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

Automatisches Kategorisieren von geschriebener Kommunikation zur Unterstützung der Informationsbereitstellung in Softwareprojekten

Automated categorization of text-based communication to support information delivery in software projects

Masterarbeit

im Studiengang Informatik

von

Nikoleta Themeliotou

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

Hannover, 25.10.2022

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 25.10.2022

Nikoleta Themeliotou

Zusammenfassung

Automatisches Kategorisieren von geschriebener Kommunikation zur Unterstützung der Informationsbereitstellung in Softwareprojekten

Kommunikation ist für den Erfolg eines Softwareprojektes essenziell. Durch die Globalisierung erhöht sich der Informationsaustausch auf digitaler Ebene, wodurch Plattformen wie Gitter und Slack immer mehr an Bedeutung gewinnen. Besonders Open Source Software Projekte greifen dabei vermehrt auf diese Art von Kommunikationstools zurück, bei denen eine große Menge an Informationen entsteht, welche von den Entwicklern und Nutzern manuell gefiltert werden müssen. Zur Unterstützung solcher Projekte wird in dieser Arbeit eine explorative Studie durchgeführt. Aus den von Lin et al. [21] herausgearbeiteten nutzenorientierten Kategorien wurden Communication, Communication with Teammates, Non-work Topics, Dev-Ops, Team Q & A und Customer Support ausgewählt. Basierend auf dem Datensatz von Parra et al. [26] wurde die automatische Kategorisierung von Nachrichten aus Gitter unter Verwendung verschiedener binärer Machine Learning Klassifikationsalgorithmen untersucht. Zur Vorverarbeitung der Daten wurden Natural Language Processing Verfahren für die geschriebene Kommunikation genutzt. Aus der Analyse der Kategorien hat sich ergeben, dass Non-work Topics und Customer Support sehr gut automatisch durch die ausgewählten Algorithmen klassifiziert werden. Dabei konnte gezeigt werden, dass der Naive Bayes Classifier mit einer Accuracy von 73% für Non-work Topics und 77% für Customer Support der zuverlässigste Klassifikationsalgorithmus ist.

Abstract

Automated categorization of text-based communication to support information delivery in software projects

Communication is essential for the success of a software project. Globalization leads to the increase in exchange of information on a digital level, which means that platforms such as Gitter and Slack are becoming more and more important. Especially open source software projects increasingly resort to these types of communication tools, where a large amount of information is generated that has to be manually filtered by developers and users. To support such projects, an exploratory study is conducted in this thesis. From the purpose categories identified by Lin et al. [21], the categories Communication, Communication with Teammates, Non-work Topics, Dev-Ops, Team Q & A and Customer Support were chosen. Based on the dataset of Parra et al. [26], the automatic categorization of messages from Gitter was investigated using various binary machine learning classification algorithms. Natural Language Processing techniques for written communication were used to preprocess the data. The analysis of the categories showed that Non-work Topics and Customer Support are classified very well automatically by the selected algorithms. It was shown that the Naive Bayes Classifier is the most reliable classification algorithm with an accuracy of 73% for Non-work Topics and 77% for Customer Support.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Akronyme

FN False Negative

FP False Positive

GIF Graphics Interchange Format

IRC Internet Relay Chat

NLP Natural Language Processing

OSD Open Source Definition

OSI Open Source Initiative

OSS Open Source Software

SVM Support Vector Machine

TF-IDF term frequency - inverse document frequency

TN True Negative

TP True Positive

Kapitel 1

Einleitung

1.1 Motivation

“Communication is key” ist eine Redewendung, welche den hohen Stellenwert von Kommunikation in der heutigen Zeit verdeutlicht. Sie ist allseits bekannt und auch die Bedeutung dahinter ist klar, die Umsetzung jedoch nicht immer einfach. Bei der Entwicklung von Software ist der Informationsaustausch unter den einzelnen Mitgliedern des Projektes sowie der Austausch mit den Stakeholdern ein essenzieller Teil des Requirement Engineering und trägt einen großen Teil zum Erfolg des Projektes bei [18]. Für diesen Austausch gibt es mehrere Kommunikationsmittel, wie zum Beispiel Meetings, sei es online oder in Präsenz, Telefonate, Messaging und E-Mails, welche verwendet werden können [36]. Außerdem haben Open Source Software (OSS) Projekte in den vergangenen 50 Jahren an größerer Bedeutung gewonnen, wodurch es immer mehr Softwareteams gibt, welche zeitlich, räumlich oder auch kulturell voneinander getrennt sind [14]. Diese Teams greifen immer häufiger auf Kommunikationstools wie Slack, Gitter sowie Twitter zurück, um Informationen miteinander zu teilen [36] [15]. Bei diesem Austausch entsteht eine große Menge an Informationen, welche von den Entwicklern selbstständig gefiltert werden müssen [18] [33]. Das Problem dabei ist, dass Informationen untergehen oder vergessen werden und dies einen negativen Effekt auf den Erfolg des gesamten Projektes haben kann [18] [33]. Bereits bekannte Arbeiten in diesem Themenfeld haben dabei verschiedene Vorschläge vorgebracht und Kategorien festgelegt, um solche Beiträge einzuordnen und den Prozess

der Suche von Informationen zu erleichtern, indem beispielsweise in alten Chatverläufen schnell und einfach Informationen wiederzufinden sind [21] [8] [11] [4]. Eine Anwendungsbeispiel dafür ist von Guzman et al. [11], die Beiträge im sozialen Netzwerk Twitter nach Verbesserungsvorschlägen für Softwareapplikationen klassifiziert haben. Damit können Beiträge von Benutzern der Applikation direkte Anforderungen an die Entwickler stellen und diese können ihnen automatisch angezeigt werden. Ein weiteres Beispiel stammt aus der Arbeit von Chowhury und Hindle [8], welche off-topic Diskussionen aus Internet Relay Chat (IRC) Unterhaltungen rausgefiltert haben. Dieses Vorgehen kann Nutzern dabei helfen, nicht durch störende Informationen abgelenkt zu werden, sondern sich auf wichtige Beiträge zu beschränken. Dies ist erforderlich, da in der heutigen Zeit, in der vermehrt auf schriftliche Kommunikation in Form von Sozialen Netzwerken, Chats und E-Mails als Austauschmedium zurückgegriffen wird, eine strukturelle Informationsbereitstellung zur Unterstützung von Softwareteams benötigt wird [36][8]. Ziel dieser Arbeit ist es herauszufinden, welche Kategorien gut geeignet sind, um die Informationen aus geschriebener Kommunikation zu klassifizieren.

1.2 Problemstellung

Schriftliche Kommunikation steht vor allem in OSS Projekten vermehrt im Fokus [15]. Da Kommunikation für den Erfolg eines Softwareprojektes eine wichtige Rolle spielt, ist es Ziel dieser Arbeit, die Informationsbereitstellung zu fördern und zu vereinfachen [18]. In Chaträumen und in Foren sammeln sich verschiedene Beiträge an, welche sich im Grad der Wichtigkeit für den Erfolg im Softwareprojekt unterscheiden [24]. Diese Menge an Daten zu überblicken ist für jeden einzelnen Entwickler eine zeitaufwändige und herausfordernde Tätigkeit [33]. Daher stellt diese Arbeit eine Lösung vor, welche eine automatisierte Kategorisierung der Beiträge von Entwicklern ermöglichen soll. Ein Beispiel hierfür ist, dass es bei einem Austausch in einem Chatraum oder etwa in einem Forum zu überflüssigen Beiträgen kommen kann, die keinen Mehrwert für die Entwicklung der Software bieten [8]. Des Weiteren ist es für das Verständnis des Entwicklungsprozesses von Vorteil, wenn in vergangenen Beiträgen Informationen über Designentscheidungen einfach aufzufinden sind [2].

Auch interessant für das Team könnte die Möglichkeit sein, etwa bei dem Support von Kunden nach bereits bestehenden Lösungen suchen zu können, um diese dann zu präsentieren. Einige Applikationen verfügen bereits über eine Taggingfunktion wie beispielsweise Stack Overflow[1]. Diese Taggingfunktion ist jedoch nur auf technische Aspekte wie Programmiersprachen oder Betriebssysteme beschränkt [4]. Eine automatische Klassifizierung mehrerer unterschiedlicher Kategorien würde daher einen Mehrwert für die Teilnehmer eines Softwareprojektes bieten.

1.3 Lösungsansatz

Im ersten Schritt sollen Kategorien herausgearbeitet werden, welche geeignet sind, Informationen aus geschriebener Kommunikation zu strukturieren. Dafür wird durch eine Recherche der zugehörigen Literatur ermittelt, wie verwandte Arbeiten ihre Datensätze gelabelt haben. Im zweiten Schritt soll analysiert werden, welche Klassifikationsmodelle eine zufriedenstellende Klassifizierung durchführen. Diesbezüglich wird eine explorative Studie durchgeführt, welche den Ansatz verfolgt, mit Hilfe von binären Machine Learning Klassifikationsmodellen ausgewählte Kategorien zu erkennen. Es werden mehrere Modelle implementiert und für jede Kategorie am Ende verglichen. Basierend auf diesen Ergebnissen soll es dann außerhalb dieser Arbeit möglich sein einen Tagging Prototypen zu implementieren, der die geschriebene Kommunikation automatisch kategorisiert. Die Kategorien sollen dabei Informationen beinhalten, die für das Team einen Mehrwert bieten. Dafür wird die geschriebenen Kommunikation in Gitter[2], einer Plattform zum Austausch zwischen OSS Entwicklern, betrachtet und der von Parra et al. [26] vorgebrachte Datensatz, bestehend aus 10000 Einträgen, für das Training und die Validierung dienen.

1.4 Struktur der Arbeit

Um sich der Fragestellung zu nähern, müssen zunächst einmal die Grundlagen in Kapitel 2 bezüglich OSS und Machine Learning, ins-

[1] https://stackoverflow.com/tags, Zugriff: 01.10.2022
[2] https://gitter.im, Zugriff: 01.10.2022

besondere Natural Language Processing (NLP), beschrieben werden. Nach der Einordnung in die Literatur kann folglich in Kapitel 3 die Abgrenzung dieser Arbeit von bereits bestehenden Studien und Implementierungen vorgenommen werden. Die beiden vorangegangenen Kapitel bieten demnach die Grundlage für das Kapitel 4, in dem die prototypische Implementierung vorgestellt wird. In diesem Kapitel soll es insbesondere darum gehen, das Konzept und die Umsetzung der explorativen Studie auf Basis des Prototypen nachzuvollziehen. Anschließend werden in Kapitel 5 die Ergebnisse der Studie vorgestellt und unter dem Gesichtspunkt des Nutzens und der Herausforderungen diskutiert. Abschließend folgt in Kapitel 7 eine kurze Zusammenfassung der Arbeit und ihrer Erkenntnisse.

Kapitel 2

Grundlagen

2.1 Open Source Software Projekte

In der Softwareentwicklung wird zwischen Closed und Open Source Software differenziert. Diese unterscheiden sich dadurch, dass Open Source Software freien Zugang zum Quellcode ermöglicht, während Closed Source Software dies nicht tut [29]. Folglich besitzen in OSS Systemen Benutzer als auch Entwickler die Erlaubnis frei, auf den Quellcode zuzugreifen und diesen zu verändern [34]. Daraus resultiert, dass der Sofwareprozess einer Closed Source Software[1] sich im Wesentlichen von dem einer Open Source Software unterscheidet. Auf Grund der besseren Verfügbarkeit fokussiert sich diese Arbeit auf OSS Projekte.
Wenngleich OSS bereits seit über 50 Jahren existiert, so hat sie insbesondere kürzlich an Aufmerksamkeit gewonnen, da immer mehr Menschen Zugang zum Internet sowie das Interesse haben, an solchen Systemen teilzunehmen [29] [14]. Um ein gemeinsames Verständnis unter den Entwicklern zu schaffen und auch eine juristische Abgrenzung zu ermöglichen, haben Prenes und Raymond die Open Source Initiative (OSI) ins Leben gerufen und dort die Open Source Definition (OSD) festgelegt [14]. Die von Prens und Raymond festgelegte OSD besteht aus 10 Punkten, welche im folgenden kurz zusammengefasst sind und in der OSI[2] für eine detaillierte Beschreibung zu finden ist. [14]

[1] Für eine detaillierte Erläuterung der Unterschiede siehe Potdar und Chang [29]
[2] `https://opensource.org/docs/osd`, Zugriff: 01.10.2022

1. **Kostenlose Weiterverteilung** der Software ohne Lizenzgebühren
2. **Quellcode** soll entweder mit der Software verfügbar oder einfach zu erhalten sein
3. **Abgeleitete Werke** oder Modifikationen der originalen Software sollen unter gleichen Lizenzbedingungen verbreitet werden dürfen
4. **Integrität des Quellcodes des Autors** muss gewährleistet sein
5. **Keine Diskriminierung von Personen oder Gruppen**
6. **Keine Diskriminierung von Tätigkeitsfeldern**
7. **Verteilung der Lizenz**
8. **Die Lizenz darf nicht produktspezifisch sein**
9. **Die Lizenz darf andere Software nicht einschränken**
10. **Die Lizenz muss technologieneutral sein**

Durch die zugrunde liegende Definition kann bereits abgeleitet werden, dass es sich bei den Entwicklern, die an einem OSS Projekt teilnehmen, um Personen handelt, welche ohne monetäre Anreize partizipieren [34]. Diese Gruppe von Personen wird auch als “community of practice” bezeichnet und setzt sich zusammen aus Entwicklern, Benutzern und “user-turned-developers” (Nutzer, die Entwickler geworden sind) [34]. Sie verfolgen alle das gemeinsame Ziel, ihr Interesse in einer bestimmten Domäne zu vertreten. Anders als bei einem klassischen Softwareprojekt sind die Rollen anders verteilt, denn sie sind nicht strikt voneinander getrennt und können sich auch überlappen [34]. Ye und Kishida [34] beschreiben dabei 8 verschiedene Rollen, welche Personen in einem OSS Projekt einnehmen können. Bei der Betrachtung dieser Rollen ist festzustellen, dass es zwar keine strenge Hierarchie gibt, jedoch einige Rollen mehr Verantwortlichkeiten besitzen als andere [34]. Die meiste Verantwortung trägt der Projektleiter, denn er ist derjenige, der das Projekt initiiert und somit das größte Interesse an dem Erfolg und an der Entwicklung hat [34]. Ihm schließen sich die Kernmitglieder an, welche das Projekt überblicken und bei der Koordination der Entwicklung helfen [34]. Kernmitgleder sind bereits über einen längeren Zeitraum ein Teil des OSS Projektes

und besitzen demnach mehr Erfahrung als andere [34]. Bei der Entwicklung übernehmen die aktiven Entwickler die Hauptrolle, denn sie tragen regelmäßig neue features oder bug fixes zu der Software bei [34]. Zusätzlich können periphere Entwickler, Bug fixer, Bug reporter, Leser oder passive Benutzer Teil eines OSS Projektes sein. Sie sind eher unregelmäßig am Projekt beteiligt oder überschauen nur einzelne Teile [34]. Nicht alle Rollen müssen in einem OSS Projekt zwingend vorkommen, denn die Teilnehmer nehmen diese nach ihrem persönlichen Empfinden selbstständig ein. Diese können sich über die Zeit verändern, wodurch auch die soziale Dynamik sowie die Struktur der Gruppe sich abwandelt [34]. Die Motivation nach Ye und Kishida [34] für Softwareentwickler an einem OSS Projekt teilzunehmen oder sogar selbst eines zu initiieren, erfolgt aus dem Willen, etwas Neues zu erlernen. Die Anerkennung in der Community ist ein Anreiz für Entwickler, an einem OSS Projekt teilzunehmen, denn es könnten sich dadurch neue Möglichkeiten ergeben, woraus persönlicher Erfolg resultiert [34].

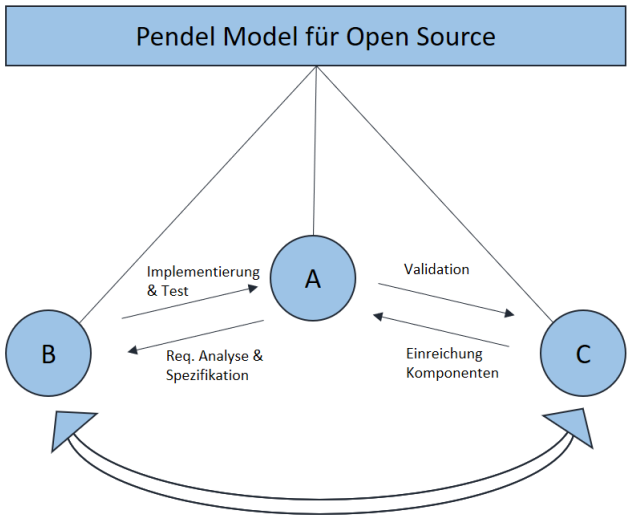

Abbildung 2.1: Das Pendel Modell für Open Source Software von Potdar und Chang [29, S. 4]

.

Da nun die Definition und die Rollen eines OSS vorgestellt wurden,

soll jetzt der Prozessablauf bei der Erstellung einer Software näher betrachtet werden. Potdar und Chang [29] haben ein solches Modell entworfen, denn auch wie die traditionellen Softwareprojekte mit ihrem Lifecycle Modell, sollte es ein solches Modell für OSS Projekte geben. Dieses nennt sich das Pendel Modell und beinhaltet alle wichtigen Phasen, welche bereits aus traditionellen Softwareprojekten bekannt sind. Das Modell ist in Abbildung 2.1 bildlich veranschaulicht [29]. Dabei stellt der Punkt A den Normalzustand des Pendels dar, welcher nämlich stillstehend auf diesem Punkt bleibt bis er durch neue Anforderungen an das System angeregt wird. Dann bewegt sich das Pendel zu B bis es dann schließlich bei C angelangt und seine Phase wieder bei A beendet.

2.2 Kommunikation in Open Source Software Projekten

In einem Softwareprojekt ist die Kommunikation besonders wichtig, weil es sich dabei um den Austausch von Informationen handelt, welche der Schlüssel für den Erfolg des gesamten Projektes sind [18] [28]. Dabei werden Themen ausgetauscht, welche die Koordination, die Besprechung von Aufgaben, die Lösung von Problemen sowie Entscheidungen über die Software zu fällen, betreffen [15]. Dabei findet die Kommunikation und die Kollaboration bei OSS Teams meistens online statt, da sie räumlich sowie zeitlich voneinander getrennt sind und somit sehr selten (wenn überhaupt) Face-to-Face Meetings haben [15] [28]. Daher wird die Kommunikation durch Tools wie E-Mails, Chats oder Foren in solchen Entwicklerteams durchgeführt [36] [15]. Besonders im Fokus stehen dabei, belegt durch eine Studie von Käfer et al. [15] Gitter und Slack, welche gegenüber traditionellen Tools, wie beispielsweise Mailing Listen immer mehr an Bedeutung gewinnen. Durch die Globalisierung der Entwicklerteams und auch durch die COVID-19 Pandemie ist die Wichtigkeit von asynchronen Kommunikationskanälen wie beispielsweise E-Mail auf synchrone Live-Chats wie Gitter umzusteigen bekannter geworden [33]. Doch diese synchrone und dynamische Art der Kommunikation beinhaltet auch einige Herausforderungen [33]. Shi et al. [33] haben folgende drei

Herausforderungen beschrieben, welche sich durch die Nutzung von Live-Chats wie beispielsweise Gitter ergeben:

1. Dialoge überschneiden sich, da mehrere Themen in einem Chatraum gleichzeitig mit verschiedenen Personen besprochen werden
2. Ein hoher Aufwand für die Teilnehmer des Chatraums fällt an, da viel Text mit komplexen und technischen Themen entsteht.
3. Durch die Menge an Daten werden auch Nachrichten verbreitet, die wenig oder keinen Mehrwert bieten und nur ein Störfaktor sind.

2.3 Maschinelles Lernen

Das Fachgebiet der künstlichen Intelligenz ist aktuell ein Forschungsgebiet, dem viel Aufmerksamkeit geschenkt wird [9]. Die Idee dabei besteht darin intelligente Software zu verwenden, um Probleme zu lösen, welche für Menschen einfach und intuitiv zu bewältigen, jedoch schwierig zu erklären sind [9]. Dazu gehören unter anderem Aufgaben aus den Bereichen der Spracherkennung oder der Gesichtserkennung [9].

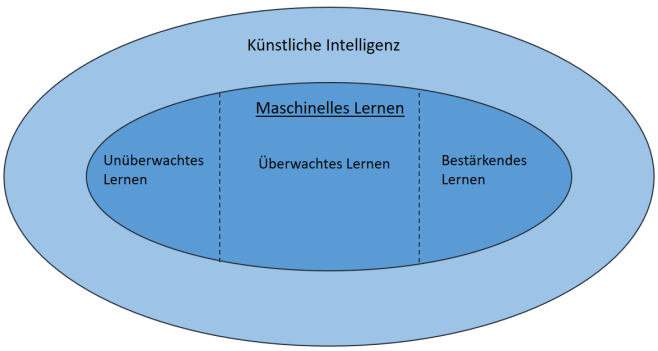

Abbildung 2.2: Künstliche Intelligenz als Obermenge mit den jeweiligen Teilmengen [16, S. 6]

.

Aus der Abbildung 2.2 geht hervor, dass das Maschinelle Lernen eine Teilmenge der künstlichen Intelligenz ist, welches auf der Idee basiert, dass ein System eigenes Wissen erlernt, indem es beispielsweise

Merkmale aus Daten extrahiert [9]. Beim Maschinellen Lernen wird zwischen drei Teilgebieten, Supervised Learning (dt. Überwachtes Lernen), Unsupervised Learning (dt. Unüberwachtes Lernen) und Reinforcement Learning (dt. Bestärkendes Lernen), unterschieden, welche im Folgenden voneinander abgegrenzt werden [16].

Supervised Learning

Die Methode des Supervised Learning basiert darauf, dass ein System mit Hilfe eines Datensatzes, welcher gelabelt ist, lernen kann. Gelabelte Daten meinen damit, dass es eine Menge X 2.1 gibt, welche für jeden Wert einen dazugehörigen Wert aus der Menge Y 2.2 besitzt und sich daraus der Datensatz $D_{labeled}$ wie in 2.3 ergibt [16]. Der Datensatz sollte dabei nochmal unterteilt werden in Trainings-, Validierungs- und Testdatensatz, um unabhängige Teilmengen von $D_{labeled}$ zu haben, mit denen das System lernt und danach ausgewertet werden kann [16].

$$X = x_1, x_2, ..., x_n \tag{2.1}$$

$$Y = y_1, y_2, ..., y_n \tag{2.2}$$

$$D_{labeled} = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) \tag{2.3}$$

Supervised Learning kann dabei in zwei Arten unterschieden werden, eine Klassifikation oder eine Regression. Ersteres beschreibt die Verbindung der Eingaben mit einer Kategorie [16]. Ein Beispiel dafür wäre, wenn die Daten aus X Bilder von verschiedenen Tieren beinhalten und diese in die Kategorien Hund, Katze oder Maus einsortiert werden würden. In diesem Fall würden unsere Labels in Y entweder Hund, Katze oder Maus beinhalten. In einem solchen Fall entspricht die Klassifikation einem “multi-class” Problem, da es mehrere Labels gibt. Bei einer binären Klassifikation von nur zwei Tieren würde lediglich zwischen Kategorie “Katze ja” mit Label 1 und “Katze nein” mit Label 0 unterschieden werden.
Eine Regression hingegen versucht, die Eingabe zu einer reellen Zahl zuzuordnen. Ein Beispiel dafür wäre es, den Wert einer Aktie vorherzusagen [16].

Unsupervised Learning

Im Gegensatz zu dem Supervised Learning wird beim Unsupervised Learning die Idee verfolgt, ohne Labels die Kategorien zu bestimmen. Diesbezüglich werden für die Bestimmung der Nähe einzelner Elemente Methoden wie Ähnlichkeitsanalyse und Clustering verwendet [16]. Beispielsweise soll versucht werden, das Verhalten eines Smartphone Users nachzuvollziehen, um ihm anhand seiner Reviews oder seiner Downloads weitere Apps vorzuschlagen, indem die Übereinstimmung mit anderen Benutzern analysiert wird.

Reinforcement Learning

Das Reinforcement Learning verfolgt einen anderen Ansatz als die zwei bereits beschriebenen Verfahren. Denn bei dieser Art des Lernens, soll die Maximierung des Erfolges erzielt werden [16]. Dabei soll das System beim Lernen in bestimmten Verhaltensweisen bestärkt oder entmutigt werden [16]. Die Anwendung dieser Methode wird bevorzugt im Bereich von Computerspielen eingesetzt, wo es darum geht, bestimmte Züge zu machen, um das Spiel zu gewinnen [16]. Beliebte Beispiele für solche Spiele sind Mühle und Go [16].

2.4 Klassifizierungsalgorithmen

Wie bereits im vorherigen Kapitel beschrieben, gibt es verschiedene maschinelle Lernverfahren. Für diese Arbeit steht die binäre Klassifikation mit Hilfe von Supervised Learning Algorithmen im Vordergrund. Aus diesem Grund werden die sieben Klassifikatoren, welche für das Kapitel 4 verwendet werden, im Folgenden kurz vorgestellt.

Logistische Regression

Die Logistische Regression verwendet eine logistische Funktion, welche einen Wahrscheinlichkeitswert zwischen 0 und 1 wiedergibt [16] [13]. Die Formel für die logistische Funktion 2.4 wird im Diagramm 2.3 dargestellt. Für ein binäres Klassifikationsproblem, welches zur Vorhersage von zwei Klassen 0 oder 1 dient, sind die Log-Odds (dt. logarithmisches Chancenverhältnis) für die Klasse, welche mit 1 gelabelt ist, der Formel

des linearen Regressionsmodells gleichgesetzt 2.5 [32] [3]. Somit ergibt sich die Formel 2.6, welche das logistische Regressionsmodell beschreibt und die Wahrscheinlichkeit wiedergibt[32].

$$\frac{1}{1 + exp^{-x}} \tag{2.4}$$

$$y = loggods = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{2.5}$$

$$p = \frac{1}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n}} \tag{2.6}$$

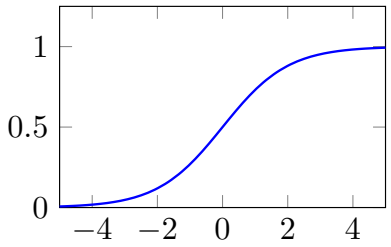

Abbildung 2.3: Sigmoidfunktion [16, S.156]

Naive Bayes Classifier

Der Supervised Learning Algorithmus Naive Bayes verwendet als Grundlage für die binäre Klassifizierung für $y_i \in (0, 1)$ das Bayes Theorem 2.7 [32][16].

$$h_{Bayes}(x) = argmax_{y \in (0,1) P(X=x|Y=y)P(Y=y)} \tag{2.7}$$

Der Naive Bayes Klassifikator unterscheidet sich dabei von den anderen Algorithmen dadurch, dass er von einer Unabhängigkeit der Merkmale ausgeht und sich daraus folgende Gleichung 2.8 für d Dimensionen ergibt [16].

$$h_{Bayes}(x) = argmax_{y \in (0,1) P(Y=y) \prod_{j=1}^{d} P(X_j=x_i|Y=y)} \tag{2.8}$$

Im Vergleich zu anderen Algorithmen trainiert der Naive Bayes Klassifikator schneller und funktioniert zuverlässig, selbst wenn nur wenige Daten zur Verfügung stehen. [32]

Decision Tree

Der Decision Tree ist ein Klassifikationsalgorithmus, welcher, wie der Name bereits sagt, eine Baumstruktur aufweist und aus Knoten sowie Blättern (terminale Knoten) besteht [19]. Dabei wird bei den Knoten zwischen dem Startknoten, den Entscheidungsknoten und terminalen Blattknoten unterschieden [3]. Die Eingabe erfolgt über den obersten Knoten, den Startknoten [19]. Die Funktionsweise des Algorithmus ist dabei sehr simpel, denn an jedem Entscheidungsknoten wird ein Test durchgeführt und überprüft, ob die Eingabe der Annahme entspricht oder nicht. Daraus resultiert eine Abzweigung in zwei Richtungen. Dabei wird unterschieden, ob ein terminales Blatt oder ein weiterer Entscheidungsknoten folgt [19] [3]. Dieser rekursive Prozess wird dann beendet, wenn ein terminales Blatt erreicht wurde und die Eingabe einer Klasse zugeordnet werden kann [19] [3]. Ein Beispiel für einen solchen Entscheidungsbaum ist in 2.4 abgebildet. Hierbei soll eine Person mit einem Alter x in eine der drei vorliegenden Kategorien eingeordnet werden. Ist die Person also jünger als 12 (Startknoten), ist sie noch ein Kind (terminaler Blattknoten), ist sie jünger als 18 (Entscheidungsknoten), ist sie ein Jugendlicher (terminaler Blattknoten) und wenn sie über 18 ist (Entscheidungsknoten), dann wird sie als Erwachsener klassifiziert (terminaler Blattknoten).

Random Forest

Ein Nachteil, welcher sich aus dem Verfahren der Decision Trees ergibt, ist, dass je mehr Daten zur Verarbeitung vorhanden sind, desto größer werden die Bäume [32]. Dies führt unter Umständen dazu, dass das Model beim Training zwar sehr gute, aber beim Testen schlechtere Werte erzielt, da es nur sehr spezifische Regeln lernt [32]. Aus diesem Grund wurde das Random Forest Verfahren entwickelt und wird dadurch ausgezeichnet, dass es aus einem Ensemble von Entscheidungsbäumen besteht [3]. Jeder Baum wird auf eine zufällige Teilmenge der Trainingsdaten oder einer zufälligen Teilmenge der Eingabemerkmale trainiert [3]. Am Ende werden diese Vorhersagen bzw. Ergebnisse kombiniert, wodurch die Gesamtgenauigkeit signifikant gesteigert werden kann. [3]

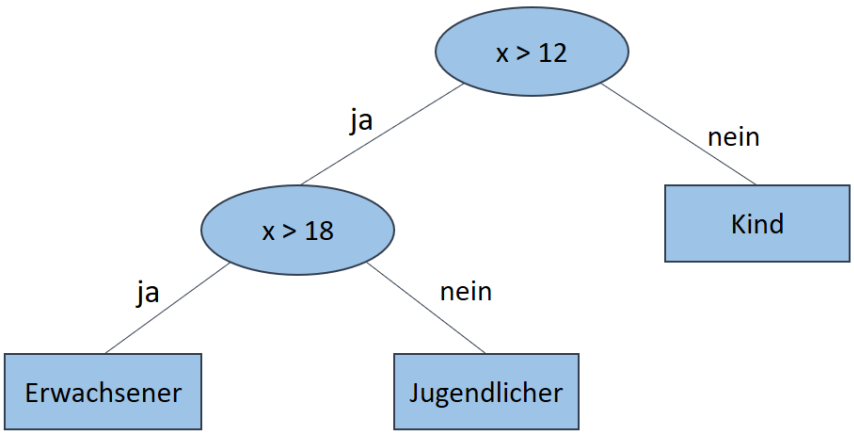

Abbildung 2.4: Beispiel für einen Entscheidungsbaum

Ensemble

Ein Ensemble beschreibt einen Klassifikationsalgorithmus, welcher sich aus einer Kombination von Klassifikatoren zusammensetzt [3]. Dieses Verfahren wird genutzt, um durch die Zusammenarbeit mehrerer Modelle bessere und zuverlässigere Ergebnisse erzielen zu können [23]. Äquivalent zum Random Forest, welcher ein Ensemble aus mehreren Decision Trees bildet, werden bei diesem Verfahren Klassifikationsalgorithmen kombiniert. Eine Art eines solchen Ensembles ist das Voting (dt. Abstimmung) [3]. Dabei unterscheiden sich die Kombinationsregeln der Klassifikatoren für die Auswertung zwischen Summe, gewichteten Summe, Median, Minimum, Maximum, sowie Produkt [3].

Support Vector Machine

Die Support Vector Machine (SVM) ist ein weiterer Supervised Learning Algorithmus [32]. Für eine binäre Klassifizierung sortiert er zukünftige Datenpunkte in eine von zwei Klassen ein [32]. Dabei separiert die SVM die gelabelten Trainingsdaten als Punkte in einem Raum, sodass sie von einer sogenannten Hyperplane (dt. Hyperebene) getrennt werden [32]. Die SVM separiert die Daten dabei nicht nur, sondern findet auch die Hyperebene, welche die optimalste Einsor-

tierung der Daten erreicht, die “maximum margin” Seperation [16]. An der “maximum margin” liegende Datenpunkte, welche durch die Hyperebene getrennt werden, werden als Support Vectors bezeichnet [16].

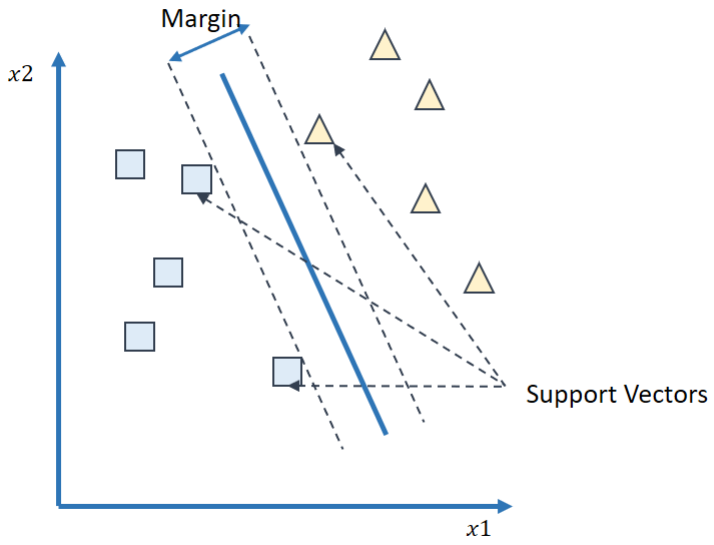

Abbildung 2.5: Separierung der Daten durch die Support Vector Machine [16, S. 68]

2.5 Metriken

Zur Auswertung binärer Klassifikationsalgorithmen gibt es mehrere Metriken, welche sich in der Literatur etabliert haben und in diesem Kapitel vorgestellt werden [16]. Es gibt vier mögliche Entscheidungsfälle, die während der Vorhersage durch den Klassifikationsalgorithmus getroffen werden können [3]. Die Abbildung 2.6 zeigt eine Konfusionsmatrix, welche diese vier Fälle beinhaltet. Dabei werden diese im Detail vorgestellt. Wenn die tatsächliche Klasse der Instanz positiv ist und ihre Vorhersage ebenfalls positiv ist, dann ist es eine True Positive (TP) (dt. richtig-positiv) Entscheidung [3]. Ist jedoch die Vorhersage für eine positive Instanz negativ, dann liegt eine Einordnung für False Negative (FN) (dt. falsch-negativ) vor [3]. Die Betrachtung der tatsächlichen Klasse von negativen Instanzen ist dabei ähnlich [3].

Wenn eine negative Instanz auch als eine solche vorhergesagt wird, dann handelt es sich um eine Einordnung in True Negative (TN) (dt. richtig-negativ) [3]. Wurde jedoch eine negative Instanz als eine positive vorhergesagt, so handelt es sich um eine False Positive (FP) (dt. falsch-positiv) Entscheidung [3].

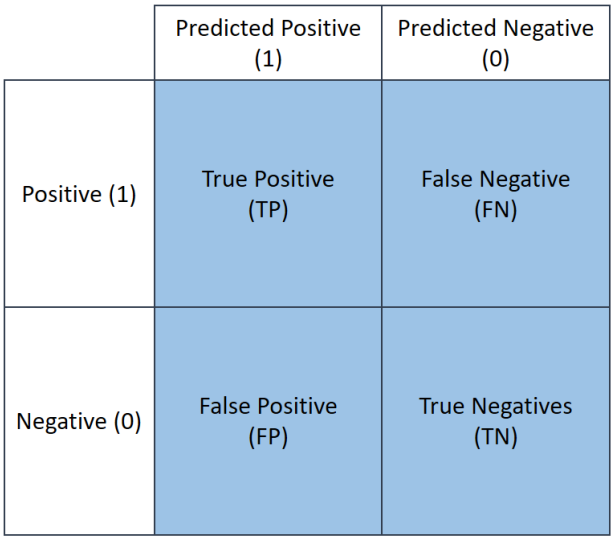

Abbildung 2.6: Konfusionsmatrix [16, S. 48]

Um später die Klassifikationsalgorithmen objektiv evaluieren zu können, bedarf es Metriken zur Messung der Aussagekraft und damit der Qualität der Modelle. In der Literatur haben sich dabei vier Metriken, Accuracy, Precison, Recall und F1-Score, etabliert.

Accuracy beschreibt die Genauigkeit, indem das Verhältnis der richtig eingeordneten Instanzen zur Summe aller Instanzen betrachtet wird [32].

$$Accuracy = \frac{TN + TP}{(TP + FN + FP + TN)} \tag{2.9}$$

Precision wird auch als postive predictive Value bezeichnet und misst den Anteil der richtig vorhergesagten Instanzen aus allen vorhergesagten Instanzen basierend auf der positiven Klasse [32][16].

$$Precision = \frac{TP}{(TP + FP)} \tag{2.10}$$

Recall wird auch als True positive rate (TRP) (dt. richtig-positiv-Rate) oder Sensitivität bezeichnet und misst den Anteil der richtig vorhergesagten positiven Instanzen [32][16].

$$Recall = \frac{TP}{(TP + FN)} \tag{2.11}$$

F1-Score beschreibt das Verhältnis zwischen Precision und Recall und wird daher als ein Gesamtmaß für Erfolg gesehen [16].

$$F1 = 2\frac{Precision \times Recall}{(Precision + Recall)} \tag{2.12}$$

2.6 Natural Language Processing

2.6.1 Natürliche Sprache

Um die Bedeutsamkeit des Teilgebietes des Maschinellen Lernen des Natural Language Processing einordnen zu können, muss zuerst die Grundlage für das Verständnis der Natürlichen Sprache geschaffen werden. Natürliche Sprache kann in gesprochene und geschriebene Sprache unterschieden werden [30][6]. Dabei ist bei ersterem der Austausch von informellen, ungeplanten und eher persönlichen Inhalten gemeint, da der Sprecher seine Adressaten meistens kennt und diese ihm auch eine direkte Antwort geben können [30]. Bei der geschriebenen Sprache, welche für diese Arbeit im Vordergrund steht und analysiert werden soll, handelt es sich meistens um einen geplanten, informativeren Austausch, da die verfassten Texte angepasst werden und somit auch bestimmte Standards erfüllen können [30].
Die Kommunikation zwischen Menschen ist über die Zeit effizienter geworden und ermöglicht es Wörter abzukürzen, wodurch ihre Bedeutung erst durch den Kontext klar wird [16]. Besonders bei geschriebener Sprache ist demnach zu beobachten, dass diese zunehmend informeller geworden ist [16].
Zuerst ist es wichtig zu verstehen, was Natürliche Sprache ist und welche Schwierigkeiten und Herausforderungen diese für die Implementierung hervorbringt. Natürliche Sprache wird dadurch bestimmt, dass sie durch die Menschen über die Zeit selbst entwickelt beziehungsweise angepasst wird [16]. Insbesondere wird sich auf einen kleineren

Wortschatz beschränkt, als die Sprache an Wörtern ermöglichen würde [16]. Daraus ergibt sich eine Mehrdeutigkeit der Wörter und der Kontext spielt eine wichtige Rolle beim Verständnis [16]. Vor allem bei geschriebener Sprache kann es vorkommen, dass die Aussagen anders als vom Sender intendiert verstanden werden oder der Empfänger die Nachricht nicht zur Kenntnis genommen hat [18].
Für ein "intelligentes" Computerprogramm besteht die Herausforderung in der Kontextabhängigkeit, wobei die Schwierigkeit vor allem in der Erkennung von Ironie, Sarkasmus, Metaphern, Humor, Grammatik sowie Satzbau liegt. Aus dem Grund werden bei der Analyse von natürlicher Sprache bestimmte Charakteristiken in Gruppen zusammengefasst [16]. Die Charakteristiken für geschriebene Sprache sind im Folgenden beschrieben [16].

Morphologie: Beschreibt die Struktur und die Form des Wortes

Lexikalische Analyse: Ist die Segmentierung von Text in Wörter

Syntax: Sind die Regeln für die Wörter in einem Satz

Semantik: Die Bedeutung des Wortes in einem Satz

Diskurs: Die Bedeutung zwischen den Sätzen

Natural Language Processing dient zur Modellierung von menschlicher Sprache und zur Extrahierung von Informationen [16]. Ziel ist es, die Sprache abzubilden, sodass morphologische, lexikalische, syntaktische, semantische oder diskursartige Charakteristiken repräsentiert werden und dann mit dem Einsatz von Machine Learning Algorithmen verarbeitet werden können [16]. Da im späteren Verlauf der Implementierung NLP Methoden eingesetzt werden, um die natürliche Sprache für die Klassifikationsmodelle umzuwandeln, werden nun fünf Verfahren vorgestellt.

2.6.2 Stemming

Bei dem Stemming Verfahren werden Wörter auf ihren Wortstamm zurückgebildet [16]. Tabelle 2.1 zeigt als Beispiel drei verschiedene Formen (teaches, teached, teachers), welche durch das Stemming alle zum Wort "teach" vereinfacht werden. Ein Nachteil dabei ist, dass es zu

mehrfachem Vorkommen des Wortes führen kann und dass der Kontext sowie die Bedeutung des Wortes verloren gehen können [16]. Ein Vorteil ist jedoch, dass Rechtschreibfehler dadurch keine große Rolle spielen, da das Wort sowieso auf seinen Wortstamm zurückgeführt wird [16].

Wort	Wortstamm (Stemming)
teaches	teach
teached	teach
teachers	teach

Tabelle 2.1: Beispiel für Stemming

2.6.3 Lemmatization

Dieses Verfahren unterscheidet sich vom Stemming darin, dass die Wörter auf Basis eines Lexikons auf ihre Wortstämme zurückgeführt werden [16]. Anders als beim Stemming entsteht so ein unterschiedlicher Wortstamm, wie in Tabelle 2.2 zu sehen ist. Folglich bleiben die Bedeutung des Wortes sowie der Kontext bestehen [16]. Der Nachteil ist jedoch, dass es sehr zeitintensiv ist, den Text mit dieser Methode anzupassen, da das Nachschlagen in Onlinebibliotheken beziehungsweise Wörterbüchern länger als das Stemming dauert [16]. Außerdem ist diese Methode anfälliger auf Rechtschreibfehler, wodurch eine Korrektur als Grundlage sinnvoll wäre [16].

Wort	Wortstamm (Lemmatization)
teaches	teaches
teached	teach
teachers	teacher

Tabelle 2.2: Beispiel für Lemmatization

2.6.4 Tokenization

Das Tokenization Verfahren dient dazu die einzelnen Sätze in ihre Wörter und Satzzeichen zu unterteilen, wie das erste Beispiel aus Tabelle 2.3 zeigt [16]. Bei der Betrachtung des zweiten Beispieles fällt

auf, dass die Tokenisierung, des Eigennamens New York City dazu führt, dass die Bedeutung des Standortes in dem Kontext verloren geht [16]. Ähnlich ist es auch bei dem dritten Beispiel aus der Tabelle 2.3. Dort befinden sich mehrere Satzzeichen im speziellen Punkte in dem Satz. Bei einer solchen Tokenisierung würden die Punkte als Satzende interpretiert werden und der Inhalt des Satz würde verloren gehen [16].

Vorher	Tokenisiert
The sun in Greece is beautiful.	|The| |sun| |in| |Greece| |is| |beautiful| |.|
I'd like to go to New York City.	|I| |'| |d| |like| |to| |go| |to| |New| |York| |City| |.|
Mrs. Simons puts 0.7ml of milk into the cake.	|Mrs| |.| |Simons| |puts| |0| |.| |7| |ml| |of| |milk| |into| |the| |cake| |.|

Tabelle 2.3: Beispiel für Tokenization

2.6.5 Stop Words

Im Kontext von Natural Language Processing wird von Stop Words gesprochen, wenn es um Begriffe geht, welche sehr häufig in einem Satz vorkommen, jedoch nur wenig oder gar keinen Mehrwert für den Inhalt bieten [16]. Die Entfernung dieser Wörter würde keine Auswirkung darauf haben, ob der Satz noch verständlich ist oder nicht [16]. Aus diesem Grund werden Wörter wie beispielsweise "the", "as", "is", "a", "an" oder "the" einfach bei der Bereinigung der Daten weggelassen [16] [32].

2.6.6 Bag of Words und TF-IDF Model

Das Bag-of-Words Modell repräsentiert jedes Dokument aus einer Reihe von Dokumenten als Vektor [16]. Dabei besteht der Vektor aus den Häufigkeiten aller Wörter, die in dem Dokument vorkommen [16]. Diese Repräsentation der Wörter als numerische Werte und der Dokumente als Vektoren ermöglicht es, Wortstellungen, Sequenzen und Grammatik außer Acht zu lassen und ein mathematisches Modell zu verwenden und wird als Count Vectorization bezeichnet [32] [16].
Der Nachteil des Bag-of-Words Modells liegt darin, dass dieses nur die Häufigkeit der Wörter in den Vektoren betrachtet, obwohl eigentlich Häufigkeit nicht zwingend mit Relevanz gleichzusetzen ist. Somit

können Wörter mit einer wichtigen Bedeutung von Informationen, die zwar häufiger vorkommen aber weniger relevant sind, überschattet werden [16]. Daher gibt es das term frequency - inverse document frequency (TF-IDF) Verfahren, welches die Gewichtung der Wörter anpasst [32] [16].

Kapitel 3

Verwandte Arbeiten

In Kapitel 2.2 wurden anhand der Ergebnisse von Käfer et al. [15] die bestehenden und an Bedeutung gewinnenden Kommunikationskanäle vorgestellt. Da es insbesondere in OSS Projekten viele Möglichkeiten für Entwickler gibt sich auszutauschen und dabei eine große Menge an Daten und Informationen auftreten können, werden im Folgenden bereits publizierte Arbeiten in diesem Bereich beschrieben und am Ende von den Inhalten dieser Arbeit abgegrenzt.

Softwareentwickler lösen oftmals Probleme in der Entwicklung, indem sie Question & Anwser (Q&A) Foren aufsuchen, um sich dort auszutauschen [15]. Stack Overflow ist ein solches Forum und bietet bereits Tags und somit die Möglichkeit, nach technischen Problemen zu suchen [4]. Diese Tags werden manuell von dem User an Hand seiner Einschätzung darüber, ob seine Frage zu einem bereits bestehenden Tag passt, eingetragen [31]. Da es mehrere Tags gibt, ist die Aufgabe, die passenden Tags zu finden, für die Entwickler zeitaufwändig [31]. Rekha et al. [31] haben daher ein hybrides auto-tagging System für Stack Overflow entwickelt, welches aus zwei grundlegenden Komponenten besteht. Erstere soll automatisch Programmiersprachen detektieren und letztere soll durch die Anwendung eines SVM Modells die Fragen klassifizieren [31]. Das System schlägt dem User demnach während er seine Frage stellt bereits Tags vor [31].
Ein weiterer Ansatz zur Automatisierung von Fragen für Stack Overflow stammt von Beyer et al. [4], welcher den Fokus bei den Tags auf den Zweck der Frage legt. Sie haben eine automatisierte

Klassifikation von Stack Overflow Posts in sieben Fragenkategorien erarbeitet und dafür 500 Posts per Hand in diese sieben Kategorien einsortiert, um sie als Datensatz zu verwenden. Für die Klassifizierung wurden Machine Learning Algorithmen verwendet und die Performance dieser am Ende verglichen.
Chatterjee et al. [7] haben ebenso ein Q&A Format untersucht, aber anstelle von Stack Overflow ihre Studie auf das Kommunikationstools Slack ausgerichtet. Die Explorative Studie, die in ihrem Paper vorgestellt wird, soll den potentiellen Nutzen und die Herausforderungen von Q&A Einträgen in Slack zur Unterstützung von Softwarewartung und Weiterentwicklung von Tools untersuchen.

Alkadhi et al. [2] haben eine explorative Studie durchgeführt, bei der sie 8700 Chatnachrichten aus drei verschiedenen Software Projeken auf "Rationale" zu untersuchten. "Rationale" meint damit Beweggründe, wieso eine bestimmte Entscheidung in einem Softwareprojekt getroffen wurde [2]. Diese Rationale entstehen durch den Austausch in einem Softwareprojekt über Probleme, alternative Lösungen und Argumentationen [2]. Dabei wurden drei wesentliche Punkte in der Studie untersucht [2]. Die Häufigkeit von "Rationale" und die Vollständigkeit dieser sowie das Potential einer automatisierten Extrahierung von "Rationale" mit Machine Learning Techniken wurden dabei betrachtet [2].
Auf Grundlage dieser Studie haben Alkadhi et al. [1] das System REACT (RationalE Annotations in ChaT messages) entworfen, als ersten Schritt zur Erfassung von "Rationale" in Entwicklerchats.
Guzman et al. [11] haben, basierend auf den Erkenntnissen einer vorangegangen Studie [10], welche aussagt, dass Twitter von Usern verwendet wird, um sich über Softwareapplikationen auszutauschen und dies für das Requirement Engineering hilfreich ist, das Tool ALERTme entwickelt. ALERTme soll Tweets automatisch klassifizieren, gruppieren und bewerten [11]. Für die Klassifikation werden Machine Learning Klassifikationsalgorithmen verwendet, um nach Verbesserungswünschen zu suchen [11].

Chowhury und Hindle [8] haben als Ansatz verfolgt, verschiedene Machine Learning Klassifikatoren zu trainieren, um Off-Topic

Unterhaltungen im IRC herauszufiltern. Dazu haben sie positive Beispiele für on-topic Unterhaltungen aus Stack Overflow und negative Beispiele für Off-topic Unterhaltungen aus Kommentaren von Youtube Videos herausgesucht. Mit diesen Bepsielen haben sie ihre Modelle trainiert und anschließend auf IRC angewendet.

Die Studie und Analyse von Parra et al. [24], basierend auf den Datensatz Gittercom [26], versucht, mit Hilfe von multi-class Machine Learning Klassifikationsalgorithmen und Deep Learning Algorithmen den Labeling Prozess von Daten aus Gitter zu automatisieren. Die Kategorien, die hierfür verwendet werden, stammen aus einer vorangegangenen explorativen Studie von Lin et al. [21], welche untersucht, wie Entwickler Slack verwenden und wie sie davon profitieren. Die Untersuchungen haben ergeben, dass Entwickler Slack für persönliche, teamweite und communityweite Zwecke verwenden [21]. Gitter ist ähnlich wie Slack und ermöglicht Softwareentwicklern, sich über Informationen bezüglich einer Software auszutauschen [24].

Das Ziel dieser Arbeit ist es einen Ansatz zu entwickeln, der als Grundlage für ein automatisiertes Tagging von Informationen aus geschriebener Kommunikation dienen soll. Dazu wird zuerst die geschriebene Kommunikation von OSS Teams auf der Plattform Gitter durch die bereits erbrachten Ergebnisse von Parra et al. [24] analysiert, um nützliche Kategorien herauszuarbeiten. Die Kategorien sollen im Speziellen aus der Informationsbereitstellung im Team und der Kommunikation über die zu entwickelnde Software bestehen. Anschließend soll eine prototypische Implementierung durchgeführt werden, bei der mehrere binäre Klassifikationsalgorithmen die einzelnen Kategorien automatisiert klassifizieren sollen. Dabei soll festgestellt werden, welche Kategorien und Algorithmen dafür gut geeignet sind. Diese Arbeit unterscheidet sich von den anderen darin, dass andere Kategorien verwendet werden und zu der Arbeit von Parra et al. [24], dass die Ergebnisse für eine Taggingsoftware dienen sollen und keine multi-class Algorithmen verwendet werden, um die Herausforderungen der einzelnen Kategorien besser einschätzen zu können.

Kapitel 4

Implementierung

4.1 Ansatz

Die Aufgabe der Arbeit liegt darin, eine automatische Klassifizierung von geschriebener Kommunikation zu erreichen, um aufbauend auf den Ergebnissen dieser Arbeit einen Tagging Prototypen zu entwickeln, der Softwareteams bei der Informationsbereitstellung unterstützt. Hierfür stehen zwei wesentliche Leitfragen im Zentrum der Implementierung.

Frage 1: Welche Kategorien können zur Klassifizierung von geschriebener Kommunikation in Softwareprojekten dienen?

Frage 2: Wie gut können diese Kategorien mit Hilfe von Machine Learning Algorithmen automatisch klassifiziert werden?

4.2 Konzept

In Abbildung 4.1 soll das Konzept zur Erarbeitung der explorativen Studie demonstriert werden. Zuerst muss eine Datenbasis geschaffen werden, welche mehrere verschiedene Kategorien von Kommunikation beinhaltet (siehe Abbildung 4.1, Phase 1). Dafür wurde der Datensatz von Parra et al. [26] Gittercom [25] verwendet und in Kapitel 4.4 vorgestellt. Aufgrund der Tatsache, dass dieser Datensatz mehrere Kategorien spezifiziert, werden mehrere kleine Datensätze für die Bearbeitung der automatischen Klassifizierung dienen. Jeder dieser

Datensätze dient dann zur binären Klassifizierung einer Kategorie und wird einzeln manuell gelabelt. Die Entscheidung über die ausgewählten Kategorien und die Herausforderung der zugrunde liegenden Daten werden in Kapitel 4.5 und 4.7 vorgestellt. Anschließend wird das Preprocessing der Daten in Kapitel 4.8 sowie Auffälligkeiten in den zugrunde liegenden Daten und der Prozess der Datenbereinigung erklärt. Diese einzelnen Datensätze dienen dann als Grundlage für das Training der Machine Learning Algorithmen, welche in Kapitel 2.4 bereits theoretisch vorgestellt wurden. Die Auswahl über die Algorithmen sowie die Implementierung werden in Kapitel 4.9 vorgestellt (siehe Abbildung 4.1, Phase 2). Zum Abschluss werden die trainierten Algorithmen mit ausgewählten Metriken aus Kapitel 2.5 ausgewertet (siehe Abbildung 4.1, Phase 3).

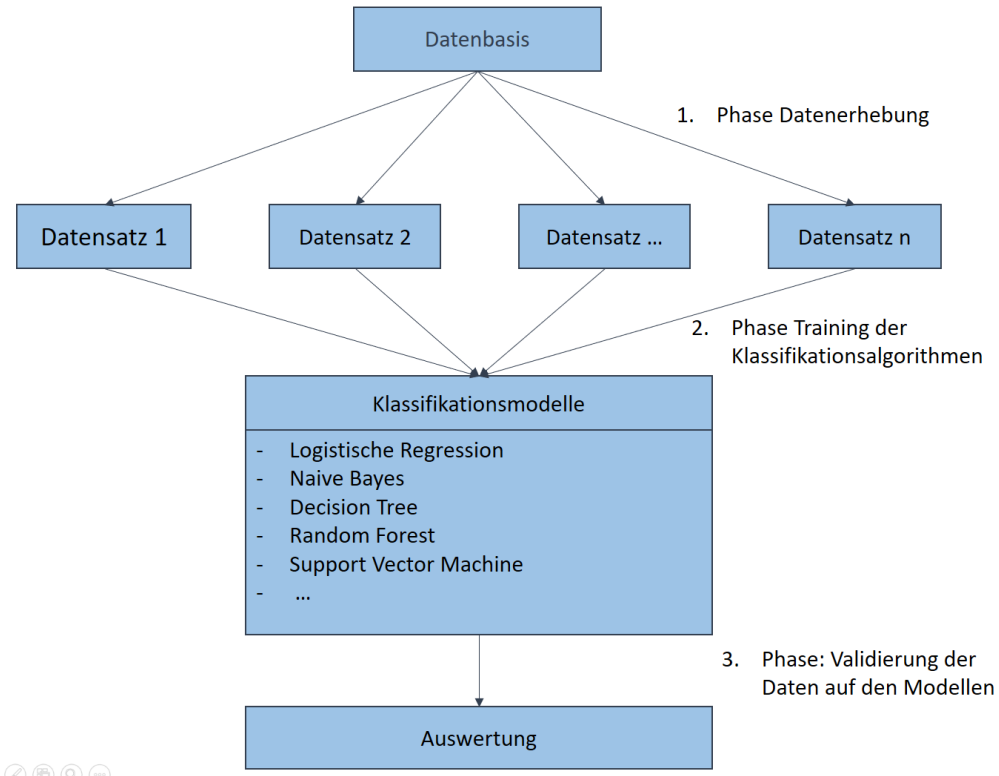

Abbildung 4.1: Konzept für die Implementierung

4.3 Verwendete Libraries und Programmiersprache

Für die Implementierung der Machine Learning Algorithmen wurde auf mehrere Open Source Libraries zurückgegriffen. Dazu wurde als Programmiersprache Python mit mehreren Packages zur Darstellung und Verarbeitung der Daten verwendet.

Python

Python[1] ist eine beliebte Open Source Programmiersprache, welche unter anderem für die Entwicklung von Webapplikationen, zur Durchführung von wissenschaftlichen und numerischen Datenanalysen sowie als unterstützende Sprache für die Softwareentwicklung durch beispielsweise Testen verwendet wird [13].

NumPy

NumPy[2] ist eine Open Source Bibliothek von Python, die ein multidimensionales Array-Objekt bereitstellt [12]. Es ermöglicht unter anderem eine Reihe an verschiedenen Operationen auf Arrays (Datenstruktur), mathematische Manipulation von Daten und grundlegende statistische Operationen.

Pandas

Pandas[3] ist eine Open Source Bibliothek, die leistungsstarke Datenstrukturen, Funktionen und Datenanalysetools für einfaches Arbeiten mit tabellarischen oder strukturierten Daten für die Programmiersprache Python bereitstellt [22].

Sklearn

Sklearn[4] ist eine Open Source Bibliothek, mit der Machine Learning Algorithmen für Python implementiert werden können [27]. Es verfügt über eine Mehrzahl an supervised learning Algorihtmen, welche für diese Arbeit implementiert werden [27].

[1] https://www.python.org, Zugriff: 17.10.2022
[2] https://numpy.org, Zugriff: 17.10.2022
[3] https://pandas.pydata.org, Zugriff: 17.10.2022
[4] https://scikit-learn.org/stable/, Zugriff: 17.10.2022

NLTK

NLTK[5] steht für Natural Language Toolkit und ist ein Tool zum Erstellen von Programmen mit Python, welche die menschliche Sprache verarbeiten sollen. Dazu enthält es lexikalische Schnittstellen wie beispielsweise WordNet und Textverarbeitungsbibliotheken für Klassifizierung, Tokenisierung, Wortstammbildung und Tagging [5].

4.4 Datenbasis

Für die automatische Kategorisierung von geschriebener Kommunikation in Softwareprojekten war es eine Herausforderung einen eigenen Datensatz zusammenzustellen, da kein Unternehmen seine Chat- oder E-Mail Verläufe öffentlich darlegt. Daher beschränkt sich die Implementierung dieser Arbeit ausschließlich auf den Informationsaustausch in OSS Projekten. Da OSS Entwickler meistens geographisch sowie zeitlich voneinander getrennt arbeiten, bieten Onlineplattformen wie Gitter und Slack die Möglichkeit, sich über Themen bezüglich der Software und dem weiteren Verlauf des Projektes auszutauschen [15][29][36].
Parras et al. [24] haben für die Analyse von geschriebener Kommunikation auf Gitter einen der größten manuell gelabelten Datensätze mit insgesamt 10.000 Nachrichten aus 10 Gitter Communities zusammengestellt [26]. Die Kategorien, welche für das Labeling der einzelnen Nachrichten verwendet wurden, stammen von Lin et al. [21], die bei einer Umfrage von Slack Usern über den Grund der Verwendung von Slack und dessen Vorteile erhoben wurden. Die Umfrage hat ergeben, dass die User den Hauptnutzen ("Purpose Types") von Slack in drei verschiedene Kategorien einteilen [21]. Die "Purpose Types" unterscheiden sich in "Personal benefits", "team-wide purposes" sowie "Community Support" und werden in der folgenden Aufzählung zusammenfassend erklärt [21].

Die Informationen und Inhalte über die einzelnen Kategorien und ihre Bedeutung stammen von Parra et al. [24] [26].

[5] https://www.nltk.org, Zugriff: 17.10.2022

Personal benefits: Nachrichten, welche einen persönlichen Nutzen für den Entwickler beinhalten, damit er seine Bedürfnisse erfüllt. Weitere drei "Purpose Categorys" werden darin beschrieben.

- *Discovery and aggregation of news and information:* Informationsweitergabe durch vertrauenswürdige und relevante Blogs
- *Networking and social acitivities:* Austausch zwischen Entwicklern von gemeinsamen Interessen und Arbeit
- *Fun:* Beitrage, in denen Graphics Interchange Format (GIF) oder Memes geteilt werden. Auch der Inhalt von gemeinsam gespielten Spielen wird in diese Kategorie mit aufgefasst.

Team-wide purpose: Der Austausch von Nachrichten, welche für die Auslieferung und Entwicklung der implementierenden Software von Bedeutung sind. Nachrichten aus dieser Kategorie können in vier Unterkategorien unterteilt werden.

- *Communication:* Diese Kategorie umfasst die gesamte Kommunikation bezüglich des Austauschs von Informationen zwischen den Teammitgliedern, etwa in Meetings, den Stakeholdern oder auch Themen, welche nicht mit der Arbeit zusammenhängen.
- *Team collaboration:* Beschreibt den Austausch von Quellcode und Dateien sowie Koordination des Teams.
- *Dev-Ops:* Hier werden Nachrichten gesammelt, welche den Status der Software beschreiben. Damit ist beispielsweise gemeint, dass kürzlich neue Änderungen vorgenommen oder Fehler behoben wurden.
- *Customer Support:* Umfasst die Einweisung neuer Benutzer in das System, indem bestimmte Funktionen erklärt werden.

Community support: Bei diesen Nachrichten handelt es sich um die Teilnahme an gemeinschaftlichen Gruppen.

- *Communities of practice:* Austausch, bei dem sich beispielsweise über neue Frameworks austauscht werden kann.

Die beschriebenen "Purpose Types" mit ihren "Purpose Categories" besitzen noch eine weitere Unterteilung in "Purpose Subcategories" [24]. Diese feinere Unterteilung in eine weitere Subkategorie wird in Tabelle 4.1 aufgezeigt. Die Tabelle orientiert sich an der Ausarbeitung von Parra et al. [24] und veranschaulicht die Zugehörigkeit der einzelnen Unterkategorien zu ihren jeweiligen Hauptkategorien.

Purpose Category	**Pupose Subcategory**
Discovery and news aggregation	Interesting/relevant blogs
Networking and social activities	Similar interests Similar jobs
Fun	Gaming Sharing gifs and memes
Communication	Communication with teammates Communication with Stakeholders Non-work topics
Team Collaboration	Team managment File and code sharing
Customer Support	Bugs Troubleshooting How-to
Dev-Ops	Development operation notification Software deployments Team Q & A
Communities of practice	Keep up with specific frameworks/communities Bouncing ideas off of other People in the communitiy Learning about new tools and frameworks

Tabelle 4.1: Purpose Categories und Subcategories der Datenbasis. Vgl. [24, S. 8]

Die Daten sind als CSV und Microsoft Excel Open XML Spreadsheet verfügbar, wobei sich in jeder Zeile eine Nachricht aus dem Forum befindet. Es gibt insgesamt sieben Eigenschaften für jeden Eintrag, bestehend aus dem Channel zu der jeweiligen Software, einer message ID, einem Zeitstempel, dem Namen des Autors, der Nachricht, des "Purpose Type", der "Purpose Category" und der "Purpose Subcategory". Ein Ausschnitt aus dem Datensatz der Microsoft Excel Tabelle ist in Abbildung 4.2 zu sehen.

Channel	messageId	time	user	message	Purpose	Category	Subcategory
Cucumber	55e714d4a75db4b375c3a921	2015-09-02T15:25:08.427Z	dkowis	If you're in ruby land, or can use ruby stuff, there's a handy tool called VCR	Community support	Participation in Communities of Practice	Bouncing Ideas off of others in the Community
Cucumber	55e714da6419c98422b8c484	2015-09-02T15:25:14.367Z	dkowis	that you can use to record/playback http requests	Team Wide	Communication	Communication with Teammates
Cucumber	55e714de17b2081605a55995	2015-09-02T15:25:18.132Z	aslakhellesoy	Take email. I need to verify that my app sends an email when X happens. I could hook my app up to a fake mail server to do that.	Team Wide	Communication	Communication with Teammates

Abbildung 4.2: Datenausschnitt aus Excel des Gittercom Datensatzes [25]

4.5 Auswahl der Kategorien

Im Kapitel zuvor wurden die “Purpose Categories” und ”Purpose Subcategories“ bereits vorgestellt. Für die automatische Klassifizierung wäre es denkbar alle Kategorien zu betrachten, da sich diese, basierend auf den Erkenntnissen von Lin et al. [21], als sinnvolle Kategorien für Benutzer und Entwickler herausgestellt haben. Auf Grundlage der

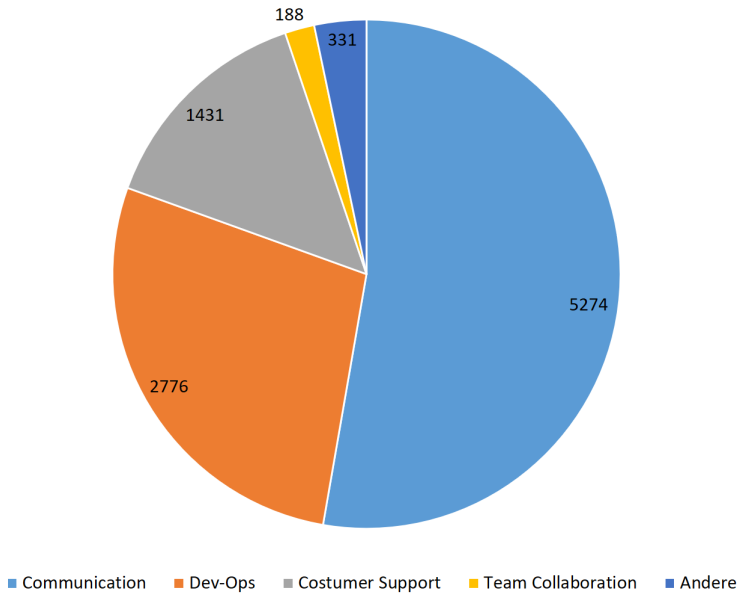

Abbildung 4.3: Aufteilung der Daten in die Kategorien [25, S.3]

bereits erbrachten Analyse von Parra et al. [24] und der Ergebnisse aus dem Diagramm 4.3 ist anzunehmen, dass die Kategorien Communication (5274 Nachrichten), Dev-Ops (2776 Nachrichten) und Customer Support (1431 Nachrichten) am häufigsten bei der Kommunikation zwischen Entwicklern auf Gitter vorkommen. Außerdem gehören diese drei Kategorien zu dem "Purpose Type" "Team-wide purpose" und beschreiben, wie bereits in Kapitel 4.4 erklärt, die Kommunikation, welche sich auf die Auslieferung und Entwicklung der Software bezieht und demzufolge ein wichtiger Teil für die Organisation des Teams und dessen Zusammenarbeit ist [24]. Die restlichen Kategorien machen außerdem insgesamt nur 5.19% der Daten aus. Dementsprechend liegt der Fokus dieser Arbeit auf den drei oben genannten Kategorien. Wie bereits erwähnt, besteht jede Kategorie aus mehreren Unterkategorien. Auch hier wäre es sinnvoll, bezüglich der automatischen Klassifizierung einzelne Unterkategorien gesondert zu betrachten. Um konsistent mit der Auswahl der Kategorien zu bleiben wird vorausgesetzt, dass eine Unterkategorie mindestens 1000 Datenpunkte enthält, damit sie neben ihrer Oberkategorie noch einmal einzeln aufgeführt wird. Die Kategorie

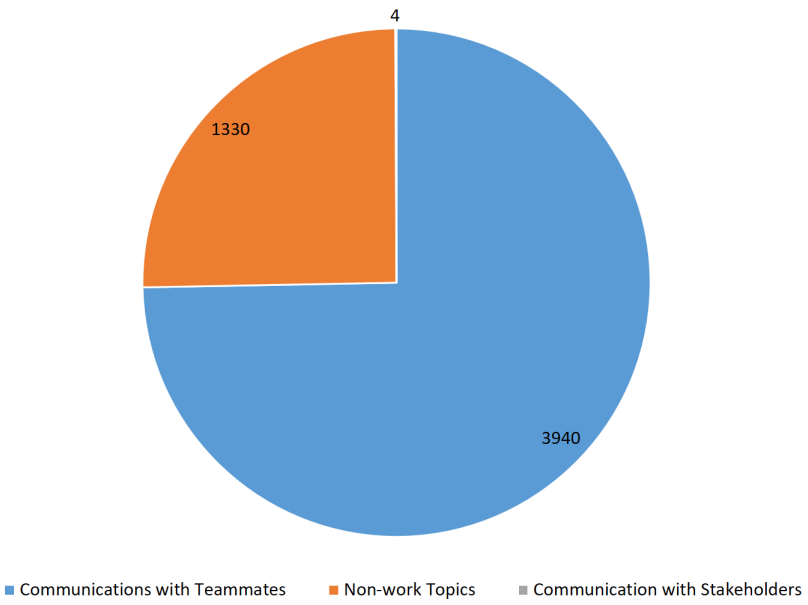

Abbildung 4.4: Aufteilung der Kategorie Communication in seine Unterkategorien

Communication enthält, wie in Abbildung 4.4 zu sehen ist, 5274 Datenpunkte. Wie zu erkennen ist, besteht die Kategorie zum Großteil aus den Unterkategorien Communications with Teammates (3940) und Non-work Topics (1330). Die Unterkategorie Communication with Stakeholders enthält dabei nur vier Datenpunkte. Dementsprechend werden die beiden erstgenannten Unterkategorien noch einmal separat betrachtet. Die Kategorie Dev-Ops enthält, wie in Abbildung 4.5

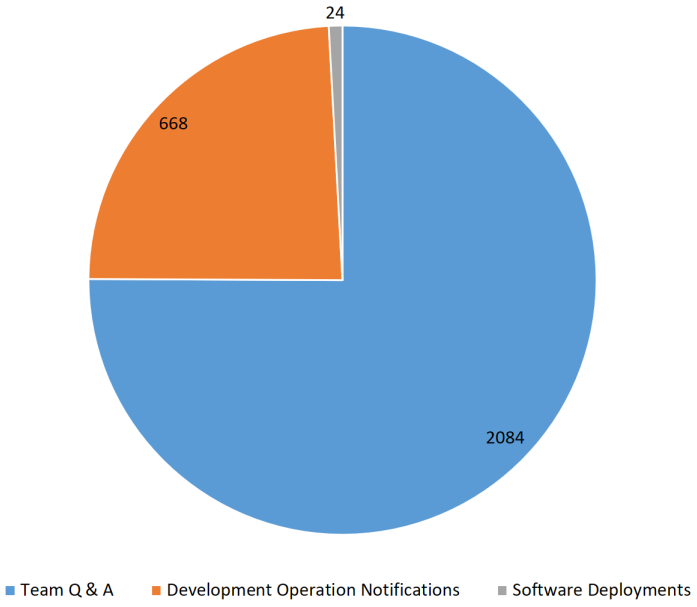

Abbildung 4.5: Aufteilung der Kategorie Dev-Ops in seine Unterkategorien

zu sehen ist, 2776 Nachrichten. Zu erkennen ist dabei, dass die Unterkategorie Team Q & A relevant genug ist, um sie hinsichtlich der Analyse gesondert zu betrachten, da sie 2084 Nachrichten enthält. Die Anzahl der Datenpunkte innerhalb der restlichen Unterkategorien Development Operation Notifications (668) und Software Deployments (24) liegen unter dem Schwellenwert von 1000 und werden dementsprechend nicht isoliert betrachtet. Die Kategorie Customer Support enthält, wie in Abbildung 4.6 zu sehen ist, 1431 Datenpunkte. In diesem Fall erreicht keine der enthaltenen Unterkategorien Trouble Shooting (485), Bug (71), Documentation (4), Example (2), How-to (860) und General (9) den Schwellenwert 1000. Dementsprechend werden keine der genannten Unterkategorien separat betrachtet.

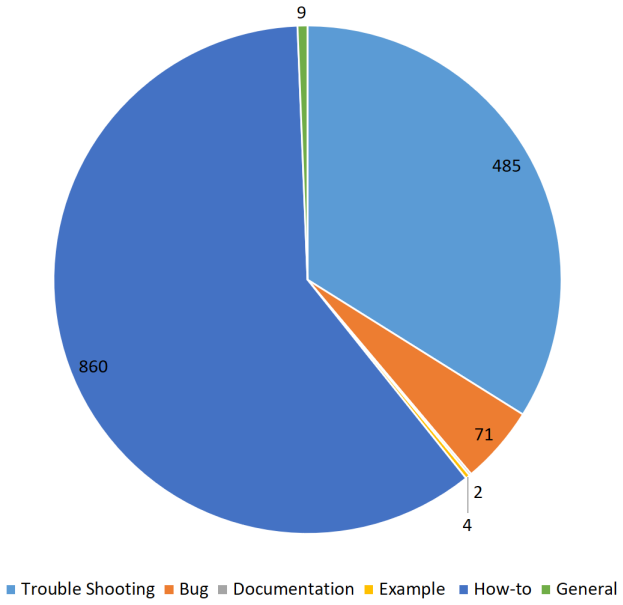

Abbildung 4.6: Aufteilung der Kategorie Customer Support in seine Unterkategorien

4.6 Labeling der Daten für die Klassifikation

Wie in Kapitel 2.3 bereits erklärt, werden für binäre Klassifikationsalgorithmen Labels benötigt, damit die Algorithmen Muster und Merkmale durch Trainingsdaten lernen können. Die Abbildung 4.7 zeigt die Aufteilung der Daten in die verschiedenen Kategorien. Dabei ergeben sich drei Datensätze für die Ebene “Purpose Category”, bestehend aus Communication, Customer Support und Dev-Ops. Für die Ebene “Purpose Subcategory” ergeben sich weitere drei Datensätze, Communication with Teammates, Non-work Topics und Team Q & A. Jede diese Kategorien bildet demnach einen Datensatz, woraus sich sechs Datensätze ergeben. In jedem Datensatz wurde eine weitere Spalte mit der Bezeichnung Sentiment hinzugefügt und ihr “1” zugewiesen, wenn es sich um eine Nachricht aus der Kategorie handelt und “0” wenn dies nicht der Fall ist. Dieses Vorgehen soll für die binäre Klassifikation der einzelnen Kategorien dienen. Demnach sind die “1” und die “0”, die Label für die Klassen. Ein beispielhafter Ausschnitt aus jedem Datensatz ist im Anhang B zu sehen.

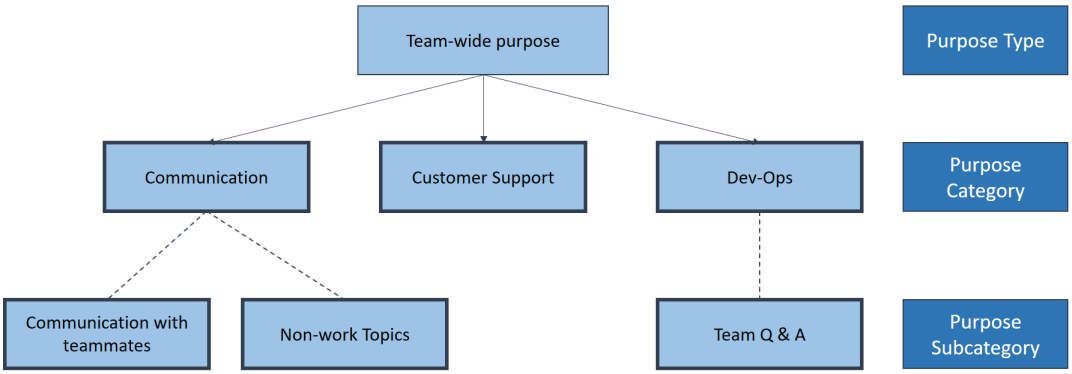

Abbildung 4.7: Aufteilung des Datensatzes in sechs Datensätze:
Die stärker umrandeten Kategorien bilden jeweils einen eigenen Datensatz für ein binäres Klassifikationsproblem

4.7 Auffälligkeiten in den Daten

Bei der manuellen Aufbereitung der Daten sind einige Auffälligkeiten in den Datensätzen festzustellen. Es sind bestimmte Satzzeichen in den Daten vorgekommen, welche keinen Mehrwert für den Text oder das Verständnis der Nachricht bieten. Unter anderem sind auch viele Unicode[6] Symbole in den Nachrichten enthalten gewesen. Auch der "newline" Command "\n" kam in den Nachrichten vor. Da es sich bei dem Datensatz um einen Nachrichtenaustausch handelt, werden auch die unterschiedlichen User referenziert mit dem Symbol "@". Durch den Nachrichtenaustausch innerhalb des Chats wurden auch Links verschickt, welche bereits bekannte Lösungsansätze beinhalten oder zu verwendende Librarys. Tabelle 4.2 dokumentiert alle erfassten Besonderheiten, wobei auf den ersten Blick die zahlreichen überflüssigen Satzzeichen und Unicodes auffallen.

[6]https://home.unicode.org, Zugriff: 20.10.2022

Auffälligkeiten	**Satz**
\u2019	shouldn \u2019 t be too tough.
\u201c	\u201cImprove Your Backbone Views with Marionette Behaviors\
\u201d	@guybedford A simple ‘\u201d postinstall \u201d:\u201dnpm install\‘ script would be much more preferable than this shell script.
\u2018	\u2018 Native Backbone Mixins now a part of Marionette’
\u2014	RESET \u2014 HARD
\u2026	I thought they changed their release stuff recently\u2026 it might hae been months ago actually...
:worried:	or I solve that...who knows :worried:
:)	Okay. I’ll try to read the docs one last time :)
\	sure, you can ignore the \packages\\and \stuff for configuring packages
@name	@guybedford sweet.
\n	i18n == ???\nFormatter == obsolete\nReporter == implement in cucumber-jvm\nResult == implement in cucumber-jvm\n

Tabelle 4.2: Auffälligkeiten im Datensatz Gittercom

4.8 Preprocessing der Daten

Das Preprocessing der Daten wurde in zwei Phasen unterteilt. Die erste Phase ist in Abbildung 4.8 dargestellt und beschreibt die Bereinigung der Daten.

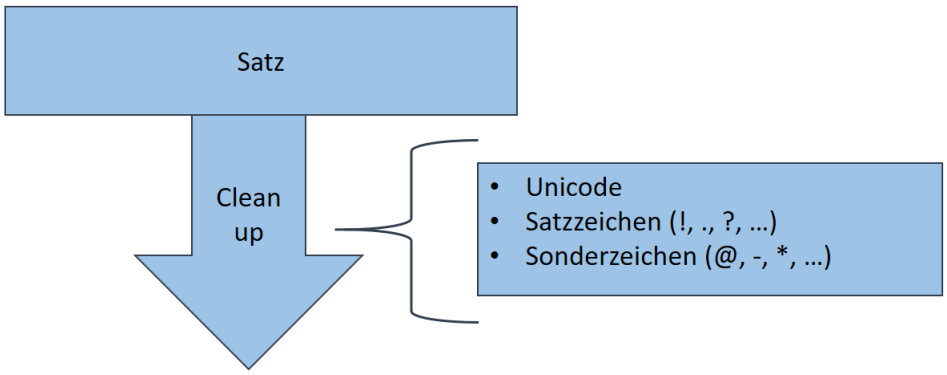

Abbildung 4.8: Preprocessing Phase 1

Clean up	**Satz**
\n vorher	How are you finding marionette, what are you using it for?\n\n
\n nacher	How are you finding marionette what are you using it for
\u2019 vorher	haven\u2019t been around, sorry!
\u2019 nachher	haven t been around sorry
:) + @ vorher	oh cool. hey there @mydigitalself :)
:) + @ nachher	oh cool hey there mydigitalself
links vorher	this is reminding me how ugly this method is: [behaviorEvents](https://github.com/marionettejs/backbone.marionette/blob/master/src/marionette.behaviors.js#L78-L97) is.
links nachher	this is reminding me how ugly this method is behaviorEvents https github com marionettejs backbone marionette blob master src marionette behaviors js L78 L97 is

Tabelle 4.3: Cleanup der Daten

Die aus dem Kapitel 4.7 herausgearbeiteten Auffälligkeiten wurden dabei mit den Regular expression operations[7] von Python entfernt. Hierbei wurden alle Unicode Zeichen (\u), Sonderzeichen sowie Zahlen gelöscht und Links angepasst. Eine Besonderheit stellt dabei das Entfernen des "newline" Commands \n dar, denn dabei werden beispielsweise auch \n aus dem Befehl \npm entfernt. Hier wurde eine Lösung gefunden, das Zeichen "\" des Befehls \npm und im Anschluss die \n zu entfernen, sodass der Befehl npm bestehen bleibt. Beispiele für das Bereinigen der herausfordernden Daten sind in der Tabelle 4.3 abgebildet.
Bei der zweiten Phase des Preprocessing, abgebildet in 4.9, werden anschließend Stop Words (siehe Anhang A) gelöscht und die übrigen Wörter mit Hilfe des WordNetLemmatizer[8], basierend auf der lexikalischen Datenbank WordNet, lexikografisch auf ihre Stämme beschränkt. In Tabelle 4.4 ist eine Nachricht aus dem Datensatz vor dem Clean up, nach der ersten Phase und abschließend nach der zweiten Phase zu sehen.

[7] https://docs.python.org/3/library/re.html, Zugriff: 20.10.2022
[8] https://www.nltk.org/_modules/nltk/stem/wordnet.html, Zugriff: 20.10.2022

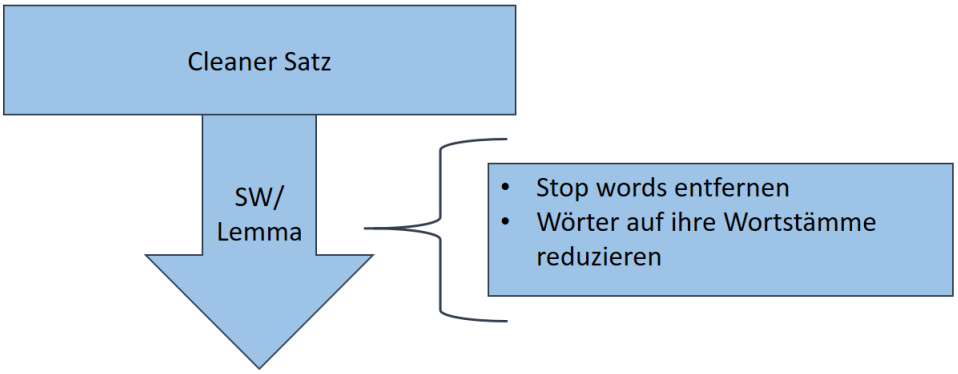

Abbildung 4.9: Phase 2

Phase	**Satz**
Vor Phase 1	yea, his comments imply that he has it in both places. He very well could be doing something evil.
Phase 1	yea his comments imply that he has it in both places He very well could be doing something evil
Phase 2	yea comment imply place He well could something evil

Tabelle 4.4: Preprocessing der Daten

4.9 Training

Trainings- und Testdaten

Zur Durchführung des Trainings müssen die Datensätze zuerst in eine Trainings- und Testmenge eingeteilt werden [16]. Aufgrund der Menge an Daten wurde auf eine Validierungsmenge verzichtet. Der Anteil an Daten, wie in Tabelle 4.5 gezeigt, ist für den Trainingsdatensatz auf 75% und der Anteil für die Testdaten auf 25% aller Daten festgelegt. Die beiden Mengen sind vollkommen unabhängig voneinander und überschneiden sich nicht. Zur Vermeidung eines "Bias" durch die Auswahl bestimmter Aussagen werden die Daten für die Mengen randomisiert ausgewählt [16].

Trainingsdaten	**Testdaten**
75%	25%

Tabelle 4.5: Datensplitting

Bevor die Daten für jede Kategorie in Trainings- und Testmenge

unterteilt werden, wird der Datensatz auf "Missing Values" untersucht [35]. "Missing Values" beschreiben fehlende Datenpunkte in einem Datensatz, welche das Ergebnis des Modells (Klassifikationsmodell) beeinflussen können [35]. Da es sich bei den im Datensatz vorliegenden fehlenden Datenpunkten um die Unterkategorien der "Purpose Category" Fun handelt und diese für alle sechs Klassifikationsprobleme als "0" gelabelt sind, wurden diese einfach aus dem Datensatz entfernt. Zusätzlich fehlt nur ein weiterer Datenpunkt, bei dem es sich jedoch um eine Nachricht handelt, welcher daher auch entfernt wird. Wie bereits in Kapitel 4.6 beschrieben, gilt für alle Datenpunkte, welche zu der Kategorie gehören, das Label "1" und für den Rest "0". Alle sechs erarbeiteten Datensätze haben eine unterschiedliche Menge an "1 - zugehörig" und "0 - nicht zugehörig". Diese ungleiche Verteilung in den Daten wird auch als "imbalanced data" (dt. unbalancierte Daten) bezeichnet [20]. Ein Verfahren, um mit solchen Daten umzugehen ist es, die Anzahl der Datenpunkte der "stärkeren" Klasse auf die Anzahl der Klasse mit den geringeren Einträgen zu beschränken [20]. In diesem Fall meint die "stärkere" Klasse diejenige, welche die meisten Datenpunkte aufweist [20]. Das Verfahren, die eine Klasse zu verkleinern, wurde auch randomisiert durchgeführt, um auch wieder in diesem Fall keinen "Bias" bei der Auswahl zu erlauben. Die Tabellen 4.6 und 4.7 zeigen dabei die Verteilung der Klassen "1" und "0" bevor und nachdem sie ausgeglichen wurden, sowohl für die drei Datensätze aus den "Purpose Categories" als auch für die drei Datensätze aus den "Purpose Subcatgories". Während bei dem Datensatz Communication die Klasse "1" um Datenpunkte verkleinert werden musste, musste bei allen anderen Datensätze die Klasse "0" um Datenpunkte verkleinert werden. Im nächsten Schritt werden die Merkmalsdaten mit dem *CountVectorizer* von *sklearn* vektorisiert und somit für das Training bereitgestellt.

	Communication		Dev-Ops		Customer Support	
	1	0	1	0	1	0
unbalanciert	5274	4689	2776	7187	1430	8533
balanciert	4689	4689	2776	2776	1430	1430

Tabelle 4.6: Verteilung der Daten der Purpose Categorys

	Communication with teammates		Non-work topics		Q & A	
	1	0	1	0	1	0
unbalanciert	3940	6023	1330	8633	2084	7879
balanciert	3940	3940	1330	1330	2084	2084

Tabelle 4.7: Verteilung der Daten der Purpose Subcategorys

4.9.1 Klassifikationsalgorithmen

Auf Grundlage der in Kapitel 2.4 beschriebenen Klassifikationsalgorithmen und unter Verwendung dieser Algorithmen in den beschriebenen Verfahren für die automatische Klassifizierung aus dem Kapitel 3, wurden sechs verschiedene binäre Klassifikationsalgorithmen implementiert. Dabei kommt das in dieser Arbeit als Single Training bezeichnete Verfahren zum Einsatz, bei dem die Klassifikationsalgorithmen einzeln für jede der sechs Kategorien (Communication, DevOps, Customer Support, Communication with Teammates, Team Q&A und Non-work Topics) trainiert werden. Im Anschluss wurden die Algorithmen erneut mit der cross-validation[9] Funktion von *sklearn* und einem K-Fold von 10 trainiert. Die Zahl 10 wird in der Literatur als die zuverlässigste Anzahl an Folds angesehen [37] [17]. Die cross-validation wird bei Klassifikationsalgorithmen angewendet, um die Leistungsfähigkeit durch eine mehrfache, also k-fache Kreuzvalidierung zu bewerten [37]. Dabei wird der Datensatz in unabhängige Teile unterteilt und das in diesem Fall 10-fach, damit durch die zufällige Aufteilung eine exaktere Genauigkeitsschätzung erfolgen kann [37].
Die folgenden Klassifikationsalgorithmen wurden trainiert:

- Logistische Regression: Aus *sklearn.linear_model* wurde der Algorithmus *LogisticRegression()* trainiert
- Naive Bayes: Aus *sklearn.naive_bayes* wurde der Algorithmus *MultinomialNB()* trainiert
- Decision Tree: Aus *sklearn.tree* wurde der Algorithmus *DecisionTreeClassifier()* trainiert
- Random Forest: Aus *sklearn.ensemble* wurde der Algorithmus *RandomForestClassifier* trainiert

[9] https://scikit-learn.org/stable/modules/cross_validation.html, Zugriff: 17.10.2022

- Support Vector Machine: Aus *sklearn* wurde der Algorithmus *svm.SVC* trainiert
- Ensemble (Voting Classifier): Aus *sklearn.ensemble* wurde der Algorithmus *VotingClassifier* trainiert

Die Funktionsweise der einzelnen Algorithmen und ihrer theoretischen Grundlagen sind in Kapitel 2.4 nachzulesen. Für den Voting Classifier[10] gibt es zwei Methoden, welche eingestellt werden können und in dieser Arbeit verwendet wurden. Die zwei Methoden werden unterschieden in "hard" und "soft" voting [27]. Wenn die Option "hard" ausgewählt wird, so wird die vorhergesagte Klasse durch eine Mehrheitsentscheidung selektiert [27]. Bei der Methode "soft" wird die Entscheidung durch den arg max der Summe der Vorhersagen durch die Algorithmen bestimmt [27]. Bei den trainierten Voting Classifier in dieser Arbeit besteht das Ensemble aus *LogisticRegression()*, *MultinomialNB()*, *DecisionTree-Classifier()*, *RandomForestClassifier* und *svm.SVC*.

[10] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html, Zugriff: 20.10.2022

Kapitel 5

Ergebnisse

5.1 Ergebnisse des Single Training Verfahrens

Im Folgenden werden die Ergebnisse aus der Trainings- und Validierungsphase der Implementierung des Single Training Verfahrens vorgestellt. Es soll ein Algorithmus gefunden werden, welcher die beste Kombination aus Effizienz und Accuracy bei der Klassifikation der Kategorien erzielt. Bei der Auswertung der Ergebnisse werden die Accuracy und der F1-Score (Siehe Kapitel 2.5) der jeweiligen Algorithmen verwendet. Für die Vorstellung der Ergebnisse werden insbesondere die verwendeten Voting Classifier den restlichen Klassifizierungsalgorithmen gegenübergestellt.
Diesbezüglich werden die Ergebnisse der Klassifikationsalgorithmen für die einzelnen Kategorien gesondert vorgestellt. Schlussendlich soll für jede Kategorie der beste Classifier herausgestellt werden. Zur weiterführenden Analyse werden außerdem die Ergebnisse der Cross Validierung aus Kapitel 5.2 hinzugezogen.

Ergebnisse der Kategorie Communication

Wie in Kapitel 4.9 erwähnt, werden die Daten für die Trainings- und Validierungsphase balanciert, wodurch insgesamt 9378 Datenpunkte für den Datensatz zur Kategorie Communication verbleiben. Dementsprechend bleiben jeweils 4689 Nachrichten, die zur Kategorie Communication (Klasse "1") beziehungsweise nicht zur Kategorie Communication (Klasse "0") gehören. Das Validierungsset besteht dabei aus 25% der Daten und wird für die unten folgende Evaluierung der

Klassifikationsmodelle verwendet. Die Tabelle 5.1 beinhaltet dabei die Werte für die Accuracy und den F1-Score jedes Klassifikationsmodelles bei der Validierung.
Beginnend mit der Accuracy liegen die Werte in einem Bereich von 0.58 bis 0.66. Dabei erzielt der Decision Tree, basierend auf dem zugrundeliegenden Datenset, den geringsten Wert bei der Accuracy mit 0.58 und der Naive Bayes gemeinsam mit dem Voting (soft) Classifier den höchsten Wert mit 0.66. Analog zu der Accuracy erreicht auch bei dem F1-Score der Decision Tree den geringsten Wert mit 0.52. Das höchste Ergebnis für den F1-Score erzielt der Voting (hard) Classifier mit 0.69.
Beim Vergleich der zwei Voting Classifier fällt auf, dass der (soft) Classifier eine höhere Accuracy aufweist, jedoch der (hard) Classifier einen höheren F1-Score erzielt.
Außerhalb der Voting Classifier erreicht die Support Vector Machine den nächsthöheren Wert für den F1-Score mit 0.68.

	Accuracy	F1-Score
Naive Bayes	**0.66**	0.67
Logistic Regression	0.65	0.66
Decision Tree	0.58	0.52
Random Forest	0.61	0.67
SVM	0.64	0.68
Voting (hard)	0.64	**0.69**
Voting (soft)	**0.66**	0.68

Tabelle 5.1: Accuracy und F1-Score der Klassifikationsalgorithmen für die Kategorie Communication

Ergebnisse der Kategorie Communication with Teammates

Analog zu der vorherigen Kategorie werden auch hier die Daten balanciert, sodass insgesamt 7880 Datenpunkte für den Datensatz zur Kategorie Communication with Teammates verbleiben. Daraus ergeben sich wieder je Klasse (“1” oder “0”) für die Kategorie Communication with Teammates jeweils 3940 Nachrichten. Das Validierungsset besteht wieder aus 25% der Daten und wird auch hier für die folgende Evaluierung der Klassifikationsalgorithmen verwendet. Die Tabelle 5.2 beinhaltet dabei die Werte für die Accuracy und den F1-Score jedes

Klassifikationsmodelles bei der Validierung.
Bei der Accuracy liegen die Werte in einem Intervall von 0.57 bis 0.61. Dabei erzielen Decision Tree und Random Forest, basierend auf dem zugrundeliegenden Validierungssatz, die geringsten Werte für die Accuracy mit 0.57 und der Voting (soft) Classifier den höchsten Wert mit 0.61. Analog zu der Accuracy erreicht auch bei dem F1-Score der Decision Tree den geringsten Wert mit 0.53. Das höchste Ergebnis für den F1-Score erzielen der Voting (hard) Classifier, der Random Forest und die Support Vector Machine mit 0.66.
Beim Vergleich der zwei Voting Classifier fällt auf, dass der (soft) Classifier wie auch bei der Kategorie Communication eine höhere Accuracy aufweist, jedoch der (hard) Classifier einen höheren F1-Score erzielt.
Außerhalb der Voting Classifier erreicht die Support Vector Machine gemeinsam mit dem Naive Bayes die höchste Accuracy mit 0.60.

	Accuracy	F1-score
Naive Bayes	0.60	0.63
Logistic Regression	0.59	0.60
Decision Tree	0.57	0.53
Random Forest	0.57	**0.66**
SVM	0.60	**0.66**
Voting (hard)	0.59	**0.66**
Voting (soft)	**0.61**	0.63

Tabelle 5.2: Accuracy und F1-Score der Klassifikationsalgorithmen für die Kategorie Communication with Teammates

Ergebnisse der Kategorie Non-work Topics

Auch hier werden die Daten balanciert, sodass insgesamt 2660 Datenpunkte für den Datensatz zur Kategorie Non-work Topics verbleiben. Dabei ergeben sich 1330 nicht arbeitsbezogene Nachrichten (Klasse "1") und 1330 arbeitsbezogene Nachrichten (Klasse "0") für die Kategorie Non-work Topics. Identisch wie bei den anderen Datensätzen besteht das Validierungsset wieder aus 25% der Daten und wird auch hier für die folgende Evaluierung der Klassifikationsalgorithmen verwendet. Die Tabelle 5.3 beinhaltet dabei die Werte für die Accuracy und den F1-Score jedes Klassifikationsmodelles bei der Validierung.

Die Werte für die Accuracy liegen für die Kategorie non-work Topics in einem Bereich von 0.63 bis 0.73. Den höchsten Wert für die Accuracy weist der Naive Bayes mit 0.73 auf. Dabei erzielt der Decision Tree, basierend auf dem zugrundeliegenden Validierungssatz, sowohl für die Accuracy (0.63) als auch für den F1-Score (0.63) die geringsten Werte. Das höchste Ergebnis für den F1-Score erreicht der Voting (hard) Classifier mit 0.76.
Die zwei Voting Classifier besitzen die gleiche Accuracy (0.72), unterscheiden sich jedoch im F1-Score, bei dem der (soft) Classifier mit 0.73 einen geringeren Wert aufweist.
Außerhalb der Voting Classifier erreicht der Naive Bayes den höchsten F1-Score mit 0.74.

	Accuracy	F1-Score
Naive Bayes	**0.73**	0.74
Logistic Regression	0.71	0.72
Decision Tree	0.63	0.63
Random Forest	0.67	0.73
SVM	0.70	0.73
Voting (hard)	0.72	**0.76**
Voting (soft)	0.72	0.73

Tabelle 5.3: Accuracy und F1-Score der Klassifikationsalgorithmen für die Kategorie Non-work Topics

Ergebnisse der Kategorie Dev-Ops

	Accuracy	F1-Score
Naive Bayes	0.61	0.61
Logistic Regression	0.61	0.61
Decision Tree	0.59	0.60
Random Forest	0.56	0.36
SVM	0.61	0.57
Voting (hard)	**0.62**	0.53
Voting (soft)	**0.62**	**0.63**

Tabelle 5.4: Accuracy und F1-Score der Klassifikationsalgorithmen für die Kategorie Dev-Ops

Analog zu den vorherigen Kategorien werden wieder die Daten balanciert, sodass insgesamt 5552 Datenpunkte für den Datensatz zur

Kategorie Dev-Ops übrig bleiben. Daraus ergeben sich wieder je Klasse ("1" oder "0") jeweils 2776 Nachrichten für die Kategorie Dev-Ops. Das Validierungsset besteht wieder aus 25% der Daten und wird auch hier für die folgende Evaluierung der Klassifikationsalgorithmen verwendet. Die Tabelle 5.4 beinhaltet dabei die Werte für die Accuracy und den F1-Score jedes Klassifikationsmodells bei der Validierung.
Bei der Accuracy liegen die Werte dabei zwischen 0.56 und 0.62. Der Random Forest erreicht den niedrigsten Wert sowohl für die Accuracy mit 0.56 als auch für den F1-Score mit 0.36. Die höchsten Werte für die Accuracy erzielen dabei die beiden Voting Classifier mit 0.62, wobei der (soft) Classifier zusätzlich den höchsten Wert für den F1-Score mit 0.63 erreicht.
Beim Vergleich der zwei Voting Classifier besitzt der Voting (hard) Classifier einen niedrigeren F1-Score (0.53).
Unabhängig von den Voting Classifiern erreichen die Support Vector Machine, der Naive Bayes sowie die Logistic Regression die höchste Accuracy mit 0.61. Analog erzielen der Naive Bayes und die Logistic Regression auch den höchsten F1-Score mit 0.61.

Ergebnisse der Kategorie Team Q & A

	Accuracy	F1-Score
Naive Bayes	**0.60**	0.61
Logistic Regression	0.59	0.59
Decision Tree	0.55	0.59
Random Forest	0.57	0.40
SVM	**0.60**	0.56
Voting (hard)	**0.60**	0.61
Voting (soft)	**0.60**	**0.62**

Tabelle 5.5: Accuracy und F1-Score der Klassifikationsalgorithmen für die Kategorie Team Q & A

Auch hier werden die Daten wieder wie in den anderen Kategorien balanciert, sodass insgesamt 4168 Datenpunkte für den Datensatz zur Kategorie Team Q & A verbleiben. Dabei ergeben sich 2084 Q & A Nachrichten (Klasse "1") und 2084, die nicht Q & A Nachrichten (Klasse "0") sind. Analog zu den anderen Datensätzen besteht das Validierungsset wieder aus 25% der Daten und wird auch hier für

die folgende Evaluierung der Klassifikationsalgorithmen verwendet. Die Tabelle 5.5 beinhaltet dabei die Werte für die Accuracy und den F1-Score jedes Klassifikationsmodells bei der Validierung.
Beginnend mit der Accuracy liegen die Werte für die Kategorie Team Q & A in einem Bereich von 0.55 bis 0.60. Die höchsten Werte für die Accuracy mit 0.60 erreichen der Naive Bayes, die Support Vector Machine sowie die beiden Voting Classifier. Dabei erzielt der Voting (hard) Classifier auch den höchsten Wert für den F1-Score mit 0.62. Mit 0.55 weist der Decision Tree den niedrigsten Accuracy Wert auf. Den kleinsten F1-Score erzielt dabei der Random Forest mit 0.40.
Die zwei Voting Classifier besitzen die gleiche Accuracy (0.60), unterscheiden sich jedoch im F1-Score, bei dem der (hard) Classifier mit 0.61 einen geringeren Wert aufweist.
Außerhalb der Voting Classifier erreicht der Naive Bayes den höchsten F1-Score mit 0.61.

Ergebnisse der Kategorie Customer Support

	Accuracy	F1-Score
Naive Bayes	**0.77**	**0.78**
Logistic Regression	0.73	0.73
Decision Tree	0.66	0.66
Random Forest	0.69	0.66
SVM	0.73	0.68
Voting (hard)	0.74	0.71
Voting (soft)	0.74	0.74

Tabelle 5.6: Accuracy und F1-Score der Klassifikationsalgorithmen für die Kategorie Customer Support

Analog zu den vorherigen Kategorien werden wieder die Daten balanciert, sodass insgesamt 2860 Datenpunkte für den Datensatz zur Kategorie Customer Support übrig bleiben. Daraus ergeben sich wieder je Klasse (“1” oder “0”) für die Kategorie Customer Support jeweils 1430 Nachrichten. Das Validierungsset besteht wieder aus 25% der Daten und wird auch hier für die folgende Evaluierung der Klassifikationsalgorithmen verwendet. Die Tabelle 5.6 beinhaltet dabei die Werte für die Accuracy und den F1-Score jedes Klassifikationsmodelles bei der Validierung.

Bei der Accuracy liegen die Werte dabei zwischen 0.66 und 0.77. Der Decision Tree erreicht den niedrigsten Wert für die Accuracy mit 0.66. Außerdem erzielt dieser gemeinsam mit dem Random Forest den niedrigsten F1-Score mit 0.66. Der Naive Bayes weist dabei sowohl für die Accuracy (0.77) als auch für den F1-Score (0.78) den höchsten Wert auf.
Beim Vergleich der zwei Voting Classifier erreichen beide eine Accuracy von 0.74, jedoch besitzt der Voting (hard) Classifier einen niedrigeren F1-Score mit 0.71.

- Decision Tree hat bei den Klassifikationsproblemen Communication, Communication with Teammates, Non-work Topics und Customer Support im Vergleich zu den anderen Algorithmen die geringsten Werte für die Accuracy und F1-Score erreicht.
- Random Forest hat bei dem Klassifikationsproblem Dev-Ops hinsichtlich der Accurcary und des F1-Score die geringsten Werte erzielt.
- Für das Klassifikationsproblem Team Q & A hat Random Forest den kleinsten Wert für den F1-Score und Decision Tree den kleinsten Wert für die Accuracy erreicht.
- Die Voting Classifier haben bei allen Kategorien konsistent hohe Werte für Accuracy und F1-Score vorweisen können. Trotzdem haben sie die Mehrheit der Kategorien nicht dominiert, in dem Sinne, dass sie sich nicht hinsichtlich beider Metriken zu stark abheben, da auch andere Klassifizierer ähnlich hohe Werte erzielt haben.
- Abgesehen von den Voting Classifiern haben Naive Bayes und Support Vector Machine die höchsten Werte hinsichtlich der Accuracy und des F1-Scores erreicht.
- Logistic Regression erzielte für keines der Klassifikationsprobleme die höchsten oder geringsten Werte bezüglich der Accuracy und des F1-Scores.

InfoBox 5.1: Zusammenfassung: Kapitel 5.1

5.2 Ergebnisse aus Training mit Cross-Validierung

Zur weiterführenden Analyse der Performance, bezüglich der Accuracy, der bereits oben aufgeführten Algorithmen wird das cross-validation[1] Verfahren angewendet. Dabei werden die gleichen Datensätze, die bereits im vorherigen Kapitel genutzt wurden, erneut verwendet. Die Ergebnissen aus dem Kapitel 5.1 zuvor haben ergeben, dass die Voting Classifier zwar hohe Werte hinsichtlich der Accuracy und F1-Score erzielt haben, jedoch diese nicht signifikant höher ausfielen als bei Verwendung eines einzelnen Algorithmus (z.B. siehe Tabelle 5.4 Naive Bayes 0.61 vs. Voting (hard + soft) 0.62). Die Voting Classifier bestehen aus mehreren Algorithmen und brauchen dementsprechend länger für das Training und die Validierung der Daten. Aus diesem Grund werden die Voting Classifier nicht für die Cross Validierung verwendet, da der zusätzliche Ressourcenaufwand wie beschrieben nur marginalen Zusatznutzen bietet und daher wenig effizient ist. Eine tiefere Diskussion dessen findet in Kapitel 6.1 statt.
Die Cross Validierung wird mit einem K-Folds von 10 durchgeführt und die Ergebnisse dieser 10 Durchgänge werden zusammengefasst betrachtet. Dabei stellt das Minimum die Accuracy des Durchgangs mit dem kleinsten Wert dar. Analog zeigt das Maximum den höchsten Accuracy Score, welcher bei den 10 Durchgängen erreicht wurde. Der "Mean" beschreibt dabei den Mittelwert der Scores über die 10 Durchgänge.

Ergebnisse der Kategorie Communication

Das Minimum für die Accuracy der Algorithmen für die Kategorie Communication liegt in einem Intervall von 0.51 für die Algorithmen Logistic Regression und Decesion Tree bis zu einem Wert 0.56, der von der Support Vector Machine erzielt wird. Bei dem Maximum erreicht auch hier der Decision Tree den kleinsten Accuracy Wert mit 0.64. Dabei hat der Naive Bayes den höchsten Wert mit 0.72 angenommen. Wie zuvor weist auch im Mittelwert der Decision Tree

[1] https://scikit-learn.org/stable/modules/cross_validation.html, Zugriff: 17.10.2022

den geringsten Wert mit 0.57 auf. Im Durchschnitt erreicht die Support Vector Machine, wie im Falle des Minimums, mit 0.61 den höchsten Wert.

	Min	Mean	Max
Naive Bayes	0.54	0.60	0.72
Logistic Regression	0.51	0.58	0.67
Decision Tree	0.51	0.57	0.64
Random Forest	0.54	0.59	0.67
SVM	0.56	0.61	0.68

Tabelle 5.7: Accuracy aus cross-validation der Klassifikationsalgorithmen für die Kategorie Communication

Ergebnisse der Kategorie Communication with Teammates

Für die Kategorie Communication with Teammates liegt die Minimum Accuracy in einem Bereich von 0.48 für den Decision Tree bis 0.52 für den Random Forest und die Support Vector Machine. Die höchste Accuracy in einem Durchgang erreichen der Random Forest und die Support Vector Machine (0.64). Wie auch für das Minimum und das Maximum erzielt die Support Vector Machine die höchste durchschnittliche Accuracy mit 0.56. Den geringsten Durchschnitt weist die Logistic Regression (0.53) auf.

	Min	Mean	Max
Naive Bayes	0.50	0.54	0.61
Logistic Regression	0.49	0.53	0.60
Decision Tree	0.48	0.54	0.62
Random Forest	0.52	0.55	0.64
SVM	0.52	0.56	0.64

Tabelle 5.8: Accuracy aus cross-validation der Klassifikationsalgorithmen für die Kategorie Communication with Teammates

Ergebnisse der Kategorie Non-work Topics

Der Naive Bayes Classifier erzielt für die Kategorie Non-work Topics im Vergleich zu den restlichen Algorithmen die höchsten Accuracy Werte für den schlechtesten (0.58) und den besten (0.73) Durchlauf

sowie auch im Durschschnitt (0.65). Den geringsten Durchschnittswert hinsichtlich der Accuracy der Durchläufe erreichen der Decision Tree und der Random Forest mit 0.61.

	Min	Mean	Max
Naive Bayes	0.58	0.65	0.73
Logistic Regression	0.56	0.63	0.72
Decision Tree	0.54	0.61	0.66
Random Forest	0.56	0.61	0.64
SVM	0.55	0.63	0.68

Tabelle 5.9: Accuracy aus cross-validation der Klassifikationsalgorithmen für die Kategorie Non-work Topics

Ergebnisse der Kategorie Dev-Ops

Für die Kategorie Dev-Ops erreicht die Support Vector Machine durchschnittlich den höchsten Accuracy Wert mit 0.56 und der Decision Tree, der Naive Bayes sowie der Random Forest die geringste mit 0.54. Die höchste Accuracy in einem Durchgang erreichen der Random Forest und die Support Vector Machine (0.63). Für den schlechtesten Durchlauf (Minimum) erzielt der Decision Tree mit 0.50 die höchste und der Naive Bayes mit 0.48 die niedrigste Accuracy.

	Min	Mean	Max
Naive Bayes	0.48	0.54	0.61
Logistic Regression	0.49	0.55	0.60
Decision Tree	0.50	0.54	0.59
Random Forest	0.49	0.54	0.63
SVM	0.49	0.56	0.63

Tabelle 5.10: Accuracy aus cross-validation der Klassifikationsalgorithmen für die Kategorie Dev-Ops

Ergebnisse der Kategorie Team Q & A

Der Naive Bayes Classifier erzielt für die Kategorie Team Q & A im Vergleich zu den anderen Klassifikationsalgorithmen die höchsten Accuracy Werte für den schlechtesten (0.49) und den besten (0.63) Durchlauf sowie auch im Durschschnitt (0.55). Den geringsten

Durchschnittswert hinsichtlich der Accuracy der Durchläufe erreicht der Decision Tree mit 0.52.

	Min	Mean	Max
Naive Bayes	0.49	0.55	0.63
Logistic Regression	0.46	0.54	0.59
Decision Tree	0.47	0.52	0.60
Random Forest	0.49	0.53	0.60
SVM	0.45	0.54	0.60

Tabelle 5.11: Accuracy aus cross-validation der Klassifikationsalgorithmen für die Kategorie Team Q & A

Ergebnisse der Kategorie Customer Support

	Min	Mean	Max
Naive Bayes	0.47	0.64	0.79
Logistic Regression	0.48	0.61	0.68
Decision Tree	0.46	0.58	0.66
Random Forest	0.50	0.59	0.80
SVM	0.52	0.63	0.76

Tabelle 5.12: Accuracy aus cross-validation der Klassifikationsalgorithmen für die Kategorie Customer Support

Den höchsten Durchschnittswert hinsichtlich der Accuracy der Durchläufe für die Kategorie Customer Support erreicht der Naive Bayes Classifier mit 0.64. Der Decision Tree erzielt im Vergleich zu den anderen Klassifikationsalgorithmen die niedrigste durchschnittliche Accuracy mit 0.58. Zusätzlich nimmt er auch die niedrigsten Werte im schlechtesten (0.46) sowie im besten (0.58) Durchlauf an. Die höchste Accuracy in einem Durchgang erreicht der Random Forest mit 0.80. Für den schlechtesten Durchlauf (Minimum) erzielt die Support Vector Machine die höchste Accuracy mit 0.52.

- Im Vergleich zu den anderen Algorithmen hat der Decision Tree bei den Kategorien Communication, Dev-Ops, Non-work Topics, Team Q & A und Customer Support im Durchschnitt aller Durchläufe bei der Cross Validierung die geringsten Werte für die Accuracy erzielt.
- Random Forest hat zusätzlich bei den Klassifikationsproblemen Non-work Topics und Dev-Ops hinsichtlich der Accurcary die geringsten Durchschnittswerte erreicht.
- Für die Kategorie Team Q & A hat die Logistic Regression den geringsten Durchschnittswert der Accuracy aufgewiesen.
- Für die Kategorien Communication, Communication with Teammates und Dev-Ops hat die Support Vector Machine die höchsten Werte im Durchschnitt erzielt.
- Der Naive Bayes Classifier hat dabei die höchsten Werte im Mittel für die Kategorien Non-work Topics, Team Q & A und Customer Support aufweisen können.

InfoBox 5.2: Zusammenfassung: Kapitel 5.2

5.3 Vergleich der Ergebnisse

Die Diagramme 5.1 bis 5.6 zeigen die Accuracies der verschiedenen Algorithmen für die jeweiligen Kategorien. Dabei werden die Ergebnisse aus dem Kapitel 5.1 mit denen aus Kapitel 5.2 verglichen. Es fällt auf, dass die Accuracy Werte beim Single Training Verfahren im Vergleich konsistent höher liegen als die Ergebnisse der Cross Validierung. Speziell bei der Betrachtung des Diagrammes 5.6 lässt sich beobachten, dass die durchschnittliche Accuracy über beide Verfahren und alle Algorithmen hinweg am höchsten ist. Wird der Fokus nun auf die Algorithmen gelegt wird ersichtlich, dass die Performance in Bezug auf die Accuracy bei dem Naive Bayes Classifier und der Support Vector Machine am besten ausfällt. Dies ist sowohl beim Single Training Verfahren als auch bei der Cross Validierung

erkennbar. Im Gegensatz dazu ist in allen Diagrammen der Trend zu erkennen, dass der Decision Tree konsistent die schlechtesten Ergebnisse für die Accuracy liefert. Wird der Fokus in jeder Kategorie jeweils auf den besten Algorithmus gelegt ist erkenntlich, dass in jedem Fall eine Accuracy von mindestens 0.60 erreicht wird. Wie in den Abbildungen 5.2 und 5.5 ersichtlich, geht die Accuracy für die Kategorien Communication with Teammates und Team Q & A nicht darüber hinaus. Im Gegensatz dazu liegt die Accuracy für die Kategorien Dev-Ops und Communication bei 0.61 beziehungsweise bei 0.66 (Siehe 5.1 und 5.4). Die am besten klassifizierten Kategorien sind Customer Support und Non-work Topics, welche sogar Accuracies über 0.70 aufweisen (siehe 5.6 und 5.3).

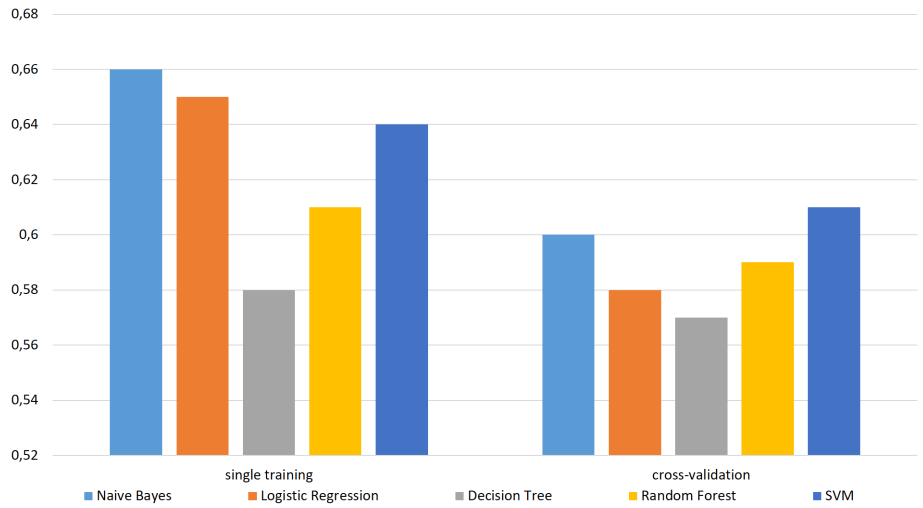

Abbildung 5.1: Vergleich der Accuracies des Single Training Verfahrens (links) zur Cross Validierung (rechts) für die Kategorie Communication: *Für das Verfahren Single Training erreicht der Naive Bayes Classifier mit 66% die beste und der Decision Tree mit 58% die schlechteste Accuracy. Bei der Cross Validierung erzielt die SVM die höchste Accuracy mit 61% und der Decision Tree bleibt auch hier wieder der schlechteste Classifier mit 57%.*

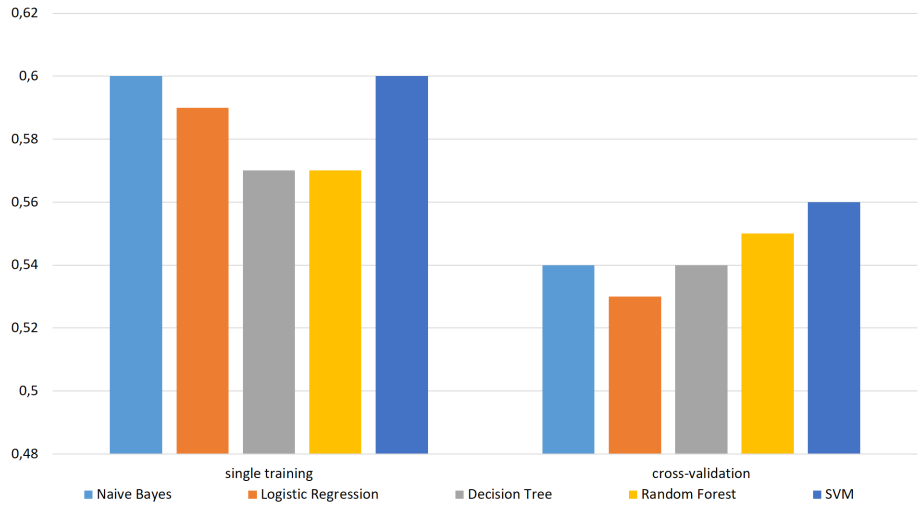

Abbildung 5.2: Vergleich der Accuracies des Single Training Verfahrens (links) zur Cross Validierung (rechts) für die Kategorie Communication with Teammates: *Für das Single Training Verfahren erreichen der Naive Bayes Classifier gemeinsam mit der SVM die höchste Accuracy mit 60% und der Decision Tree sowie der Random Forest mit 57% die schlechteste Accuracy. Bei der Cross Validierung erzielt die SVM die höchste Accuracy mit 56% und die Logisitc Regression ist hierbei der schlechteste Classifier mit 53%.*

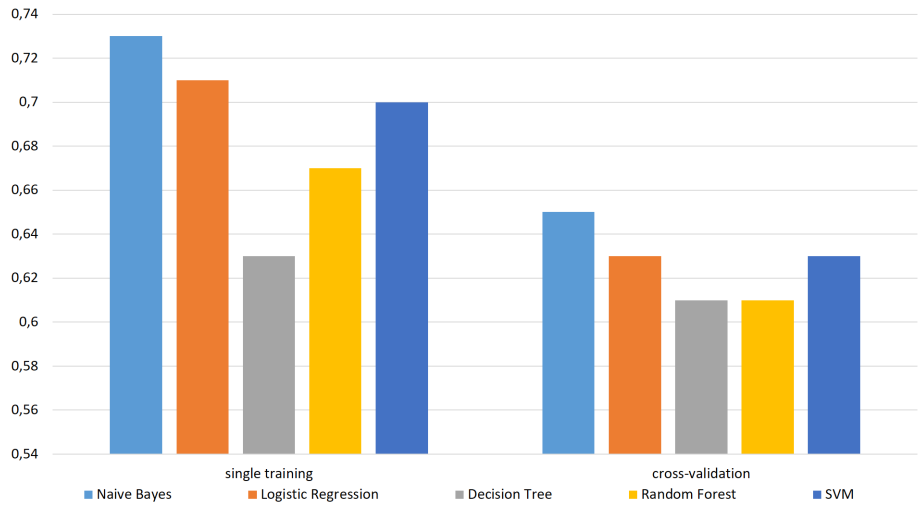

Abbildung 5.3: Vergleich der Accuracies des Single Training Verfahrens zur Cross Validierung für die Kategorie Non-work Topics: *Der Naive Bayes Classifier erzielt für das Single Training Verfahren und die Cross Validierung die besten Accuracies mit 73% und 65%. Der Decision Tree erreicht für beide Verfahren die schlechtesten Werte mit 63% und 61%, wobei der Random Forest für die Cross Validierung auch nur 61% annimmt.*

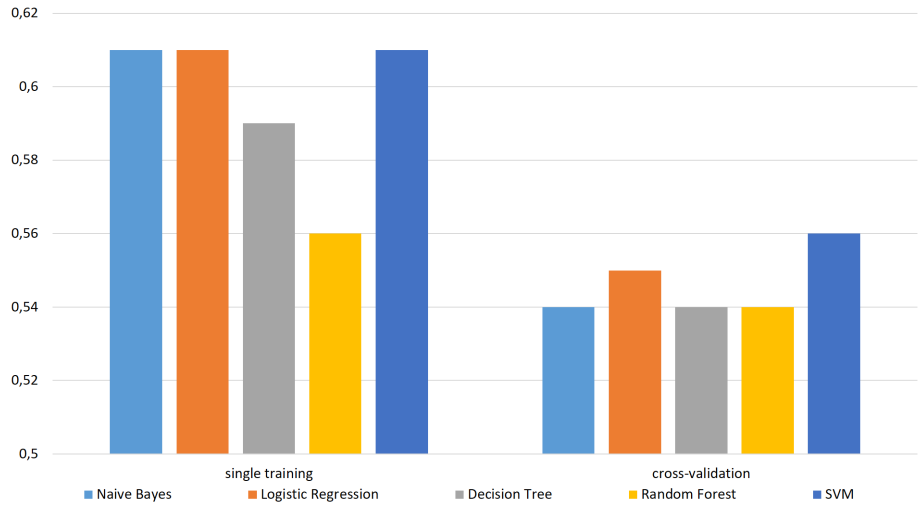

Abbildung 5.4: Vergleich der Accuracies des Single Training Verfahrens zur Cross Validierung für die Kategorie Dev-Ops: *Der Naive Bayes Classifier, die Logistic Regression sowie die SVM erzielten gemeinsam die besten Accuracy Werte mit 61% für das Single Training Verfahren. Die schlechtesten Ergebnisse für dieses Verfahren hat der Random Forest mit 66% erreicht. Für die Cross Validierung hat die beste Accuracy mit 56% die SVM und die schlechteste Naive Bayes, Decision Tree und Random Forest mit 54%.*

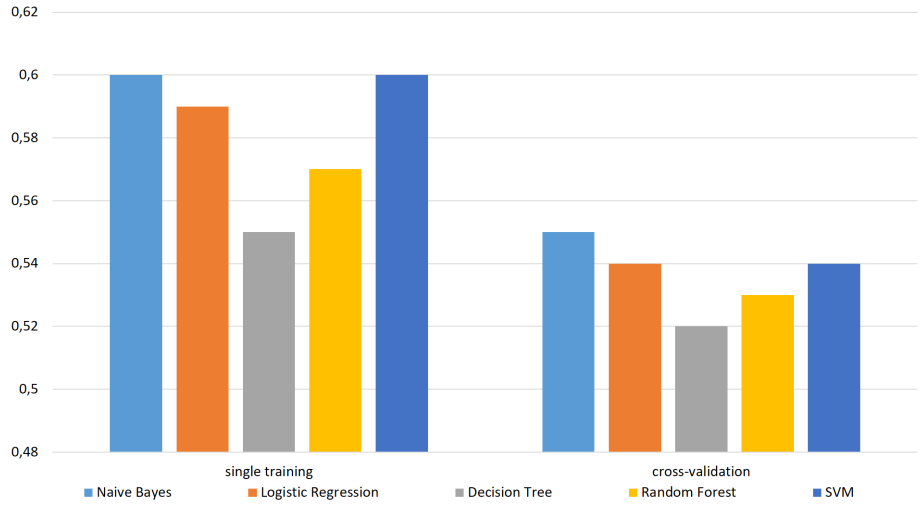

Abbildung 5.5: Vergleich der Accuracies des Single Training Verfahrens zur Cross Validierung für die Kategorie Team Q & A: *Für beide Verfahren hat die schlechtesten Werte für die Accuracy der Decision Tree mit 55% und 52% erreicht. Die SVM und der Naive Bayes Classifier haben für das Single Training mit 60% die besten Accuracies erreicht, wobei der Naive Bayes auch für die Cross Validierung die höchste Accuracy mit 55% erzielt hat.*

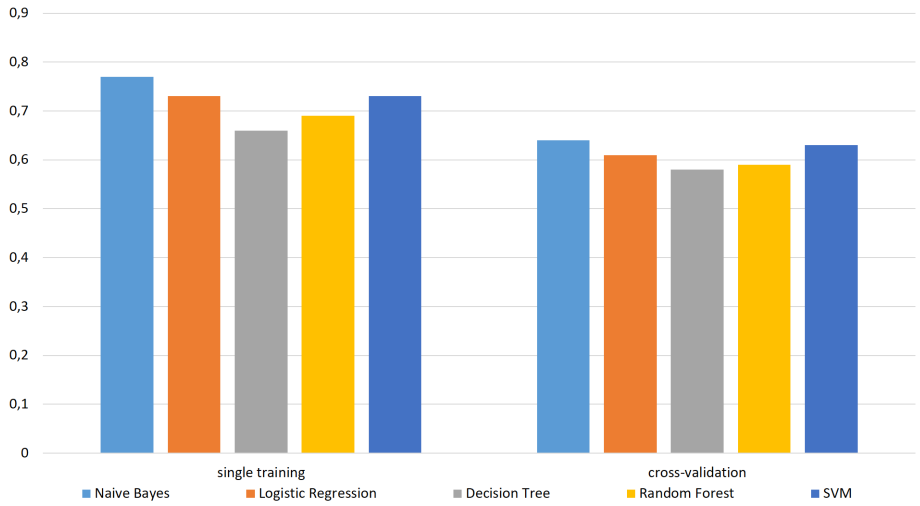

Abbildung 5.6: Vergleich der Accuracies des Single Training Verfahrens zur Cross Validierung für die Kategorie Costumer Support: *Für beide Verfahren hat die besten Werte für die Accuracy der Naive Bayes Classifier mit 77% und 64% erreicht und die schlechtesten Werte der Decision Tree mit 66% und 58%.*

Kapitel 6

Diskussion

6.1 Diskussion der Ergebnisse

Frage 1: Welche Kategorien können zur Klassifizierung von geschriebener Kommunikation in Softwareprojekten dienen?

Für die Beantwortung von Frage 1 wurde in Kapitel 4.5 eine Analyse der Daten durchgeführt, aus der sich die folgenden Kategorien ergeben haben: Communication mit den Unterkategorien Communication with Teammates und Non-work Topics, Dev-Ops mit der Unterkategorie Team Q & A, Customer Support. Basierend auf diesen Kategorien wurden in Kapitel 5 verschiedene Klassifikationsalgorithmen ausgewertet.

Frage 2: Wie gut können diese Kategorien mit Hilfe von Machine Learning Algorithmen automatisch klassifiziert werden?

In der folgenden Diskussion der Ergebnisse liegt der Fokus auf dem Single Training Verfahren. Wie bereits in Kapitel 5.3 beobachtet werden kann, erzielen in beiden Verfahren (Single Training und Cross Validation) ohnehin die gleichen Algorithmen die besten Accuracy Werte, welche für die Diskussion am relevantesten sind.
In den Ergebnissen hat sich herausgestellt, dass bei der Kategorie Team Q & A die Algorithmen schlechte Werte erzielen. Die höchste Accuracy erreichten der Naive Bayes Classifier und die Support Vector Machine mit nur 0.60. Daraus lässt sich schließen, dass es den Modellen schwer fällt, das Q & A Format aus den Daten richtig zuzuordnen. Zur

Klärung dieser Annahmen werden im Folgenden zwei Begründungen formuliert. Ein erster Grund für die schwierige Zuordnung könnte im Labeling der Daten aus dem originalen Datensatz Gittercom liegen. Parra et al. [24, S. 12] führen den Labelingprozess chronologisch durch, wodurch die Fragen und Antworten entsprechend aufeinander folgen. Im Gegensatz dazu werden beim Training die Nachrichten mit den dazugehörigen Labels in einer randomisierten Reihenfolge gelernt. Entsprechend könnte es dem Modell sehr schwer fallen, zufällige Antworten ohne den Kontext einer zuvor gestellten Frage richtig zuzuordnen. Ein zweiter Grund könnte sein, dass bereits beim Labelingprozess von Parra et al. [24, S. 14] Konflikte dahingehend auftraten, dass Nachrichten fälschlicherweise zur Kategorie Customer Support zugeordnet wurden, obwohl diese zur Kategorie Team Q & A gehörten. Daraus kann geschlossen werden, dass sich ähnliche Schwierigkeiten bei der Zuordnung der verschiedenen Kategorien für die Modelle ergeben könnten. Das genannte Problem wird im folgenden Beispiel veranschaulicht. Dabei handelt es sich um Textnachrichten aus dem zugrunde liegenden Datensatz.

Beispiel 1:
EisenbergEffect: @guybedford We have a bunch of windows users that consistently have the issue command \u201c unzip \u201d \u201d when doing a jspm install. Any ideas?
Guybedford: @EisenbergEffect yes this can be improved it is because of

In diesem Fall hat die erste Nachricht das Label Dev-Ops (Team Q & A), die zugehörige Antwort jedoch das Label Communication (Communication with Teammates) erhalten. Der Naive Bayes Classifier hat die Antwort aus dem Beispiel als Dev-Ops kategorisiert.
Analog zu der Kategorie Team Q & A erzielte dessen Oberklasse Dev-Ops auch schlechte Ergebnisse hinsichtlich der Accuracy. Der Naive Bayes, die Logistic Regression und die Support Vector Machine waren die besten Modelle mit einer Accuracy von 0.61. Auch hier lässt sich darauf schließen, dass die Modelle Schwierigkeiten haben könnten, die entsprechenden Nachrichten der Kategorie Dev-Ops zuzuordnen. Ein Grund dafür könnte sein, dass diese Kategorie durch seine Unterkategorie Team Q & A dominiert

wird. Wie in Kapitel 4.5 erwähnt wird, liegen bereits 2084 der 2776 Dev-Ops Elemente in der Unterkategorie Team Q & A. Entsprechend könnten die herausgestellten Probleme, die für die Kategorie Team Q & A vorliegen, ebenso hier ins Gewicht fallen. Des Weiteren wird die Kategorie Dev-Ops, wie in Kapitel 4.4 dargestellt, durch Nachrichten dominiert, die eine nähere Verbindung mit dem Schwerpunkt Entwicklung haben. Daher könnte zu erwarten sein, dass Textnachrichten, die entwicklungsrelevante Schlagwörter wie beispielsweise Programmnamen, Releases und Tools enthalten, häufig falsch der Kategorie Dev-Ops zugeordnet werden. Es folgt ein Beispiel, welches das beschriebene Problem verdeutlicht.

Beispiel 2:
Cobbweb: This guy if you haven \u2019t seen, mostly a ripoff of Marionette.Module but with a few personal tweaks (namely module args)

Hierbei handelt es sich um eine Nachricht aus der Kategorie Communication (Communication with teammates). Der Naive Bayes Classifier ordnete diese Nachricht fälschlicherweise der Kategorie Dev-Ops zu. Ein Grund dafür könnte die Erwähnung des Systems *Marionette.Module* sein.
Aus einer weiteren Beobachtung hat sich ergeben, dass die Kategorie Communication with Teammates ebenfalls eine schlechte Performance hinsichtlich der Accuracy aufweist. Dabei konnten de Naive Bayes Classifier und die Support Vector Machine noch den höchsten Wert mit 0.60 erreichen. Auch hier stellt sich wieder die Frage, ob die Modelle Schwierigkeiten bei der Zuordnung der Nachrichten für die Kategorie Communication with Teammates haben. Ein potentieller Konfliktpunkt dafür könnte die Unterscheidung zwischen einem Austausch von Informationen in einem allgemeinen Kontext zu beispielsweise dem Q & A Format aus der Kategorie Dev-Ops sein.
Im Gegensatz zu den vorherigen Kategorien erzielen die Modelle für Non-work Topics eine hohe Accuracy. Dabei erreichte der Naive Bayes Classifier sogar eine Accuracy von 0.73, was dafür spricht, dass es dem Modell gut gelingt, die Kategorie Non-work Topics richtig zu klassifizieren. Ein Grund dafür könnte sein, dass sich im Kontrast zu den oben besprochenen Kategorien Dev-Ops (Team Q & A) und

Communication with Teammates die Nachrichten aus dem Non-work Bereich gut von den arbeitsrelevanten Nachrichten abgrenzen lassen.
Für die Oberkategorie Communication bestehend aus den Unterkategorien Communication with Teammates und Non-work Topics erzielte der Naive Bayes Classifier eine Accuracy von 0.66. Wie in Kapitel 4.5 erwähnt, besteht die Kategorie Communication vornehmlich aus den beiden genannten Unterkategorien. Bis auf vier Datenpunkte lassen sich alle einer der beiden Unterkategorien zuordnen. Daraus ließe sich schließen, dass die beiden Unterkategorien die Performance der Kategorie Communication stark beeinflussen. Dementsprechend befindet sich die Accuracy zwischen den Ergebnissen der beiden Unterkategorien.
Zuletzt lässt sich beobachten, dass die Accuracy für die Kategorie Customer Support am höchsten ist. Hier erreichte der Naive Bayes Classifier eine Accuracy von 0.77. Wie bei der Kategorie Non-Work Topics lässt sich auch hier vermuten, dass das Modell die Kategorie gut klassifizieren kann. Mögliche Gründe könnten Satzstrukturen oder Schlagwörter (Allgemeine Ansprachen, Fragen nach Hilfe, Problembeschreibungen oder das Bedanken) sein, welche generell von Nutzern der jeweiligen Systeme verwendet werden, um Hilfestellungen zu erhalten. Exemplarisch lässt sich das mit Hilfe des folgenden Beispiels beobachten.
Abschließend lässt sich erkennen, dass der Naive Bayes Classifier die besten Ergebnisse hinsichtlich der Klassifikation der ausgewählten Kategorien liefert. Ein Grund dafür könnte sein, dass sich dieser, wie in Kapitel 2.4 zu Naive Bayes beschrieben, bereits mit einer geringeren Anzahl von Datenpunkten schnell trainieren lässt und trotzdem noch zuverlässig für die Validierung funktioniert [32]. Insbesondere lässt sich dies für die Kategorien Non-work Topics und Customer Support erkennen, da diese mit nur 2660 und 2860 die kleinsten Datensätze darstellen und der Naive Bayes Classifier für diese sehr gute Accuracies erzielt.

Beispiel 3.1:
mykola-tarchanyn: hey guys, I am new in cucumber, but I need to use this(cucumber-jvm + appium) in my project. [...] Can somebody explain this, or at least send me some helpful articles, I read a lot, but still stuck on this Thank you very much!
Beispiel 3.2:
joanaaraujo88: Hello all! I am having a problem \Result was missing for this step! Does anyone can helps me?

- Team Q & A und Communication with Teammates werden am schlechtesten klassifiziert. Potentielle Gründe dafür könnten sein:
  - Kontext im Q & A Format wichtig
  - Unterscheidung zwischen beiden Kategorien schwierig
- Non-work Topics und Customer Support werden am besten klassifiziert. Ein potentieller Grund dafür könnte sein:
  - Die beiden Kategorien beinhalten eindeutige Nachrichten mit relevanten Schlagwörtern
- Naive Bayes Classifier hat die Kategorien am besten klassifiziert. Ein potentieller Grund dafür könnte sein:
  - Naive Bayes Classifier trainiert schneller und funktioniert zuverlässiger, selbst wenn nur wenige Daten zur Verfügung stehen

InfoBox 6.1: Zusammenfassung: Kapitel 6.1

6.2 Grenzen der Arbeit

6.2.1 Grenzen der Daten

Durch die Verwendung des Datensatzes (Gittercom) von Parra et al. [25], welcher dementsprechend nicht für diese Arbeit selbst erstellt wurde, entstehen potentielle Konfliktpunkte. Ein erster Punkt ist der Labelingprozess, der zwar im Detail erklärt ist, jedoch Unklarheiten

offen lässt. Wie die Ergebnisse widerspiegeln, ist die Performance der Modelle für die Kategorien Team Q & A und Communication with Teammates schlechter als bei den restlichen Kategorien. Die Definitionen der Kategorien erschweren die Zuordnung von Nachrichten für den Labelingprozess in "Purpose Type", "Purpose Category" und "Purpose Subcategory". Entsprechend könnten die beschriebenen Unklarheiten zu einer Minderung der Performance der Modelle führen. Für das Training der binären Klassifikationsalgorithmen wurden die einzelnen Datensätze bestehend aus Communication, Communication with Teammates, Non-work Topics, Dev-Ops, Team Q & A und Customer Support präpariert, indem die Daten balanciert wurden. Dadurch soll sichergestellt werden, dass die Klassen ("1" = enthalten und "0" = nicht enthalten) aus gleich vielen Nachrichten bestehen. Die Balancierung der Datenpunkte wurde durch das randomisierte Entfernen von Nachrichten aus der jeweils größeren Klasse garantiert. Dieses Vorgehen führt dazu, dass den Modellen automatisch weniger Daten zur Verfügung stehen. Alternativ hätte es die Möglichkeit gegeben, eine Balancierung der Daten durch Datamining mit Hilfe des GitterComCrawlers[1] durchzuführen, wodurch mehr Datenpunkte für die kleinere Klasse zur Verfügung gestanden hätten. Dabei wäre es eine große Herausforderung gewesen, eine weiterhin konsistente Einordnung in "Purpose Type", "Purpose Category" und "Purpose Subcategory" zu garantieren.

Wie bereits in Kapitel 4.4 erwähnt, stammen die Daten aus Konversationen innerhalb von OSS Projekten. In Kapitel 2.1 wurde herausgestellt, dass es zwar Rollen gibt, die mehr Verantwortung haben (z.B. Projektleiter), es jedoch keine strenge Hierarchie gibt. Das bedeutet auch, dass es jeder Person möglich, ist sich mit verschiedenen Rollen zu identifizieren. Dementsprechend fällt es schwer, zwischen Nutzern des Systems und Entwicklern dessen zu unterscheiden. Dies setzt sich in der Kategorisierung von entwicklungsinternem Austausch (Communication with Teammates und Team Q & A) und Benutzerfragen (Customer Support) fort [24, S. 14]. Der zugrunde liegende Datensatz besteht aus englischen Nachrichten. Dadurch, dass sich OSS Projekte durch die Globalisierung der Entwicklerteams auszeichnen, kann es dahingehend zu Problemen kommen, dass es sich

[1] Teil des Gittercom Datensatzes [25]

bei Englisch nicht bei jedem Projektmitglied um die Muttersprache handelt. Exemplarisch kann in Beispiel 3.2 aus dem vorherigen Kapitel beobachtet werden, dass es dadurch zu Fehlern in Rechtschreibung und Grammatik kommen kann. Diese werden zwar durch das Preprocessing der Daten versucht zu bereinigen, jedoch ist dieser Prozess subjektiv und wurde von keiner weiteren Person geprüft. Das Modell ist in dem Sinne beschränkt, dass es sich in erster Linie nur auf OSS Projekte anwenden lässt, da die Daten nur den Austausch innerhalb von OSS Projekten betrachten. Folglich ist die Übertragbarkeit der Ergebnisse aus den Daten auf Closed Source Projekte fraglich.

6.2.2 Grenzen der Modelle

Bei den unterschiedlichen Klassifikationsmodellen aus Kapitel 2.4, welche zur Kategorisierung genutzt wurden, handelt es sich um binäre Classifier. Entsprechend bedeutet dies, dass die Modelle einzeln für jede Kategorie unabhängig trainiert wurden. Eine Alternative dazu wäre ein multi-class Klassifikationsalgorithmus, welcher alle Kategorien gleichzeitig betrachtet und klassifiziert [24]. Ein solches Verfahren nutzen Parra et al. [24] bei ihrer Analyse der Daten.
Wie in Kapitel 5.2 erläutert, wurden die Algorithmen Naive Bayes, Logistic Regression, Decision Tree, Random Forest und Support Vector Machine in den Fokus gestellt. Dabei wurden die Voting Classifier nicht weiter betrachtet, da sie einen hohen Ressourcenaufwand aufwiesen, jedoch im Vergleich zu Modellen wie Naive Bayes und Support Vector Machine nicht eindeutig bessere Ergebnisse hinsichtlich der Accuracy erzielten. Da das Konstruieren der Voting Classifier auf allen Algorithmen aufbaut (siehe Kapitel 2.4), beeinflussen auch die schlechten Modelle die Qualität beider Voting Classifier.
In dem Kapitel 4.5 wurde sich subjektiv für die Kategorien Communication, Communication with Teammates, Non-work Topics, Dev-Ops, Team Q & A und Customer Support entschieden. Das Hauptargument für die Auswahl dieser war dabei, dass pro Kategorie mindestens 1000 Datenpunkte vorhanden sind.

6.3 Ausblick

Die Analyse der vorgestellten Modelle zur automatischen Kategorisierung von geschriebener Kommunikation zur Unterstützng der Informationsbereitstellung in Softwareprojekten aus den Abschnitten zuvor hat ergeben, dass die Klassifizierung bei einigen Kategorien (Customer Support und Non-work Topics) bereits gut funktioniert. Andere Kategorien wie Communication (Communication with Teammates) und Dev-Ops (Team Q & A) bergen jedoch noch mehr Herausforderungen, an denen in der Zukunft weitergearbeitet werden kann. In Bezug darauf könnten neue Kategorien geschaffen beziehungsweise existierende weiter spezifiziert werden. Dies könnte den Labelingprozess erleichtern, was zu einer höheren Qualität des Datensatzes führen könnte, welche sich auch positiv auf die Performance der Klassifikationsmodelle auswirken könnte. Dabei könnte der GitterComCrawler eingesetzt werden, um neue Daten zu erhalten und diese manuell mit den neuen Kategorien zu labeln. Dies würde einen Vergleich der Modelle hinsichtlich der verschiedenen Kategorien ermöglichen.
Wie bei den Grenzen des Modells erwähnt, wurde sich auf Kategorien beschränkt, die mindestens 1000 Datenpunkte enthalten. Um für die übrigen Kategorien sinnvolle Ergebnisse produzieren zu können, könnten diese durch Datamining mit Hilfe des GitterComCrawler mit Daten erweitert werden.
Die größte Herausforderung, die bei der Analyse aufgetreten ist, scheint das Klassifizieren des Q & A Formats zu sein. Für zukünftige Forschung könnten Fragen und die dazugehörige Antworten gebündelt betrachtet werden. Ein Beispiel dafür wäre der E-Mail Verkehr zwischen zwei Personen, in dem die Referenzierung der vorherigen Nachricht chronologisch fortgeführt wird. Zur Vermeidung von Unklarheiten bei den Rollen verschiedener Personen (Entwickler, Nutzer) könnte diese Information der jeweiligen Nachricht angehängt werden. Auf Grundlage der beiden zuvor genannten Aspekte, die jeder Nachricht mehr Kontext hinzufügen würden, könnte eine ähnliche Analyse durchgeführt werden, welche mit den aktuellen Ergebnissen verglichen werden kann.
Insgesamt hat sich ergeben, dass der Naive Bayes und die Support Vector Machine die Kategorien am besten klassifizieren können.

Dementsprechend könnte für zukünftige Forschung der Fokus auf diese Algorithmen gelegt werden, um die Perfomance dieser noch weiter zu verbessern. Eine Möglichkeit dafür wäre es, einen neuen Voting Classifier mit nur diesen beiden Algorithmen zu implementieren und diese Ergebnisse für die Klassifizierung auszuwerten. Zusätzlich kann aus den beiden Algorithmen (Naive Bayes und Support Vector Machine) jeweils ein multi-class Classifier konstruiert werden. Dies lässt eine Überprüfung der Sinnhaftigkeit einer Bündelung der Kategorien zu. Dabei kann untersucht werden, ob diese neuen multi-class Classifier zu besseren Ergebnissen führen, anstatt die Kategorien als binäre Klassifikationsprobleme isoliert zu betrachten.
Die Klassifizierung für Non-work Topics sowie für Customer Support funktioniert sehr gut mit dem Naive Bayes Classifier. Dazu könnte in einer zukünftigen Arbeit ein Tagging Prototyp entwickelt werden, welcher die Nachrichten nach Non-work Topics und Customer Support einordnet. Die restlichen Nachrichten könnten dabei als entwicklungsinterne Kommunikation interpretiert werden.

Kapitel 7

Fazit

Ziel dieser Arbeit war es herauszufinden, welche Kategorien gut geeignet sind, um die Informationen aus geschriebener Kommunikation automatisch zu klassifizieren, um Softwareprojekte zu unterstützen. Im zweiten Schritt sollten dann binäre Klassifikationsalgorithmen aus dem Bereich des Maschinellen Lernens angewandt werden, mit dem Ziel herauszufinden, welcher Algorithmus am besten dafür geeignet ist. Als Datenbasis dafür diente der Gittercom Datensatz [26], welcher aus Nachrichten von OSS Projekten besteht. Daher wird in dieser Arbeit nur die Kommunikation aus OSS Projekten analysiert und klassifiziert. Der Gittercom Datensatz wurde dabei bereits in nutzenorientierte Kategorien für die Benutzer und Entwickler der Softwareprojekte aufgeteilt (siehe Lin et al. [21]). In dieser Arbeit wurden sechs dieser Kategorien beziehungsweise Unterkategorien ausgewählt, wobei im Folgenden für jede ein eigener Datensatz konstruiert wurde, welche jeweils für das Training und die Validierung der binären Klassifikationsalgorithmen genutzt wurden. Die Daten wurden für das Training mit Hilfe von verschiedenen NLP Verfahren vorbereitet und bereinigt. Anschließend wurden zwei Trainingsverfahren verwendet, das Single Training und die cross-validation, wobei letzteres als Bestätigung für die Ergebnisse des ersten Verfahrens diente. Für das Single Training Verfahren wurden für jede Kategorie die Klassifikationsalgorithmen hinsichtlich ihrer Accuracy und ihres F1-Scores ausgewertet. Ein direkter Vergleich mit der Accuracy der cross-validiation hat dabei ergeben, dass der Naive Bayes Classifier und die Support Vector Machine am besten für die Klassifizierung der ausgewählten Kategorien

geeignet sind. Die besten Ergebnisse erzielte der Naive Bayes Classifier für die Kategorien Customer Support und Non-work Topics mit einer Accuracy von 0.77 beziehungsweise 0.73. Für entwicklungsrelevante Kategorien wie Communication (Communication with Teammates) und Dev-Ops (Team Q & A) fiel es den Modellen im Vergleich schwieriger, diese richtig zu klassifizieren. Anhand der Ergebnisse dieser Arbeit lässt sich erkennen, dass noch immer Potential besteht für die in dieser Arbeit definierten, nutzenorientierten Kategorien bezüglich der Analyse weiterer Algorithmen und neuer Definitionen zusätzlicher Kategorien. Die Ergebnisse dieser Arbeit bieten eine gute Grundlage für die Entwicklung eines Tagging Prototypen. Wie oben ausgeführt, gelingt es den hier verwendeten Algorithmen bereits sehr gut Non-work Topics und Customer Support Nachrichten zu erkennen. Die restlichen Nachrichten können dabei als entwicklungsinterne Kommunikation zusammengefasst klassifiziert werden. Diese Aufteilung erlaubt es Entwicklern Non-work Topics auszublenden, um sich auf arbeitsrelevante Themen zu fokussieren. Im Bereich des Customer Support könnte es der Tagging Prototyp Entwicklern erlauben, speziell nach Nachrichten zu filtern und damit Nutzern schneller zu helfen.

Anhang A

A.1 Stop Words Liste

Liste der Stop Words aus *nltk.corpus.stopwords.words('english')* [1]
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

[1] https://www.nltk.org/search.html?q=stopwords&check_keywords=yes&area=default, Zugriff: 17.10.2022

Anhang B

B.1 Ausschnitt des Datensatzes für Communication

Channel	messageId	time	user	message	Purpose	Category	Subcategory	Sentiment
Cucumber	5601ba3af254567e29c43751	2015-09-22T20:29:46.585Z	demos74dx	Gherkin3	Team Wide	Dev-Ops	Team Q&A	0
Cucumber	5601ba54a0ecf0e07a8e3ac9	2015-09-22T20:30:12.199Z	mattwynne	hi @demos74dx. As far as I know there isn't yet. Are you interested in helping out with it?	Team Wide	Dev-Ops	Team Q&A	0
Cucumber	5601ba7d463feefb419dd22e	2015-09-22T20:30:53.564Z	demos74dx	@mattwynne yes, I'm actually digging in the code now to figure out where the matching actually happens and plan to do a PR once I	Team Wide	Dev-Ops	Development Operation Notifications	0
Cucumber	5601ba93c588a0de6edfcef0	2015-09-22T20:31:15.844Z	demos74dx	I think it needs a pretty big refactor to accomplish though	Team Wide	Communication	Communication with Teammates	1
Cucumber	5601baa1a0ecf0e07a8e3ae3	2015-09-22T20:31:29.760Z	mattwynne	you could do worse than use http://github.com/cucumber/cucumber-ruby-core for reference	Personal benefits	Discovery and aggregate news and	Reliably interesting/relevant blogs	0
Cucumber	5601baaf5c1379fe645904f6	2015-09-22T20:31:43.315Z	mattwynne	it was a major project to do that for the ruby one	Team Wide	Communication	Non-work Topics	1
Cucumber	5601bab3463feefb419dd23e	2015-09-22T20:31:47.956Z	mattwynne	took us about 18 months!	Team Wide	Communication	Non-work Topics	1
Cucumber	5601bb09c588a0de6edfcf0b	2015-09-22T20:33:13.453Z	demos74dx	I'll take a look thanks	Team Wide	Communication	Communication with Teammates	1
Cucumber	5601bb18c588a0de6edfcf0f	2015-09-22T20:33:28.531Z	demos74dx	So, I've been wondering though, or maybe I'm \cuking it wrong\	Team Wide	Communication	Communication with Teammates	1
Cucumber	5601bb4b40f4ecd92f713677	2015-09-22T20:34:19.250Z	demos74dx	wouldn't it be better to possibly create a PickleRunner for each Pickle pulled from the feature, then run each pickle as a child	Team Wide	Dev-Ops	Team Q&A	0

Abbildung B.1: Ausschnitt des Datensatzes für Communication. *Zehn Zeilen aus dem Datensatz*

B.2 Ausschnitt des Datensatzes für Communication with Teammates

Channel	messageId	time	user	message	Purpose	Category	Subcategory	Sentiment
MarionetteJS	5330d446e419207542000e85	2014-03-25T00:56:38.591Z	cobbweb	@jasonLaster Miminal Marionette.Module ripoff	Team Wide	Dev-Ops	Team Q&A	0
MarionetteJS	5330d4563c6d36c1640003f4	2014-03-25T00:56:54.853Z	jasonLaster	Where do I find it?	Team Wide	Dev-Ops	Team Q&A	0
MarionetteJS	5330d45f5d62367723000411	2014-03-25T00:57:03.006Z	cobbweb	https://github.com/cobbweb/ocky	Team Wide	Dev-Ops	Team Q&A	0
MarionetteJS	5330d46c473aa37923000ea8	2014-03-25T00:57:16.371Z	jasonLaster	makese sens :)	Team Wide	Communication	Communication with teammates	1
MarionetteJS	5330d478e419207542000e8a	2014-03-25T00:57:28.734Z	cobbweb	Main difference is the module definition parameters	Team Wide	Dev-Ops	Team Q&A	0
MarionetteJS	5330d48c5d62367723000412	2014-03-25T00:57:48.566Z	cobbweb	Marionette.Module will receive 6 parameters, in this order:\n\nThe module itself\nThe Application	Team Wide	Communication	Communication with teammates	1
MarionetteJS	5330d49d473aa37923000eb0	2014-03-25T00:58:05.399Z	samccone	@cobbweb per the release titles	Team Wide	Communication	Communication with teammates	1
MarionetteJS	5330d4a0473aa37923000eb2	2014-03-25T00:58:08.657Z	samccone	i duno	Team Wide	Communication	Communication with teammates	1
MarionetteJS	5330d4a5473aa37923000eb3	2014-03-25T00:58:13.832Z	cobbweb	Ocky will receive 1 parameter, in this order:\n\nThe module itself\nAny custom arguments	Team Wide	Communication	Communication with teammates	1
MarionetteJS	5330d4b7473aa37923000eb6	2014-03-25T00:58:31.318Z	cobbweb	@samccone I think it was better with the version numbers in there TBH	Team Wide	Communication	Communication with teammates	1

Abbildung B.2: Ausschnitt des Datensatzes für Communication with Teammates. *Zehn Zeilen aus dem Datensatz*

B.3 Ausschnitt des Datensatzes für Non-work Topics

Channel	messageId	time	user	message	Purpose	Category	Subcategory	Sentiment
Freezing Moon - Ancient Beast	56db50cf19834f3c35351a72	2016-03-05T21:34:07.056Z	DreadKnight	funny seeing different spellings of the same thing	Team Wide	Communication	Non-work topics	1
Freezing Moon - Ancient Beast	56db510aa54928411668a606	2016-03-05T21:35:06.104Z	tukkek	@DreadKnight dark priest	Team Wide	Communication	Non-work topics	1
Freezing Moon - Ancient Beast	56db5121a54928411668a60c	2016-03-05T21:35:29.732Z	DreadKnight	oh, that was my second guess	Team Wide	Communication	Non-work topics	1
Freezing Moon - Ancient Beast	56db5135b0cc3f1b41505cc5	2016-03-05T21:35:49.557Z	DreadKnight	good, I'm confident you'll manage; he's most used after all, so it's important xD	Team Wide	Team collaboration	Team management	0
Freezing Moon - Ancient Beast	56db5149817dfa1e41eca567	2016-03-05T21:36:09.058Z	DreadKnight	his changes aren't too big, but have a lot of impact on gameplay	Team Wide	Team collaboration	Team management	0
Freezing Moon - Ancient Beast	56db5154126367383571361b	2016-03-05T21:36:20.571Z	tukkek	it's almost done :)	Team Wide	Communication	Communication with teammates	0
Freezing Moon - Ancient Beast	56db5157a54928411668a622	2016-03-05T21:36:23.393Z	Zephyrum	@DreadKnight you're in the wrong neighborhood https://i.gyazo.com/75a78c223ed7aecdacc64f	Personal benefits	Fun	Sharing gif or memes	0
Freezing Moon - Ancient Beast	56db516dddfe3d431627f9a6	2016-03-05T21:36:45.365Z	Zephyrum	Oh, nice; the chat inlines image links	Team Wide	Communication	Non-work topics	1
Freezing Moon - Ancient Beast	56db517919834f3c35351a99	2016-03-05T21:36:57.181Z	Zephyrum	And doesn't crop it!	Team Wide	Communication	Non-work topics	1
Freezing Moon - Ancient Beast	56db517b68ddef776468ea6a	2016-03-05T21:36:59.799Z	DreadKnight	Yeah, nice upgrade compared to most IRC clients	Team Wide	Communication	Non-work topics	1

Abbildung B.3: Ausschnitt des Datensatzes für Non-work Topics. *Zehn Zeilen aus dem Datensatz*

B.4 Ausschnitt des Datensatzes für Dev-Ops

Channel	messageId	time	user	message	Purpose	Category	Subcategory	Sentiment
TheHolyWaffle	55102b9ec57069db7e0a6b76	2015-03-23T15:05:02.583Z	rogermb	Ah, wonderful. Then it's just not documented. Or I didn't find the complete list of installed programs	Team Wide	Communication	Communication with Teammates	0
TheHolyWaffle	55102c00630684d9583429cd	2015-03-23T15:06:40.526Z	rogermb	Wonderful, I even got to kick the hideous profiles out of the pom again	Team Wide	Communication	Communication with Teammates	0
TheHolyWaffle	55102c1d3bd19ad97eadf9b6	2015-03-23T15:07:09.999Z	TheHolyWaffle	actually a passphrase is optional	Team Wide	Communication	Communication with Teammates	0
TheHolyWaffle	55102c26bb9b35940a30b61d	2015-03-23T15:07:18.532Z	TheHolyWaffle	when creating gpg keys	Team Wide	Communication	Communication with Teammates	0
TheHolyWaffle	55102c2dbb9b35940a30b61f	2015-03-23T15:07:25.103Z	rogermb	Cool, even better then ^^	Team Wide	Communication	Communication with Teammates	0
TheHolyWaffle	55102c46630684d9583429d3	2015-03-23T15:07:50.052Z	rogermb	How should I call the environment variables for username and password?	Team Wide	Dev-Ops	Team Q&A	1
TheHolyWaffle	55102c4dc57069db7e0a6b9f	2015-03-23T15:07:57.522Z	TheHolyWaffle	OSSRH_USERNAME	Team Wide	Dev-Ops	Team Q&A	1
TheHolyWaffle	55102c78e336ad920a5f7efe	2015-03-23T15:08:40.546Z	rogermb	Is there no way to use some kind of OAuth token for that? It seems kind of sloppy to me that one would have to use a raw username	Team Wide	Dev-Ops	Team Q&A	1
TheHolyWaffle	55102c8ce336ad920a5f7f00	2015-03-23T15:09:00.286Z	rogermb	Especially for the system Sonatype uses, which even requires GPG signatures	Team Wide	Communication	Communication with Teammates	0
TheHolyWaffle	55102c9cbb9b35940a30b631	2015-03-23T15:09:16.970Z	TheHolyWaffle	I've looked but found nothing	Team Wide	Dev-Ops	Team Q&A	1

Abbildung B.4: Ausschnitt des Datensatzes für Dev-Ops. *Zehn Zeilen aus dem Datensatz*

B.5 Ausschnitt des Datensatzes für Team Q & A

Channel	messageId	time	user	message	Purpose	Category	Subcategory	Sentiment
Cucumber	55649744220dcade3c4d5503	2015-05-26T15:54:44.800Z	DEllster	@Danon9111 You should integrate Cucumber with another testing framework which can allow you use some APIs to write	Team Wide	Dev-Ops	Team Q&A	1
Cucumber	556498018f5532b4396ac6d1	2015-05-26T15:57:53.346Z	DEllster	@aslakhellesoy ... When reading \ The Cucumber Book\ i found the code below in Ruby . there is any way to convert it to Java ?	Team Wide	Dev-Ops	Team Q&A	1
Cucumber	5565704b8f5532b4396adb01	2015-05-27T07:20:43.093Z	aslakhellesoy	@Sidkiyassine just call the methods directly!	Team Wide	Dev-Ops	Team Q&A	1
Cucumber	5565d8a31d1bf91146cb914f	2015-05-27T14:45:55.343Z	priyankshah217	I had intergrated cucumber+appium https://github.com/priyankshah217/AppiumCucumberTest	Team Wide	Dev-Ops	Development Operation Notifications	0
Cucumber	5565d956dacb05b3394a1ade	2015-05-27T14:48:54.545Z	priyankshah217	@Sidkiyassine i think you dont need to write annotations manually, it will be taken care if you run feature using cucumber test runner	Team Wide	Dev-Ops	Team Q&A	1
Cucumber	5581c1c8bb2c3e7c158666cb	2015-06-17T18:51:52.481Z	codewitme	Hello guys, need a help.I want to call the run my featute file by calling the cucumber.cli.Main form a java class by	Team Wide	Communication	Communication with Teammates	0
Cucumber	558912436f7465873a365c38	2015-06-23T08:01:07.939Z	analghin	Hello guys,\nIs it possible to get test coverage using jacoco for android tests?\ndefaultConfig {\n	Team Wide	Dev-Ops	Development Operation Notifications	0
Cucumber	558bdddb8762506301760dd2	2015-06-25T10:54:19.926Z	Simon-Kaz	hey everyone, can someone help me out with running a feature file with specific tags included and excluded?	Team Wide	Dev-Ops	Team Q&A	1
Cucumber	558bdde9e6702c3a576477c9	2015-06-25T10:54:33.743Z	Simon-Kaz	say i want to run tags @Android but not @IOS	Team Wide	Communication	Communication with Teammates	0
Cucumber	558bde1dad7e376201fe92aa	2015-06-25T10:55:25.441Z	Simon-Kaz	mvn clean test -Dspring.profiles.active=ios -Dcucumber.options=\src/test/resources/com/mttnow/steps --tags @Settings,	Team Wide	Team Collaboration	File and Code Sharing	0

Abbildung B.5: Ausschnitt des Datensatzes für Team Q & A. *Zehn Zeilen aus dem Datensatz*

B.6 Ausschnitt des Datensatzes für Customer Support

Channel	messageId	time	user	message	Purpose	Category	Subcategory	Sentiment
UIKit	54b7d9397ae76eb525e99fe0	2015-01-15T15:14:01.532Z	malles	I'm afraid you are going to make that now :smile:	Team Wide	Customer support	How-to	1
UIKit	54b7d9637ae76eb525e99fe3	2015-01-15T15:14:43.874Z	malles	Maybe @aheinze has a quick example, but he seems to be in a 'grid-lock' right now	Team Wide	Customer support	How-to	1
UIKit	54b7d9e48fc5af4a068136d1	2015-01-15T15:16:52.070Z	Ilya-Zhulin	I understand you. But where I have to type a code? Atributes only defined here http://getuikit.com/docs/addons_htmleditor	Team Wide	Customer support	How-to	1
UIKit	54b7da8eb33b188c06dc18a9	2015-01-15T15:19:42.636Z	malles	get an instance of the editor:\n```\njQuery(function ($) {\n var editor = UIkit.htmleditor($('#myEditor'));\n	Team Wide	Team collaboration	File and code sharing	0
UIKit	54b7de9a8fc5af4a06813787	2015-01-15T15:36:58.383Z	aheinze	@Ilya-Zhulin try this:\n\n```javascript\n$(document).on('init.uk.component', '[data-uk-htmleditor]',	Team Wide	Team collaboration	File and code sharing	0
UIKit	54b7df3c8fc5af4a068137a7	2015-01-15T15:39:40.115Z	malles	yeah, or that :)	Team Wide	Customer support	How-to	1
UIKit	54b809aa73b182a117e9c70f	2015-01-15T18:40:42.842Z	Ilya-Zhulin	@aheinze , thank you, it works.	Team Wide	Customer support	How-to	1
UIKit	54b80a1f573df0f92345267c	2015-01-15T18:42:39.062Z	Ilya-Zhulin	But, why, when I click to list button in HTML mode, editor wirtes `<li>` tags only? What about `<ul>`?	Team Wide	Customer support	Troubleshooting	1
UIKit	54b8dc1b73b182a117e9e572	2015-01-16T09:38:35.926Z	robertotrem onti	@aheinze data-uk-grid-match is not fixed in 2.16.2 yet	Team Wide	Dev-ops	Development Operation Notifications	0
UIKit	54b8dc2a73b182a117e9e574	2015-01-16T09:38:50.572Z	robertotrem onti	release notes says it is, but it doesn't work	Team Wide	Communication	Communication with teammates	0

Abbildung B.6: Ausschnitt des Datensatzes für Customer Support. *Zehn Zeilen aus dem Datensatz*

Literaturverzeichnis

[1] R. Alkadhi, J. O. Johanssen, E. Guzman, and B. Bruegge. React: An approach for capturing rationale in chat messages. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 175–180, 2017.

[2] R. Alkadhi, T. Lata, E. Guzmany, and B. Bruegge. Rationale in development chat messages: An exploratory study. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 436–446, 2017.

[3] E. Alpaydin. *Maschinelles Lernen* -. Walter de Gruyter GmbH Co KG, Berlin, 2022.

[4] S. Beyer, C. Macho, M. Pinzger, and M. Di Penta. Automatically classifying posts into question categories on stack overflow. In *Proceedings of the 26th Conference on Program Comprehension*, ICPC '18, page 211–221, New York, NY, USA, 2018. Association for Computing Machinery.

[5] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[6] W. Chafe and D. Tannen. The relation between written and spoken language. *Annual Review of Anthropology*, 16:383–407, 1987.

[7] P. Chatterjee, K. Damevski, L. Pollock, V. Augustine, and N. A. Kraft. Exploratory study of slack qa chats as a mining source for software engineering tools. In *Proceedings of the 16th International Conference on Mining Software Repositories*, MSR '19, page 490–501. IEEE Press, 2019.

[8] S. A. Chowdhury and A. Hindle. Mining stackoverflow to filter out off-topic irc discussion. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 422–425, 2015.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[10] E. Guzman, R. Alkadhi, and N. Seyff. A needle in a haystack: What do twitter users say about software? In *24th IEEE International Requirements Engineering Conference, RE 2016, Beijing, China, September 12-16, 2016*, pages 96–105. IEEE Computer Society, 2016.

[11] E. Guzman, M. Ibrahim, and M. Glinz. A little bird told me: Mining tweets for requirements and software evolution. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 11–20, 2017.

[12] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[13] T. Haslwanter. *Python*, pages 5–42. Springer International Publishing, Cham, 2016.

[14] M. Henley and R. Kemp. Open source software: An introduction. *Computer Law Security Review*, 24(1):77–85, 2008.

[15] V. Käfer, D. Graziotin, I. Bogicevic, S. Wagner, and J. Ramadani. Communication in open-source projects-end of the e-mail era? In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, ICSE '18, page 242–243, New York, NY, USA, 2018. Association for Computing Machinery.

[16] U. Kamath, J. Liu, and J. Whitaker. *Deep Learning for NLP and Speech Recognition* -. Springer, Berlin, Heidelberg, 2019.

[17] J.-H. Kim. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics Data Analysis*, 53(11):3735–3745, 2009.

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

[19] S. W. Knox. *Machine Learning - a Concise Introduction*. John Wiley Sons, New York, 2018.

[20] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30:25–36, 11 2005.

[21] B. Lin, A. Zagalsky, M.-A. Storey, and A. Serebrenik. Why developers are slacking off: Understanding how software teams use slack. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, CSCW '16 Companion, page 333–336, New York, NY, USA, 2016. Association for Computing Machinery.

[22] W. McKinney and K. Lichtenberg. *Datenanalyse mit Python: Auswertung von Daten mit Pandas, NumPy und IPython*. Animals. O'Reilly, 2018.

[23] J. P. Mueller and L. Massaron. *Maschinelles Lernen mit Python und R für Dummies* -. Wiley-VCH, Weinheim, 2017.

[24] E. Parra, M. Alahmadi, A. Ellis, and S. Haiduc. A comparative study and analysis of developer communications on slack and gitter. *Empirical Softw. Engg.*, 27(2), mar 2022.

[25] E. Parra, A. Ellis, and S. Haiduc. Gittercom - dataset. `https://figshare.com/s/9b3df36e22a8a8f77169`.

[26] E. Parra, A. Ellis, and S. Haiduc. Gittercom: A dataset of open source developer communications in gitter. In *Proceedings of the 17th International Conference on Mining Software Repositories*, MSR '20, page 563–567, New York, NY, USA, 2020. Association for Computing Machinery.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] M. Pinzger and H. C. Gall. *Dynamic Analysis of Communication and Collaboration in OSS Projects*, pages 265–284. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[29] V. Potdar and E. Chang. Open source and closed source software development methodologies. *IET Conference Proceedings*, pages 105–109(4), January 2004.

[30] G. Redeker. On differences between spoken and written language. *Discourse Processes*, 7(1):43–55, 1984.

[31] V. S. Rekha, N. Divya, and P. S. Bagavathi. A hybrid auto-tagging system for stackoverflow forum questions. In *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, ICONIAAC '14, New York, NY, USA, 2014. Association for Computing Machinery.

[32] D. Sarkar. *Text Analytics with Python - A Practitioner's Guide to Natural Language Processing.* Apress, New York, 2019.

[33] L. Shi, X. Chen, Y. Yang, H. Jiang, Z. Jiang, N. Niu, and Q. Wang. A first look at developers' live chat on gitter. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2021, page 391–403, New York, NY, USA, 2021. Association for Computing Machinery.

[34] R. Spauwen and S. Jansen. Towards the roles and motives of open source software developers. *CEUR Workshop Proceedings*, 987:62–73, 01 2013.

[35] E. W. Steyerberg. *Missing Values*, pages 127–155. Springer International Publishing, Cham, 2019.

[36] M. R. Thissen, J. M. Page, M. C. Bharathi, and T. L. Austin. Communication tools for distributed software development teams.

In *Proceedings of the 2007 ACM SIGMIS CPR Conference on Computer Personnel Research: The Global Information Technology Workforce*, SIGMIS CPR '07, page 28–35, New York, NY, USA, 2007. Association for Computing Machinery.

[37] T.-T. Wong and P.-Y. Yeh. Reliable accuracy estimates from k -fold cross validation. *IEEE Transactions on Knowledge and Data Engineering*, PP:1–1, 04 2019.