegen gehört zu den Ersten, die ein Konzept verwenden, um Äußerungen in Klassen einzuordnen, die auf den Kategorien des act4teams®-SHORT-Kodierschema basieren. Trotz aller Unterschiede sei zu erwähnen, dass diese Arbeit auf ausgewählte Implementierungen und Erkenntnissen einiger zuvor genannter Arbeiten aufbaut.

Kapitel 4

Konzept zur Erkennung destruktiver Äußerungen

In Softwareprojekten wird häufig in Meeting kommuniziert. Destruktive Äußerungen können den Ausgang eines Meetings bestimmen und sogar einen negativen Einfluss auf den Projekterfolg haben. Daher ist es wichtig, diese Art von Äußerungen auf ein Minimum zu halten. Im Rahmen dieser Arbeit wird die Kommunikation in Softwareprojekt-Meetings auf destruktive Äußerungen untersucht. Kapitel 4 beschreibt das entwickelte Konzept, um destruktive Aussagen automatisch zu erkennen. Abschnitt 4.1 zeigt anhand von Beispielen die Merkmale destruktiver Äußerungen, die für eine automatische Klassifikation verwenden werden. In Abschnitt 4.2 wird die Auswahl der Klassifikatoren begründet. Eine Beschreibung des gesamten Prozesses der automatischen Klassifikation von destruktiven Aussagen befindet sich unter Abschnitt 4.3.

4.1 Merkmale destruktiver Äußerungen

In Abschnitt 2.3 wurde der Begriff der destruktiven Äußerung wie folgt definiert: Fünf Unterkategorien aus dem act4teams®- und act4teams®-SHORT-Kodierschema bilden die Kategorie der destruktiven Äußerungen. Alle Äußerungen, die andere beleidigen, abwerten oder zurechtweisen, zählen zur Unterkategorie *Tadel*. Wenn der negative Ist-Zustand betont oder die Unveränderlichkeit einer Situation hervorgehoben wird, wird das als *jammern* bezeichnet. Die dritte Unterkategorie *Schuldigensuche* umfasst alle Aussagen, die getroffen werden, um jemandem die Schuld für Probleme zu geben. Äußerungen, die ausdrücken, dass jemand keine Veränderung wünscht, fallen in die Kategorie *Kein Interesse an Veränderung*. Zudem ist das Unterbrechen anderer ebenfalls als destruktive Äußerung zu werten.

Im Beispiel 4.1 sind drei Äußerungen, die Tadel ausdrücken. Auffallend

Das Design ist echt grottenhässlich.
Warum liest du nicht deine dummen E-Mails?
Was du gemacht hast, ist einfach Mist.

Beispiel 4.1: Äußerungen für Tadel

ist die häufige Verwendung von Diffamien[1], um andere abzuwerten. Die negative Polarität dieser Wörter kann daher als Unterscheidungsmerkmal verwendet werden. Um die Polarität eines Wortes zu bestimmen, wird ein Lexikon verwendet, das Sentiment-Wörter mit Polaritätswerten enthält. Für die Domäne des Software Engineering wurde das frei verfügbare Sentiment-Lexikon SentiStrength-SE entwickelt [28]. Positive Wörter werden mit Ganzzahlen zwischen +1 und +5 bewertet, wobei ein Wert von +1 eine neutrale Stimmung und ein Wert von +5 eine stark positive Stimmung ausdrückt [28]. Bei negativen Wörtern wird analog dazu mit negativen Zahlen gearbeitet. Ein Wert von -1 drückt eine neutrale Stimmung aus und ein Wert von -5 eine stark negative Stimmung. Ein Vergleich mit dem Original SentiStrength zeigte, dass SentiStrength-SE bei der Erkennung von Emotionen in Software Engineering Texten deutlich überlegen ist [28]. Für diese Arbeit kann SentiStrength-SE jedoch nicht verwendet werden, da es sich um ein englischsprachiges Lexikon handelt. Remus et al. [59] entwickelten das erste deutschsprachige Sentiment-Lexikon SentimentWortschatz (kurz. SentiWS), das frei verfügbar ist. Es beinhaltet positive und negative Sentimentwörter mit Sentiment-Werten, die in einem Intervall zwischen [-1;1] liegen. Positive Wörter bekommen einen positiven Wert zugewiesen und negative Wörter einen negativen Wert. Leider wurde dieses Lexikon nicht spezifisch für das Software Engineering erstellt.

Es ist doch immer das Gleiche.
Jedes Mal gibt es Stress.
Und schon wieder müssen wir warten.

Beispiel 4.2: Äußerungen für Jammer

An den Sätzen für Jammer aus Beispiel 4.2 ist erkennbar, dass bestimmte Situationen verallgemeinert werden. Dies kann durch Wörter wie “immer” oder durch Satzteile wie “jedes Mal” ausgedrückt werden. Solche Sätze können eine Hilflosigkeit oder die Unveränderlichkeit einer Situation ausdrücken. Daher können diese Satzglieder verwendet werden, um Aussagen zu erkennen, die Jammer ausdrücken. Darüber hinaus können Wörter mit negativer Polarität

---

[1] Beleidungen, Beschimpfungen

auftreten, die ebenfalls als Merkmal eingesetzt werden können.

> Wir müssen immer später anfangen, weil du niemals pünktlich bist.
>
> Deinetwegen müssen wir Extra-Arbeit erledigen.
>
> Das ist alles deine Schuld.

Beispiel 4.3: Äußerungen, die einen Schuldigen suchen

Sätze, die geäußert werden, um einen Schuldigen zu suchen, wie in Beispiel 4.3, scheinen ähnliche Merkmale aufweisen, wie die beiden zuvor genannten Kategorien. Es können Wörter vorhanden sein, die einen Sachverhalt generalisieren, in diesem Fall "immer" oder "niemals". Letzteres hat zusätzlich eine negierende Wirkung auf das positive Wort "pünktlich", wodurch der Polaritätswert ins Negative gekehrt wird.

> Wir machen das auf gar keinen Fall.
>
> Wollen wir uns wirklich diesen unnötigen Aufwand machen?
>
> Das ist doch viel zu schwierig.

Beispiel 4.4: Äußerungen, die kein Interesse an Veränderungen vermitteln

Sätze aus der Kategorie *Interesse an Veränderungen*, wie aus Beispiel 4.4, können auch Wörter mit negativer Polarität enthalten, wie die Wörter "unnötigen", "Aufwand" oder "schwierig".

Einige statistische Merkmale, wie die Anzahl oder Relation der Interpunktionen, um beispielsweise Emoticons zu erfassen, können in der gesprochenen Sprache nicht angewendet werden. Stattdessen könnten Partikeln betrachtet werden, die in der gesprochenen Sprache sehr häufig vorkommen. Sie können unter anderem eine subjektive Einstellung oder Emotionen ausdrücken [69]. Das Wort "doch" im dritten Satz der Beispiel 4.4 drückt aus, dass der Sprecher den Sachverhalt für selbstverständlich hält und somit könnte diese Äußerung einen leicht ärgerlichen Unterton bekommen. Das häufige Verwenden von Partikeln könnte ein Hinweis auf die Stimmung der Äußerung geben und wird daher als Merkmal verwendet.

Unterbrechungen unterscheiden sich grundsätzlich von den zuvor genannten Kategorien, da der Inhalt der Äußerungen sowohl positiv als auch negativ sein kann. Dies gilt für die Aussagen aller Sprecher, die an der Sprecherüberschneidung involviert sind. Unterbrochene Sätze können also inhaltlich nicht in destruktive und nicht destruktive Äußerungen aufgeteilt werden und können daher die Klassifizierung des Machine-Learning-Algorithmus verfälschen.

Aus diesem Grund dürfen unterbrochene Sätze nicht Teil des Datensatzes sein, mit dem die Modelle trainiert werden. Jedoch kann die syntaktische Satzstruktur einer unterbrochenen Aussage als Unterscheidungsmerkmal dienen und sollte anhand dessen vor der Anwendung von Machine-Learning-Algorithmen identifiziert und aus dem Datensatz entfernt werden. Um dies zu erreichen, wird der Datensatz mit Hilfe von *Rule-based Matching* auf Satzstrukturen, die auf unterbrochene Sätze hindeuten, untersucht.

Die Erkennung unterbrochener Äußerungen ist nicht trivial. In der Theorie heißt es zwar, dass ein vollständiger deutscher Satz aus mindestens einem Subjekt und einem Prädikat bestehen muss [11], jedoch gibt es rhetorische Stilmittel, die es erlauben, beispielsweise Satzteile auszulassen. Somit könnte es Sätze ohne ein Prädikat geben, die trotzdem korrekt sind. Ein Beispiel dafür wäre das Sprichwort “Erst die Arbeit und dann das Vergnügen”. Hier kann das Prädikat weggelassen werden, ohne dass der Satz unterbrochen wird. Daher werden verschiedene Regeln aufgestellt, die vollständige Sätze erkennen können. Beispiele für diese Regeln werden in Unterabschnitt 4.3.3 vorgestellt.

4.2 Klassifizierer

In Unterabschnitt 2.5.3 wurden sechs Klassifizierer vorgestellt, die im Bereich der Sentiment Analyse zum Einsatz gekommen sind. Bei der **Sentiment Analyse** handelt es sich um ein Teilgebiet des Text Mining, das sich mit der Extraktion von Stimmungen und Meinungen aus Texteinheiten beschäftigt [1]. So kann aus einer Texteinheit eine entweder positive, negative oder neutrale Stimmung erkannt werden [54]. Die Polaritätswerte bestimmter Wörter kann als Merkmal für den Klassifizierer verwendet werden. Mit Hilfe von Lexika, die stimmungstragende Wörter mit ihrem zugehörigen Polaritätswert (Sentiment-Score) beinhalten, wird der Gesamtwert einer Texteinheit errechnet und für die Klassifikation herangezogen [64].

In einem Literatur-Review von Obaidi und Klünder [49] wurden verschiedene wissenschaftliche Publikationen aus dem Bereich der Sentiment Analyse im Software Engineering betrachtet, die Machine-Learning-Algorithmen unter anderem hinsichtlich ihrer Performanz verglichen. Da die Klassifizierer SVM und Gradient-Boosting die besten Resultate zeigten, werden sie für die Klassifikation von destruktiven Äußerungen verwendet. Zusätzlich wurde untersucht, wie häufig die Klassifizierer eingesetzt wurden [49]. Das Ergebnis zeigte, dass SVM und Bayes am meisten genutzt wurden, daher wird zusätzlich Naive Bayes für dieses Konzept verwendet. Im Rahmen dieser Arbeit soll die Performanz der drei ausgewählten Klassifizierer bezüglich der Unterscheidung von destruktiven und nicht destruktiven Äußerungen gegenübergestellt werden.

4.3 Klassifikationsprozess

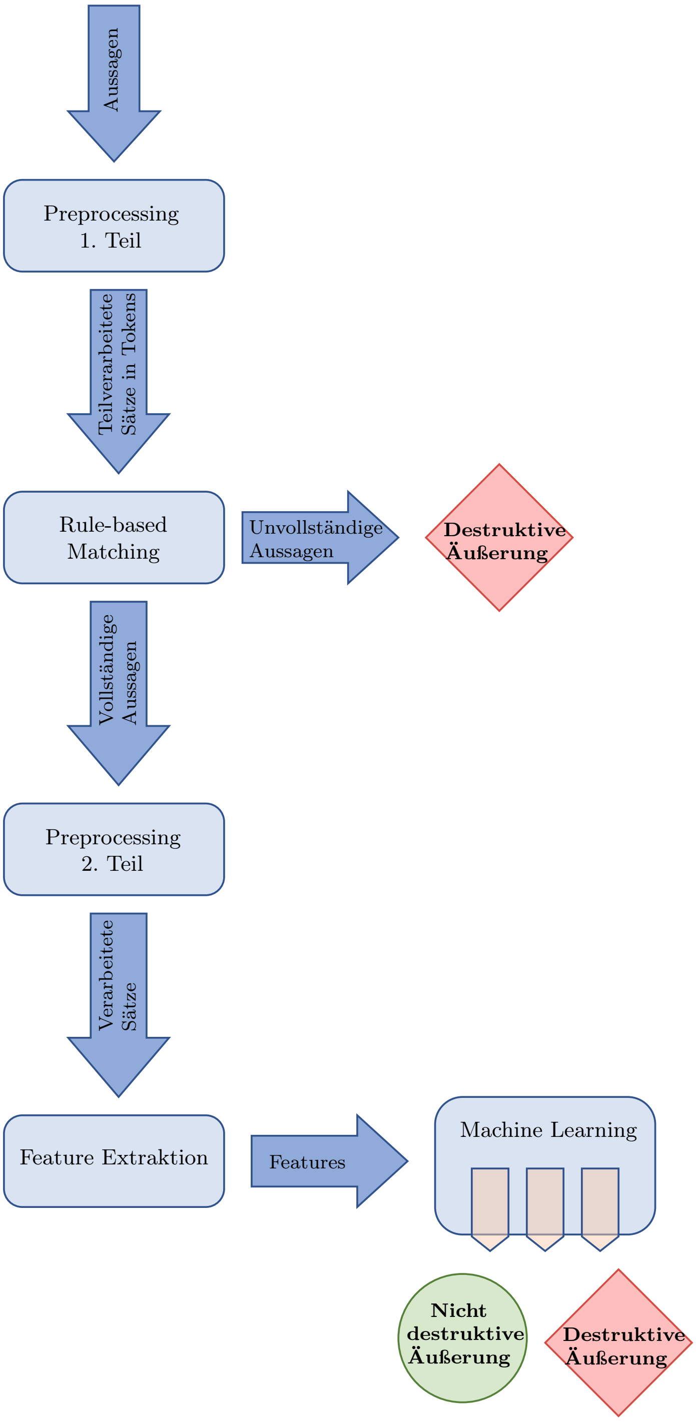


Abbildung 4.1: Prozess zur Erkennung destruktiver Äußerungen

Abbildung 4.1 stellt den Ablauf der automatischen Klassifikation von destruktiven Äußerungen dar. Die Schritte des Prozesses werden in den folgenden Unterkapiteln genauer beschrieben. Zu Beginn wird ein Datensatz ausgewählt und aufbereitet, sodass dieser als Eingabe für den ersten Schritt **Preprocessing** eingesetzt werden kann. Nachdem der erste Teil der Vorverarbeitung durchgeführt wurde, werden unterbrochene Sätze durch **Rule-based Matching** herausgefiltert. Darauf folgt der zweite Teil des **Preprocessing**, bevor die Merkmale extrahiert (engl. **Feature Extraction**) werden. Der Vektor mit allen relevanten Merkmalen wird für die Klassifikation durch die Modelle, die zuvor trainiert und validiert wurden, verwendet. Im letzten Schritt **Machine Learning** werden die Aussagen den Klassen *destruktiv* oder *nicht destruktiv* zugeordnet.

4.3.1 Datenselektion

Für die Umsetzung des Konzepts werden Datensätze zum Trainieren der Klassifikatoren und zum Validieren aller Komponenten benötigt. Die Aussagen für diesen Datensatz stammen aus einer von Schneider et al. [61] durchgeführten Studie mit 32 studentischen Entwicklerteams im Rahmen der jährlich stattfindenden Veranstaltung *Software-Projekt* der Leibniz Universität Hannover. Die Meetings aus der Anfangsphase des Projekts wurden nach dem act4team-Kodierschema analysiert. Die aktuelle Version des Datensatzes umfasst 42 Transkripte, die die Aussagen der Teilnehmer und die Kategorien, mit denen sie kodiert wurden, beinhalten [34]. Aus diesem deutschsprachigen Ausgangsdatensatz wurden zwei Datensätze erstellt.

Der erste Datensatz, der keine unterbrochenen Aussagen beinhalten darf, wird zum Trainieren der Klassifikatoren benötigt. Zum Zeitpunkt der Klassifikation durch Machine-Learning sollten unvollständige Sätze bereits erkannt[2] und aus dem Datensatz entfernt worden sein, anderenfalls könnte die Klassifikation verfälscht werden. Ein zweiter Datensatz wird benötigt, um das gesamte Konzept vor der Evaluation zu validieren. Auf diese Weise können Schwachstellen aufgedeckt und behoben werden. Für die Klassifikation durch Machine-Learning sollte auf ein ausgeglichenes Verhältnis zwischen destruktiven und nicht destruktiven Aussagen geachtet werden, da ein unausgeglichener Datensatz Fehlklassifikationen begünstigt [12].

4.3.2 Preprocessing

Der generierte Datensatz unterläuft zunächst einer Teilvorverarbeitung, wie in Abbildung 4.2 dargestellt. Die Sätze werden auf Rechtschreibfehler überprüft und in Tokens aufgeteilt. Zusätzlich werden die Tokens mit POS-Tags und gemäß ihrer Stellung im Satz markiert. Eine Modifikation der

---

[2] Durch Rule-based Matching

Sätze wird noch nicht durchgeführt. Dies ist wichtig, damit die Äußerungen im nächsten Schritt **Rule-based Matching** auf bestimme Muster bzw. Satzstrukturen untersucht werden können.

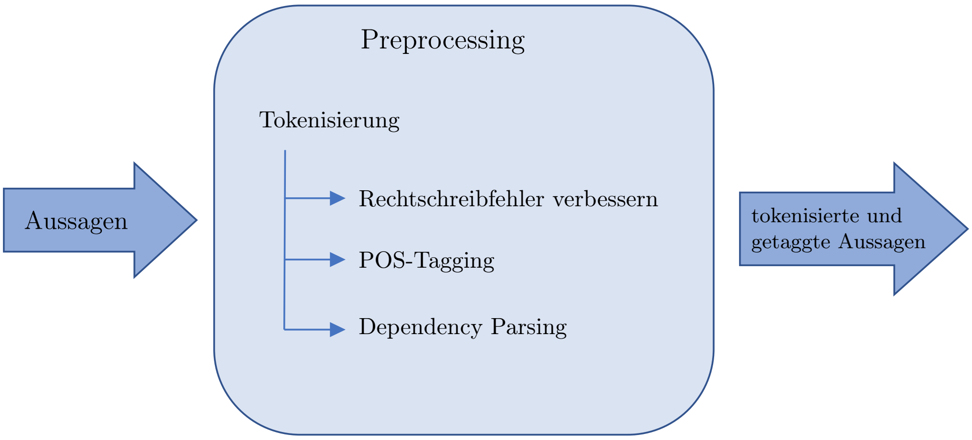


Abbildung 4.2: Der erste Teil der Vorverarbeitung

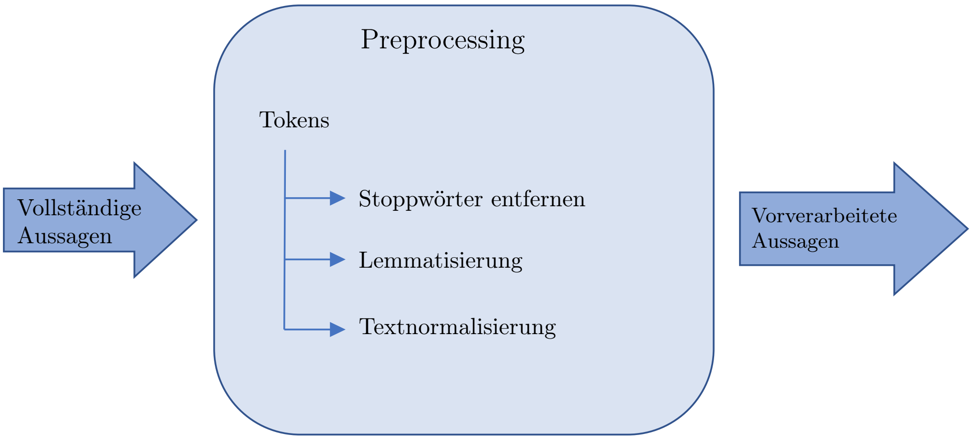
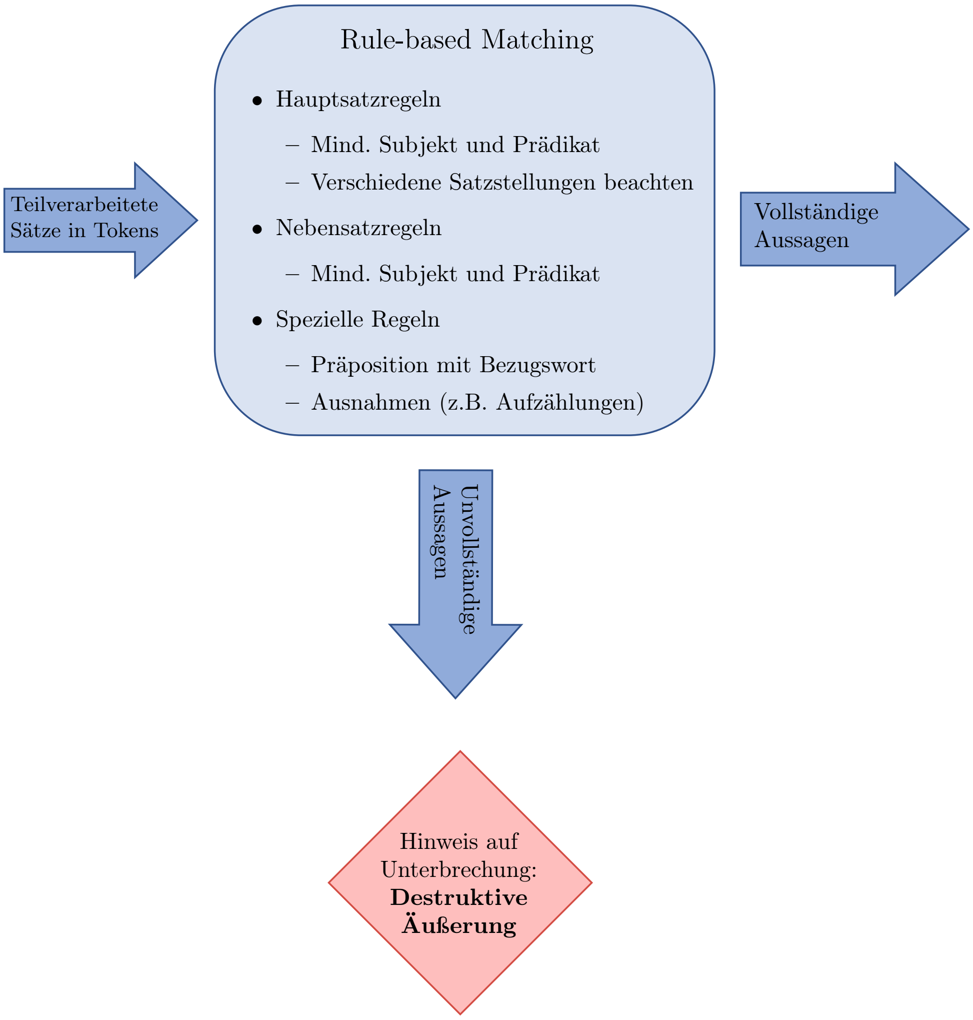


Abbildung 4.3: Der zweite Teil der Vorverarbeitung

Nachdem unterbrochene Aussagen aus dem Datensatz herausgefiltert wurden, wird der zweite Teil der Vorverarbeitung durchgeführt. Abbildung 4.3 zeigt die Vorgänge in diesem Schritt. Es werden alle Stoppwörter und Interpunktionen entfernt. Die übrig gebliebenen Wörter werden auf ihre Grundformen zurückgeführt. Anschließend werden die Aussagen normalisiert. Das bedeutet, dass alle Sonderzeichen wie Umlaute konvertiert und die Texteinheiten in Kleinschreibung geändert werden.

4.3.3 Rule-based Matching



In diesem Schritt wird die Satzstruktur der teilverarbeiteten Sätze überprüft. Mit Hilfe von erstellten Regeln werden Muster gesucht, die auf einen vollständigen Satz hindeuten. Die Herausforderung besteht darin, Regeln für gesprochene Sätze zu formulieren, da die gesprochene Sprache andere grammatikalische Besonderheiten gegenüber der geschriebenen Sprache besitzt (vgl. Unterabschnitt 2.4.3). Die folgende Aufzählung gibt eine Übersicht über die verwendeten Regeln:

Ein Satz ist vollständig, wenn:

- ein Hilfsverb[3] verwendet wird und ein Hauptverb vorhanden ist.
  Beispiel: Ich werde gehen.

---

[3] sein, haben, werden

- ein Hilfsverb verwendet wird und ein Adjektiv vorhanden ist.
  Beispiel: Ich bin fröhlich.
- ein Hilfsverb verwendet wird und ein (Pro-)Nomen oder substantiviertes Verb/Adjektiv vorhanden ist.
  Beispiel: [Das] Hab' ich.
- ein (Pro-)Nomen oder substantiviertes Verb/Adjektiv verwendet wird und ein Hauptverb vorhanden sind.
  Beispiel: Ich gehe.
- ein (Pro-)Nomen oder substantiviertes Verb/Adjektiv verwendet wird und ein Hilfsverb und ein (Pro-)Nomen oder substantiviertes Verb/-Adjektiv vorhanden sind.
  Beispiel: Ich habe ein Haus.
- ein (Pro-)Nomen oder substantiviertes Verb/Adjektiv verwendet wird und ein Hilfsverb und ein Adjektiv vorhanden sind.
  Beispiel: Ich bin müde.
- ein Imperativ verwendet wird. Beispiel: Geh!
- in einem Nebensatz ein (Hilfs-)Verb am Ende steht.
  Beispiel: [Ich finde es schön,] dass du bleibst.

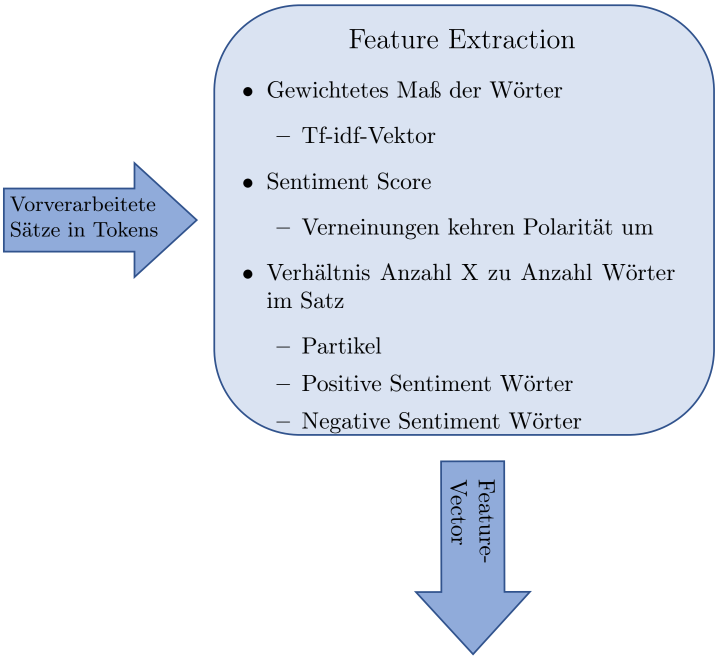
Für jede dieser einfachen Grundregeln werden in der Implementierung mehrere Pattern erstellt, da sich die Position des Verb je nach Tempus des Satzes ändern kann. Zudem werden Pattern nach folgenden speziellen Regeln erstellt. Diese Regeln werden benötigt, da einige Fälle durch die oben genannten Regeln nicht abgedeckt werden. Einige Spezialregeln können bei Nichterfüllung auch unvollständige Sätze erkennen.

Spezielle Regeln:

- Präpositionen brauchen ein Bezugswort
  Beispiel: [Ich gehe] über die Straße.
- Auf einen Artikel am Ende des Satzes folgt ein Nomen.
  Beispiel: [Ich gehe über] die Straße.
- Infinitivkonstruktionen mit “zu” stehen meist am Ende eines Satzes
  Beispiel: [Das ist] einfacher zu merken.
- Wenn nur (Pro-)Nomen und Konjunktionen verwendet werden, ist es höchstwahrscheinlich eine Aufzählung.
- Ellipsen sind vollständige Sätze.
  Beispiele: Ich auch. Ich nicht.

- Fragepronomen und einige Konjunktionen, wie “und”, “oder” und “aber” können auch alleine stehen.
  Beispiele: Oder? Warum?

4.3.4 Feature Extraktion



Aus den vorverarbeiteten und vollständigen Sätze sollen nun Merkmale herausgefiltert werden. Der Sentiment-Score ist ein Wert, der mit Hilfe eines Lexikons für jede Aussage errechnet wird. Jeder Token wird mit dem Lexikon abgeglichen und der Polaritätswert bestimmt. Anschließend wird der Durchschnitt der Werte berechnet. Dabei müssen Verneinungen unbedingt berücksichtigt werden, da sie die Polarität einer Aussage komplett umdrehen können. Man betrachte den folgenden Satz:

> Das ist gar nicht schlecht.

Das Sentiment-Lexikon *SentiWS* bewertet das Wort “schlecht” mit dem Polaritätswert -0.7706. Dies entspricht jedoch nicht der positiven Stimmung, die der Satz eigentlich ausdrückt. Durch die Berücksichtigung der Verneinung “nicht” würde der Polaritätswert von “schlecht” umgekehrt und das Ergebnis wäre ein positiver Sentiment-Score.

Unsere Zusammenarbeit hat gut funktioniert.
Unsere Zusammenarbeit hat schlecht funktioniert.
Unsere Zusammenarbeit war nicht gut.

Die oben genannten Sätze sollen dieses Prinzip an einem anderen Beispiel verdeutlichen. Die Aussagen sehr ähnlich, jedoch ist die Stimmung unterschiedlich. Die erste Aussage ist positiv und die beiden letzten Aussagen haben eine negative Stimmung. Der Wortabgleich mit dem Sentiment-Lexikon ergibt folgende Werte:

0.0	0.0893	0.0	0.3716	0.0040	$= 0.4649/3$
Unsere	Zusammenarbeit	hat	gut	funktioniert.	$= 0.155$
0.0	0.0893	0.0	-0.7706	0.0040	$= -0.6773/3$
Unsere	Zusammenarbeit	hat	schlecht	funktioniert.	$= -0.226$
0.0	0.0893	0.0	0.0	0.3716	$= 0.4609 * (-1)/3$
Unsere	Zusammenarbeit	war	nicht	gut.	$= -0.4609/3$
					$= -0.154$

Die grün geschriebenen Worte “Zusammenarbeit”, “gut” und “funktioniert” haben nach *SentiWS* positive Werte und drücken daher eine positive Stimmung aus. Das rot geschriebene Wort “schlecht” hat einen negativen Sentiment-Wert und weist damit auf eine negative Stimmung hin. Die lilafarbene Verneinung “nicht” hat zwar keinen Sentiment-Wert, jedoch müssten dadurch ein oder mehrere Sentiment-Werte umgekehrt werden. Die oben genannten Beispiele zeigen, wie der Sentiment-Wert einer Aussage berechnet wird. Aus den drei positiven Sentiment-Werten des ersten Satzes wird der arithmetische Mittelwert gebildet und erhält dadurch einen Gesamt-Sentiment-Wert von 0.155. Dieser Wert liegt im positiven Bereich und würde somit auch der positiven Polarität des Satzes entsprechen. Die zweite Äußerung besitzt zwei Wörter mit positiver Polarität und eins mit negativer Polarität. In diesem Fall überwiegt der negative Wert des Wortes “schlecht” und der insgesamt negative Score spiegelt die negative Stimmung der Aussage wider. Die letzte Äußerung besitzt ebenfalls eine negative Stimmung. Es ist jedoch kein Wort mit negativer Polarität enthalten. Wenn die Verneinung “nicht” berücksichtigt wird, wird der Sentiment-Score negativ. In diesem Fall wird das Vorzeichen des Polaritätswert umkehrt.

Die Untersuchung eines Satzes auf Partikeln kann hilfreich sein, falls eine Aussage keine Sentiment-Wörter aufweist. Da Partikeln Träger von Emotionen sein können [69], könnte die Häufigkeit der Nutzung ein Hinweis auf destruktive Aussagen sein. Für jede Aussage wird die Anzahl der Partikeln im Verhältnis zu der Anzahl verwendeter Wörter im Satz gesetzt. Wörter, die Situationen verallgemeinern, wie zum Beispiel “immer” oder “ständig” werden ebenso berücksichtigt. Auch für positive und negative Sentimentwörter wird der prozentuale Anteil im Satz berechnet. Aus den verwendeten negativen Sentimentwörtern und Partikeln wird ein gewichteter Vektor erstellt, der das Tf-idf-Maß nutzt. Das nachfolgende Beispiel veranschaulicht das Ergebnis der Merkmalsextraktion für die Äußerungen. Aus Gründen der besseren Lesbarkeit zeigt die Tabelle 4.5 den Merkmalsvektor ohne den Tf-idf-Vektor in einer leicht veränderten Darstellung.

(1) Unsere Zusammenarbeit hat schlecht funktioniert.

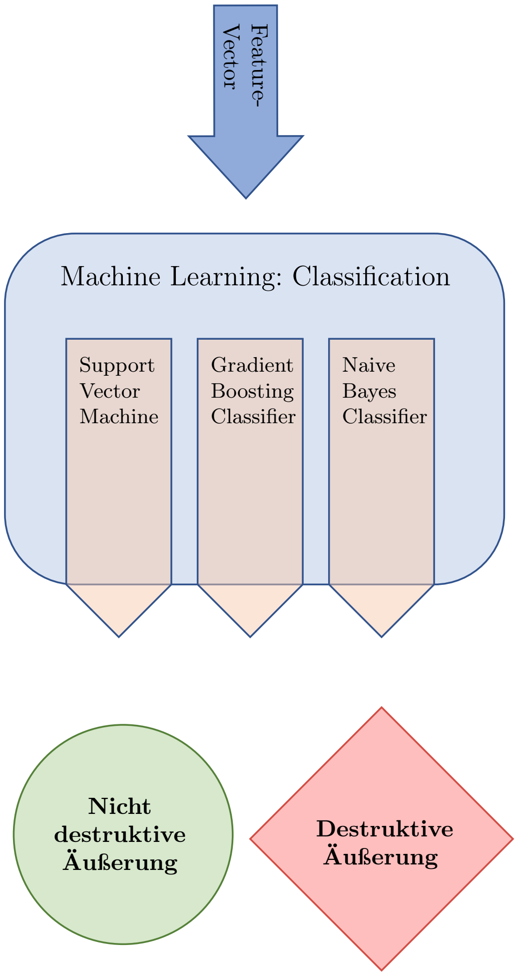
(2) Hast du die Mail etwa schon wieder vergessen?

(3) Ständig muss ich deine Arbeit erledigen.

Äußerung Merkmal	(1)	(2)	(3)
Sentiment-Score	0.0	0.0	0.0
Partikel-Ratio	0.0	0.375	0.0
Allgemein-Ratio	0.0	0.0	0.167
Positive Sentimentwort-Ratio	0.4	0.0	0.0
Negative Sentimentwort-Ratio	0.2	0.0	0.0

Tabelle 4.5: Merkmalsvektor ohne Tf-idf-Vektor

4.3.5 Machine Learning



Die drei Klassifizierer SVM, Gradient-Boosting und Naive Bayes werden zunächst mit dem zuvor erstellten Datensatz, der keine unvollständigen Sätze enthält, trainiert. Es wird ein Testset aus 30% der vorverarbeiteten Daten erstellt, um das trainierte Modell zu validieren. Die verbliebenen 70% der vorverarbeiteten Daten bilden das Trainingsset. Die trainierten Modelle werden für eine spätere Anwendung, wie etwa eine Evaluation, gespeichert.

Der Klassizifierer wird die Sätze, die nach dem Rule-based Matching verblieben sind, mit Hilfe der extrahierten Merkmale den Klassen *destruktiv* und *nicht destruktiv* zuordnen. Liegt ein gelabelter Datensatz vor, können die Ergebnisse ausgewertet werden. Bei einem ungelabelten Datensatz können die Sätze und ihre vorhergesagten Klassen angezeigt werden. Zusätzlich wird eine Information über den prozentualen Anteil an destruktiven Äußerungen ausgegeben.

Kapitel 5

Prototypische Implementierung

Kapitel 5 stellt die Implementierung des Prototypen vor. Zunächst wird in Abschnitt 5.1 die verwendete Programmiersprache vorgestellt und eine Übersicht über die verwendete Bibliothek für *Natural Language Processing* gegeben. In Abschnitt 5.2 wird anschließend erläutert, welche Datenquelle verwendet wurde und wie der Datensatz erstellt wurde. Abschnitt 5.3 beschreibt auf welche Weise Daten eingegeben werden können, um sie zu klassifizieren. Zusätzlich wird die Vorverarbeitung der Texteinheiten und *Rule-based Matching* mit der Bibliothek *spaCy* in Abschnitt 5.4 dargestellt. Abschnitt 5.5 beschäftigt sich mit der Extraktion der Merkmale und der Implementierung der Klassifizierer.

5.1 Programmiersprache

Für die Implementierung des Prototypen wurde Python als Programmiersprache ausgewählt, aufgrund der Auswahl an frei zugänglichen Open Source Bibliotheken. Die verwendete Python-Version 3.8.2 erfüllt die Mindestanforderungen der verwendeten Bibliotheken.

In Python ist die weitverbreitetste Bibliothek für Natural Language Processing *Natural Language Toolkit* (kurz NLTK). Diese ist zwar in der Lage deutsche Texte zu verarbeiten, jedoch gibt es für die deutsche Sprache keinen POS-Tagger oder Lemmatisierer. Daher wurde die Bibliothek *spaCy* in der aktuellen Version 3.1.0 verwendet, die deutsche Texte tokenisieren, lemmatisieren und mit dem POS markieren kann.

Bibliotheken

Die folgende Übersicht zeigt, welche Bibliotheken für den Prototypen verwendet wurden.

Spellchecker wird verwendet, um die Rechtschreibung zu korrigieren.

SpaCy wird für die natürliche Sprachverarbeitung im *Preprocessing* und für das *Rule-based Matching* eingesetzt.

Aus **sklearn** werden die Klassifikatoren sowie der Vectorizer zur Erstellung eines Merkmalsvektors benutzt.

Matplotlib bietet zusammen mit **sklearn** die Möglichkeit Konfusionsmatrizen grafisch darzustellen.

Pickle wird zum Speichern und Laden der trainierten Modelle verwendet.

Deepspeech, **pyaudio**, **webrtcvad** und **scipy** werden für die Verarbeitung von Audiospuren verwendet und wurden aus der Implementierung von Herrmann [21] übernommen.

5.2 Datenselektion

Als Datenquelle werden Transkripte aus den *Software-Projekt*-Meetings der Leibniz Universität Hannover verwendet [61]. Die Aussagen aus diesen Meetings wurden manuell von erfahrenen Psychologinnen nach dem act4teams®-Kodierschema klassifiziert [34]. Für diese Arbeit werden zwei Datensätze erzeugt. Zunächst wurden alle destruktiven Aussagen für den ersten Datensatz herausgefiltert. Von den ungefähr 10.000 Aussagen waren nur etwa 400 bis 500 Aussagen destruktiv. Um einen ausgeglichenen Datensatz zu erzeugen, wurden daher etwa genau so viele nicht destruktive Aussagen zufällig ausgewählt. Zum Schluss wurden die Kategorien des Datensatzes auf Korrektheit überprüft und einige Klassifikationen nach eigenem Empfinden angepasst. Dieser Datensatz wird für die Validierung der Klassifizierer und des Pattern Matching verwendet.

Der zweite Datensatz beinhaltet destruktive Aussagen, aber keine unterbrochenen Sätze. Auch für diesen Datensatz werden nicht destruktive Aussagen zufällig ausgewählt, um einen ausgeglichenen Datensatz zu erzeugen. Die Klassifizierer sollen mit diesem Datensatz trainiert werden, da sie Sätze mit positiver Stimmung von Sätzen mit negativer Stimmung unterscheiden sollen. Unterbrochene Sätze können sowohl positiv als auch negativ sein, daher könnten sie die Klassifikation verfälschen.

5.3 Eingabe

Um Aussagen zu klassifizieren, benötigt der Prototyp ein Transkript im *csv*-Format oder eine Audio-Aufzeichnung im *WAV*-Format. Alternativ besteht die Möglichkeit das Mikrofon in einem Meeting zu verwenden, um die

Aussagen vor Ort zu klassifizieren. Der Prototyp verwendet für die Sprach- und Texteingabe die Implementierung von Herrmann, die im Rahmen seiner Bachelorarbeit *Automatische Klassifikation von Aussagen in Meetings von Entwicklungsteams* [21] entstand. Das Tool von Herrmann [21] kann Audioaufzeichnungen sowie eine Audiospur über Mikrofon transkribieren unter anderem mit Hilfe von *Deepspeech* und *Microphone VAD Streaming* für die Sprechpausenerkennung. Das Transkript wird im nächsten Schritt für die Vorverarbeitung mit *spaCy* verwendet.

5.4 spaCy

SpaCy ist eine Open Source Bibliothek in Python für Natural Language Processing [14]. Mit *spaCy* ist es möglich, Texte zu tokenisieren und zu lemmatisieren. Zusätzlich gibt es für die deutsche Sprache einen POS Tagger und Dependency Parser, um Sätze syntaktisch zu analysieren. Zudem kann mit *spaCy* kann ein regelbasierter Abgleich (engl. rule-based Matching) durchgeführt werden. Dadurch können Texte auf Vorkommen bestimmter Muster untersucht werden.

Um Tagger oder Parser verwenden zu können, muss zunächst eine trainierte Pipeline geladen werden, die diese Komponenten beinhaltet. Zur Auswahl stehen verschiedene Pipelines je nach Sprache und verwendeten Trainingsdatensatz. Die hier verwendete deutschsprachige Pipeline *de_core_news_sm* wurde mit einem Datensatz aus dem Bereich Nachrichten trainiert und beinhaltet alle verfügbaren Komponenten. Die Trainingsdatensätze aller Pipelines, die für die deutsche Sprache zur Verfügung stehen, stammen aus dem Bereich der geschriebenen Sprache.

5.4.1 POS Tagger & Dependency Parser

Der POS-Tagger markiert jedes Token mit seiner Wortart, wie beispielsweise Nomen, Verb oder Adjektiv. Diese Tags werden für die syntaktische Analyse im Rahmen des *Rule-based Matching* verwendet.

Der Dependency Parser ist nicht nur in der Lage, Wörter mit der entsprechenden Satzstellung zu markieren. Er besitzt zudem die Fähigkeit, Abhängigkeiten zwischen Tokens zu erkennen. Dies ist nützlich für die Berücksichtigung von Verneinungen. Die folgende Beispielaussage besitzt zwei positive Wörter und eine Verneinung, daher ist es wichtig zu wissen, ob die Polarität eines Wortes oder des gesamten Satzes umgekehrt werden soll. In Abbildung 5.1 und Abbildung 5.2 wird die Beispielaussage „Dieses Ergebnis ist nicht korrekt, aber alles andere ist super.“ mit den Tags und Abhängigkeiten grafisch dargestellt.

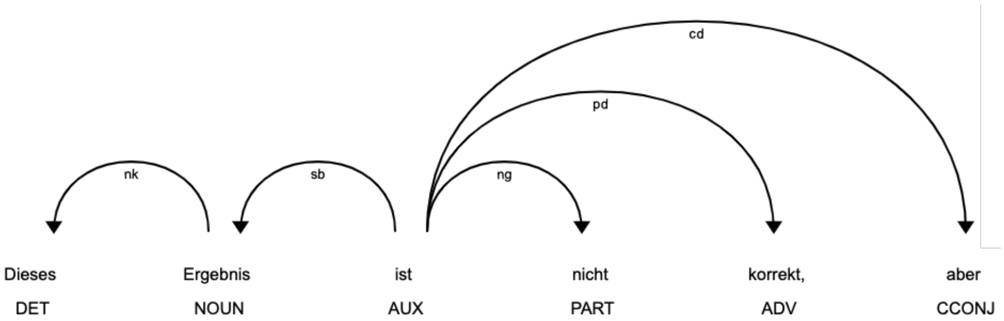


Abbildung 5.1: Visualisierung des ersten Hauptsatzes mit dem Dependency Parser (DET: Artikel, NOUN: Nomen, AUX: Hilfsverb, PART: Partikel, ADV: Adverb, CCONJ: Konjunktion, nk: Nominalphrase, sb: Subjekt, ng: Negation, pd: Prädikat, cd: Konjunktion)

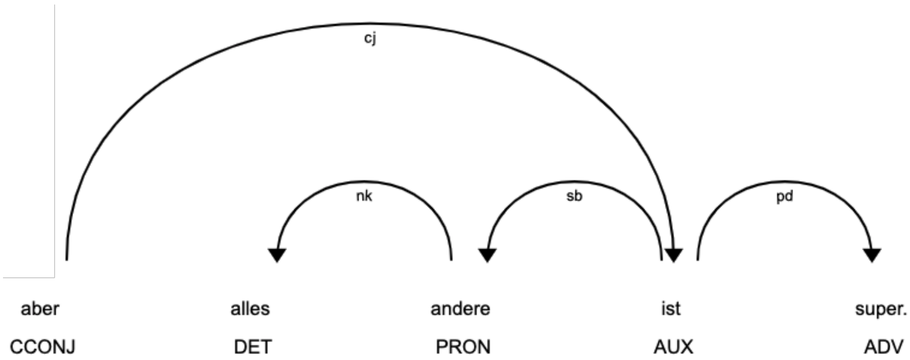


Abbildung 5.2: Visualisierung des zweiten Hauptsatzes mit dem Dependency Parser (PRON: Pronomen, cj: Konjunktion)

Abbildung 5.1 und Abbildung 5.2 zeigen, dass es zwischen dem ersten und zweiten Hauptsatz keine Abhängigkeiten gibt, da keine Pfeile die beiden Hauptsätze verbinden. Die Wurzel des ersten Hauptsatzes bildet das Wort “ist” und es ist zu beobachten, dass alle anderen Wörter davon abhängig sind mit Ausnahme des Artikels “Dieses”. Um herauszufinden, worauf sich die Verneinung “nicht” bezieht, könnte eine indirekte Abhängigkeit gesucht werden, wenn die Verneinung und das Bezugswort denselben Vorgänger oder diesselbe Wurzel besitzen. Es ist auch möglich, dass zwischen Verneinung und Bezugswort eine direkte Abhängigkeit besteht. In diesem Fall bezieht sich die Verneinung “nicht” auf das Wort “korrekt”, da sie beide denselben Vorgänger haben. Es folgt eine Übersicht mit dem Beispielsatz und Polaritätswerten aus SentiWS.

0.0	0.0	0.0	0.0	0.0040	0.0	0.0	0.0	0.0	0.5012
Dieses	Ergebnis	ist	nicht	korrekt,	aber	alles	andere	ist	super.

Die oben genannte Beispielaussage besitzt eine eher positive Stimmung, wenn sie als Ganzes betrachtet wird. Würde man mit der Verneinung “nicht” die gesamte Porarität des Satzes umkehren, wäre der Polaritätswert insgesamt negativ und würde eigentlich auf eine negative Stimmung hinweisen. Mit dem Dependency Parser ist es möglich, eine Verbindung zwischen “nicht” und “korrekt” herzustellen, wodurch der Polaritätswert des Wortes “korrekt” gezielt umgekehrt werden kann. Der positive Polaritätswert des Wortes “super” würde überwiegen, sodass die Stimmung des gesamten Satzes positiv bleibt.

5.4.2 Rule-based Matching

SpaCys Rule-based Matching erlaubt es, Regeln zu erstellen, um Texte auf das Vorkommen von Mustern zu durchsuchen. Es ist nicht nur möglich, Stringabfolgen zu suchen, sondern auch bestimmte Abfolgen von Wortarten oder eine Satzstellung. Auf diese können Aussagen auf syntaktische Satzstrukturen untersucht werden. Im Prototyp werden nach dem Erstellen eines Matchers sogenannte Pattern erstellt, um diese in den Texteinheiten zu suchen. Sobald die erstellten Pattern zum Matcher hinzugefügt wurden, können Texte auf die gewünschten Muster durchsucht werden. Der Matcher gibt an, welche Pattern an welchen Stellen gefunden wurden.

So würde man normalerweise keinen Satz mit einer Präposition beenden, da die Präpositionen vor einem Bezugswort stehen. Ein Präfix eines Verbs, das bei der Konjugation getrennt werden kann, wie beispielsweise “aufbauen”, wird nicht als Präposition markiert, sondern als Teil eines Verbs. Um zu überprüfen, ob auf eine Präposition stets ein Bezugswort folgt, wird ein Pattern erstellt.

```
praepos = [{"TAG": {"IN":["APPR","APPRART"]}}, {"OP":"?"},{"OP":"?"}, {"OP":"?"}, {"OP":"?"}, {"OP":"?"}, {"POS": {"NOT_IN":["PUNKT","DET"]}}]
```

`praepos` ist die Bezeichnung des Pattern, das einen Token sucht, dessen Parsing-Tag auf eine Präposition hindeutet (`{"TAG": {"IN":["APPR","APPRART"]}}`) gefolgt von einem beliebigen Token, das keine Interpunktion oder Artikel sein darf (`{"POS": {"NOT_IN":["PUNCT","DET"]}}`). Bezugswörter werden nicht immer zwangsläufig als Nomen erkannt, daher ist jede Wortart erlaubt außer die beiden Genannten. `{"OP":"?"}` sind optionale Token, die vorkommen können. Ein Pattern ohne optionale Token würde nur ein Match finden, wenn Präposition und Bezugswort direkt hintereinander vorkommen. Da jedoch durchaus Adjektive zwischen Präposition und Bezugswort auftreten können, sind die optionalen Token notwendig.

Wenn eine Texteinheit eine Präposition enthält, aber das Präpositionsmuster nicht erfüllt, wird die Texteinheit als unvollständiger Satz behandelt. In dem nachfolgenden Beispielcode enthalten zwei Sätze Präpositionen und sollen auf Vollständigkeit untersucht werden. Der erste Satz ist offensichtlich unvollständig, da auf die Präposition "über" noch ein Bezugswort folgen müsste. Der Matcher gibt an, dass im zweiten Satz zwei Treffer für das Muster *praepos* gefunden wurden. Da im ersten Satz keine Treffer gefunden wurden, würde dieser als unvollständig gekennzeichnet werden.

```python
import spacy
from spacy.matcher import Matcher

nlp = spacy.load("de_core_news_sm")
matcher = Matcher(nlp.vocab)

praepos = [{"TAG": {"IN":["APPR","APPRART"]}}, {"OP":"?"},{"OP":"?"}, {"OP":"?"}, {"OP":"?"}, {"OP":"?"}, {"POS": {"NOT_IN":["PUNCT","DET"]}}]

matcher.add("praepos", [praepos])

doc = nlp("Wir sollten ueber. Wir sollten ueber das Tool sprechen.")

matches = matcher(doc)

for match_id, start, end in matches:
    string_id = nlp.vocab.strings[match_id]
    span = doc[start:end]
    print(string_id, start, end, span.text)

# praepos 6 9 ueber das Tool
# praepos 6 10 ueber das Tool sprechen
```

Alle erkannten unvollständigen Sätze werden als destruktive Äußerungen klassifiziert und aus dem Datensatz entfernt. Nach der Durchführung des *Rule-based Matching* werden die Sätze, die Muster eines vollständigen Satzes aufweisen, im zweiten Teil des Preprocessing weiterverarbeitet, um sie für die Merkmalsextraktion vorzubereiten.

5.5 Klassifizierer

Nach dem *Rule-based Matching* wird der zweite Teil des *Preprocessing* durchgeführt. Zunächst werden Stoppwörter aus den Texteinheiten entfernt. Dabei wird nicht wie üblicherweise eine Stoppwortliste verwendet, da Verneinungen, die wichtig für die Berechnung des Sentiment-Werts sind, ansonsten entfernt werden. Aus diesem Grund werden Stoppwörter mittels ihrer POS-Tags identifiziert und entfernt. Danach werden die Token lemmatisiert und normalisiert.

Um den Polaritätswert einer Aussage zu berechnen, wird der deutschsprachige SentimentWortschatz (kurz SentiWS) Version 2.0 verwendet [59]. Da SentiWS leider nicht spezifisch für das Software Engineering ist, wurde das Lexikon angepasst. Das Wort "Problem" ist laut SentiWS ein negatives Wort. Nach act4teams® ist die Kommunikation, um Probleme anzusprechen und zu diskutieren, frei von destruktivem Verhalten [32]. Daher wurde das Wort "Problem" aus dem Wortschatz entfernt. Es wurden auch Wörter hinzugefügt, die im heutigen Sprachgebrauch verwendet werden, um eine negative Stimmung auszudrücken, jedoch nicht im Wortschatz vertreten sind. Beispiele dafür sind "grottenhässlich" oder "schwachsinnig". Für die Sentiment-Werte hinzugefügter Worte wurden Synonyme aus dem Duden-Wörterbuch bestimmt [10]. Sofern die Synonyme im Wörterbuch vorhanden waren, wurde der Median aus diesen Sentiment-Werten für das zu hinzufügende Wort ausgewählt. Bei einer geraden Anzahl an Sentiment-Werten wurde von den beiden mittleren Werten nach eigenem Empfinden der passendere Wert ausgewählt. Es wurden keine neuen Sentiment-Werte generiert.

Als statistisches Merkmal wurde das Verhältnis aus der Anzahl verwendeter Partikeln zu der Anzahl an Wörtern im Satz berechnet. Gleiches wurde für die positiven und negativen Sentimentwörter sowie für verallgemeinernde Wörter durchgeführt. Zu den inhaltlichen Merkmalen zählt der Tf-idf-Vektor. Die Wörter, die für diesen Vektor verwendet wurden, waren die Partikeln und die negativen Sentimentwörter. Das Auslassen positiver Wörter für den Tf-idf-Vektor wirkte sich positiv auf die Klassifikation aus.

Die Klassifikatoren SVM, Gradient-Boosting und Naive Bayes werden zunächst mit einem Datensatz ohne unvollständige Sätze trainiert und gespeichert. Dafür wird die Machine-Learning-Bibliothek in Python *scikit-learn* verwendet. Die trainierten Modelle werden geladen und treffen eine Vorhersage für jede Äußerung mit Hilfe des Vektors bestehend aus den extrahierten Merkmalen. Jede Aussage wird entweder als *destruktiv* oder *nicht destruktiv* klassifiziert.

Kapitel 6

Evaluation

Das in Kapitel 4 beschriebene und implementierte Konzept wird nun evaluiert. Dafür wird die Performanz nach bestimmten Metriken gemessen und die Anwendung anhand der Ergebnisse bewertet. In Kapitel 6 wird zunächst der Ablauf der Evaluation erläutert und die Ergebnisse vorgestellt. Abschnitt 6.1 beschreibt den Datensatz, der für die Evaluation verwendet wurde. Die Metriken, die für die Messung der Performanz eingesetzt werden, werden in Abschnitt 6.2 angegeben. Eine Erläuterung der Schritte, die bei der Evaluation und Validierung durchlaufen werden, erfolgt in Abschnitt 6.3. Die Ergebnisse aus der Validierung der Klassifikatoren und des Konzepts werden in Abschnitt 6.4 vorgestellt. Abschnitt 6.5 stellt die Ergebnisse der Performanzmessungen für die Klassifikatoren vor und zieht diese für einen Vergleich heran.

6.1 Datenquelle

Für die Evaluation wurde ein Transkript aus einem Meeting verwendet, welches im Rahmen der Veranstaltung *Software-Projekt* der Leibniz Universität Hannover stattgefunden hat [21]. Aufgrund der SARS-CoV2-Pandemie wurden sämtliche Veranstaltungen online angeboten. Dies führte dazu, dass die Meetings des *Software-Projekts* ebenfalls online angehalten wurden. Die Kommunikation der Teilnehmer erfolgte über den Onlinedienst Discord, der über Möglichkeiten für Chat, Sprach und Videokonferenzen verfügt [21]. Herrmann [21] zeichnete die Audiospur des Meetings über einen Bot auf dem Discord Server auf und transkribierte es manuell. Die Teilnehmer haben eine Einverständniserklärung, das eine Weiternutzung der aufgezeichneten Daten für die wissenschaftliche Forschung erlaubt, unterschrieben. Herrmann [21] nahm für seine Bachelorarbeit jedoch ein andere Klassifikation vor. Daher wurden die Aussagen des Transkripts für diese Arbeit von einer erfahrenen objektiven Person manuell klassifiziert.

In Abbildung 6.1 sind die Anteile destruktiver und nicht destruktiver Äu-

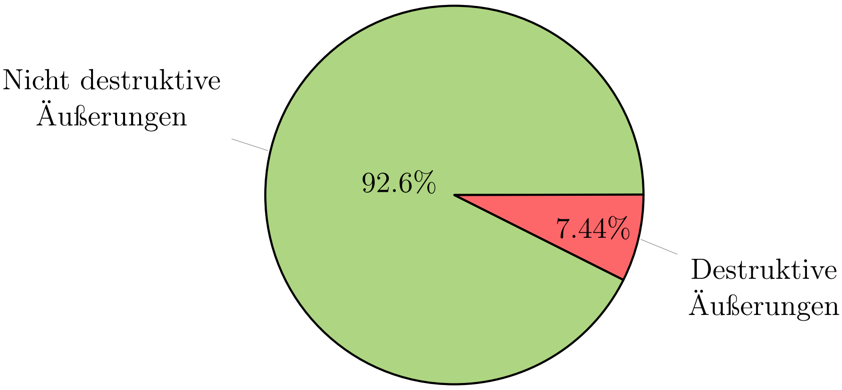


Abbildung 6.1: Anteile der Kategorien im Meeting

ßerungen in einem Kreisdiagramm grafisch dargestellt. Auffällig ist der mit 7.44% geringe Anteil an destruktiven Äußerungen. Das manuell klassifizierte Transkript wird vom Tool eingelesen und für die erste Vorverarbeitung von Rechtschreibfehlern befreit.

6.2 Metriken zur Evaluation

Um die Performanz der Klassifizierer zu messen und zu vergleichen, werden die folgenden Metriken verwendet. Die **Accuracy** drückt die Genauigkeit eines Klassifizierers aus und gibt an, wie groß der Anteil an richtigen Vorhersagen ist. Wie in der unten stehenden Formel 6.1 dargestellt wird dazu die Anzahl korrekter Vorhersagen durch die Anzahl aller Vorhersagen geteilt. Dieses Maß allein reicht nicht aus, da ein Klassifikator eine hohe Genauigkeit erzielen könnte, obwohl dieser nicht in der Lage ist, eine Klasse zu erkennen.

$$\text{Accuracy} = \frac{\text{Korrekte Vorhersagen}}{\text{Alle Vorhersagen}} \tag{6.1}$$

Aus diesem Grund sollte ebenfalls die Konfusionsmatrix (engl. **Confusion Matrix**) betrachtet werden. Die Abbildung zeigt, wie viele der Vorhersagen auch der tatsächlichen Klassifizierungen entsprechen. In dieser Arbeit werden die Konfusionsmatritzen, die in Abschnitt 6.5 vorgestellt werden, wie in Abbildung 6.2 dargestellt.

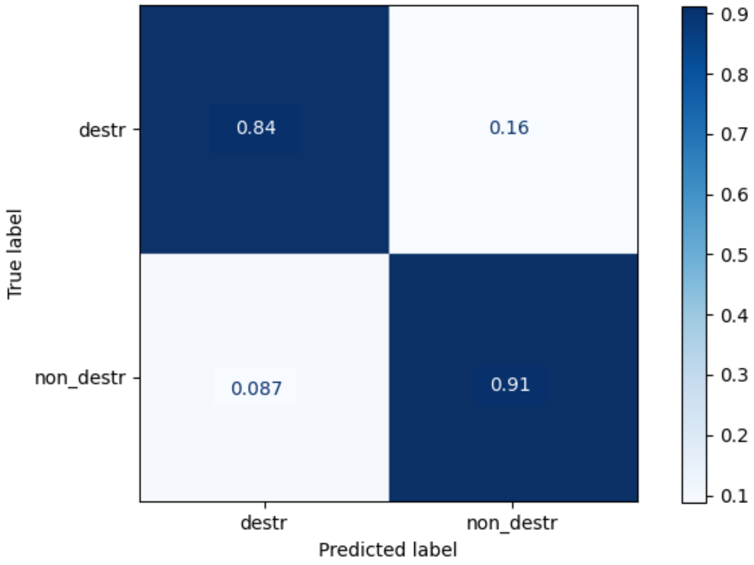


Abbildung 6.2: Beispiel einer Confusion Matrix

Die tatsächlichen Klassen befinden auf der linken Seite und die vorhergesagten Klassen sind auf der unteren Seite zu erkennen. Die Färbung der Quadranten soll den prozentualen Anteil grafisch hervorheben. Die Skala für die Färbung findet sich rechts neben der Confusion Matrix. Aus dem ersten Quadranten der Confusion Matrix lässt sich herauslesen, dass 84% Aussagen aus der vorhergesagten Klasse *destr* auch der tatsächlichen Klasse entsprechen. Demnach wurden 16% der Aussagen der Klasse *destr* fälschlicherweise als *non_ destr* vorhergesagt.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.2}$$

Die Präzision (engl. precision) gibt an, wie viele Vorhersagen einer Klasse korrekt vorhergesagt wurden. Um die Präzision zu berechnen wird die Anzahl der korrekt vorhergesagten Fälle (*True Positives*, kurz: TP) durch die Anzahl aller Vorhersagen einer Klasse (*True Positives* + *False Positives*, kurz: TP + FP) geteilt, wie in der Formel 6.2 angegeben.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.3}$$

Die Sensitivität (engl. recall) aus der Formel 6.3 beschreibt den Anteil korrekt vorhergesagter Fälle einer Klasse. Die Anzahl an korrekt vorhergesagten Fällen wird geteilt durch die Anzahl aller Fälle einer Klasse, das der Summe aus korrekt vorhergesagter Fälle und falsch vorhergesagter Fälle entspricht (*True Positives* + *False Negatives*, kurz: TP + FN).

$$\mathrm{F}_1 = 2 * \left(\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}\right) \tag{6.4}$$

Das F_1-Maß bildet das harmonische Mittel aus Precision und Recall und gewichtet beide Werte gleich. Dadurch ist es möglich, die Perfomance von Klassifizierern anhand eines Wertes zu vergleichen. Die Gleichung 6.4 gibt die Formel zur Berechnung des F_1-Maßes an.

6.3 Vorgehensweise der Evaluation und Validierung

Das eingelesene Transkript durchläuft den ersten Teil der Vorverarbeitung und wird danach auf unvollständige Sätze überprüft. Diese werden zunächst aus dem Datensatz entfernt. Nach dem zweiten Teil der Vorverarbeitung, werden die Merkmale extrahiert und für die maschinelle Klassifikation vorbereitet. Die trainierten Modelle *Support Vector Machine*, *Gradient Boosting* und *Naive Bayes* werden geladen und treffen anhand der Merkmale eine Vorhersage bezüglich der Klassenzugehörigkeit. Die klassifizierten Aussagen aus dem *Rule-based Matching* und *Machine Learning* werden vereint, damit sie zusammen ausgewertet werden. Sowohl die Validierung als auch die Evaluation des Konzepts werden auf diese Weise durchgeführt. Im Abschnitt 6.5 werden die Ergebnisse der Performanzmessung für das gesamte Konzept anhand der in Abschnitt 6.2 genannten Metriken vorgestellt.

6.4 Validierung der Klassifizerer und des Konzepts

Vor der Evaluation wurden beide Ansätze zur Klassifikation validiert. Die beiden in Abschnitt 5.2 beschriebenen Datensätze wurden in Trainings- und Validierungsdatensätze aufgeteilt. Die Vorhersagen der Klassifizierer wurden mit dem Machine-Learning-Datensatz nach dem Training validiert, um Fehler und Schwächen aufzudecken und zu beheben. Auf diese Weise konnte eine Verbesserung der Performanz durch Anpassung des Sentiment-Lexikons festgestellt werden. Zusätzlich wurden für jeden Klassifizierer Hyperparameter eingestellt, um die bestmögliche Performanz zu erreichen.

	Accuracy	Klasse	Precision	Recall	F1-Score
DUMMY	60.67	destr	0	0	0
		non_destr	0.61	1.00	0.76
SVM	74.67	destr	0.74	0.54	0.63
		non_destr	0.75	0.88	0.81
Gradient Boosting	74.00	destr	0.78	0.47	0.59
		non_destr	0.73	0.91	0.81
Bayes	69.33	destr	0.68	0.42	0.52
		non_destr	0.70	0.87	0.77

Tabelle 6.1: Performanzwerte nach dem Training der Klassifizierer

Nach dem Training der Klassifizierer wurde eine Validierung mit dem Testset durchgeführt. Die Ergebnisse sind in Tabelle 6.1 dargestellt. Um zu gewährleisten, dass die ausgewählten Klassifikatoren besser sind als eine willkürliche Klassifikation, wurde zusätzlich ein DUMMY-Klassifikator hinzugefügt. Dieser sagt für jede Äußerung die am häufigsten vorkommende Klasse innerhalb des Datensatzes vorher. In diesem Fall wird jede Aussage vom DUMMY-Klassifikator als nicht destruktiv vorhergesagt. Bei einem ausgeglichenen Datensatz müsste daher der DUMMY-Klassifikator eine Accuracy von ca. 50% erreichen. Da das Testset 30% der Texteinheiten aus dem Gesamtdatensatz bekommt, ist es zu erwarten, dass nicht exakt 50% destruktive Äußerungen enthalten sind. Dies würde bedeuten, dass bei einer konstanten Klassifikation des Labels *non_destr* die Accuracy von 60.67% auch dem tatsächlichen Anteil an nicht destruktiven Äußerungen im Testset entspricht. Der DUMMY-Klassifikator erreicht einen Recall-Wert von 100%, da jede nicht destruktive Aussage auch als solches klassifiziert wird. Doch die Präzision liegt nur bei 61% wegen allen destruktiven Aussagen, die fälschlicherweise als nicht destruktiv erkannt wurden. Aus dem Precision- und Recall-Wert für die Klasse *non_destr* ergibt sich ein F1-Wert von 76%. Doch für die Klasse *destr* wird ein F1-Wert von 0 berechnet, da der DUMMY-Klassifikator kein einziges Mal die Klasse *destr* vorhergesagt hat. An den Accuracy-Werten der Tabelle 6.1 ist erkennbar, dass die Klassifikatoren SVM, Gradient Boosting und Naive Bayes eine bessere Accuracy haben als der DUMMY-Klassifikator und somit zunächst besser scheinen als eine willkürliche Klassifikation. Anhand der F1-Werte für die Klasse *non_destr* erkennt man, dass die drei Klassifikatoren mindestens genauso gut sind wie der DUMMY. Die F1-Werte der Klasse *destr* zeigen, dass die drei Klassifikatoren eine wesentlich bessere Leistung als der DUMMY erbringen. Betrachtet man die F1-Scores der Klassen für jeden Klassifikator ist ersichtlich, dass SVM und Gradient Boosting eine sehr ähnliche Performanz besitzen. Bayes besitzt mit einem F1-Score von 0.52 für die Klasse *destr* und 0.77 für die Klasse *non_ destr* etwas geringere Werte als die beiden anderen Klassifikatoren. Mit der Erkenntnis, dass die drei ausgewählten Klassifikatoren eine bessere Performanz erreichen als der DUMMY, werden die trainierten Modelle für die nächste Validierung gespeichert.

	Accuracy	Klasse	Precision	Recall	F1-Score
SVM	69.75	destr	0.77	0.57	0.65
		non_destr	0.65	0.83	0.73
Gradient Boosting	69.75	destr	0.82	0.51	0.63
		non_destr	0.64	0.88	0.74
Bayes	70.25	destr	0.80	0.55	0.65
		non_destr	0.65	0.86	0.74

Tabelle 6.2: Performanzmaße des Konzepts mit dem Validierungsdatensatz

Die Ergebnisse der Tabelle 6.2 zeigen die Gesamtperformanz, da die Klassifikationen des Rule-based Matching und des Machine Learning zum Schluss wieder vereint wurden. Aus diesem vereinten Datensatz mit den Vorhersagen wurden die Performanzwerte berechnet.

Aus den Ergebnissen ist erkennbar, dass alle Klassifikatoren eine Accuracy von ca. 70% besitzen. Auch bei den F1-Werten gibt es keine starken Unterschiede. Bezüglich der Klasse *destr* besitzt Gradient-Boosting einen etwas geringeren F1-Wert als die beiden anderen Klassifikatoren, deren F1-Werte gleich sind. Die F1-Werte der Klasse *non_ destr* zeigen, dass Bayes und Gradient-Boosting die gleiche Leistung erbringen. Der SVM-Klassifikator hat einen minimal geringeren Wert. Da sich der verwendete Datensatz vom zuvor Verwendeten nur um die unvollständigen Sätze unterscheidet, sollte bei dieser Validierung festgestellt werden, ob die unvollständigen Sätze die Gesamtperformanz negativ beeinflusst. Vergleicht man die Performanzwerte der Tabelle 6.1 und Tabelle 6.2, ist eine geringe Verschlechterung der Performanz festzustellen. Dies ist zu erwarten, wenn unvollständige Sätze nicht zu 100% entfernt werden.

Für das *Rule-based Matching* wurden keine Performanzwerte wie für die Klassifikatoren berechnet, da das Matching nicht alle destruktiven Sätze erkennen soll. Dennoch ist es möglich, einen Wert wie die Accuracy zu berechnen. Eine Messung zeigt, dass 87.78% der als destruktiv erkannten

Sätze der tatsächlichen Klasse *destr* entsprachen. Es wird also nur ein geringer Teil der Aussagen fälschlicherweise aus dem Datensatz entfernt.

6.5 Ergebnisse der Evaluation

Für die bisherige Validierung des Konzepts wurde ein leicht abgewandelter Trainingsdatensatz verwendet, der zusätzlich unterbrochene Sätze enthält. Nun soll für die Evaluation ein Datensatz verwendet werden, der sich vom Validierungsdatensatz gänzlich unterscheidet. In Abschnitt 6.1 wurde beschrieben, woher die Aussagen stammen und wie dieser Datensatz generiert wurde.

	Accuracy	Klasse	Precision	Recall	F1-Score
SVM	73.74	destr	0.14	0.50	0.22
		non_destr	0.95	0.76	0.84
Gradient Boosting	77.02	destr	0.16	0.50	0.24
		non_destr	0.95	0.79	0.86
Bayes	74.18	destr	0.15	0.53	0.23
		non_destr	0.95	0.76	0.84

Tabelle 6.3: Messwerte der Evaluation für das Konzept

Die Tabelle 6.3 gibt eine Übersicht der Messwerte für die drei Klassifizierer. Die Accuracy der Klassifizierer liegt in einem Bereich zwischen 73.74% und 77.02%. Die höchste Accuracy mit 77.02% erreicht die Klassifikation mit dem Gradient-Boosting-Algorithmus. Die Werte für Precision, Recall und F1 bezüglich der Klasse *non_destr* unterscheiden sich bei SVM und Bayes nicht. Gradient Boosting erreicht einen etwas besseren Recall für die Klasse *non_destr*, da es weniger Fehlklassifikationen gibt. Dies kann aus den Konfusionsmatrizen der Abbildungen 6.3 bis 6.5 beobachtet werden, da der Gradient-Bootsing-Klassifikator mit einem Anteil von 20% weniger nicht destruktive Aussagen irrtümlich als destruktiv erkennt. Für die Klasse *destr* erreicht der Gradient-Boosting-Klassifikator ebenfalls den höchsten F1-Score von 0.25. Dieser klassifiziert zwar nicht so viele destruktive Aussagen korrekt wie der Bayes-Klassifikator, dafür gibt es weniger Fehlvorhersagen (vgl. Abbildungen 6.3 bis 6.5) und hat damit eine nur geringfügig bessere Performanz als der Naive Bayes. Zusammenfassend lässt sich sagen, dass der Gradient-Boosting-Algorithmus bei der Accuracy und den beiden F1-Scores leicht höhere Werte erzielt als die anderen beiden Klassifikatoren. Der Bayes-und SVM-Klassifikator liegen in etwa auf dem gleichen Niveau bezüglich der Performanz, doch Bayes erreicht minimal höhere Accuracy- und F1-Werte.

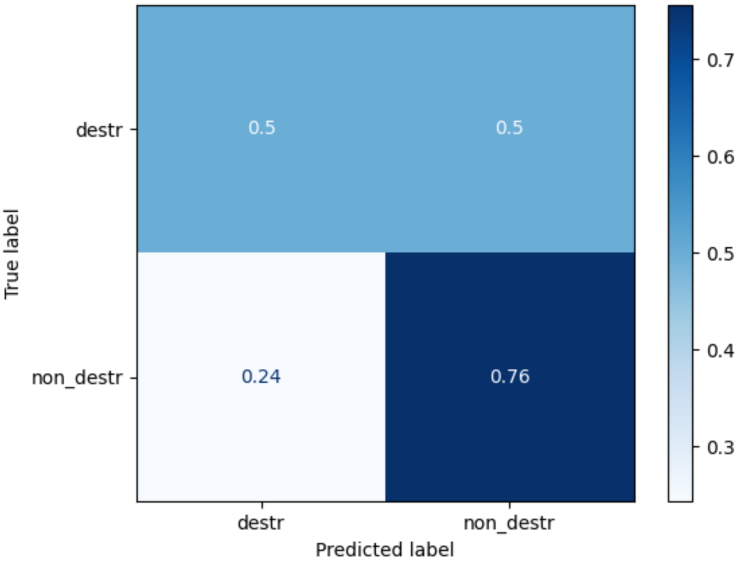


Abbildung 6.3: Confusion Matrix für den SVM-Klassifikator

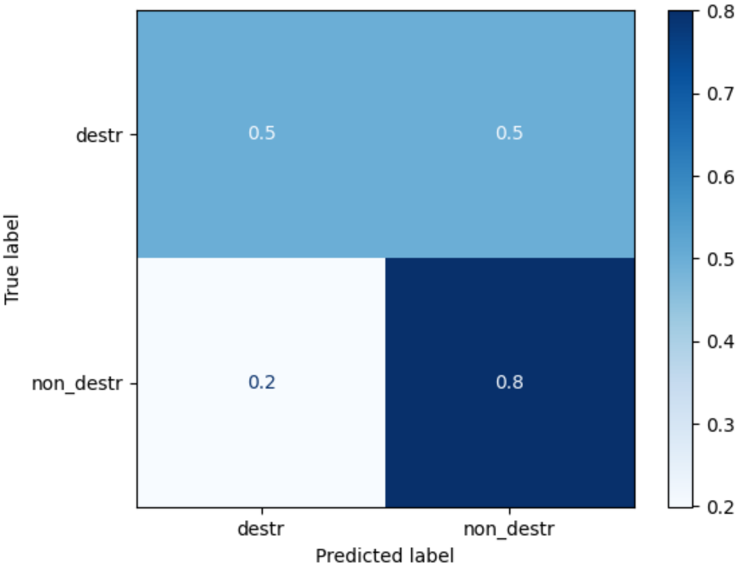


Abbildung 6.4: Confusion Matrix für den Gradient-Boosting-Klassifikator

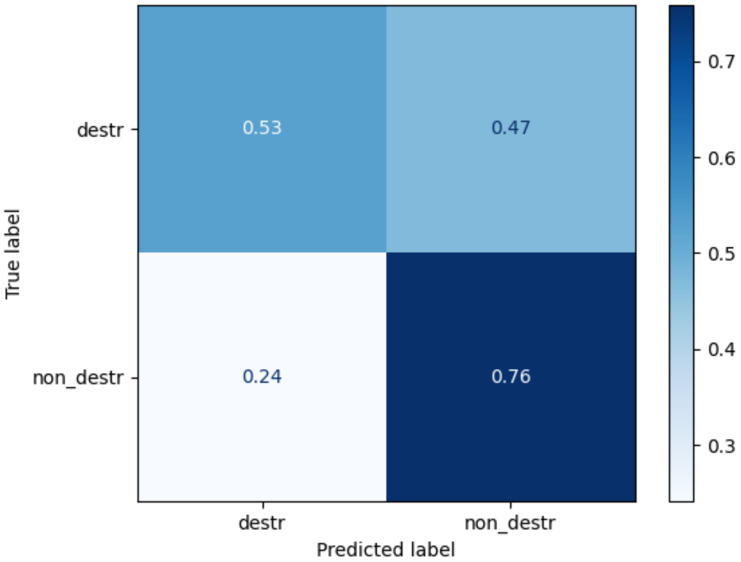


Abbildung 6.5: Confusion Matrix für den Bayes-Klassifikator

Kapitel 7

Diskussion

Das Kapitel Diskussion greift die Ergebnisse aus der Evaluation auf, um sie im Hinblick auf die Anwendbarkeit zu interpretieren. In Abschnitt 7.1 wird betrachtet, wie sich die Messergebnisse aus der Evaluation zu den Ergebnissen aus der Validierung verhalten und welche Schlüsse daraus gezogen werden können. Weiterhin wird diskutiert, in welcher Weise die Subjektivität einen Einfluss auf den Ausgang der Klassifikation nimmt. Abschnitt 7.2 gibt an, welche Faktoren die Allgemeingültigkeit der Ergebnisse gefährden können und welche Vorkehrungen getroffen wurden, um die Schwächen zu reduzieren.

7.1 Interpretation der Ergebnisse

Das Ziel dieser Arbeit ist es, destruktive Äußerungen in Softwareprojekt-Meetings automatisch zu erkennen. In Kapitel 6 wurden die Ergebnisse aus der Validierung und der Evaluation vorgestellt, die nun näher betrachtet und gedeutet werden sollen. Die drei Klassifizierer erreichten bei der Validierung des Konzepts F_1-Werte im Bereich zwischen 0.63 und 0.74 für beide Klassen. Die gleiche Vorgehensweise mit einem anderen Datensatz (für die Evaluation) brachte die F_1-Werte im Bereich zwischen 0.22 und 0.86 für beide Klassen. An den höheren F_1-Werten für die Klasse *nicht destruktiv* ist erkennbar, dass sich die Performanz bezüglich der Erkennung nicht destruktiver Aussagen leicht verbessert hat. Die Fähigkeit destruktive Äußerungen zu erkennen, hat jedoch stark abgenommen. Dies wird deutlich aus den F_1-Werten für die Klasse *destruktiv*, die von ca. 0.63 auf ca. 0.23 gefallen sind.

Für die Verminderung der Performanz bezüglich der Erkennung destruktiver Aussagen kann es mehrere Ursachen geben. Der Validierungsdatensatz wurde von Experten nach dem act4teams®-Kodierschema analysiert und kategorisiert. Anhand dieser Kategorisierung wurden die Aussagen in die zwei Klassen *destruktiv* und *nicht-destruktiv* eingeteilt. In Abschnitt 2.3 wurde definiert, welche act4teams®-Kategorie als destruktiv gewertet wird. Da der Datensatz nur wenige hundert destruktive Aussagen hervorbrachte, wurden sie alle für den zu erstellenden Datensatz ausgewählt und noch einmal überprüft, ob die Kategorisierung angemessen ist. Für einen ausgeglichenen Datensatz wurden in etwa genau so viele nicht destruktive Aussagen zufällig ausgewählt. Auch diese Aussagen wurden auf eine korrekte Kategorisierung überprüft. Alle Änderungen bezüglich der Klassen, die vorgenommen wurden, wurden von mir nach eigenem Empfinden vorgenommen. Der Evaluationsdatensatz musste manuell neu klassifiziert werden, da die von Herrmann [21] vorgenommene Klassifikation einer anderen Definition entsprach. Daher nahm eine erfahrene objektive Person die Neuklassifikation vor. Die Vermutung liegt nahe, dass die Klassifikation stark von der Subjektivität abhängig ist, da die zwei Datensätze von zwei verschiedenen Personen klassifiziert wurden.

Um diese Annahme zu untersuchen, wurde ein Auszug mit ca. 200 Äußerungen aus dem Evaluationsdatensatz entnommen und von fünf Beobachtern (engl. Rater) klassifiziert. Den Ratern wurde eine Datei mit den zu klassifizierenden Aussagen und eine Anleitung mit der Definition der Klassen und Beispielen bereitgestellt. Um die Übereinstimmung unter den fünf Ratern zu ermitteln, wird aus den Ergebnissen Fleiss' Kappa [16] berechnet. Fleiss' Kappa berechnet den Grad der Zuverlässigkeit der Übereinstimmung zwischen mehr als zwei Ratern [16]. Dabei wird auch berücksichtigt, dass Rater zufällig übereinstimmen könnten und dass diese zufallsbedingten Übereinstimmungen keinen Einfluss auf den berechneten Wert nehmen soll. Fleiss' Kappa wurde nach der Gleichung 7.1 [16] berechnet. Für die Berechnung wird von der relativen Übereinstimmung (p_0) die zufallsbedingte Übereinstimmung (p_e) subtrahiert und durch die maximal erreichbare Übereinstimmung ($1 - p_e$) dividiert.

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{7.1}$$

Die Auswertung der Ergebnisse ergab einen κ-Wert von 0.19. Der Grad der Übereinstimmung ist nach der von Landis und Koch [41] vorgeschlagenen Interpretation aus der Tabelle 7.1 schwach bzw. geringfügig (*slight*). Daraus lässt sich schlussfolgern, dass eine Klassifikation von Äußerungen anhand eines Transkripts von der subjektiven Wahrnehmung abhängig ist. Zu einer ähnlichen Erkenntnis kamen Imtiaz et al. [26], die in ihrer wissenschaftlichen Studie Sentiment und Politeness Analysis Tools untersuchten. Bei der Generierung eines Goldstandards aus Nachrichten von Entwicklern durch menschliche Rater konnte eine geringe Übereinstimmung der menschlichen Rater nachgewiesen werden. Letztlich wurde gezeigt, dass die verfügbaren Tools unzuverlässig sind und dass sogar die Klassifikation der menschlichen Rater widersprüchlich sind [26]. Eine weitere Schwierigkeit stellt die Erkennung einer Stimmung einer transkribierten Äußerung dar. Die Vermutung liegt nahe, dass die Betonung einzelner Wörter oder der Tonfall einer Aussage eine bestimmte Stimmung oder Bedeutung vermitteln können.

Kappa Statistic	Strength of Agreement
<0.00	Poor
0.00-0.20	Slight
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Substanial
0.81-1.00	Almost Perfect

Tabelle 7.1: Interpretation der κ-Werte nach Landis und Koch [41]

Auch Mimik und Gestik könnten ein Rolle spielen. Beispielsweise kann die Aussage „Das ist ja wunderbar.“ eine negative Stimmung vermitteln, wenn dabei die Augen gerollt werden oder ein sarkastischer Ton gewählt wird. Da diese Metainformationen den Ratern jedoch nicht vorliegen, muss eine Entscheidung nach eigenem Ermessen getroffen werden. Der Vorteil einer Klassifiaktion von Aussagen durch einen Beobachter gegenüber einer automatischen Klassifikation, ist die Fähigkeit diese Metainformationen bei der Entscheidung berücksichtigen zu können. Diese fehlen in Transkripten jedoch gänzlich. Die Erkennung von Ironie und Sarkasmus ist eine bekannte Herausforderung in der Sentiment Analyse und ist in der Forschung unter dem Begriff *Automatic Sarcasm Detection* bekannt[6, 15, 29, 44].

Darüber hinaus ist das *Rule-based Matching* stark von der eigenen Wahrnehmung abhängig, da die Regeln selbst erstellt wurden. Sätze, die mit Hilfe der Regeln identifiziert werden, sind nur aus der Sicht der Person unvollständig, die diese Regeln erstellt hat. Unvollständige Sätze sind schwierig aus Transkripten zu erkennen, da Sprecher sich auch selbst unterbrechen können. In diesem Fall würde kein destruktives Verhalten vorliegen. Aus einem Transkript jedoch ist sehr schwierig zu erkennen, ob ein Sprecher seinen Satz absichtlich nicht beendet hat oder aufgrund einer Unterbrechung nicht beenden konnte. Die Erkennung unvollständiger Sätze stellte sich für die deutsche Sprache als sehr schwierig heraus. Die in Unterabschnitt 2.4.3 beschriebenen “Standard-Problemfälle” erschwerten die Erstellung von Regeln für vollständige Sätze, da es mehrere Ausnahmen gibt, wie beispielsweise das Weglassen eines Prädikats in einem Satz. Obwohl diese Sätze per Definition unvollständig wären, gelten sie trotzdem als vollständig. Zudem funktionieren diese Regeln nur, wenn die Parsing- und POS-Tags absolut korrekt sind. Da der Parser und Tagger jedoch auf einem statistischen Modell beruhen, könnten Tags falsch gesetzt werden. Dies hätte zur Folge, dass ein vollständiger Satz fälschlicherweise als unvollständig erkannt wird. Auch der umgekehrte Fall ist möglich. Aus diesem Grund war die Klassifikation des Matchers schon während der Validierung nicht fehlerfrei. Die Vielfalt und Komplexität der deutschen Grammatik hat gezeigt, dass das Pattern Matching für die Erkennung unvollständiger

Sätze wenig geeignet scheint. In der zukünftigen Forschung sollten andere Alternativen untersucht werden.

7.2 Threats to Validity

Die Interpretation der Evaluation zeigt, dass es Faktoren gibt, die einen Einfluss auf die Validität der Ergebnisse haben. Im Folgenden werden die Faktoren genannt, die die Allgemeingültigkeit der Ergebnisse gefährden können.

Construct Validity

Der Evaluationsdatensatz entstand im Rahmen der Arbeit von Herrmann [21]. Nach seinen Angaben kannten sich vier der sechs Teilnehmer bereits und waren befreundet. Häufig lässt sich beobachten, dass die Ausdrucksweise sich je nach persönlicher Beziehung zum Gesprächspartner ändert. Wenn Gesprächspartner, die sich gut kennen, eine für sie normale derbe Ausdrucksweise wählen, könnte das die Klassifikation negativ beeinflussen. Zudem wäre es möglich, dass sich die Teilnehmer anders verhalten haben, da sie wussten, dass das Meeting aufgezeichnet wird [21]. Es wurde sogar von einem Teilnehmer selbst angemerkt. Insofern würde das Meetingtranskript für die Evaluation nicht der Realität entsprechen. Um eine Beeinflussung der Ergebnisse durch eine einzelne Person zu verhindern, wurde der Evaluationsdatensatz von einer anderen Person klassifiziert. Wie bereits erwähnt waren die Datensätze nur auf studentische Entwicklerteams beschränkt. Für eine bessere Konstruktvalidität sollte der Datensatz von Entwicklerteams aus verschiedenen Bereichen stammen. Zudem wurde in dieser Arbeit das destruktive Verhalten nur mit einer Methode gemessen. Die Äußerungen wurden nur durch das implementierte Tool klassifiziert. Daher liegt ein *mono-method bias* vor. Herrmann [21] merkte in seiner Arbeit an, dass das zu untersuchende Meeting während der SARS-CoV2-Pandemie stattfand und daher online über Discord abgehalten wurde. Da jeder Teilnehmer ein eigenes Mikrofon besitzt, kann die Tonqualität der einzelnen Teilnehmer variieren. Eine schlechte Tonqualität könnte sich negativ auf das Ergebnis der automatischen Spracherkennung auswirken und somit auch die Klassifikation verschlechtern. Um dieses Problem vorzubeugen, wurde für die Evaluation ein manuell erstelltes Transkript verwendet.

Internal Validity

Die subjektive Wahrnehmung hat großen Einfluss auf die Klassifikation. So wurden der Trainingsdatensatz, die Regeln zur Erkennung von unvollständigen Sätzen und das Sentiment-Lexikon SentiWS nach eigenem Empfinden erstellt oder modifiziert. Die Klassifikation des Evaluationstranskripts durch eine andere Person brachte daher merkbare Abweichungen bezüglich der Performanzergebnisse in der Validierung und der Evaluation.

Conclusion Validity

Der Datensatz für das Training und die Validierung wurde zwar von Experten kategorisiert, doch diese Kategorisierung wurde zum Teil modifiziert. Dieser Datensatz und der Evaluationsdatensatz wurden jeweils von nur einer Person klassifiziert. Darüber hinaus wurde die Klassifikation des Tools nur an einem Meetingtranskript durchgeführt und evaluiert. Wie zuvor angemerkt, wurde das Meeting aus dem *Software-Projekts* online über Discord abgehalten, da die Veranstaltung während der SARS-CoV2-Pandemie stattfand. Man beobachtet, dass sich Online-Meetings von Präsenz-Meetings unterscheiden. Bei einem Präsenz-Meeting befinden sich alle Teilnehmer im gleichen Raum und teilen daher dieselben Faktoren aus der Umgebung, wie beispielsweise Geräusche. Die räumliche Trennung der Teilnehmer von Online-Meetings sorgt dafür, dass jeder unterschiedlichen Umgebungsfaktoren ausgesetzt ist und dadurch die Aufmerksamkeit und Konzentration einzelner Teilnehmer leichter gestört werden kann.

All die genannten Faktoren können einen negativen Einfluss auf die statistische Validität (Conclusion Validity) haben.

External Validity

Alle verwendeten Datensätze zum Trainieren, Validieren oder Evaluieren stammen aus Meetings studentischer Entwicklerteams. Es wäre möglich, dass sich die Kommunikation in beruflichen Entwicklerteams unterscheidet. Auch das Alter der Entwickler könnte die Kommunikation beeinflussen. Es fällt auf, dass junge Erwachsene sich anders artikulieren als Menschen mittleren Alters, da sich die Sprache mit der Zeit ein wenig verändert. Aus den beiden zuvorgenannten Gründen könnten sich die Ergebnisse bei der Anwendung des Tool auf berufliche Softwareprojekt-Meetings unterscheiden. Um eine bessere Allgemeingültigkeit zu erreichen, wäre die Verwendung eines größeren und vielfältigeren Datensatzes empfehlenswert. Die Ergebnisse könnten auch dann eine geringere Allgemeingültigkeit besitzen, wenn Entwicklerteams eine andere Toleranzgrenze für destruktive Aussagen haben als andere Entwicklerteams. Objektiv betrachtet könnten die Entwickler destruktive Äußerungen tätigen, doch sie selbst empfinden es als normalen Umgangston.

Das Ziel der Einzelfallstudie war, die Anwendbarkeit des Verfahren zu zeigen ohne dabei Rückschlüsse für andere Meetings zu ziehen. Die Ergebnisse zeigen, dass das Verfahren anwendbar ist und dass es Potential zu Verbesserungen gibt.

Kapitel 8

Fazit

Dieses Kapitel soll in einem Fazit erläutern, welche Erkenntnisse aus dieser Arbeit im Hinblick auf das Forschungsziel gewonnen werden konnten. Dafür wird zunächst in Abschnitt 8.1 das entwickelte Konzept und die Ergebnisse aus dessen Analyse zusammengefasst. Zum Abschluss bietet Abschnitt 8.2 einen Ausblick über zukünftige Arbeiten für diesen Forschungsbereich.

8.1 Zusammenfassung

Meetings bieten in Softwareprojekten eine der effektivsten Arten der Kommunikation. Durch die Kommunikation in Meetings können auf schnelle und unkomplizierte Art Informationen ausgetauscht oder Aufgaben koordiniert werden. Analysen von Kommunikation in Meetings haben gezeigt, dass es auch eine negativ beitragende Art der Kommunikation gibt. Diese destruktiven Äußerungen können den Erfolg eines Meeting gefährden und zu einer negativen und pessimistischen Haltung der Entwickler führen. Ferner können Meetings mit einem Übermaß an destruktivem Verhalten sogar gesundheitsgefährdend sein. Diese Probleme gefährden im schlimmsten Fall den Erfolg des gesamten Projekts. Im Bereich des Software Engineering kann mit act4teams®-SHORT die Kommunikation in Meetings analysiert werden. Ein Beobachter wohnt dem Meeting bei und ordnet in Echtzeit relevante Aussagen in eine der neun Kategorien ein. Unter anderem kann damit auch der Anteil an destruktivem Verhalten in einem Meeting gemessen werden. Durch diese Methode kann jedoch nicht jede Aussage bewertet werden, da ein Mensch die dafür nötige kognitive Leistung nicht erbringen kann. Aus diesem Grund wurde in dieser Arbeit ein Konzept entwickelt, destruktive Äußerungen automatisiert zu erkennen.

Bei der Untersuchung der Texteigenschaften, die destruktive Äußerungen aufweisen, wurde klar, dass die Verwendung einer alleinigen Methode für den Klassifikationsprozess nicht ausreichend ist. Daher wurde ein hybrider Ansatz gewählt, bei dem *Rule-based Matching* und *maschinelle Lernverfahren* eingesetzt werden. Der Klassifikationsprozess beginnt mit der Vorverarbeitung der Texteinheiten durch *Natural Language Processing*. Da sich Unterbrechungen durch unvollständige Sätze auszeichnen, werden sie mit Hilfe von Pattern Matching auf charakteristische syntaktische Strukturen untersucht. Als unvollständig erkannte Sätze werden als destruktiv klassifiziert. Aus den vollständigen Sätzen werden Merkmale für die *Machine-Learning-Classifier* SVM, Gradient-Boosting und Naive Bayes extrahiert. Die vortrainierten Modelle klassifizieren die Äußerungen mit Hilfe des Merkmalsvektors als *destruktiv* oder *nicht destruktiv*. Der implementierte Prototyp wurde mit einem Meeting-Transkript eines studentischen Entwicklerteams evaluiert. Der Prototyp zeigte eine leicht verbesserte Performanz bezüglich der Klassifikation nicht destruktiver Aussagen des Evaluationsdatensatz. Der F1-Wert stieg von ca. 0.74 in der Validierung auf ca. 0.84 in der Evaluation. Bei der Klassifikation destruktiver Aussagen allerdings verschlechterte sich die Performanz des Prototyps bei der Evaluation. Der F1-Wert sank von ca. 0.64 in der Validierung auf ca. 0.23 in der Evaluation. Für diese Beobachtung konnten mehrere Gründe identifiziert werden. Da der Evaluations- und Validierungsdatensatz von zwei verschiedenen Personen gelabelt wurden, liegt die Vermutung nahe, dass die Subjektivität einen Einfluss auf die Güte der Klassifikation haben könnte. Eine manuelle Klassifikation eines Auszugs aus dem Meeting-Transkript durch menschliche Rater zeigte, dass die Rater eine schwache Übereinstimmung in Bezug auf ihre Klassifikationen hatten. Diese Beobachtung unterstützt die Annahme, dass Subjektivität die Klassifikation beeinflussen kann. Eine weitere Schwachstelle stellt das Pattern Matching dar, das ebenso von der subjektiven Wahrnehmung unvollständiger Sätze abhängig ist. Die deutsche Sprache ist so vielseitig, dass es schwierig wäre, für jede spezielle syntaktische Besonderheit eine Regel zu erstellen. Zudem ist aus einem Transkript nicht erkennbar, ob ein Sprecher unterbrochen wird oder sich selbst unterbricht, insbesondere wenn die Interpunktion wegen der automatischen Sprecherkennung nicht vorhanden ist. Das Pattern Matching ist zudem stark abhängig von den Tags des Dependency-Parsers und des POS-Taggers. Da beide jedoch auf statistischen Modellen beruhen, ist es möglich, dass Tag falsch gesetzt werden und somit zu fehlerhaften Ausgaben des Matchers führen. Die Evaluation zeigte, dass das Konzept anwendbar ist, aber auch Potenzial für Verbesserung bietet. Beispielsweise könnten zur Erkennung von Unterbrechungen die Häufigkeit der Sprecherüberschneidungen bei der automatischen Spracherkennung ermittelt werden. Auf diese Weise könnten Unterbrechungen zuverlässiger erkannt werden.

8.2 Ausblick

Bei der Untersuchung des Konzepts konnten mehrere Möglichkeiten für Verbesserungen entdeckt werden. Die Methode zur Erkennung unvollständiger Sätze durch Pattern Matching scheint zu sehr von der subjektiven Wahrnehmung abzuhängen. Die Vielfalt der deutschen Sprache lässt eine eindeutige Erkennung unvollständiger Sätze durch die Analyse syntaktischer Satzstrukturen kaum zu. Aus diesem Grund wäre die Betrachtung der Sprecherüberschneidungen einer Alternative dazu. Die automatische Sprecherkennung könnte eine Möglichkeit bieten, Sprecherüberschneidungen automatisch zu erfassen.

Für dieses Konzept wurden nur Unigramme betrachtet, da eine wissenschaftliche Publikation [66] zeigte, dass mit höherem n bezüglich N-Gramme schlechtere Genauigkeiten bei der Klassifikation erzielt wurden. Dies betrifft jedoch nicht deutschsprachige Texte. Es könnte daher untersucht werden, ob dies für die deutsche Sprache ebenso der Fall ist. Es ist zu beobachten, dass die Worte ”schon“ oder “wieder” nicht überwiegend in destruktiven oder nicht destruktiven Äußerungen verwendet werden. Doch bei der Kombination “schon wieder” könnte der Eindruck einer tadelnden Äußerung entstehen. Daher könnte in Zukunft die Klassifikation mit N-Grammen verschiedener n-Größen untersucht werden.

Auch die Klassifikation durch maschinelle Lernverfahren ist durch die gelabelten Trainingsdatensätze abhängig. Die Datensätze könnten von mehreren Personen klassifiziert werden, doch auch dies garantiert keine Verbesserung, da sich menschliche Rater häufig uneinig sein können. Darüber hinaus beeinflusst das verwendete Sentiment-Lexikon ebenso die Klassifikation. Das verwendete SentiWS ist ein deutschsprachiges Lexikon, das jedoch nicht spezifisch für das Software Engineering ist. Momentan existieren nur englischsprachige Alternativen für das Software Engineering. Im Rahmen dieser Arbeit zeigten kleine manuelle Modifikationen des Lexikons Verbesserung bei der Klassifikation. Daher wäre es möglich, eine noch bessere Genauigkeit zu erzielen, wenn ein deutschsprachiges Sentiment-Lexikon aus dem Software-Engineering-Bereich verwendet wird.

In der zukünftigen Forschung soll eine automatisierte Erkennung von Äußerungen in Softwareprojekt-Meetings angestrebt werden, um den Arbeits- und Zeitaufwand zu reduzieren. Dafür sollen diese Äußerungen allen act4teams®-SHORT-Kategorien zugewiesen werden können. Diese Arbeit, die zwischen destruktiven und nicht destruktiven Aussagen unterscheidet, stellt den ersten Ansatz dar, um das übergeordnete Ziel zu erreichen. Die in dieser Arbeit gewonnenen Forschungsergebnisse dienen daher als Grundlage für zukünftige Arbeiten, die eine Unterscheidung mehrerer Kategorien anstreben.

Anhang A

Modifikaton des Sentiment-Lexikons

Für eine bessere Genauigkeit der Klassifikation mit maschinellen Lernverfahren wurde das Sentiment-Lexikon *SentiWS* angepasst. Die folgende Aufzählung gibt eine Übersicht über hinzugefügte und entfernte Wörter. Bei den hinzugefügten Wörtern sind zusätzlich die Synonyme mit ihren Sentiment-Werten in Klammern angegeben.

Positive Wörter

Hinzugefügt:

+ mega (riesig 0.4554)
+ geil (riesig 0.4554)

Negative Wörter

Hinzugefügt:

+ Aufwand (Anstrengung -0.0048)
+ Dummi (Dummkopf -0.0048)
+ Abfuck (Scheiß -0.3381)
+ Schwachsinn (Unsinn -0.4969)
+ Quatsch (Unsinn -0.4969)
+ Blödsinn (Unsinn -0.4969)
+ Stuss (Unsinn -0.4969)
+ grottenhässliche (hässlich -0.4387)
+ fuck (scheiße -0.2945)
+ shit (scheiße -0.2945)

+ umbringen (töten -0.5203)
+ schwachsinnig (unsinnig -0.3516)
+ ätzend (ärgerlich -0.3111)

Entfernt:

– Abnahme
– Dilemma
– Ende
– Fehler
– Fehlermeldung
– Krieg
– Problem
– Risiko
– abnehmen
– alt
– auseinandersetzen
– ausschließen
– einstellen
– fallen
– farblos
– grob
– kippen
– klein
– knapp
– krass
– kurz
– leer
– leider
– löschen
– problematisch
– rückwärts
– trennen
– unten
– verfälschen
– wegfallen

Literatur

[1] Yassine Al Amrani, Mohamed Lazaar und Kamal Eddine El Kadiri. “Random forest and support vector machine based hybrid approach to sentiment analysis”. In: *Procedia Computer Science* 127 (2018), S. 511–520.

[2] Mehdi Allahyari et al. “A brief survey of text mining: Classification, clustering and extraction techniques”. In: *arXiv preprint arXiv:1707.02919* (2017).

[3] Scott Ambler. *Agile modeling: effective practices for extreme programming and the unified process.* John Wiley & Sons, 2002.

[4] J. Appel et al. *Gabler Büro Lexikon.* Gabler Verlag, 2013. ISBN: 9783322837721. URL: `https://books.google.de/books?id=RyPTBgAAQBAJ`.

[5] David Baccarini. “The logical framework method for defining project success”. In: *Project management journal* 30.4 (1999), S. 25–32.

[6] Rushlene Kaur Bakshi et al. “Opinion mining and sentiment analysis”. In: *2016 3rd international conference on computing for sustainable global development (INDIACom).* IEEE. 2016, S. 452–455.

[7] Fabio Calefato et al. “Sentiment polarity detection for software development”. In: *Empirical Software Engineering* 23.3 (2018), S. 1352–1382.

[8] Wallace Chafe und Deborah Tannen. “The relation between written and spoken language”. In: *Annual review of anthropology* 16.1 (1987), S. 383–407.

[9] Graham Crookes. “The utterance, and other basic units for second language discourse analysis”. In: *Applied linguistics* 11.2 (1990), S. 183–199.

[10] *Duden online.* [Online; accessed 02-Oktober-2021]. o.J. URL: `https://www.duden.de/`.

[11] Christa Dürscheid und Jan Georg Schneider. “Satz, Äußerung, Schema”. In: *E. Felder & A. Gardt (Éd.), Handbuch Sprache und Wissen* (2014), S. 167–194.

[12] Ali Ekramipooya, Davood Rashtchian und Mehrdad Boroushaki. "Kernelled Naïve Bayes Using a Balanced Dataset for Accurate Classification of the Material Toxicity". In: *Advanced Journal of Chemistry-Section A* 4.2 (2021), S. 138–151.

[13] Kristian Enns. "Entwicklung einer Webanwendung für Interaktionsanalyse in Meetings". Bachelor's thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2021.

[14] AI Explosion. *Industrial-strength natural language processing.* [Online; accessed 19-Oktober-2021]. 2015. URL: `https://spacy.io/usage/spacy-101`.

[15] DI Hernández Farias und Paolo Rosso. "Irony, sarcasm, and sentiment analysis". In: *Sentiment Analysis in Social Networks.* Elsevier, 2017, S. 113–128.

[16] Joseph L Fleiss. "Measuring nominal scale agreement among many raters." In: *Psychological bulletin* 76.5 (1971), S. 378.

[17] Jerome H Friedman. *The elements of statistical learning: Data mining, inference, and prediction.* springer open, 2017.

[18] Anandasivam Gopal, Tridas Mukhopadhyay und Mayuram S Krishnan. "The role of software processes and communication in offshore software development". In: *Communications of the ACM* 45.4 (2002), S. 193–200.

[19] Amit Gupte et al. "Comparative study of classification algorithms used in sentiment analysis". In: *International Journal of Computer Science and Information Technologies* 5.5 (2014), S. 6261–6264.

[20] Ann-Christin Haak. "Verbesserung der Meetinganalyse in Entwicklerteams unter Verwendung von praktischen Tutorials". In: (2020).

[21] Marc Herrmann. "Automatische Klassifikation von Aussagen in Meetings von Entwicklungsteams". Bachelor's thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2020.

[22] Hajo Hippner und René Rentzmann. "Text mining". In: *Informatik-Spektrum* 29.4 (2006), S. 287–290.

[23] Ludger Hoffmann. *Deutsche Syntax: Ansichten und Aussichten.* Walter de Gruyter GmbH & Co KG, 2019.

[24] Julian Horstmann. "Computer-gestützte Analyse des Kommunikationsverhaltens in Entwicklerteams unter Berücksichtigung digitaler Medien". Magisterarb. Master's thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2019.

[25] Andreas Hotho, Andreas Nürnberger und Gerhard Paaß. "A brief survey of text mining." In: *Ldv Forum.* Bd. 20. 1. Citeseer. 2005, S. 19–62.

[26] Nasif Imtiaz et al. “Sentiment and politeness analysis tools on developer discussions are unreliable, but so are people”. In: *2018 IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*. IEEE. 2018, S. 55–61.

[27] S Inzalkar und Jai Sharma. “A survey on text mining-techniques and application”. In: *International Journal of Research In Science & Engineering* 24 (2015), S. 1–14.

[28] Md Rakibul Islam und Minhaz F Zibran. “Leveraging automated sentiment analysis in software engineering”. In: *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE. 2017, S. 203–214.

[29] Aditya Joshi, Pushpak Bhattacharyya und Mark J Carman. “Automatic sarcasm detection: A survey”. In: *ACM Computing Surveys (CSUR)* 50.5 (2017), S. 1–22.

[30] Yusmadi Yah Jusoh et al. “Communication management in global software development projects”. In: *2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP)*. IEEE. 2018, S. 1–7.

[31] Simone Kauffeld. *Kompetenzen messen, bewerten, entwickeln: Ein prozessanalytischer Ansatz für Gruppen*. Bd. 128. Schäffer-Poeschel, 2006.

[32] Simone Kauffeld und Nale Lehmann-Willenbrock. “Meetings matter: Effects of team meetings on team and organizational success”. In: *Small group research* 43.2 (2012), S. 130–158.

[33] Simone Kauffeld, Nale Lehmann-Willenbrock und Annika L Meinecke. “21 The Advanced Interaction Analysis for Teams (act4teams) Coding Scheme”. In: (2018).

[34] Jil Klünder. *Analyse der Zusammenarbeit in Softwareprojekten mittels Informationsflüssen und Interaktionen in Meetings*. Logos Verlag Berlin GmbH, 2019.

[35] Jil Klünder. *Kapitel 6 - Interaktionsanalysen in Meetings*. Vorlesungsskript. Fachgebiet Software Engineering, 2020.

[36] Jil Klünder et al. “Do you just discuss or do you solve? Meeting analysis in a software project at early stages”. In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 2020, S. 557–562.

[37] Arnd Christian König und Eric Brill. “Reducing the human overhead in text categorization”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, S. 598–603.

[38] Robert E. Kraut und Lynn A. Streeter. “Coordination in Software Development”. In: *Commun. ACM* 38.3 (März 1995), S. 69–81. ISSN: 0001-0782. DOI: `10.1145/203330.203345`. URL: `https://doi.org/10.1145/203330.203345`.

[39] Eklekta Kristo. “Computer-Aided Analysis of Video Comments for Requirements Analysis”. In: *Bachelor thesis, Leibniz Universität Hannover* (2021).

[40] Beate Kuhnt und Andreas Huber. “IT-Projektmanagement”. In: *Eine kommunikative Herausforderung. Herausgegeben von Forum InformatikerInnen für Frieden und gesellschaftliche Verantwortung eV (1/07). Online verfügbar unter www. fiff. de, zuletzt geprüft am.* Bd. 26. 2007, S. 2008.

[41] J Richard Landis und Gary G Koch. “The measurement of observer agreement for categorical data”. In: *biometrics* (1977), S. 159–174.

[42] Bin Lin et al. “Sentiment analysis for software engineering: How far can we go?” In: *Proceedings of the 40th International Conference on Software Engineering.* 2018, S. 94–104.

[43] Katharina Manderscheid. “Text Mining”. In: *Handbuch Methoden der empirischen Sozialforschung.* Hrsg. von Nina Baur und Jörg Blasius. Wiesbaden: Springer Fachmedien Wiesbaden, 2019, S. 1103–1116. ISBN: 978-3-658-21308-4. DOI: `10.1007/978-3-658-21308-4_79`. URL: `https://doi.org/10.1007/978-3-658-21308-4_79`.

[44] Diana G Maynard und Mark A Greenwood. “Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis”. In: *Lrec 2014 proceedings.* ELRA. 2014.

[45] Walaa Medhat, Ahmed Hassan und Hoda Korashy. “Sentiment analysis algorithms and applications: A survey”. In: *Ain Shams engineering journal* 5.4 (2014), S. 1093–1113.

[46] Metro. *Cyberpunk 2077 preview and interview.* [Online; accessed 29-September-2021]. 2009. URL: `https://archive.ph/20190617143332/https://metro.co.uk/2019/06/17/cyberpunk-2077-preview-and-interview-i-just-love-my-work-9973737/#selection-501.0-501.36`.

[47] Christian Meyer. “Verbesserung von evolutionären Algorithmen zur Klassifikation von schriftlicher Kommunikation in Entwicklungsteams”. Magisterarb. Master’s thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2020.

[48] Gabriel Murray. “Using speech-specific characteristics for automatic speech summarization”. Diss. Citeseer, 2008.

[49] Martin Obaidi und Jil Klünder. “Development and Application of Sentiment Analysis Tools in Software Engineering: A Systematic Literature Review”. In: *Evaluation and Assessment in Software Engineering* (2021), S. 80–89.

[50] Ilan Oshri, Julia Kotlarsky und Leslie P Willcocks. “Global software development: Exploring socialization and face-to-face meetings in distributed strategic projects”. In: *The Journal of Strategic Information Systems* 16.1 (2007), S. 25–49.

[51] Bo Pang, Lillian Lee und Shivakumar Vaithyanathan. “Thumbs up? Sentiment classification using machine learning techniques”. In: *arXiv preprint cs/0205070* (2002).

[52] Gerald Petz. *Opinion Mining Im Web 2.0.* Springer, 2019.

[53] Tobias Powelske. “Analyse von Stimmungen in Open-Source-Projekten anhand von text-basierter Kommunikation”. Magisterarb. Master’s thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2020.

[54] Federico Pozzi et al. *Sentiment analysis in social networks.* Morgan Kaufmann, 2016.

[55] Rudy Prabowo und Mike Thelwall. “Sentiment analysis: A combined approach”. In: *Journal of Informetrics* 3.2 (2009), S. 143–157.

[56] Nils Prenner. “Analyse von direkter und dokumentenbasierter Kommunikation in agilen Entwicklerteams”. In: *Leibniz Universität Hannover, Fachgebiet Software Engineering, Masterarbeit* (2017).

[57] Nils Prenner, Jil Klünder und Kurt Schneider. “Making meeting success measurable by participants’ feedback”. In: *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering.* 2018, S. 25–31.

[58] Gisela Redeker. “On differences between spoken and written language”. In: *Discourse processes* 7.1 (1984), S. 43–55.

[59] Robert Remus, Uwe Quasthoff und Gerhard Heyer. “SentiWS - A Publicly Available German-language Resource for Sentiment Analysis”. In: Jan. 2010.

[60] David Zafer Jörg Schiller. “Untersuchung von Zusammenhängen zwischen Stimmungen und Interaktionen von Meetings in Softwareprojekten”. Bachelor’s thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2020.

[61] Kurt Schneider et al. “Positive affect through interactions in meetings: The role of proactive and supportive statements”. In: *Journal of Systems and Software* 143 (2018), S. 59–70.

[62] Eva-Maria Schulte, Tatjana Fenner und Simone Kauffeld. “Nicht ohne Nebenwirkungen: Gesundheitsrisiko Meeting”. In: *Personal Quarterly* 2 (2013), S. 8–15.

[63] Ken Schwaber und Jeff Sutherland. “The scrum guide”. In: *Scrum Alliance* 21.19 (2011), S. 1.

[64] Melanie Siegel und Melpomeni Alexa. “Wörter in der Sentiment-Analyse”. In: *Sentiment-Analyse deutschsprachiger Meinungsäußerungen*. Springer, 2020, S. 37–48.

[65] NK Sreeja und A Sankar. “Pattern matching based classification using ant colony optimization based feature selection”. In: *Applied Soft Computing* 31 (2015), S. 91–102.

[66] Abinash Tripathy, Ankit Agrawal und Santanu Kumar Rath. “Classification of sentiment reviews using n-gram machine learning approach”. In: *Expert Systems with Applications* 57 (2016), S. 117–126.

[67] S Vijayarani, Ms J Ilamathi, Ms Nithya et al. “Preprocessing techniques for text mining-an overview”. In: *International Journal of Computer Science & Communication Networks* 5.1 (2015), S. 7–16.

[68] John M. Wiemann und Mark L. Knapp. “Turn-taking in Conversations”. In: *Journal of Communication* 25.2 (Feb. 2006), S. 75–92. ISSN: 0021-9916. DOI: `10.1111/j.1460-2466.1975.tb00582.x`. eprint: `https://academic.oup.com/joc/article-pdf/25/2/75/22327852/jjnlcom0075.pdf`. URL: `https://doi.org/10.1111/j.1460-2466.1975.tb00582.x`.

[69] Angelika Wöllstein. *Duden - die Grammatik : unentbehrlich für richtiges Deutsch ; Duden Band 4, Die Grammatik, Der Aufbau der deutschen Sprache vom Laut über das Wort und den Satz bis hin zum Text und zu den Merkmalen der gesprochenen Sprache.* Die Grammatik, Der Aufbau der deutschen Sprache vom Laut über das Wort und den Satz bis hin zum Text und zu den Merkmalen der gesprochenen Sprache. Berlin: Dudenverlag; 2016. ISBN: 3411040491. URL: `https://www.tib.eu/de/suchen/id/TIBKAT%3A850577144`.

[70] Bei Yu et al. “Machine learning and pattern matching in physical design”. In: *The 20th Asia and South Pacific Design Automation Conference*. IEEE. 2015, S. 286–293.

[71] G. Zifonun, L. Hoffmann und B. Strecker. *Grammatik der deutschen Sprache*. Schriften des Instituts für Deutsche Sprache. De Gruyter, 2011. ISBN: 9783110872163. URL: `https://books.google.de/books?id=8dYZYYSJu04C`.