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

Computer-gestützte Analyse des Kommunikationsverhaltens in Entwicklerteams unter Berücksichtigung digitaler Medien

Tool-supported analysis of communication behavior in development teams considering digital media

Masterarbeit

im Studiengang Informatik

von

Julian Horstmann

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

Hannover, 08.10.2019

Zusammenfassung

Da die Komplexität in der Softwareentwicklung stetig zunimmt, wird in dezentralen Softwareentwicklerteams gearbeitet. Eine erfolgreiche Kommunikation ist für den Projekterfolg unerlässlich, die Kommunikation kann dabei schriftlich oder mündlich erfolgen. Bisher wurde vor allem die Kommunikation in Meetings untersucht, die mündlich stattfindet. Das in dieser Arbeit vorgestellte Konzept analysiert die textuelle digitale Kommunikation außerhalb der Meetings. Dazu wird mittels „natural language processing“ (NLP) die textuelle Kommunikation analysiert, um über verschiedene Metriken unter Nutzung maschineller Lernverfahren die Stimmung in der Kommunikation in die drei Klassen **positiv**, **negativ** und **neutral** einzuordnen. Die Ergebnisse der Klassifizierung werden aufbereitet und in einer Trendanalyse visualisiert, sodass Rückschlüsse auf die Stimmung im Team möglich sind. Zur Validierung des Konzeptes wurde mit einem Unternehmen aus der Privatwirtschaft kooperiert, sodass das Konzept auf die digitale textuelle Kommunikation eines dezentral arbeitenden Softwareentwicklerteams angewendet werden konnte. Aus datenschutzrechtlichen Gründen war es nicht möglich, eine Crossvalidation der Trainingslabel durchzuführen, welche manuell von nur einer Person gelabelt wurden. Deshalb lag die Genauigkeit des manuellen Labelns lediglich bei 66%, jedoch war das maschinelle Lernverfahren mit 63% Genauigkeit in der Erkennung der richtigen Emotion nur 4% schlechter. Damit konnte gezeigt werden, dass es möglich ist, die Stimmung eines Softwareentwicklerteams durch eine Analyse der digitalen textuellen Kommunikation abzuleiten. Allerdings zeigt die Analyse auch, dass die Erkennung der Emotionen selbst für Menschen schwierig ist. Weitere auf die Analyse aufbauende Anwendungsfälle sind denkbar.

Abstract

Tool-supported analysis of communication behavior in development teams considering digital media

Since the complexity of software development is constantly increasing software projects are in most cases operated by decentralized software development teams. This team work requires an appropriate amount of communication, either written or verbal, i.e. face-to-face. Up to now, communication in meetings that takes place orally has frequently been subject to research. Therefore, the concept presented in this thesis analyzes textual and digital communication outside of meetings. For this purpose, „natural language processing“ (NLP) is used to analyze textual communication in order to classify the mood resonating in the written statements into the three classes **positive**, **negative** and **neutral** using various metrics and machine learning methods. The results of the classification are processed and visualized in a trend analysis so that conclusions can be drawn about the mood in the team. To validate the concept the digital textual communication of a decentralized software development team in industry was analyzed. For data protection reasons, it was not possible to cross-validate the training labels, which were manually labelled by only one person. Therefore, the accuracy of manual labeling was only 66%, but the machine learning process was only 4% worse with 63% accuracy in recognizing the right emotion. This showed that it is possible to derive the mood of a software development team by analyzing digital and textual communication. However, the analysis also shows that the recognition of emotions is difficult even for humans. Further use cases based on the analysis are possible.

Inhaltsverzeichnis

1. Einleitung **1**
- 1.1. Motivation . . . 2
- 1.2. Problemstellung . . . 2
- 1.3. Lösungsansatz . . . 3
- 1.4. Struktur der Arbeit . . . 4

2. Grundlagen **5**
- 2.1. Digitale Medien . . . 5
- 2.2. Kommunikation im Softwareentwicklungsteam . . . 8
- 2.3. Natural Language Processing . . . 8
- 2.4. Evolutionäre Algorithmen . . . 9

3. Verwandte Arbeiten **11**
- 3.1. Zusammenarbeit in Teams . . . 11
- 3.2. Textanalyse zur Erkennung von Emotionen . . . 12
- 3.3. Entwicklernetzwerke . . . 13
- 3.4. Kommunikations-Analyse . . . 14
- 3.5. Abgrenzung der Arbeit . . . 15

4. Konzept **17**
- 4.1. Problemstellung und Ziele . . . 17
- 4.2. Idee . . . 18
- 4.3. Metriken . . . 19
- 4.4. Klassifizierung . . . 20
- 4.5. Zusammenfassung . . . 21

5. Umsetzung **23**
- 5.1. Programmiersprache . . . 24
- 5.2. Datenquelle . . . 24
- 5.3. Vorverarbeitung der Kommunikationsdaten . . . 26
- 5.4. Metriken . . . 27
- 5.5. Klassifizierung . . . 34
- 5.6. Auswertung . . . 38

6. Evaluation **39**
- 6.1. Ziel der Studie . . . 39
- 6.2. Datenherkunft . . . 40
- 6.3. Datenaufbereitung . . . 45
- 6.4. Lernen des Modells . . . 53
- 6.5. Evaluierung des Models . . . 56

6.6. Ergebnisse für die Konversationen 58

7. Diskussion **63**
7.1. Threats to Validity 64
7.2. Weitere Anwendungsfälle 66

8. Zusammenfassung und Ausblick **67**
8.1. Zusammenfassung 67
8.2. Ausblick 68

Literaturverzeichnis **71**

A. Literaturverzeichnis **71**

Abbildungsverzeichnis

2.1. Funktionsweise eines **Evolutionäre Algorithmus** 9

4.1. Konzeptioneller Ablauf des Analyseverfahrens 21

5.1. Pipeline des Analyseverfahrens der digitalen textuellen Kommunikation eines Softwareentwicklerteams 23
5.2. Mind-Map der verwendbaren Metriken 27
5.3. Beispielhafte Darstellung der Klassifizierung mit einem Random Forest 35
5.4. Beispielhafte Darstellung der Klassifizierung mit einer SVM 36
5.5. Pipeline der Klassifizierung einer Datenquelle 37

6.1. Darstellung der Oberfläche des Windowstools von Zulip (Abgerufen von der https://zulipchat.com/apps/windows am 04.09.2019) 42
6.2. Die Geschlechterverteilung in dem untersuchten Softwareentwicklungsteam 45
6.3. Wordcloud der Kommunikation im Entwicklerteam 47
6.4. Verteilung der fünf Klassen in den Kommunikationsdaten 50
6.5. Verteilung der drei Klassen in den Kommunikationsdaten 51
6.6. Kovarianzmatrix der Merkmale ohne die *Bag of Words* Merkmale 54
6.7. Optimierung durch den EA 55
6.8. Darstellung des Gefühlsniveaus nach den manuell gelabelten Daten 59
6.9. Häufigkeit der Kommunikation in der Trend-Darstellung 60
6.10. Darstellung des Gefühlsnievaus nach der Vorhersage durch das Modell 61
6.11. Darstellung der Schwankungen in der Emotionalität im Softwareentwicklungsprojekt . . 62

Tabellenverzeichnis

2.1. Kommunikationskanäle nach MST und MRT 7

5.1. Ergebnisse der Berechnung der statistischen Metriken (Die Metriken sind auf das Intervall von -1 bis 1 skaliert) 29
5.2. Beispiele für die Klassifizierung von Waltinger [40] 30
5.3. Beispiele für die Polarität von Remus et al. [28] 31
5.4. Anwendung der Metriken an dem Beispielsatz 31
5.5. Beispielsatz als mono-gram und bi-gram. 32
5.6. Beispielsatz im trainierten CountVectorizer 33

6.1. Datenfelder der Antwort für das Abrufen der Nachrichten 44
6.2. Relevante Attribute einer Zulipnachricht 44
6.3. Beispiel für eine erfolgreiche Nachrichtentrennung 48
6.4. Beispiel für eine schlechte Nachrichtentrennung 48
6.5. Emotionen den Labelklassen zugeordnet 50
6.6. Konfusions-Matrix auf dem Validierungsdatensatz mit fünf Klassen 52
6.7. Konfusions-Matrix auf dem Validierungsdatensatz mit drei Klassen 53
6.8. Konfusions-Matrix des Modells auf dem Testdatensatz 56
6.9. Klassifizierungsreport für das Testset 57
6.10. Beispielsätze für die Vorhersagen des Klassifizierungsmodells 58

1

Einleitung

So wie die Herausforderungen und Prozesse von Unternehmen [6] stets komplexer werden, nimmt die Komplexität insbesondere der Softwareprojekte stetig zu [15]. Softwareentwicklung setzt dabei Teamarbeit voraus, um den komplexeren Anforderungen gerecht zu werden [18]. Dabei lebt die Teamarbeit von der Kommunikation zwischen den Teammitgliedern, weshalb die Ansprüche an die Kommunikation gesteigert werden [24]. Da bisher häufig die Kommunikation in Meetings analysiert [16, 27, 31] wurde, weil dort die entscheidende Kommunikation vermutet wurde, legt diese Arbeit den Fokus auf die digitale textuelle Kommunikation von Softwareentwicklerteams. Besonders in Zeiten verteilter Teams geschieht diese Kommunikation und damit die Informationsweitergabe meist über digitale Medien [26], was auch auf Teams zutrifft, die an einem Standort arbeiten. Es wird somit nicht nur in Meetings kommuniziert [26]. Aufgrund dieser Tatsache betrachtet das vorgestellte Analyseverfahren die digitale Kommunikation außerhalb von Meetings. Dazu wird digitale textuelle Kommunikation mit Hilfe maschineller Lernverfahren analysiert, um dem Anspruch einer computergestützten Analyse des Kommunikationsverhaltens in Entwicklerteams unter Berücksichtigung digitaler Medien gerecht zu werden. In diesem Zusammenhang wird die zentrale Forschungsfrage

Zentrale Forschungsfrage

Wie lässt sich das Kommunikationsverhalten in Entwicklerteams analysieren, um Rückschlüsse über die Stimmung abzuleiten?

zu klären sein. Exemplarisch werden dabei Kommunikationskanäle wie Skype oder Slack betrachtet, da digitale Kommunikationskanäle für Entwicklerteams zunehmend an Bedeutung gewinnen und Aufschluss über das Verhalten und Empfinden im Entwicklerteam geben können [26].

1.1. Motivation

Jedes größere Vorhaben der Welt beruht auf der Zusammenarbeit im Team. Je größer und aufwändiger das Vorhaben, desto mehr Menschen werden von diesem indirekt oder direkt beeinflusst oder haben daran mitgewirkt. In global verteilten Softwareentwicklerteams werden dabei nicht nur häufig Ländergrenzen sondern auch Ozeane und Zeitzonen überwunden [13]. Besonders durch die Verwendung von digitalen Tools in der Softwareentwicklung sind die Voraussetzungen für ein dezentrales Arbeiten geschaffen [9]. Der Schlüssel zum Erfolg eines Vorhabens oder Projekts ist die erfolgreiche Kommunikation im Team [31] und mit allen beteiligten/betroffenen Personen. Um Faktoren und Kennzeichen erfolgreicher Kommunikation oder problematische Entwicklungen oder Situationen festzustellen, ist eine Analyse des Kommunikationsverhaltens unausweichlich. In dezentralen Softwareentwicklungsprojekten kommunizieren die Teilnehmer meist über moderne digitale Medien, die die Entwicklung unterstützen, wie Skype, Slack, Jira, Mail und weitere. Diese digitalen Medien haben den Vorteil, dass sämtliche Interaktionen leicht zu fixieren sind. Somit bietet sich die Möglichkeit an, das Kommunikationsverhalten computergestützt auszuwerten. Der Einsatz von digitalen Medien ist allerdings nicht einzig und allein der Softwareentwicklung vorbehalten. Auch außerhalb der Softwareentwicklung werden in dezentral agierenden Teams ähnliche Tools und Kommunikationskanäle für Teamarbeit verwendet. Die Untersuchung der Kommunikation kann somit auf viele verschiedene Teams aus unterschiedlichen Fachdisziplinen angewendet werden. Speziell Softwareentwicklerteams begegnen häufig komplexen und schnelllebigen Anforderungen, weshalb eine reibungsfreie Kommunikation unverzichtbar ist. Zusätzlich ist das Mitwirken eines Entwicklers in mehreren Softwareprojekten eine besondere Herausforderung bei Softwareentwicklungsprojekten. So wird nicht nur fachliches Wissen in der Softwareentwicklung benötigt, sondern auch das jeweilige domänenspezifische Wissen. Für ein erfolgreiches Softwareentwicklungsprojekt ist es somit unverzichtbar, eine ausgezeichnete Kommunikation zwischen den Entwicklern, von den Entwicklern zu den fachlichen Experten und dem Kunden zu gewährleisten [27]. Durch eine Analyse des Kommunikationsverhaltens ist es möglich, Fehler, Lücken und Missstände in der Kommunikation aufzudecken und Lösungsansätze zu finden, um somit den wirtschaftlichen Erfolg zu sichern und zu maximieren. Denn wie bereits festgehalten, ist eine gute Kommunikation entscheidend für eine gute Teamarbeit.

1.2. Problemstellung

Die Probleme in der Kommunikation oder in dem Kommunikationsverhalten aufzudecken, ist kein triviales Problem. Es erfordert sowohl eine analytische als auch eine psychologische Herangehensweise. Hierzu existieren viele Theorien über die Kommunikation und die Probleme bei eben jener. Viele dieser Theorien arbeiten mit einem Sender- und Empfängermodell, wobei die meisten Probleme oft bei der Übermittlung der eigentlichen Botschaft zum Empfänger auftreten. Diese Probleme

können aus verschiedenen kulturellen Kontexten, Wissensständen, Empathie, Apathie und weiteren Faktoren herrühren. Soziale Komponenten, wie beispielsweise mangelnde Sympathie zwischen zwei Team-Mitgliedern, wirken sich zusätzlich erschwerend aus. Da diese Arbeit sich mit der computergestützten Analyse der Kommunikation beschäftigt, werden die psychologischen Theorien zugunsten einer technischeren Analyse der Kommunikationskanäle auf Grundlage verschiedener Ansätze nur angerissen. Gleichwohl kommt es bei der Entwicklung im Team selbstverständlich auch auf die soziale Komponente an, denn Kommunikation ist sozial. Ziel dieser Arbeit wird es aber sein, Rückschlüsse auf Basis des vorgestellten Analyseverfahrens auf das Kommunikationsverhalten zu ziehen. Aspekte der Analyse könnten die Nachrichtenlänge, der Inhalt und die Aussage der Nachrichten, die Emotionalität, der Sprachstil und die Häufigkeit der Kommunikation sein.

1.3. Lösungsansatz

Kommunikation ist anspruchsvoll und fehlerhafte Kommunikation kann den Projekterfolg gefährden [12]. Um der fehlerhaften Kommunikation entgegenzuwirken, wurde häufig die Kommunikation in Meetings analysiert [16, 27, 31]. Da die Kommunikation in vielen Fällen auch digital außerhalb der Meetings stattfindet [26], ist eine Untersuchung eben dieser entscheidend. Die Methoden zur Meetinganalyse lassen sich nicht für die Auswertung der digitalen Kommunikation verwenden, weshalb neue Verfahren für diese Kommunikationskanäle erforderlich sind. Aus diesem Grund wird ein Verfahren zur computergestützten Analyse des digitalen Kommunikationsverhaltens in Entwicklerteams in dieser Arbeit entwickelt. Dafür ist es hilfreich, die textuelle Kommunikation mittels „natural language processing“ (NLP) zu analysieren. Doch bevor eine Textanalyse auf diesem Niveau durchgeführt werden kann, ist es erforderlich, sich ausführlich mit der Datenlage zu beschäftigen. So kann festgestellt werden, welche Merkmale der textbasierten Kommunikation relevant sind, um daraus Rückschlüsse auf die Kommunikation abzuleiten. Für diesen Zweck wurden neben diversen statistischen auch inhaltliche Merkmale aus den Texten extrahiert. Durch die Vielzahl von Metriken, die abgeleitet werden können, entsteht ein Optimierungsproblem: Die Suche nach den wichtigsten Merkmalen, um ein Klassifizierungsverfahren optimal zu trainieren. Über einen evolutionären Algorithmus (EA) wird dieses Problem gelöst. Mit den geeigneten Merkmalen wird in verschiedenen Klassifizierungsverfahren ein Model gelernt, welches in der Lage ist, Nachrichten in drei Klassen zu klassifizieren (**positiv**, **negativ** und **neutral**). Um Wertungen einzelner Nachrichten nicht auf ein Teammitglied zurückverfolgen zu können, werden die Ergebnisse der Klassifizierung abschließend über Zeitintervalle und Personengruppen kumuliert. Die kumulierten Klassifizierungsergebnisse sollen so die Möglichkeit zulassen, Aussagen über das Kommunikationsverhalten im Entwicklerteam zu treffen.

1.4. Struktur der Arbeit

Die Arbeit gliedert sich neben der Einleitung in sieben Kapitel. Es empfiehlt sich, die Arbeit im Ganzen oder zumindest kapitelweise durchzulesen. Zu Beginn der Arbeit wird auf Grundlagen und Verwandte Arbeiten eingegangen, um später die Ergebnisse dieser Arbeit abzugrenzen. Im weiteren Verlauf wird dann auf das Konzept und in dem Kapitel Umsetzung auf dessen Implementierung eingegangen. Im Anschluss werden in einer Studie die Anwendbarkeit des Konzeptes in der Evaluation überprüft, sowie dessen Ergebnisse in einer Diskussion eingeordnet. Schlussendlich werden die Ergebnisse dieser Arbeit in einem Resümee zusammen mit einem Ausblick festgehalten.

2

Grundlagen

Diese Arbeit beschäftigt sich mit der Analyse digitaler Medien in Softwareentwicklerteams. Um eine Analyse der Kommunikationsdaten durchzuführen, werden jedoch noch einige Grundlagen benötigt, die im Folgenden abgehandelt werden. So wird für die Visualisierung der Kommunikationsnetzwerke die FLOW-Methode verwendet. Um die Kommunikation an sich besser zu verstehen, werden Grundlagen der Kommunikationstheorie und sozialer Netzwerke angeführt. Darüber hinaus wird „natural language processing“ (NLP), mit dessen Hilfe eine Analyse des Inhalts und Stimmung durchgeführt wird, näher beschrieben. Bevor die Analyse und Visualisierung der Daten durchgeführt werden kann, müssen zunächst die von Entwicklern verwendeten digitalen Medien bestimmt werden.

2.1. Digitale Medien

In der Softwareentwicklung werden hohe Ansprüche an die Kommunikation und die Zusammenarbeit gestellt [25]. Aus diesem Grund ist die Wahl des geeigneten Kommunikationkanals entscheidend. In den meisten Entwicklerteams stellen räumliche Trennungen, kulturelle Unterschiede und sprachliche Schwierigkeiten zusätzliche Hürden bei der Kommunikation dar. Daher ist es noch entscheidender, das richtige Medium für die Kommunikation zu wählen [25]. In einer empirischen Studie haben Niinimäki et al. [25] mit dem Ziel die besten Kommunikationswege zu finden, festzustellen versucht, welche Medien in Entwicklerteams verwendet werden. In dem Paper von Daft und Lengel [3] wird generell die Face-to-Face Kommunikation als informationsreichster Kommunikationsweg beschrieben. Da die Face-to-Face Kommunikation ohne ein digitales Medium auskommt, ist sie nicht relevant für diese Arbeit. Als nächstbester Kommunikationskanal wird die Kommunikation über das Telefon angesehen. Da das Paper von Daft und Lengel von 1983 ist, betrachtet es strenggenommen keine digitale Kommunikation. Allerdings kommen Niinimäki et al. [25] und Daft und Langel [3] zum selben grundlegenden Ergebnis: Je schneller und direkter die Kommunikation erfolgen kann, desto mehr Nutzen bietet sie.

Im Vergleich zum Jahr 1983 haben sich die Kommunikationsmethoden stark weiterentwickelt, sodass auch Videokonferenzen, Instantmessages, Email und andere Kommunikationswege für eine dezentrale Zusammenarbeit in Betracht gezogen werden können. Im Widerspruch zur *Media-Richness-Theorie* (MRT) von Daft und Langel [3] stehen die Erkenntnisse der Studie von Niinimäki et al. [25], denn demnach müssten Videokonferenzen nach der Face-to-Face Kommunikation der beliebteste Kommunikationkanal sein, weil eine Videokonferenz der Face-to-Face Kommunikation am nächsten kommt und damit den nächstbesten Informationsgehalt bietet. Nach der MRT sollten „reiche“ Kommunikationsmedien für komplexe oder unsichere Aufgaben verwendet werden. In der Praxis scheitert eine Videokonferenz allerdings häufig an den fehlenden Raumkapazitäten mit entsprechender Ausstattung. Ein weiteres Hindernis für eine Videokonferenz stellt die Voraussetzung dar, dass die Gegenstelle kompatibel zur Ausgangstelle ausgestattet sein muss. Für jeden Teilnehmer muss somit entsprechende Hardware und Raumkapazität bereitstehen. Wie Niinimäki et al. [25] auch festgestellt haben, ist die Wahl des Kommunikationswegs abhängig von den sprachlichen Fähigkeiten der Teammitglieder. So wird häufig über textbasierte Kommunikationkanäle kommuniziert, wenn das sprachliche Niveau sehr unterschiedlich ist. Ebenfalls wurde festgestellt, dass sich die Wahl des Mediums von nicht technischen Teammitgliedern zu technischeren Mitgliedern in der Form unterscheidet, dass technisch erfahrenere Teammitglieder eher über sprachbasierte Medien kommunizieren.

Bei der Wahl des Kommunikationsmediums ist zusätzlich noch die *Media synchronicity Theorie* (MST) von Dennis und Valacich [5] zu beachten. Nach dieser Theorie gibt es fünf Unterscheidungen für ein Kommunikationsmedium: Unmittelbarkeit des Feedbacks (immediacy of feedback), Parallelität (parallelism), Symbolvielfalt (symbol variety), Überprüfbarkeit (rehearsability) und Wiederaufbereitbarkeit (reprocessability). Mit Kommunikationsmedien in global verteilten Entwicklerteams beschäftigt sich Niinimäki et al. [25] im Bezug auf MST. Auch Noll [26] hat sich in seiner Bachelorarbeit mit den Kommunikationsmedien in Entwicklerteams beschäftigt. Er legte dabei den Fokus auf Instant-Messaging-Dienste und die Teammeetinganalyse. Aus den in verschiedenen Veröffentlichungen erwähnten Kommunikationsmedien wurde die Tabelle 2.1 zusammengestellt.

Tabelle 2.1.: Kommunikationskanäle nach MST und MRT

Kommunikationskanal	Methode	MST	MRT
Telefon-Gespräch	Sprachbasiert	Unmittelbarkeit des Feedbacks;	Hoch
Telefon-Konferenz	Sprachbasiert	Unmittelbarkeit des Feedbacks;	Hoch
Video-Gespräch	Sprachbasiert und Videobasiert	Unmittelbarkeit des Feedbacks; Parallelität;	Sehr hoch
Video-Konferenz	Sprachbasiert und Videobasiert	Unmittelbarkeit des Feedbacks; Parallelität;	Sehr hoch
Email	Textbasiert	Parallelität; Symbolvielfalt; Überprüfbarkeit; Wiederaufbereitbarkeit	Sehr gering
Whatsapp	Textbasiert	Unmittelbarkeit des Feedbacks; Parallelität; Symbolvielfalt; Überprüfbarkeit; Wiederaufbereitbarkeit	Normal
Slack	Textbasiert	Unmittelbarkeit des Feedbacks; Parallelität; Symbolvielfalt; Überprüfbarkeit; Wiederaufbereitbarkeit	Normal
Skype-Textnachrichten	Textbasiert	Parallelität; Symbolvielfalt; Überprüfbarkeit; Wiederaufbereitbarkeit	Normal
JIRA-Kommentare	Textbasiert	Parallelität; Symbolvielfalt; Überprüfbarkeit; Wiederaufbereitbarkeit	Gering

2.2. Kommunikation im Softwareentwicklungsteam

Dass Kommunikation eine wichtige Rolle spielt und das nicht nur in der Softwareentwicklung, zeigt eine kurze Abfrage bei Google Scholar. Für den Suchbegriff „communication“ wurden circa 6.540.000 Ergebnisse gefunden. Für die Erweiterung der Suche um den Suchtherm „problems“ gab es immer noch 4.640.000 Ergebnisse. Somit beschäftigen sich noch mehr als 70% der Ergebnisse mit Kommunikation und Problemen. Wird der Suchtherm noch konkreter auf „communication problems in teams“ eingegrenzt, so bleiben noch 4.090.000 Ergebnisse übrig. Folglich scheint es einen Zusammenhang zwischen Kommunikation, Problemen und Teams zu geben. Schränkt man die Abfrage nun sehr spezifisch auf Softwareentwicklerteams ein („communication problems in software development teams“), so bleiben noch 1.500.000 Ergebnisse übrig, was nahezu 25% aller Ergebnisse zu dem Schlagwort „communication“ entspricht.[1]

Diese kleine Abfrage unterstützt die These, dass besonders im Zusammenhang mit Softwareentwicklungsprojekten die Kommunikation sehr entscheidend und viel diskutiert ist. Mangelnde, schlechte oder fehlerhafte Kommunikation kann zu Fehlannahmen, missverstandenen Anforderungen, Mehraufwand in der Entwicklung oder dem Scheitern des gesamten Projektes führen [17]. Bei der Kommunikation im Team kommt es nicht nur auf ausgetauschte Informationen wie Anforderungen, Fehler, Architekturentscheidungen und weitere an, sondern auch auf die Wahl des richtigen Kommunikationsweges [17].

Soziale Konflikte innerhalb des Teams können den Erfolg der Kommunikation ebenfalls beeinflussen. Zu dieser Erkenntnis kam Sawyer [30]: er konnte einen empirischen Zusammenhang zwischen teaminternen Konflikten und dem Projekterfolg herstellen. Hierzu konnte er auf Kommunikationsdaten von 40 Softwareentwicklerteams zurückgreifen.

2.3. Natural Language Processing

Natural Language Processing (NLP) ist ein Verfahren, Texte computergestützt auszuwerten. NLP eignet sich für mehrere Aufgaben, darunter Information Retrieval (IR), Informationsextraktion, Fragenbeantwortungsaufgaben und Textverständnis (Artificial Intelligence) [22]. Bei der Analyse mit NLP gibt es verschiedene Stufen (*Levels*) der Verarbeitung [21]. Das Verarbeiten von Sprache ist ein sehr dynamisches Unterfangen. Bei dem Versuch einen Text zu lesen, wird nicht nur auf Informationen zurückgegriffen, welche aus dem Text hervorgehen, sondern auch auf Kontextinformationen. Liddy [21] erläutert dies, indem die Kontextinformation genutzt wird, dass das Buch welches gelesen wird, biologische Themen behandelt, um Doppeldeutigkeiten aufzulösen. Folgende Stufen der Verarbeitung existieren dabei [20]:

[1] Abfragen am 19.05.2019 durchgeführt.

- Phonologie: Interpretation von Sprache und Wörtern
- Morphologie: Analyse der Wortteile Präfix, Suffix und Hauptteil
- Lexikal: Analyse der Wörter mit deren Bedeutung und **Part-of-Speech**-Analyse
- Syntaktik: Analyse der Wörter, um die Grammatik des Satzes zu erkennen
- Semantik: Die Aussagen und Sinn eines Satzes ermitteln mit der Auflösung von Mehrdeutigkeiten
- Diskussion: Die Bedeutung und Struktur von längeren Texten erkennen
- Pragmatik: Das geschickte Verwenden von Sprache, mit dem Ziel einen bestimmten Zweck zu verfolgen. Dazu bedarf es Weltwissen.

Mit steigender Verarbeitungsstufe steigt auch der Aufwand und der Nutzen der Analyse. Die Komplexität der Verarbeitungsstufe ist an die jeweilige Aufgabe geknüpft. Eine einfache Informationsextraktion kurzer Texte mittels NLP ist bereits mit wenigen Verarbeitungsschritten möglich. Zunächst muss der zu analysierende Text in einzelne Wörter aufgeteilt werden. In diesem Schritt wird jedes Wort hinsichtlich der **Part-of-Speech** (POS) analysiert. POS meint dabei das Erkennen von Wörtern mit ähnlichen grammatikalischen Eigenschaften. Nach dieser Vorarbeit müssen alle Wörter in ihre Grundform gebracht werden. Dieser Vorgang wird Lemmatisierung genannt. Nach der Lemmatisierung müssen Stoppwörter, wie „und“, „als“, „der“ und weitere entfernt werden. Die übriggeblieben Wörter könnten mit einer Wissensdatenbank abgeglichen werden, um so eine Aussage über die Bedeutung zu treffen. Im späteren Verlauf dieser Arbeit wird NLP verwendet, um die Syntax der eingelesenen Nachrichten zu analysieren und so weitere Analyseschritte darauf aufzubauen.

2.4. Evolutionäre Algorithmen

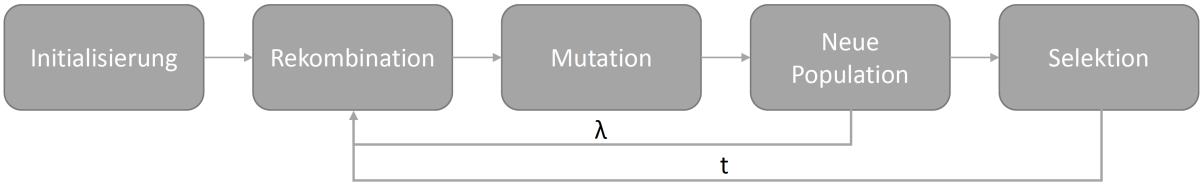

Abbildung 2.1.: Funktionsweise eines **Evolutionäre Algorithmus**

Für diese Arbeit gibt es eine Vielzahl von Möglichkeiten, die Modelle zur Bestimmung der Emotionalität zu bestimmen. Durch die Vielzahl von Hyperparametern, die Parameter die das Optimierungsverfahren für die Optimierung verändern kann, stellt das Finden der optimalen Lösung ein Optimierungsproblem dar. Dieses zu lösen obliegt dem **Evolutionäre Algorithmus** (EA). Angesichts der großen Zahl von möglichen Lösungsräumen und komplexen Optimierungsproblemen [29] wird es

immer wichtiger, das Optimum zu finden. Ein mögliches Verfahren, den Lösungsraum zu explorieren, sind **Evolutionäre Algorithmen** [23]. Dieses Optimierungsverfahren ist angelegt an die Evolutionstheorie von Charles Robert Darwin [4], nach der nur die am besten angepasste Art überlebt und sich eine Art durch Mutationen an den Lebensraum anpassen kann [4]. Die Güte einer Lösung für ein Optimierungsproblem wird dabei über eine Fitnessfunktion bestimmt. Das Verfahren des EA arbeitet in Generationen. So werden in jeder Generationen durch Mutation der Eltern neue Kinder erzeugt. Nach der Berechnung der Fitness jeder Lösung werden die n-besten Lösungen (die Lösungen mit der besten Fitness) ausgewählt und als Eltern in der nächsten Generation verwendet. Für die Selektion und die Mutation kann es verschiedene Verfahren der Implementierung geben. So kann für die Selektion nur „survival of the fittest“ gelten. Es sind aber auch Situationen möglich, in denen es sinnvoll sein kann, ebenfalls schlechtere Lösungen zuzulassen, damit der Lösungsraum nicht zu zielgerichtet durchquert wird und damit der Algorithmus nicht in einem lokalen Optimum feststeckt. Außerdem gibt es verschiedene Möglichkeiten, die Mutationen anzupassen. So kann es beispielsweise zu guten Ergebnissen führen, wenn die Mutationsrate mit zunehmender Anzahl von Generationen abnimmt, um nicht kleine Optima im Lösungsraum zu überspringen. Unabhängig von der Implementierung folgt jeder EA dem in Abbildung 2.1 gezeigten Ablauf: Initialisierung, Rekombination, Mutation, neue Populationen und Selektion [8]. Einfache Implementierungen eines EA wären beispielsweise ein 1-1 EA. Bei diesem wird der Schritt der Rekombination ausgelassen, da in jeder Generation nur eine Lösung erzeugt und diese bei der Selektion mit der vorherigen verglichen wird. **Evolutionäre Algorithmen** können durch die Möglichkeit, den Lösungsraum zu explorieren, auch zu Problemen mit unbekannten Lösungsräumen optimale Lösungen finden. Allerdings arbeiten EA mit einer Heuristik bei der Mutation und explorieren den Lösungsraum zufällig, was dazu führen kann, dass das Optimum nicht oder erst nach vielen Generation gefunden wird.

3

Verwandte Arbeiten

Einzelne Ansätze, die in diesem Kapitel vorgestellt werden, wurden in anderen Publikation behandelt: Zusammenarbeit in Teams, Textanalyse zur Erkennung von Emotionen, Entwicklernetzwerke und die Kommunikations-Analyse. Am Ende dieses Kapitels und jedes Abschnittes findet noch eine Abgrenzung und Einordnung der verwandten Arbeiten zu dem Ansatz dieser Arbeit statt.

3.1. Zusammenarbeit in Teams

Calefato et al. [2] führten eine empirische Studie durch, in der die textbasierte Kommunikation in Requirements-Workshops untersucht wurde. Dabei sind sie auf die textbasierte Kommunikation mit Face-to-Face Kommunikation in verschiedenen Requiremts-Aufgaben (Elicitation und Negotiation) bei verteilten Projekten eingegangen. Zusätzlich wurde noch der Effekt von Computer-gestützter Kommunikation (CMC) im Bezug auf die Zufriedenheit analysiert. Dabei zeigten erste Ergebnisse, dass die Zufriedenheit bei der Verwendung von CMC für die Elicitation höher ist als bei der Negotiation und dass generell die Face-to-Face Kommunikation der CMC vorgezogen wird.

Ebert und Neve [7] beschäftigten sich in einer Fallstudie mit den Vor- und Nachteilen von verteilter Zusammenarbeit in globalen Teams. Dabei führten sie als Vorteile das Arbeiten in mehreren Zeitzonen und die Kostenersparnis durch die Auslagerungen von Teilaufgaben in „günstigere“ Länder an. Als nachteilig wurde der höhere Verwaltungsaufwand und die eventuell auftretenden soziale Konflikte benannt.

Herbsleb und Mockus [13] suchten in einer empirischen Studie nach Gründen für Verzögerungen in global verteilten Softwareentwicklerteams. Dabei wurde mit Daten aus Umfragen und Programmcode-Versionierungen gearbeitet. Die Studie kam zu dem Ergebnis, dass die Entwicklung vergleichbarer Programmstücke in verteilten Teams im Vergleich eineinhalb bis zweimal so lange brauchte, wie nicht dezentrale Softwareentwicklerteams. Als Ursache führten Herbsleb und Mockus [13] an, dass in verteilten Teams mehr Personen beteiligt sind, als in nicht verteilten Teams.

Diese Arbeit beschäftigt sich mit der computergestützten und textbasierten Analyse digitaler Kommunikation in Entwicklerteams, da die Zusammenarbeit in Teams eine entscheidende Rolle spielt. In den vorgestellten Forschungsarbeiten zur Zusammenarbeit werden die Vor- und Nachteile der Arbeit in globalen Teams und die Herausforderungen, die diese Arbeitsweise mitbringt, behandelt. Diese Arbeit bedient sich lediglich des Wissens um die Schwierigkeiten dieser Arbeitsweise und der genutzten Kommunikationswege. Dennoch zeigen die genannten Quellen die zusätzlichen Herausforderungen, die dezentrale Teams mit sich bringen und sie unterstreichen damit die Relevanz des in der Arbeit vorgestellten Ansatzes.

3.2. Textanalyse zur Erkennung von Emotionen

Felbo et al. [10] führten eine Textanalyse auf 1,2 Milliarden Twitter „tweets“ mittels NLP durch. Ziel der Analyse war es, die Emoticons in den Nachrichten anhand des Inhalts und der Ausdrucksweise vorherzusagen. Dazu wurden zunächst die Emoticons aus den „tweets“ entfernt und als Label verwendet. Mittels des DeepMoji Models wurde anschließend versucht, die Emotion zu ermitteln. DeepMoji ist das trainierte Modell von Felbo et al. [10], welches über die „tweets“ trainiert wurde. Dabei wurden gute Erkennungsraten festgestellt. So liegt die Übereinstimmung bei 82,4%. Die Autoren veröffentlichen nicht nur sämtlichen Quellcode und Quelldaten, sondern stellten auch eine Onlinedemo zur Verfügung, bei der die eigenen Kurznachrichten überprüft werden können. Leider ist das Modell auf die englische Sprache trainiert, sodass bei der Verwendung von deutschen „tweets“ zu einer sehr geringen Genauigkeit kommt. Da Sprachen sehr spezifische Eigenarten besitzen, ist dieses Modell nicht für diese Arbeit zu verwenden, weil Nachrichten in deutscher Sprache ausgewertet werden sollen. Für englische Texte kommen die Autoren allerdings zu guten Ergebnissen.

Alghalibi et al. [1] haben sich ebenfalls mit Daten von Twitter versucht, um die Stimmung in „tweets“]mit einem NLP-Ansatz zu analysieren. Dabei verfolgten sie das Ziel, eine Echtzeit Bestimmung der Stimmung in den Kurznachrichten zu realisieren. Dieses würde nach Alghalibi et al. [1] ein großes Feld an Anwendungen bieten, beispielsweise das Voraussagen des Benutzerverhaltens oder eine Verbesserung beim Marketing.

Grewe und Hu [11] haben ULearn entwickelt, eine Lern-Plattform für Studenten. Die Plattform erkennt dabei die Frustration der Studenten bei der Suche nach Inhalten und sucht mittels NLP Suchanfragen verwandte Inhalte heraus und versucht Lern-Tipps zu geben.

Suvethan et al. [38] beschäftigen sich mit einem Ansatz, die Institute von Universitäten zu entlasten, indem sie einen automatischen Studenten-Ratgeber entwickelt haben, der automatisch auf die Anfragen der Studenten antwortet. Dieses Projekt entstand aus der Erkenntnis, dass viele Studenten stets identische Anfragen an die Institute stellten. Somit verfolgt das Projekt das Ziel, nicht mehr manuell auf dieselben Fragen der Studierenden zu antworten. Dazu wurde eine Wissensdatenbank

aufgebaut. Mittels NLP werden die Anfragen der Studierenden analysiert und die Antwort aus der Wissensdatenbank gesucht. Sollte eine Frage nicht beantwortet werden, leitet das System die Anfrage an einen Menschen weiter.

Seedall et al. [35] haben ein SafeChat für Kinder entwickelt, bei dem mittels NLP und einer Wissensdatenbank Gefahren in Echtzeit erkannt werden sollen. Somit sollen die Kinder vor Gefahren wie Cyber-Mobbing, Radikalisierungen und Betrug geschützt werden.

Iqbal et al. [14] haben sich ebenfalls mit der Aufdeckung von kriminellen Aktivitäten im Internet beschäftigt. Dazu analysierten sie die Chatverläufe aus sozialen Netzwerken mittels NLP. Diese Art der Ermittlung ist nach Iqbal et al. [14] eine sehr effektive Unterstützung bei den Ermittlungsarbeiten von Behörden.

Entscheidend für die Beantwortung der Forschungsfrage nach den Rückschlüssen auf die Zusammenarbeit ist das Erkennen der Emotionen aus der textbasierten Kommunikation. Generell beschäftigen sich die hier vorgestellten Arbeiten nur mit der Erkennung von Emotionen oder Inhalten, dazu verwenden die Publikationen, wie diese Arbeit auch, „natural language processing“. Entscheidende Unterschiede liegen jedoch in den verwendeten Datenquellen und dem avisierten Nutzen. So beziehen viele Ansätze die Kommunikationsdaten aus Kurznachrichtendiensten wie Twitter, andere analysieren ganze Webinhalte auf unangemessene Inhalte. Wieder andere verfolgen das Ziel, eine Art virtuellen Mitarbeiter zu erzeugen. Keiner der vorgestellten Ansätze versucht hingegen, die Zusammenarbeit in einem Team auf Basis der textuellen Kommunikation zu analysieren.

3.3. Entwicklernetzwerke

Neben den Ansätzen der sprachlichen Analyse gibt es noch Ansätze, die mittels der Metadaten der Kommunikation die Möglichkeit bieten, diese zu analysieren und Aussagen über die Zusammenarbeit zu treffen.

Stapel et al. [37] befassten sich mit FLOW-Mapping, einem systematischem Ansatz zur Planung und Verwaltung von Informationsflüssen in verteilten Teams. Die FLOW-Methode dient zur Visualisierung und Analyse von Kommunikationsnetzwerken. In einer Fallstudie haben Stapel et al. [37] gezeigt, dass FLOW-Mapping bei der Etablierung eines verteilten Teams und der Erstellung einer Kommunikationsstrategie unterstützt. Die Konformität, mit der der Strategie gefolgt wurde, lag zwischen 79% und 88%.

Krigel [19] befasst sich in seiner Masterarbeit mit Kommunikationsnetzwerken basierend auf Daten aus Jira und Confluence. Seine Netzwerke stellen Entwickler, Tickets und Dokumente als Konten und die Verbindungen als Relationen dar. Durch diese Darstellungsweise ist es möglich, sich schnell einen Überblick über die Zusammenarbeit und Abhängigkeiten zu verschaffen. Diese automatisierte

Darstellungsweise ermöglicht, wie Krigel [19] beschreibt, eine Vielzahl von konkreten Anwendungsfällen. So wäre es über das Netzwerk einfach möglich, Entwickler zu finden, die sich mit ähnlichen Aufgaben beschäftigt haben.

Andere Ansätze, die nicht mittels NLP versuchen, eine Kommunikationsanalyse durchzuführen, arbeiten mit graphbasierten Verfahren. Diese Verfahren visualisieren die Prozesse der Kommunikation und erlauben über Metriken auf Basis der Metadaten der Kommunikationsdaten Rückschlüsse auf die Kommunikation. Abgesehen von dem Ziel und der Datenquelle unterscheiden sich diese Ansätze gravierend von dem Verfahren, welches in dieser Arbeit vorgestellt wird. Denn es werden nicht die Inhalte der Kommunikation betrachtet und es wird auch nicht mit computergestützten Lernverfahren gearbeitet. Dennoch sind die Ziele ähnlich, nämlich die Kommunikation in Softwareentwicklerteams zu verbessern.

3.4. Kommunikations-Analyse

Die Analyse von Kommunikation in Entwicklerteams war ebenfalls oft Bestandteil von Forschung.

Shakeri et al. [36] haben die Anwendung *ELICitation Aid tool* (ELICA) entwickelt, die Requirements Analysten und Stakeholder bei der Elicitation-Phase helfen soll. Um die anwendungsspezifische Domäne besser zu verstehen, werden dem Analysten dynamische domänenspezifische Informationen zu dem betrachteten Sachverhalt gegeben. So sollen Missverständnisse zwischen Analysten und Stakeholdern verringert werden. Die Erkennung und Extraktion von Information wird mittels eines auf NLP basierenden Modells umgesetzt. Außerdem werden nicht-sprachliche Informationen wie Vertrauensverhältnis und Emotionen in die Analyse aufgenommen. Der Nutzen von ELICA wurde in einer Fallstudie bewiesen.

Schuh und Dreiseitl [33] haben eine Technik analysiert, die aggressive Kommunikation im Internet erkennen soll. Dabei haben sie diese Technik auf zwei Datensätzen zur Validierung angewendet und festgestellt, dass lediglich drei bis 18 Merkmale erforderlich sind, um eine gute Erkennung sicherzustellen. Dabei wurden neben statischen Merkmalen auch Merkmale aus der Grammatik, der Interpunktion und der lexikalischen Analyse verwendet. Auch wenn Schuh und Dreiseitl [33] sich nicht mit Softwareteams beschäftigt haben, so zeigt diese Arbeit die Möglichkeiten der Erkennung.

Einige der vorgestellten Publikationen fungieren als unterstützende Tools zur Anforderungserhebung. Für diesen Zweck ist der hier vorgestellte Ansatz ungeeignet, denn primär soll der Ansatz dieser Arbeit helfen, eine Konversation zu bewerten und nicht inhaltliche Merkmale zu extrahieren. Dennoch zeigen die vorgestellten Verfahren das Potential der NLP-Analyse.

3.5. Abgrenzung der Arbeit

Diese Arbeit grenzt sich auf den Ebenen der Zielsetzung, Verfahrensweise und der Datenquelle eindeutig von den hier vorgestellten Publikationen ab. Dennoch lässt sich die Arbeit in einzelnen Punkten in die thematischen Unterpunkte einordnen. Das Konzept des Analyseverfahrens ist durch die aufgeführten Arbeiten beeinflusst worden.

4

Konzept

Für die automatische Analyse digitaler Medien wird in dieser Arbeit das folgende Konzept zu Grunde gelegt, welches sich insbesondere mit Merkmalen aus der textbasierten Konversation in Entwicklerteams befasst.

4.1. Problemstellung und Ziele

Wie bereits in den Grundlagen verdeutlicht, setzt Teamarbeit Kommunikation voraus. Ebenso wurde festgestellt, dass erfolgreiche Teams auch erfolgreich kommunizieren. Daher die Forschungsfrage:

Zentrale Forschungsfrage

Wie lässt sich das Kommunikationsverhalten in Entwicklerteams analysieren, um Rückschlüsse über die Stimmung abzuleiten?

Um diese Frage zu beantworten, wird im Folgenden die Idee ausgeführt, das digitale Kommunikationsverhalten im Softwareentwicklerteams zu analysieren. Bisherige Ansätze zur Analyse des Kommunikationsverhaltens haben mit erheblichem Aufwand bei der Erfassung und Verarbeitung der Daten zu kämpfen gehabt [27]. Diese Ansätze umfassten Methodiken wie das Filmen von Meetings, die Nutzung von Fragebögen oder das Durchführen von Interviews. Der hohe Zeitaufwand für die Analyse und die zusätzliche zeitliche Belastung der Teammitglieder machten diese Praktiken für die Praxis eher unattraktiv. Gleichwohl ist, wie bereits dargestellt, die Analyse des Kommunikationsverhaltens und dessen damit einhergehende Optimierung entscheidend. Somit bleibt der Bedarf nach einer zeitsparenden und automatisierten Methode zur Analyse des Kommunikationsverhaltens bestehen. Im Idealfall kommt diese Methode ohne zusätzliche zeitliche Belastung des Entwicklerteams aus. Wie in der Einleitung herausgestellt, arbeiten insbesondere dezentral arbeitende Softwareentwicklerteams mit digitalen Medien. Teilweise sind diese Teams nicht auf Ländergrenzen beschränkt, häufig ist daher eine Face-to-Face Kommunikation nicht möglich. Aus diesem Grund werden erhöhte Anforderungen

an die Kommunikation gestellt, was zur Folge hat, dass digitale Kommunikationsmedien eingesetzt werden. Eben diese Kommunikationsmedien zu analysieren soll Ziel dieses Konzeptes sein. Unter Berücksichtigung genannter Faktoren wurde dieses Konzept erarbeitet.

4.2. Idee

Im Folgenden wurde versucht, ein Konzept zu entwickeln, welches den Ansprüchen, den Zielen und Problemen gerecht wird, die Kommunikation in Softwareentwicklungsprojekten automatisch und computergestützt auszuwerten. Um generell automatisch computergestützte Aussagen über eine Kommunikation abzuleiten, ist es erforderlich, ein Verfahren zu entwickeln, welches die Fähigkeiten besitzt, die Nachrichten in einem Kommunikationskanal zu bewerten. Möglich ist dieses über die Klassifizierung der Nachrichten. Als Entscheidungsgrundlage für die Klassifizierung dienen Merkmale, die durch Metriken aus den Nachrichten extrahiert werden können, da ein Klassifizierungsverfahren nicht direkt mit natürlicher Sprache arbeiten kann. Für ein Klassifizierungsproblem mit natürlicher Sprache gibt es bereits, wie in den Verwandte Arbeiten erwähnt, einige Ansätze. Aus den vorgestellten Ansätzen und den Überlegungen aus der „Problemstellung und Ziele“ ergeben sich folgende Ansprüche an ein geeignetes Verfahren:

- Automatisches Einlesen und Filtern der Kommunikationskanäle
- Verwendung von Metriken zur automatischen Extraktion relevanter Merkmale
- Automatische Klassifizierung der Nachrichten in den Kommunikationskanälen
- Automatisiertes Evaluieren der Ergebnisse

Aus den erarbeiteten Ansprüchen ergeben sich vor der Klassifizierung einige Verfahrensschritte, darunter die automatische Filterung und Extraktion der Merkmale. Am Ende der Verarbeitung soll das Verfahren in der Lage sein, eine Aussage über das Kommunikationsverhalten in dem Softwareentwicklungsprojekt zu treffen. Mögliche Aussagen über die Kommunikation könnten sein, dass die Kommunikation in dem betrachteten Zeitraum bezogen auf deren inhaltliche und emotionale Aspekte **positiv**, **negativ** oder **neutral** war. Mithilfe der Betrachtung dieser Ergebnisse soll es beispielsweise möglich sein, negative Tendenzen in dem Projektverlauf zu erkennen und ihnen entgegenzuwirken.

An dieser Stelle wird auf die Anforderung aus der Idee nicht weiter eingegangen. Die konkrete Umsetzung wird in dem Kapitel Umsetzung näher beschrieben, da diese sehr stark von den Kommunikationskanälen abhängt. Jedoch bleibt die Tatsache weiterhin wichtig, dass auch bei diesem Schritt kein zusätzlicher manueller Aufwand für eine Person entstehen soll. Die Idee verfolgt somit das Ziel, ohne zusätzlichen manuellen Aufwand einer Person das Kommunikationsverhalten eines Entwicklerteams einschätzen zu können. Einzig die Beurteilung dieser Einschätzung und das Ziehen von Schlussfolgerungen soll noch einem Menschen zufallen.

4.3. Metriken

Bevor eine Klassifizierung der Kommunikation möglich ist, müssen Metriken gebildet werden, um relevante Eingenschaften aus der natürlichen Sprache zu extrahieren. Um aus Sprache für einen Algorithmus verwendbare Eigenschaften zu extrahieren, gibt es verschiedene Möglichkeiten. Da natürliche Sprache nicht eindeutig ist, wie die Kommunikationsmodelle von Friedemann Schulz von Thun [34] zeigen, ist es keineswegs trivial, die relevanten Informationen aus der Sprache zu extrahieren. In diesem Zusammenhang werden einige mögliche Informationsquellen in der Sprache angeschnitten. Die aufgeführten Metriken können in ihrer Relevanz stark mit dem Verwendungszweck variieren. Jede Metrik muss dabei einzeln für jede Nachricht und jeden Nachrichtenkanal berechnet werden. Daher sollte besonders bei größeren Datensätzen mit vielen Metriken auf die Berechnungszeiten der Metriken geachtet werden. Die für dieses Konzept relevanten Metriken lassen sich in zwei Oberklassen gruppieren.

4.3.1. Statistische Metriken

Statische Metriken beschränken sich in diesem Zusammenhang auf die äußere Form einer Textnachricht. Diese Metriken lassen sich leicht berechnen, da sie häufig durch einfaches Zählen und Bilden des Durchschnitts ermittelt werden können. Generell sind bei dem Erdenken der Metriken keine Grenzen gesetzt. Allerdings sollten stets Relevanz und Kosten für die Berechnung in einem angemessenen Verhältnis stehen. Insbesondere Häufigkeiten und Eigenschaften einzelner Wörter und die Verhältnisse von Wörtern zu anderen Satzbestandteilen wie der Interpunktion können interessant für Metriken sein. Das häufige Vorkommen von langen Wörtern in den Nachrichten könnte für eine sehr fachliche und komplexe Wortwahl sprechen. Die Verwendung von sehr kurzen Wörtern könnte hingegen auf die Benutzung von etablierten Abkürzungen in dem Entwicklerteam hindeuten. Die Verwendung häufiger Interpunktion kann ein Hinweis auf eine informelle und ungezwungen Konversation sein, da hier Emoticons verwendet werden können. Auch die Verwendung bestimmter Wortarten, wie beispielsweise von Adjektiven, kann einen großen Einfluss auf die Wirkung einer Nachricht haben. Die Länge einer Nachricht kann in Kombination mit der Verwendung von Emoticons beispielsweise auch Informationen hinsichtlich der Beziehung zwischen Autor und Empfänger liefern. So kann in einem Entwicklerteam eine lange Nachricht mit vielen Emoticons die freudige Mitteilung eines Arbeitserfolgs sein. Hingegen kann eine lange Nachricht ohne eine auffällig hohe Anzahl an Punktierungszeichen, aber mit sehr langen Wörtern, auf einen Arbeitsauftrag des Teamleiters an den Entwickler hindeuten.

4.3.2. Inhaltliche Metriken

Eine weitere relevante Quelle für Eigenschaften aus Sprache ist eine inhaltliche Betrachtung. Eine solche Betrachtungsweise ist im Gegensatz zu vergleichsweise einfachen Betrachtungen der äußerli-

chen Merkmale wesentlich komplexer. Eine inhaltliche Betrachtung setzt voraus, dass ein gewisses Wissen über die Satzstruktur, den sogenannten **Part of Speech**, vorhanden ist. Eine Betrachtung auf dieser Ebene wird mittels NLP durchgeführt. Dabei wird die Satzstellung eines jeden Wortes ermittelt und für eine inhaltliche Analyse verwendet. Somit ist es möglich, explizite Eigenheiten der Sprache zu erkennen und die Beziehungen zwischen Wörtern herzustellen. Aufgrund dessen ist es möglich, inhaltsbezogene Informationen aus den Nachrichten zu extrahieren. Die dadurch höhere Komplexität der Metriken macht sich auch in der Berechnungszeit der Metriken bemerkbar. In diesem Zusammenhang ist es weiterhin möglich, den emotionalen Wert einer Nachricht zu ermitteln. Dem Interpretationsraum einer Nachricht stehen mit dieser Form der Metriken Informationen einer wesentlich höheren Verarbeitungsstufe zur Verfügung. So kann aus der Verwendung der dritten Person Plural-Dativ und der Versalien-Schreibweise von Nomen und Eigennamen ein Grad für die Formalität der Nachricht gebildet werden. Auch eine Betrachtung der Wortwahl kann auf die Intention und beabsichtigte Wirkung einer Nachricht hindeuten. Werden Wörter gewählt, die mit Angst oder Hass in Verbindung gebracht werden, so kann dies auf ein negatives Verhältnis zwischen den Mitgliedern des Softwareentwicklerteams hindeuten.

Hinweis: In dem Kapitel zur Umsetzung sind die Metriken, die in dieser Arbeit verwendet werden, ausformuliert zu finden und in ihrer Implementierung beschrieben. Die Implementierung wurde bewusst von dem Konzept getrennt, um zunächst einen Überblick über das Analyseverfahren geben zu können.

4.4. Klassifizierung

Durch die Anwendung der Metriken ist es möglich, die relevanten Eigenschaften aus den Nachrichten zu extrahieren, welche über die betrachteten Kommunikationskanäle ausgetauscht werden. Aus diesen Eigenschaften gilt es nun, ein Modell zu lernen, welches in der Lage ist, die Nachrichten zu klassifizieren. Der Klassifizierungsalgorithmus versucht Muster, in den Eigenschaften zu finden. Entscheidend ist, dass nicht alle Merkmale gleich relevant für das gewünschte Ziel sind. Im Vorfeld wurde die Wahl der Metriken durch die Zielsetzung bereits in eine Richtung getrieben. Dennoch ist es ist durchaus möglich, dass einige Metriken keine Relevanz für eine der gewählten Klassen haben. Ein Klassifizierungsalgorithmus würde im Optimalfall solche nicht relevanten Merkmale erkennen und entsprechend nicht für die Entscheidungsfindung heranziehen. Allerdings macht jede zusätzliche Metrik das Klassifizierungsproblem komplexer und es wird schwieriger, ein Modell zu trainieren. Bei der Wahl der geeignetsten Merkmale kann der Klassifizierungsalgorithmus dabei über das vorgestellte Optimierungsverfahren unterstützt werden. Für eine Klassifizierung ist ebenfalls das Wählen geeigneter Klassen notwendig, die bestmöglich den gewünschten Zielen entsprechen. In diesem Fall sind das Klassen, die Rückschlüsse auf die Zusammenarbeit zulassen. Die Idee sieht dazu vor, dass Nachrichten durch den Klassifizierer entsprechend bewertet werden und abschließend eine Zusammenfassung

der Kommunikation erstellt wird. Diese Zusammenfassung soll es ermöglichen, dem Anwender des Verfahrens konkrete Informationen zur Zusammenarbeit zu liefern, um gegebenenfalls Maßnahmen ergreifen zu können.

4.5. Zusammenfassung

Die vorgestellten Probleme zu lösen, ist keine triviale Aufgabe. Besonders deutlich wird dies daran, dass es bereits viele Ansätze gibt, die Kommunikation zu analysieren. Wie in der Abbildung 4.1 zu

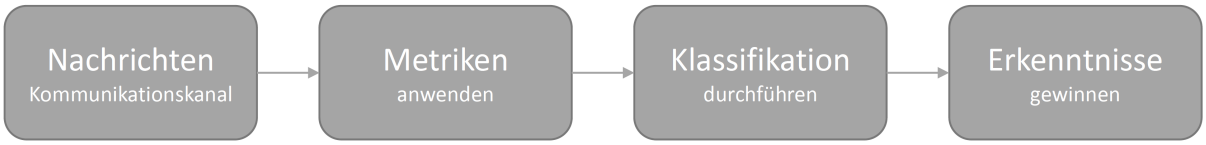

Abbildung 4.1.: Konzeptioneller Ablauf des Analyseverfahrens

sehen ist, lassen sich die grundlegenden Schritte, die in dem Konzept vorgestellt werden, zusammenfassen. Zunächst müssen die Nachrichten von den Komunikationskanälen abgefangen werden, um anschließend durch die Metriken analysiert zu werden. Dabei werden die relevanten Eigenschaften aus den Nachrichten extrahiert. Diese extrahierten Eigenschaften werden im nächsten Schritt für die Klassifizierung genutzt. Nach der Klassifizierung werden die gewonnenen Informationen für eine Zusammenfassung über mehrere Nachrichtenkanäle verwendet. Somit können die in Abschnitt 4.2 vorgestellten Ansprüche an das Analyseverfahren theoretisch erfüllt werden. Im weiteren Verlauf der Arbeit wird das Konzept durch eine Implementierung näher konkretisiert und abschließend im Anhang einer Studie validiert.

5

Umsetzung

Dieses Kapitel beschäftigt sich mit der Umsetzung des zuvor beschriebenen Konzepts. Dabei werden einzelne Punkte weiter ausgeführt. Dabei soll das Kapitel keine Entwicklungsdokumentation darstellen und weniger auf die Ebene des Programmcodes gehen, sondern eher die Idee hinter der Implementierung erläutern. Auf die Implementierung wird dafür in der kurzen Entwicklerdokumentation im Projektordner eingegangen. Zunächst beschäftigt sich die Umsetzung mit den verfügbaren Datenquellen und der Vorverarbeitung der Daten. Anschließend geht es auf die konkrete Umsetzung der Metriken ein, um im Anschluss die Klassifizierung und die Auswertung näher zu beschreiben.

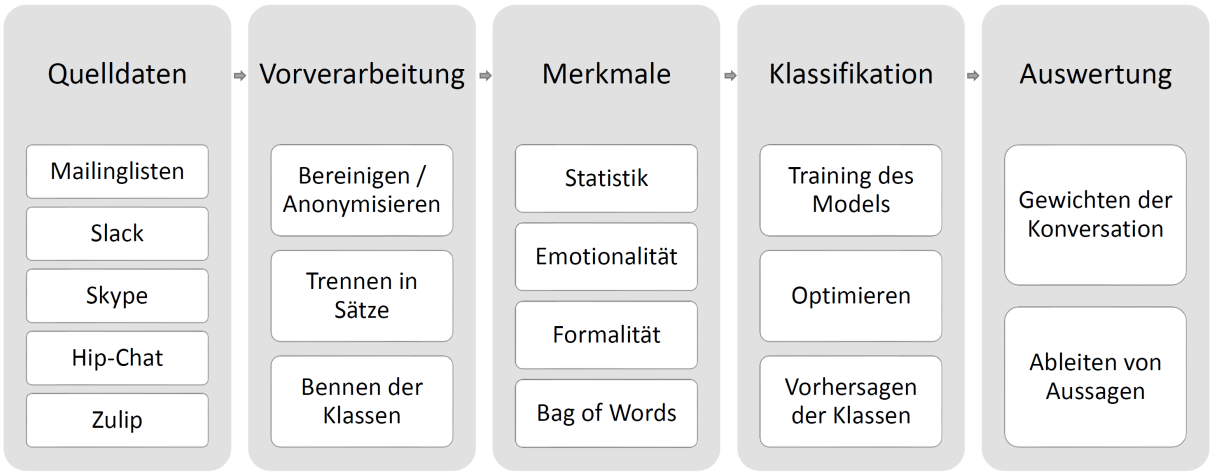

Abbildung 5.1.: Pipeline des Analyseverfahrens der digitalen textuellen Kommunikation eines Softwareentwicklerteams

In der Abbildung 5.1 wird diese Vorgehensweise konkreter dargestellt. Die Arbeitsschritte sind aus diesem Konzept abgeleitet. Werden sie durchlaufen, so wird es nach dem Konzept möglich sein, eine Aussage über die Zusammenarbeit im Softwareentwicklungsteam auf Basis der textuelle Kommunikation abzuleiten.

5.1. Programmiersprache

Für die Entwicklung der Analyse-Software wird in dieser Arbeit die Programmiersprache **Python** verwendet. **Python** findet besonders häufig Anwendung im Bereich Datascience[2], da die Sprache sehr gut lesbar, einfach zu verwenden und zu installieren ist. Für die Implementierungen dieser Arbeit wird die **Python** Version 3.7.3 verwendet. Neben der **Python**-Installation sind noch diverse andere Pakete über *pip*[3] nachzuladen:

- **zulip:** Das Paket ermöglicht einen komfortablen Zugriff auf den Service von **Zulip**[4].
- **sklearn:** Bietet eine Vielzahl an Implementierungen von maschinellen Lernverfahren und Datascience-Tools an.
- **spacy:** Ermöglicht die Analyse von Texten mittels NLP auch für die deutsche Sprache,
- **pyspellchecker:** Ist ein Paket, mit dem es möglich ist, Wörter auf die sprachliche Richtigkeit zu überprüfen.
- **stop_words**: Enthält für verschiedene Sprachen, unter anderem Deutsch, eine Liste mit Stopp-Wörtern.
- **nltk**: Arbeitet wie *spacy* und analysiert die menschliche Sprache.
- **panda**: Bietet Tools zur Analyse von großen Datenmengen an.
- **Jinja2**: Erstellt mit Vorlagen automatisch Dokumente.
- **seaborn:** Visualisiert statistische Daten in verschieden Darstellungen.
- **matplotlib:** Ermöglicht das Zeichnen von Graphen.
- **wordcloud:** Erzeugt Wordclouds von Dokumenten und dient zur Datenexploration.

5.2. Datenquelle

Die Umsetzung des Konzeptes wird bedingt durch die Datenquellen, die später analysiert werden sollen. Im Gegengensatz zu dem Konzept ist die Schnittstelle, durch die die Kommunikationsdaten den Analyseverfahren zugeführt werden können, weniger generalisierbar. Im Folgenden sind drei mögliche Datenquellen vorgestellt, auf die das Konzept angewendet werden kann. Dabei beschränkt sich die Auswahl auf Softwareentwicklungsprojekte, für die das Konzept explizit entwickelt wurde. Jedes der aufgeführten Beispiele unterscheidet sich in der Homogenität der Zusammensetzung des Ent-

[2] JetBrains führte 2018 zu der Verwendung von **Python** eine Umfrage unter Entwicklern durch. Nähere Informationen unter https://www.jetbrains.com/research/python-developers-survey-2018/
[3] *pip* ist der Package-Installer von **Python**
[4] Das Programm Zulip wird in dem Abschnitt zu Zulip näher beschrieben.

wicklerteams und betrachtet eine andere Herausforderung in der Zusammenarbeit. Für die sinnvolle Anwendung des Konzepts ist die Wahrung der für das Konzept festgelegten Ansprüche an das Softwareentwicklungsteam zu beachten. Insbesondere steht hier die Verwendung eines digitalen, textuellen Kommunikationsmediums und die Tatsache der Dezentralität des Teams im Vordergrund.

5.2.1. Studentische Softwareprojekte

Das Fachgebiet Software Engineering der Leibniz Universität Hannover bietet jedes Jahr mehrere Veranstaltungen an, bei denen Studierende praktische Erfahrungen in der Softwareentwicklung sammeln können. Die Verarbeitung der Kommunikationsdaten eines Projektes aus einer dieser Veranstaltungen bietet durchaus einen Reiz. Denn hierbei müssen die Ansprüche an das Entwicklerteam in den Punkten der Dezentralität und des einheitlichen Kommunikationsmittels mit einer Schnittstelle zum Datenexport, Datenschutz und Schutz der Privatsphäre gewahrt werden. Dafür wird eine sehr bunte Mischung an Erfahrungsniveaus, kulturellen Hintergründen, Altersklassen und Geschlechtern sowie Fachdisziplinen geboten. In diesem Zusammenhang bietet das Fachgebiet für Software-Engineering die Lehrveranstaltung **Softwareprojekt** an. Eine immer im Wintersemester stattfindende Veranstaltung, welche alle genannten Voraussetzungen an das Entwicklerteam aus dem Konzept erfüllt. In dieser Veranstaltung werden den Studierenden verschiedene Projekte zur Auswahl gestellt. Anschließend werden Entwicklerteams nach Fähigkeiten und Projektwunsch gebildet. Die Kommunikation eines solchen Projektteams würde Erkenntnisse über das Sozialverhalten in jungen und unerfahrenen Softwareentwicklerteams liefern. Limitierend wirkt sich allerdings aus, dass kein einheitliches Kommunikationsmedium seitens des Fachgebietes in der Lehrveranstaltung vorgegeben wird und damit die Kommunikation nur sehr eingeschränkt verwendet werden könnte.

5.2.2. Öffentliche Projekte

Neben der Möglichkeit der Verwendung eines studentischen Projektes bietet sich die Möglichkeit, freie und öffentlich verfügbare Kommunikation für die Analyse in Betracht zu ziehen. Dies ist im Hinblick auf die Zusammensetzung des Entwicklerteams sehr interessant, da sich die Teammitglieder in einem sehr breiten Spektrum bewegen, was Erfahrungen, Alter und zeitlichen Einsatz angeht. Problematisch für die Vergleichbarkeit der Ergebnisse sind allerdings die nicht eindeutigen Bedingungen unter welchen die Kommunikation entstanden und ob die verfügbare Kommunikation vollständig ist. Eine weitere Hürde stellt multilinguale Kommunikation dar, da einige Merkmale sehr sprachspezifisch sind.

5.2.3. Privatwirtschaftliche Projekte

Eine dritte Möglichkeit besteht darin, die Kommunikationsdaten eines Unternehmens aus der Privatwirtschaft zu analysieren. Ein solches Softwareentwicklungsteam würde sich im Gegensatz zu den zuvor dargestellten Projekten in der Homogenität der Entwickler sehr unterscheiden. Denn in einem Unternehmen finden häufig Verhaltensregeln, eine gemeinsame Leitkultur, Code-Konventionen und eine einheitlichere Tool-Landschaft Anwendung. Dieses und besonders der Anspruch der Professionalität dürften einen entscheidenden Einfluss auf die Kommunikation haben, weshalb die Analyse eines solchen Entwicklerteams spannend ist.

Wie einleitend erwähnt, ist das Konzept nicht auf einen Typ von Datenquelle festgelegt, sondern hat im Gegenteil den Anspruch zu generalisieren. Dennoch besteht je nach Projektwahl Anpassungsbedarf an die Quelldaten.

5.3. Vorverarbeitung der Kommunikationsdaten

Das Konzept ist, wie im vorherigen Abschnitt bereits erwähnt, auf verschiedene Datenquellen und Teams anwendbar. Unabhängig von der Wahl der Datenquelle ist es erforderlich, die Kommunikationsdaten aus dem Medium zu exportieren und weiterzuverarbeiten. Die Vorverarbeitung ist sehr stark abhängig von der Wahl des Kommunikationsmediums. Generell sind folgende Schritte notwendig, um die Daten in einer adäquaten Qualität bereitzustellen.

Zunächst müssen die Daten von allem, was nicht natürliche Sprache ist (darunter fallen beispielsweise Programmcode oder HTML-Tags und Links oder Verweise), bereinigt werden. Nachdem die Nachrichten entsprechend behandelt wurden, ist es möglich, Namen in den Texten, Absender sowie Empfänger zu anonymisieren. Dies dient dem Schutz der Privatsphäre der Teammitglieder und dritter Personen.

Da eine Nachricht als ein zusammenhängender String aus mehreren Sätzen bestehen kann, ist es auch erforderlich, die Nachrichten in einzelne Sätze und damit Strings zu teilen. Dies geschieht vor dem Hintergrund, dass einzelne Sätze sehr verschiedene Wirkungen haben können. Somit ist es möglich, die Analyse detailreicher durchzuführen, wenn eine Nachricht nicht als eine Einheit bewertet wird.

Damit Lernverfahren „lernen“ können, ist es erforderlich, jeden Satz einer vordefinierten Klasse zuzuordnen. Dieser Vorgang wird als Labeln bezeichnet. Auf den Vorgang des Labelns wird in der Evaluation näher eingegangen.

5.4. Metriken

Der wohl entscheidenste Punkt für die Analyse ist das Extrahieren geeigneter Merkmale (Features) über Metriken, die möglichst relevante Informationen über die gegebene Nachricht liefern. Die hier generierten Merkmale sind die einzige Datenquelle, aus denen die Lernverfahren Muster erlernen können, um so die Nachrichten zu klassifizieren. Die Liste der aufgeführten Metriken erhebt keinen Anspruch auf Vollständigkeit, sondern wurde durch Brainstorming und den Verwandte Arbeiten inspiriert.

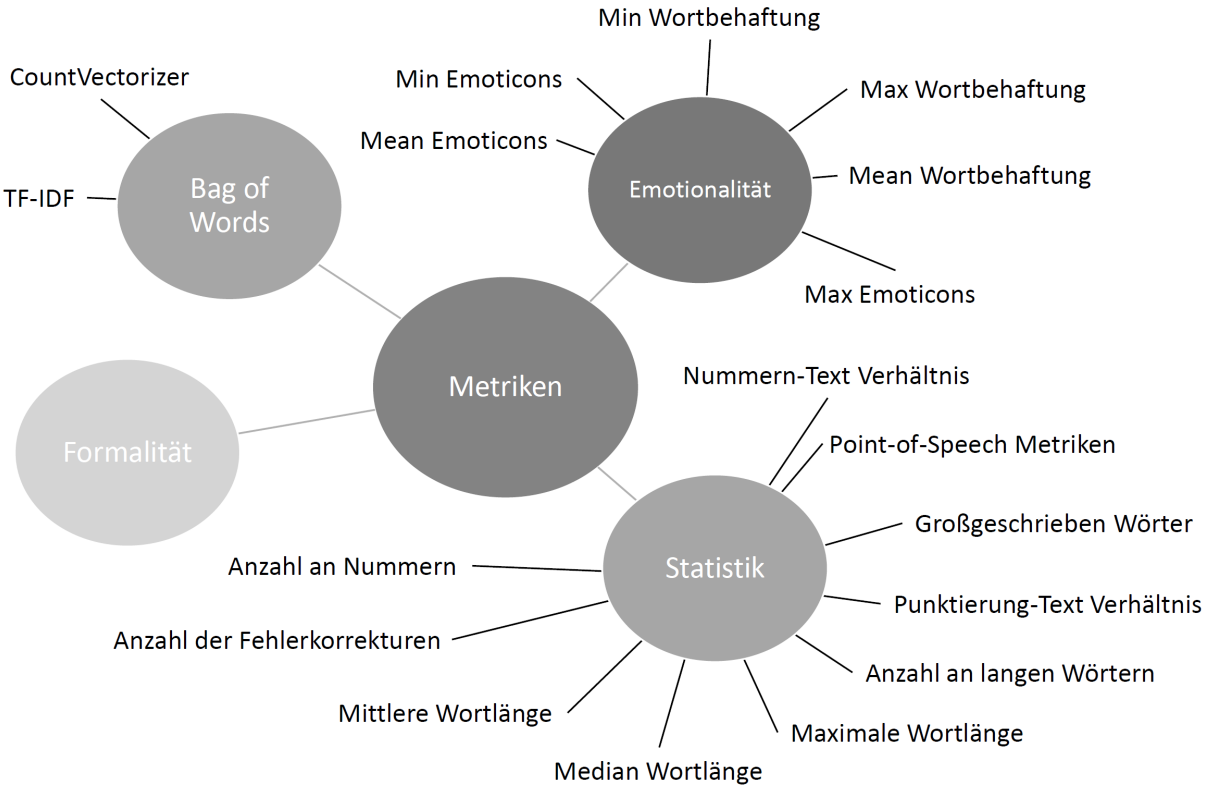

Abbildung 5.2.: Mind-Map der verwendbaren Metriken

Wie die Abbildung 5.2 zeigt, ergaben sich vier Kategorien, in die sich die Merkmale gruppieren lassen. Im Folgenden wird konkreter auf die Implementierung der einzelnen Metriken eingegangen. Dazu werden die Metriken auf folgenden fiktiven Satz angewendet:

> Sie sollten mit der Vorstellung der Ergebnisse nicht wieder bis zum Mittagessen warten ;)

Der Beispielsatz wird für die Anwendung der Metriken weiter verarbeitet, denn alle Metriken (außer einigen statistischen Features) filtern die Eingabesätze. Bei der Filterung werden Stoppwörter, wie

Artikel und Satzzeichen sowie Wörter, die Zahlen enthalten, entfernt. Dies hat den Hintergrund, dass solche Satzbestandteile sehr häufig vorkommen und damit wenige Informationen für die Analyse bieten oder für die Analyse nicht relevant sind. Für einige Metriken sind diese Informationen relevant, daher werden diesen Metriken auch die vollständigen Sätze zur Verfügung gestellt. Neben der Bereinigung der Sätze durch die Filterung werden die übrigen Wörter automatisch sprachlich korrigiert, lemmatisiert und kleingeschrieben. Bei der Lemmatisierung werden die Wörter in ihre Grundform gebracht. Dies vereinfacht die Suche in Wortdatenbanken und minimiert die Anzahl der Variationen. Aus der Verarbeitung ergibt sich somit aus dem Beispielsatzes folgende Wortreihung:

> sollen vorstellung ergebnis mittagessen warten

5.4.1. Statistische Features

Unabhängig von dem Inhalt einer Nachricht können wichtige Informationen mit einfachen statistischen Metriken extrahiert werden. Diese Features beschreiben vor allem die Form der Sätze wie beispielsweise die Anzahl der langen Wörter. Ein Wort wird dabei als lang markiert, wenn es mehr als sieben Buchstaben enthält. Ein durchschnittliches deutsches Wort hatte 2009 5,7 Buchstaben [32] in 1,4 Milliarden Wortformen, die die Volltextdatenbank des Duden zählt. Somit wurde für diese Arbeit festgelegt, dass ein Wort der Länge sieben lang ist. Bezogen auf die Anzahl der Buchstaben wird noch die Länge des längsten Wortes gemessen. Ebenfalls als Merkmal fließt die durchschnittliche Länge der Wörter und der Median als Merkmal ein. Betrachtet man die Wortebene, so wird die Anzahl von Interpunktion, Freistellen, Zahlen und Wörter, die geführt werden, durch Versalien ins Verhältnis zu der Anzahl der Wörter gesetzt. Die Textlänge wird ebenfalls als Merkmal aufgenommen.

Neben diesen sehr einfach zu ermittelnden Metriken sind noch komplexere Metriken, wie beispielsweise ein Maß für die Anzahl der Rechtschreibfehler, implementiert. Diese Metrik bedient sich der Bibliothek **pyspellchecker**, die mit der Levenshtein-Distanz zum deutschen und englischen Wörterbuch arbeitet, um wahrscheinlich falsch geschriebene Wörter zu ermitteln. Diese automatische Textkorrektur birgt die Gefahr der falschen Klassifizierung und benötigt auf Grund der Berechnung der Levenshtein-Distanz von jedem Wort zu jedem anderen in den Wörterbüchern viele Leistung.

Eine weitere komplexere Metrik bedient sich der **Part-Of-Speech**-Analyse. Denn diese stellt das Verhältnis zwischen Verben, Adjektiven und Nomen zu anderen Wörtern dar.

In der Tabelle sind die Ergebnisse der statistischen Metriken für den Beispielsatz dargestellt. Alle ermittelten Werte der Metriken werden auf das Intervall von -1 bis 1 transformiert. Dies hängt mit den Eingangsvektoren für die Lernverfahren zusammen.

Tabelle 5.1.: Ergebnisse der Berechnung der statistischen Metriken (Die Metriken sind auf das Intervall von -1 bis 1 skaliert)

Metrik	Ermittelter Wert
Anzahl der Fehlerkorrekturen	-1
Textlänge	-0.15
Mittlere Wortlänge	0.123
Median Wortlänge	0
Anzahl der Wörter	0.3
Maximale Wortlänge	-0.266
Anzahl an langen Wörtern	-0.4
Anzahl an Freistellen	-0.717
Nummern-Text Verhältnis	-1
Interpunktion-Text Verhältnis	-0.4
Großgeschriebene Wörter	-0.6

Metrik	Ermittelter Wert
ADVRatio	-0.9
ADJRatio	-0.9
VERBRatio	-0.8
NOUNRatio	-0.9
PARTRatio	-0.9
AUXRatio	-1
ADPRatio	-0.7
DETRatio	-0.8
PROPNRatio	-1
SCONJRatio	-1

5.4.2. Emotionalität

Die Ermittlung der Emotionalität geschieht im Wesentlichen über zwei Varianten. Beide Ansätze arbeiten mit der Färbung von Textausdrücken. Dabei meint die Färbung eines Wortes im Kontext dieser Arbeit die Emotionalität. Der erste Ansatz versucht die emotionale Färbung über Muster in der Interpunktion, den sogenannten Emoticons, zu erkennen. Der zweite Ansatz arbeitet auf Grundlage der Färbung der Wörter mit Textkorpora.

5.4.2.1. Emoticons

Wie bereits angesprochen, wird mit erstem Ansatz versucht, die Verwendung von Emoticons in die Analyse aufzunehmen. Dazu haben Wang und Castanon [41] einen Zusammenhang zwischen der Verwendung von Emoticons und der Intention bei menschlicher Kommunikation herstellen können. Wang und Castanon [41] haben festgestellt, dass ein Emoticon nicht immer genau einer Klasse (**positiv**, **negativ** und **neutral**) zugeordnet werden kann und daher eine Liste mit Emoticons präsentiert, die eine Liste mit den Wahrscheinlichkeiten für die Zugehörigkeit zu einer der Klassen angibt. Für diese Arbeit wurde die höchste Klassenwahrscheinlichkeit eines jeden Icons gewählt. So sind Emoticon wie **:D** zu 90% als **positiv** gewertet. Hingegen wurde nach Wang und Castanon [41] die Zeichenkombination **:(** zu 95% als **negativ** klassifiziert. In dieser Arbeit wurden die Emoticons auf Grundlage der in der Studie ermittelten Werte zur höchsten Klassenwahrscheinlichkeit bewertet. Bewusst wurden die Klassen **neutral** und unbekannt von Wang und Castanon [41] ignoriert, um das Model einfacher zu halten. Die Wahrscheinlichkeiten zur negativen Klasse wurden dabei mit einem negativen Vorzeichen versehen. Alle Emoticons in einer Nachricht wurden mit der so erstellten Liste abgeglichen und der

Durchschnitt der Wahrscheinlichkeit wurde auf diese Weise für den Satz gebildet. Der ermittelte Wert fließt als Metrik in das Model ein.

5.4.2.2. Wortfärbung

Eine emotionale Färbung ist nicht nur in Emoticons zu erkennen. Vielmehr besteht, wie auch Wang und Castanon [41] festgestellt haben, ein Zusammenhang zwischen der Wortwahl und den verwendeten Emoticons. Emoticons sind im Gegensatz zu Wörtern nicht an die Sprache, sondern an das kulturelle Verständnis gekoppelt. Das Verständnis von Emoticons ist in den westlichen Ländern wesentlich homogener als die verwendeten Sprachen. So gibt es alleine in der Europäischen Union 28 Länder, von denen fast jedes Land eine eigene Sprache besitzt. Zwar wird für die internationale Kommunikation die englische Sprache bevorzugt, allerdings wird mehrheitlich in der Amtssprache des jeweiligen Landes kommuniziert. Um die emotionale Färbung der einzelnen Wörter in einem deutschen Projekt festzustellen, ist eine Datenbank, in der diese Färbung angegeben wurde, notwendig. Waltinger [40] hat eine solche Datenbank erstellt, in der die deutschen Wörter entsprechend ihrer emotionalen Färbung in die Klassen **positiv**, **negativ** und **neutral** eingeordnet wurden. Neben Waltinger haben sich auch Remus et al. [28] mit der Erstellung einer Datenbank mit positiven und negativen deutschen Wörtern beschäftigt. Allerdings unterscheiden sich die Datensätze in dem Punkt, dass Remus et al. [28] auch die einzelnen Wörter entsprechend ihrer Polarität gewichtet haben. Somit ist eine feinere Differenzierung zwischen den Wörtern möglich. Der Datensatz von Remus et al. [28] umfasst 1827 negativ und 1644 positiv behaftete Wörter. Waltinger [40] verfügt über einen größeren Datensatz mit 5929 negativen, 1229 neutralen und 3632 positiven Wörtern. Auf Grund der unterschiedlichen Datensätze wird aus beiden Datensätzen jeweils eine eigene Metrik erstellt.

Für die Metrik zum Datensatz von Waltinger [40] werden alle Wörter einer Nachricht mit der Datenbank abgeglichen und bei Vorkommen entsprechend der Klasse (bei **negativ** mit -1, bei **positiv** 1 und bei **neutral** mit 0) bewertet. Für jeden Satz wird der Durchschnitt bestimmt und als Metrik für das Modell verwendet, dabei werden im Datensatz nicht gefundene Wörter ignoriert. Im Folgenden sind einige Beispiele für **positive**, **negative** und **neutrale** Wörter aus dem Datensatz, in der Tabelle 5.2, angeführt.

Tabelle 5.2.: Beispiele für die Klassifizierung von Waltinger [40]

positiv	**negativ**	**neutral**
zustimmen	willkürlich	objektiv
überzeugt	konfus	vollauf
Innovation	defizitär	Gedanke
Gewaltlosigkeit	Autokrat	Argumentation

Die Metrik zum Datensatz von Remus et al. [28] arbeitet prinzipiell sehr ähnlich wie die zuvor vorgestellte Metrik. Es wird ebenfalls Wort für Wort einer Nachricht mit dem Datensatz abgeglichen. Allerdings wird statt der groben Gewichtung des obigen Ansatzes hier die Polarität des Wortes verwendet. Abschließend wird auch der Durchschnitt gebildet und als Metrik in das Modell genommen. Zusätzlich werden bei diesem Ansatz auch noch das Minimum und das Maximum bestimmt und als weitere Metriken in das Modell aufgenommen. Im Folgenden werden auch zu diesem Datensatz einige Beispiele aufgeführt, wie in Tabelle 5.3 zu lesen ist.

Tabelle 5.3.: Beispiele für die Polarität von Remus et al. [28]

Wort	Polarität
zustimmen	0.0040
willkürlich	-0.3481
überzeugt	0.2381
Innovation	0.0040
defizitär	-0.4535

5.4.2.3. Beispiel für die Emotionalität

Um die Emotionalität zu erfassen, wurden drei Ansätze vorgestellt. Im Folgenden werden anhand des zuvor ausgewählten Beispielsatzes die Ergebnisse der Metriken gezeigt.

Tabelle 5.4.: Anwendung der Metriken an dem Beispielsatz

Metrik	Wert
∅ Emoticon	0.75
min. Emoticon	0.75
max. Emoticon	0.75
∅ Waltinger	-0.333
∅ Remus	0
min. Remus	0
max. Remus	0

Wie in der Tabelle Tabelle 5.4 zu sehen ist, ist für diesen Satz lediglich die Metrik für die Emoticons und die des Waltinger Datensatzes [40] aussagekräftig. Dies liegt daran, dass die Metrik auf Basis von Waltingers [40] Datensatz ein negatives, kein positives und zwei neutrale Worte feststellt. Somit ist der Wert -0.333 und damit leicht **negativ**. Für die Metrik, die auf den Datensatz von Remus et al. [28] basiert, sind keine Wörter in dem Datensatz zu finden. Folglich liefert die Metrik den Wert 0.

5.4.3. Bag of Words

Ein weiteres Feature sind Bag of Words. Dieses Feature repräsentiert Wörter als Vektoren, die das Vorkommen eines Wortes angeben. Für diese Bag of Words gibt es verschiedene Möglichkeiten zur Erzeugung der Repräsentationeeine wie inn. In dieser Arbeit wird auf zwei Verfahren näher eingegangen. Für die Bag of Words Metriken werden die Sätze bereinigt betrachtet.

5.4.3.1. CountVectorizer

Die erste vorgestellte Bag of Words Metrik ist der CountVectorizer. Der CountVectorizer ist eine Implementierung in dem Package **Sklearn**, um Texteingaben als nummerische Vektoren darzustellen. Dabei wird für jedes neu vorkommende Wort der Vektor erweitert. Wird nach dem Bau des Vektors eine Evaluierung eines Textes mit dem CountVectorizer durchgeführt, so wird das Ergebnis ein Vektor sein, der stets die Länge der zuvor trainierten einzigartigen Wörter hat. Der Vektor wird für jedes in dem Satz vorkommende Wort eine Eins enthalten und die nicht vorkommenden Wörter mit Nullen füllen. Somit erhält man eine Repräsentation, die angibt, welche Wörter in der Eingabe vorkommen. Allerdings ist eine Rekonstruktion des Eingabesatzes nicht vollständig, wenn die Wortreihenfolge nicht beachtet wird. Außerdem werden zuvor nicht trainierte Wörter in der Repräsentation ignoriert. Um mehr Informationen über den Kontext abzuleiten, können N-Gramm verwendet werden, bei denen die Wortpaare und nicht einzelne Wörter in dem Vektor repräsentiert werden. Je mehr Kontextinformationen abzuleiten versucht werden, desto spezifischer und größer wird der Eingabevektor. Bei kleinen Trainingssets kann die Bildung von N-Gramm sich **negativ** auf die Aussagekraft der Repräsentation auswirken. Für den Beispielsatz sind in der Tabelle 5.5 die Darstellung für die wortweise Betrachtung und ein bi-gram aufgezeigt.

Tabelle 5.5.: Beispielsatz als mono-gram und bi-gram.

mono-gram				
sollen	vorstellung	ergebnis	mittagessen	warten
bi-gram				
sollen vorstellung	vorstellung ergebnis	ergebnis mittagessen	mittagessen warten	

Wenn nun die Nachricht „Heute stellen wir die Ergebnisse beim Mittagessen vor.“ (*heute stellen ergebnisse beim mittagessen*) mit dem zuvor trainierten CountVectorizer repräsentiert werden soll, würde sich für das mono-gram und bi-gram eine wie in Tabelle 5.6 dargestellte Repräsentation ergeben.

Tabelle 5.6.: Beispielsatz im trainierten CountVectorizer

mono-gram

sollen	vorstellung	ergebnis	mittagessen	warten
0	0	1	1	0

bi-gram

sollen vorstellung	vorstellung ergebnis	ergebnis mittagessen	mittagessen warten
0	0	0	0

An der Darstellung ist zu sehen, dass insbesondere im Bi-Gramm keine Übereinstimmung zu finden ist, obwohl die Sätze ähnlich wirken.

5.4.3.2. TF-IDF Vectorizer

TF-IDF (term frequency-inverse document frequency) Vectorizer ist wie der CountVectorizer ebenfalls bereits in dem Package **Sklearn** vorimplementiert. Der TF-IDF Vectorizer liefert eine ähnliche Rückgabe wie der CountVectorizer. Beide geben eine Vektorrepräsentation zurück. Im Gegensatz zum CountVectorizer liefert der TF-IDF Vectorizer keine binäre Repräsentation, sondern einen Vektor, der die Relevanz der Wörter angibt. Das Verfahren arbeitet mit dem Produkt aus zwei Termen, dem *term frequency* und dem des *inverse document frequency*. Der Term *term frequency* definiert sich wie folgt

$$\mathrm{tf}(t, D) = \frac{\#(t, D)}{\max_{t' \in D} \#(t', D)}$$

wobei t für den Term steht, t' steht für einen beliebigen Term aus dem Dokument und D für das Dokument, in dem die Häufigkeit des Vorkommens gesucht ist. Der Term *inverse document frequency* ist folgendermaßen definiert

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

wobei t für den Term steht und N für die Anzahl der Dokumente. Der *term frequency* bestimmt die Häufigkeit des Terms im Dokument. Im Gegenzug zum *inverse document frequency* wird hier die Spezifität des Terms über alle Dokumente gemessen. Das Produkt aus diesen beiden Termen gewichtet dann die Häufigkeit des Terms mit seiner Relevanz:

$$\mathrm{tf.idf}(t, D) = \mathrm{tf}(t, D) \cdot \mathrm{idf}(t)$$

Durch diese Art der Gewichtung werden markante Wörter, die selten vorkommen, hervorgehoben und häufig vorkommende Wörter weniger präsent. Zusätzlich kann eine minimale Häufigkeit der

Wörter angegeben werden, um zu seltene Begriffe nicht zu sehr hervorzuheben. Die Anwendung des Beispielsatzes ist für diese Metrik wenig relevant, da nur mit einem Satz trainiert werden würde.

5.4.4. Formalität der Konversation

Die Formalität einer Konversion kann den Umgangston stark beeinflussen, weshalb das Erkennen der Formalität von Relevanz sein könnte. Für die Detektion der Kommunikation in der Höflichkeitsform wurden zwei markante Eigenschaften der Höflichkeitsform im Deutschen ausgewählt. Zum einen ist es die Verwendung der dritten Person Plural und die Versalien-Schreibweise von Nomen und Namen. Zusätzlich wird angenommen, dass bei der Verwendung der Höflichkeitsform auf die Rechtschreibung geachtet wird, weshalb die Anzahl der sprachlichen Autokorrekturen mit in die Metrik einfließt. Die Metrik dazu arbeitet für die Erkennung mit **Part-Of-Speech**, um Nomen und Personalpronomen zu detektieren. Die höfliche Anrede beginnt stets mit *Sie*, *Ihrer* und *Ihnen*. Wie bereits erwähnt, besteht die Analyse der Förmlichkeit aus drei Teilen. Im ersten Teil wird das Verhältnis zwischen formalen und nicht formalen Personalpronomen gebildet. Im nächsten Schritt wird das Verhältnis von großgeschriebenen zu kleingeschriebenen Nomen gebildet. Zuletzt fließt noch das Verhältnis von korrigierten zu nicht korrigierten Wörtern in die Berechnung ein. Angewendet auf den Beispielsatz in seiner unveränderten Form ergibt sich daraus ein Wert für die Formalität von 0.333. Der Wert deutet an, dass die Kommunikation eher formaler Natur ist.

5.5. Klassifizierung

Nachdem die Metriken für die Generierung der Features definiert und implementiert sind, muss nun das Model trainiert werden. Dazu werden Lernverfahren genutzt, die Sätze Klassen zuordnen. Für diesen Ansatz wurden die Klassen **positiv**, **negativ** und **neutral** gewählt, je nach Verwendungszweck kann die Wahl der Klassen angepasst werden. Es gibt eine Vielzahl von Klassifikatoren, die verwendet werden können. **Sklearn** enthält Implementierungen für diverse Klassifikatoren, unter anderem Implementierungen eines **Random Forests**, einer **Support Vector Machine** und eines **Naive Bayes**. Für die Klassifizierung können auch mehre Klassifikatoren kombiniert werden, beispielsweise über einen **VotingClassifier**. Bei diesem Verfahren würden alle Klassifikatoren bestimmt werden und abschließend würde der **VotingClassifier** gewichtet die mehrheitliche Klasse wählen.

5.5.1. Lernverfahren

Im Folgenden werden die drei erwähnten Klassifikatoren kurz umrissen. Ein **Random Forest** ist ein Klassifikator, der nach dem Zufallsprinzip Entscheidungsbäume aufbaut. Bei der Klassifizierung klassifiziert jeder Baum selbst. Nach der Klassifizierung der einzelnen Bäume wird die mehrheitliche

Klasse gewählt. Die einzelnen Entscheidungsbäume sind unabhängig und nicht zusammenhängend. Dies wird durch die zufällige Featurewahl beim Aufbau des Baumes sichergestellt. Für jeden neuen Knoten in einem Baum wird ein zufälliges Subset der Merkmalsmenge gebildet und durch Bagging das beste Merkmal ausgewählt. Auf diese Weise kann eine Vielzahl von verschiedenen Bäumen gebildet und eine stabile Klassifizierung sichergestellt werden. Durch die Unabhängigkeit der einzelnen Bäume kann das Training mit einem **Random Forest** sehr einfach parallelisiert werden, was ein schnelles Lernen ermöglicht. Mit wachsender Anzahl der Bäume wird allerdings die Klassifizierung erheblich langsamer als bei anderen Lernverfahren, da jeder Entscheidnungsbaum berechnet werden muss. Durch die häufig verwendeten Pfade, die bei der Klassifizierung gegangen werden, können die relevantesten Merkmale bestimmt werden. Damit ist es möglich, Entscheidungen der Bäume bedingt nachzuvollziehen. **Sklearn** bietet für die in dem Package enthaltende Implementierung des **Random Forest** eine Vielzahl von Parametern an. Zu den wichtigsten zählt die Anzahl von Bäumen und deren Tiefe. In der Abbildung 5.3 wird eine abstrakte Darstellung des verwendeten **Random Forest** dargestellt. In der Abbildung sind die einzelnen Entscheidungsbäume zu sehen, die die Eingabe jeweils selbst klassifizieren, sodass später über das Bilden einer Mehrheit eine Klasse als Ergebnis angesetzt werden kann. Die einzelne Bäume müssen nicht die gleiche Tiefe haben und auch nicht ausbalanciert sein, sodass die Klassifizierung, wie in der Darstellung angedeutet, in den schwarzen Knoten endet. In jedem Knoten eines Entscheidungsbaums wird über ein Feature gesplittet.

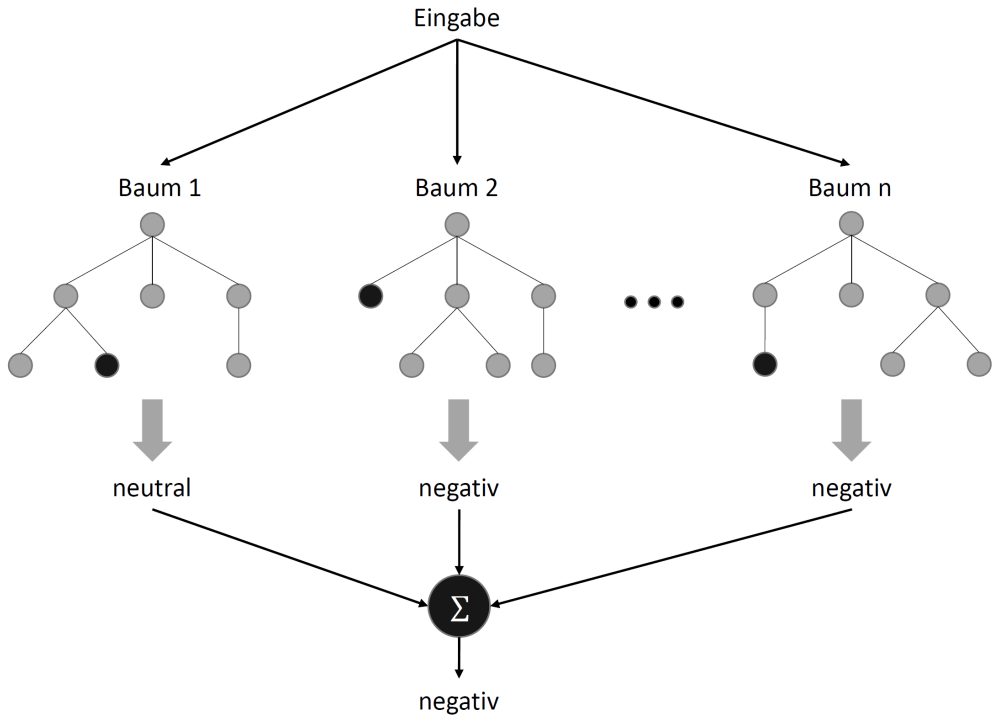

Abbildung 5.3.: Beispielhafte Darstellung der Klassifizierung mit einem Random Forest

Eine **Support Vector Machine** (**SVM**) arbeitet anders als ein **Random Forest** nicht mit Entscheidungsbäumen, sondern durch die Separierung des Merkmalraumes. Bei der Suche nach einer möglichst klaren Trennung des multidimensionalen Merkmalraumes existieren mehrere Ansätze diesen zu trennen. Unterschieden wird zwischen linear separierbaren Problemen, bei denen es möglich ist, den Merkmalraum durch eine lineare Funktion zu trennen und nicht linear separierbaren Problemen, bei denen ein minimaler Fehler bei der Trennung in Kauf genommen wird, indem eine nichtlineare Funktion den Merkmalsraum trennt. In der Praxis sind die meisten Probleme nicht linear separierbar. Das Beispiel in Abbildung 5.4 zeigt einen einfachen zweidimensionalen Merkmalraum. Bei der Verwendung der vorgestellten Metriken hätte der Merkmalraum eine vierstellige Dimensionzahl, weshalb er nicht wie in diesem Beispiel durch einen Strich, sondern durch eine Hyperebene geteilt werden könnte. Die grauen und schwarzen Punkte stellen in diesem Beispiel **neutral** und **negativ** gelabelte Datenpunkte dar. Der gestrichelt dargestellte Punkt gilt, klassifiziert zu werden. Durch die lineare Trennung des Merkmalsraums würde die **Support Vector Machine** dem Datenpunkt die Klasse **negativ** zuweisen. Jeder Datenpunkt in der Abbildung 5.4 stellt dabei einen Satz dar, für den die Metriken berechnet wurden.

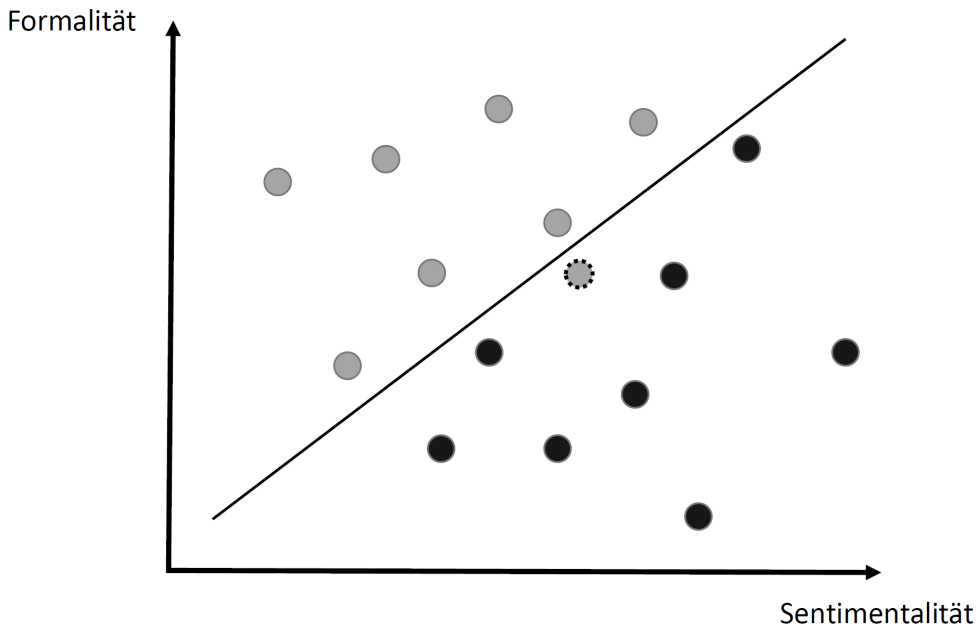

Abbildung 5.4.: Beispielhafte Darstellung der Klassifizierung mit einer SVM

Der **Navie Bayes** Klassifikator ist benannt nach dem Mathematiker Thomas Bayes und arbeitet im Gegensatz zu den anderen vorgestellten Verfahren mit Wahrscheinlichkeitsfunktionen, die angeben, ob ein Punkt im Merkmalraum zu einer Klasse gehört. Dieses ist nur über eine Annäherung möglich, da die Dichtefunktionen für die Wahrscheinlichkeiten unbekannt sind. Der **Navie Bayes** Klassifikator

lässt sich schnell berechnen, hat allerdings durch die eindeutige Zuordnung Probleme mit zu stark korrelierenden Merkmalen.

Für diese Arbeit werden die drei vorgestellten Klassifikatoren über einen **VotingClassifier** zusammengefasst. Über die Kombination aus mehreren Klassifikatoren soll versucht werden, die Klassifizierung zu verbessern und robuster gegen ein Rauschen in den Daten zu machen. Somit ergibt sich für die Klassifizierung der Nachrichten der Prozess, der in Abbildung 5.5 gezeigt wird. Um die Modelle zu trainieren, wurden die Eingangsdaten im Trainings- und Testset in dem Verhältnis **90:10** getrennt.

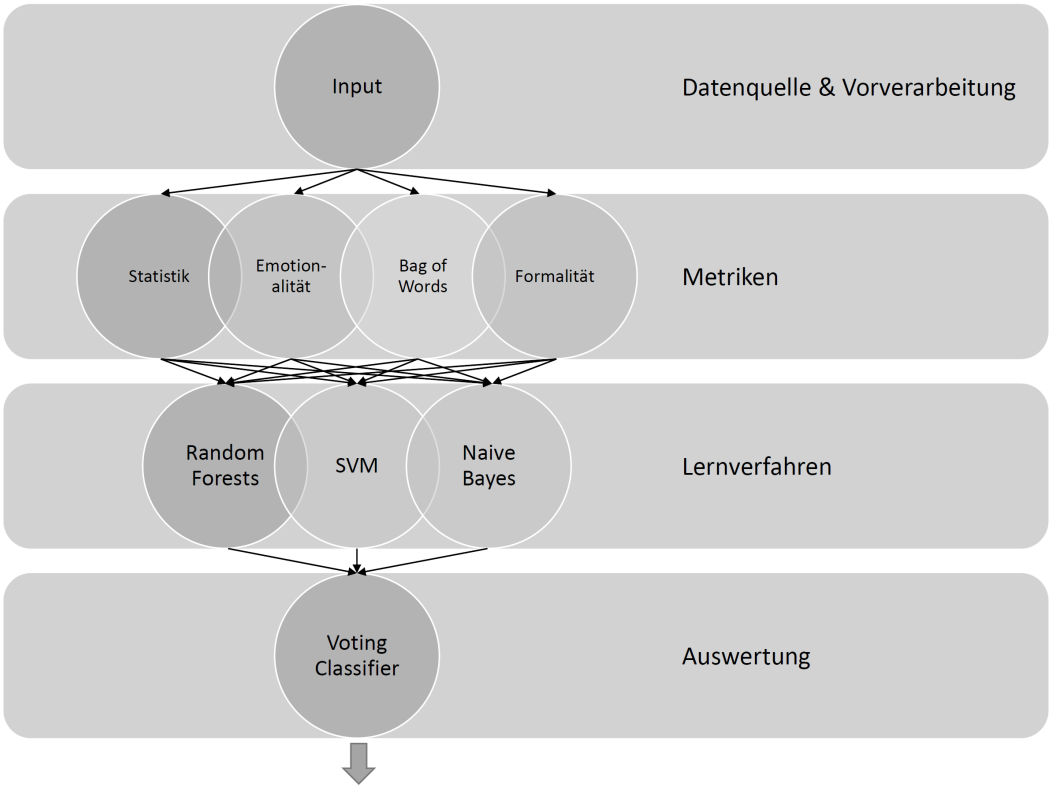

Abbildung 5.5.: Pipeline der Klassifizierung einer Datenquelle

5.5.2. Optimierung des Lernverfahrens

Die Klassifizierung der Sätze führt zu einer Vielzahl von Herausforderungen, wie die einzelnen Analyseschritte zeigen. Besonders die Wahl der richtigen Metriken und die Wahl der optimalen Parameter für die Lernverfahren stellt ein Optimierungsproblem dar. **Evolutionäre Algorithmen** können dabei helfen, eine gute Lösung für dieses Optimierungsproblem zu finden. Aus diesem Grund wird eine einfache Variante eines **Evolutionäre Algorithmus**, wie er in Abbildung 2.1 dargestellt wird, verwendet. Die Anzahl der neuen Populationen wurde auf eine reduziert, sodass ein 1-1 EA verwendet wird. Mit jeder Mutation wird das Hyperparameterset durch einen einfachen Bitflip-Operator mutiert, um

so neue Lösungen zu erzeugen. Wie in Kapitel 2 bereits angesprochen, optimiert ein evolutionärer Algorithmus die Lösung auf Basis einer Fitnessfunktion, die die Güte einer Lösung angibt. In diesem Fall wird die durchschnittliche Genauigkeit bei der Klassifizierung als solche gewählt. Dafür wird für jedes Hyperparameterset der Lernprozess 20 mal mit unterschiedlichen Aufteilungen des Test- und Trainingssets berechnet. Durch die mehrfache Ausführung mit verschiedenen Aufteilungen des Test- und Trainingssets wird eine Crossvalidation durchgeführt.

5.6. Auswertung

Für die Auswertung wird das trainierte Model verwendet, um eingegebene Nachrichten zu klassifizieren. Abschließend werden die Nachrichten über einen Zeitraum zusammengefasst, um die Güte der Konversation zu bestimmen. Die Wahl des Zeitraums kann abhängig von der Frequenz der Nachrichten gewählt werden. Für die Güte der Konversation ist die häufigste durchschnittliche Klasse sowie die Standardabweichung zu bestimmen. Dabei werden die Klassen für die Berechnungen gewichtet. So wird die **neutrale** Klasse mit 0 gewertet, während die **negative** Klasse mit -1 und die **positive** Klasse mit 1 gewichtet werden. Die Weiterverarbeitung variiert stark zu dem gewünschten Verwendungszweck. So könnte gezielt ausgewertet werden, welches Teammitglied sich momentan in einer angespannten Situation befindet, konsequent unangemessen kommuniziert, momentan einen Erfolg verzeichnet oder stets einen sehr freundlichen Umgang pflegt. Neben der Klassifizierung der emotionalen Auswertung können noch die Metadaten, wie die Häufigkeiten der Kommunikation oder die Darstellung der Informationsweitergabe in einem FLOW-Graphen die Auswertungen weiter unterstützen. So können konkretere Erkenntnisse über die Kommunikation im Softwareentwicklungsteam erzielt werden.

6

Evaluation

Zur Validierung des Konzepts wurde eine Studie durchgeführt, in der das Konzept mit den Kommunikationsdaten aus einem Softwareentwicklungsprojekt verarbeitet wurde. In diesem Zusammenhang wird der Prozess der Wahl eines passenden Projekts und die Evaluierung des trainierten Modells erörtert.

6.1. Ziel der Studie

Die Studie verfolgt das Ziel, die Anwendbarkeit des Konzepts im Hinblick auf die in dem Konzept beschriebenen Ansprüche zu zeigen und die Beantwortung der zentralen Forschungsfrage aus der Einleitung zu liefern, in der nach einer Möglichkeit gefragt wird, die Stimmung in einem Softwareentwicklungsteam automatisch auf Basis der Kommunikation im Team abzuleiten. Im Wesentlichen sieht das Konzept vor, eine automatische Auswertung der digitalen textuellen Kommunikation eines Softwareentwicklerteams ohne zusätzlichen manuellen Aufwand des Teamleiters zu ermöglichen. Für diesen Zweck wurde ein Softwareentwicklungsprojekt ausgewählt, mit einer ausreichenden Größe ausgewählt, damit die Studie eine wissenschaftliche Aussagekraft besitzt. Dabei wurde die Erreichung des folgenden Forschungsziels angestrebt, das in Anlehnung an das **Goal-Definition-Template** von Wohlin et al. [42] wie folgt formuliert wurde:

Forschungsziels

Analysiere die Kommunikation im Softwareentwicklerteam
mit dem Ziel, Verbesserungspotenzial aufzudecken,
in Bezug auf die Zusammenarbeit und Kommunikation im Team
aus Sicht eines Teamleiters
im Rahmen von einer Einzelfallstudie.

6.2. Datenherkunft

Bevor mit der Evaluierung begonnen werden konnte, mussten einige Voraussetzungen geschaffen werden. Darunter fällt auch die Suche nach einem geeigneten Entwicklungsprojekt zur Validierung des Konzeptes. Für die automatische Analyse der Telekommunikation ist es unerlässlich, Kommunikationsdaten eines geeigneten Teams zu erhalten. Geeignet für dieses Verfahren sind vor allem Entwicklerteams, die dezentral arbeiten und somit wesentliche Teile der Kommunikation über digitale Medien durchführen. Wie eingangs dargestellt, gibt es bereits einige Vorüberlegungen zu geeigneten Kommunikationskanälen. Nicht jedes der vorgestellten digitalen Medien eignet sich gleichermaßen für die Analyse. Ein besonderer Faktor bei der Analyse ist die Existenz geeigneter Schnittstellen, um die Nachrichten abzurufen. Wesentliche Punkte bei der Analyse der Daten von Telekommunikation sind der Datenschutz, der Geheimnisschutz und der Schutz der Privatsphäre. Aus den genannten Gründen ist das Finden eines geeigneten Projektes zusammen mit geeigneten Kommunikationsmedien und entsprechender Wahrung der schützenswerten Bedürfnisse nicht trivial.

All diese Punkte in einem Softwareentwicklungsteam zu vereinen, stellt eine Herausforderung dar. Die bevorzugte Datenquelle für die Studie ist eine Kooperation mit einem Softwareunternehmen aus der Privatwirtschaft. Die Vorteile gegenüber Projekten aus dem universitären Umfeld sind vielfältig, denn studentische Projekte haben häufig Probleme mit der Kontinuität in der Arbeitsweise und Zusammenarbeit. Dieser Umstand und die Tatsache der geringeren Professionalität würden die Ergebnisse bei der Analyse verzerren. Auch gegenüber den Kommunikationsdaten eines OpenSource-Softwareprojekts, welche frei im Internet verfügbar sind, sind die Kommunikationsdaten aus einem Projekt eines privatwirtschaftlichen Unternehmens vorzuziehen. Somit wurde für diese Arbeit eine Kooperation mit einem Softwareentwicklungsunternehmen eingegangen.

Eine solche Kooperation bringt jedoch zusätzliche Hürden und einen gewissen Mehraufwand mit sich. Neben der Planung von Terminen mit Interessenvertretern des Betriebsrates, des Datenschutzes und des Managements, die erforderlich waren, um Bedenken zu zerstreuen, war die Suche nach einem Unternehmen, welches sich generell bereit zeigte, hochsensible Kommunikationsdaten für die Evaluierung des Konzeptes bereitzustellen, eine Herausforderung. Für die Kooperation fand sich schließlich ein Softwareberatungshaus, welches hohe Anforderungen an den Datenschutz, den Geheimnisschutz sowie den Schutz der Privatsphäre stellt. Zusammen mit dem Unternehmen wurden die Vorgaben für die Kooperation ausgearbeitet.

Eine wesentliche Vorgabe des Unternehmens war es, dass sämtliche Teammitglieder über ein Transparenzschreiben[5] informiert werden. In diesem Schreiben wurden sowohl das generelle Konzept als auch die Bedingungen der Zusammenarbeit sowie der Verarbeitung der Kommunikationsdaten des Entwicklerteams vorgestellt. Die Teammitglieder hatten zwei Wochen lang die Möglichkeit, Beden-

[5]Das Transparenzschreiben wurde der Arbeit als Anlage beigefügt.

ken, Anregungen und Widersprüche zu äußern. Keines der Teammitglieder hat dem Vorhaben widersprochen. Hätte ein Teammitglied widersprochen, so wären sämtliche Konversationen, an denen es beteiligt war, von der Analyse ausgenommen worden. Teil dieses Transparenzschreibens war die Zusicherung, dass keine Ergebnisse zum Nachteil einzelner Personen ausgelegt werden. Zusätzlich war eine wesentliche Bedingung seitens des Unternehmens, dass sämtlicher Zugriff auf die Kommunikation nur innerhalb der Geschäftsräume des Unternehmens stattfindet, die Daten nur auf hauseigenen Systemen verarbeitet werden und Kommunikationsdaten sowie die Ergebnisse nur nach abschließender Moderation exportiert werden dürfen. Die in dieser Arbeit dargestellten Ergebnisse wurden im Nachgang der Analyse nicht bearbeitet oder angepasst. Die Moderation seitens des Unternehmens erfolgte nicht mit dem Ziel, ein möglichst glorifizierendes Bild des Unternehmens darzustellen, sondern diente dazu, die Rechte der Mitarbeiter und Interessen des Unternehmens zu wahren, insbesondere den Geheimnisschutz. Aus diesen Gründen wurde auch der Veröffentlichung des gesamten Datensatzes widersprochen. Dem Erstprüfer wurde zur Zertifizierung der Ergebnisse allerdings der gesamte Datensatz bei der Abgabe übergeben.

6.2.1. Informationen zum Entwicklerteam und dem Projekt

Das für die Kooperation gewonnene Unternehmen bietet für die Evaluierung des Konzeptes eine Zugriffsmöglichkeit auf das Kommunikationstool, das in dem für diese Arbeit betrachteten hauseigenen Entwicklungsprojekt eingesetzt wird. Nach Auskünften des Unternehmens arbeiten bis zu 80 Mitarbeiter, die in Standorten deutschlandweit verteilt sind, an dem Softwareprojekt. Somit sind durch die Größe des Projektteams, die Tatsache der dezentralen Arbeit und dem Einsatz eines digitalen Kommunikationsmediums die Voraussetzungen an das Entwicklerteam, die durch das Konzept skizziert wurden, erfüllt. Das genutzte digitale Kommunikationsmedium ist das OpenSource-Projekt **Zulip**. **Zulip** ermöglicht die Einrichtung verschiedener Kommunikationsstreams, die thematisch gegliedert werden können.

Neben der Verwendung des Tools **Zulip** sind in dem Projekt für die Kommunikation (abgesehen von analogen Bürogesprächen und den Meetings) auch noch folgende Kommunikationsmedien in Verwendung: Telekommunikation via Telefon, die Kommunikation per Mail und das Chattool Jabber[6]. Als offizielles digitales Kommunikationstool ist jedoch **Zulip** im Einsatz.

6.2.2. Zulip

Das Chattool **Zulip**[7] wurde bereits kurz angeschnitten. Allerdings ist es entscheidend, einen kurzen Eindruck in die Bedienung und Datenhaltung des Programms zu bekommen, um ein besseres Verständnis der Kommunikationsdaten zu erhalten. Daher folgt eine kurze Einführung in **Zulip**.

[6] Jabber ist ein Instant Messaging Tool. Nähere Informationen unter https://www.jabber.de
[7] Das Programm **Zulip** ist unter https://zulipchat.com verfügbar.

Zulip ist zu 100% eine OpenSource-Software, welche von der „Vibrant-Community“ mit mehreren hundert, über die ganze Welt verteilten Entwicklern entwickelt wird. Es gibt verschiedene Lizenzmodelle, die kostenpflichtig sind. Allerdings wird die Software kostenlos zum Download angeboten und kann auf einem eigenen Server kostenlos betrieben werden. **Zulip** selbst hebt die eigene Dokumentation mit mehr als 120.000 Wörtern als umfangreich hervor[8].

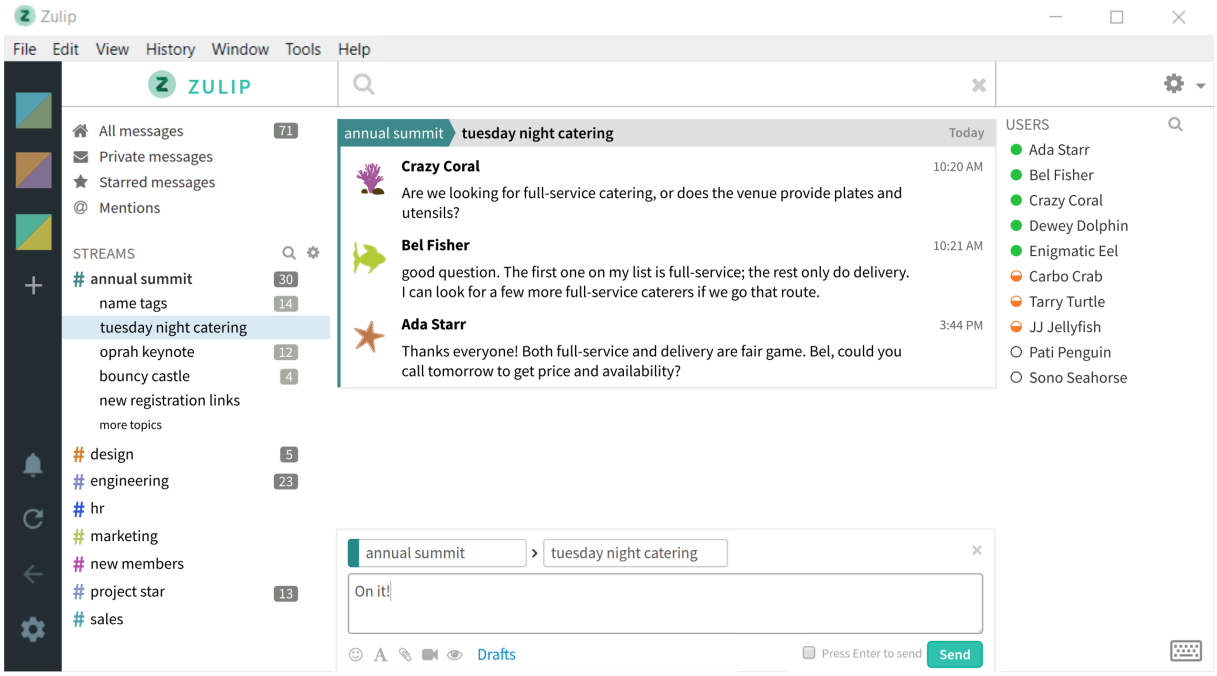

Abbildung 6.1.: Darstellung der Oberfläche des Windowstools von Zulip (Abgerufen von der https://zulipchat.com/apps/windows am 04.09.2019)

Zulip bietet Gruppenchats (sogenannte Streams) an, die thematisch gegliedert werden können. So kann für jedes größere Feature ein Stream angelegt werden. Ein Nutzer von **Zulip** hat die Möglichkeit selbst zu entscheiden, welche Streams er abonnieren möchte. Diese Vorgehensweise hat den Vorteil, dass der Nutzer nicht mit für die eigene Arbeit unnötigen Informationen überflutet wird. Zusätzlich hat jeder Stream verschiedene Topics, die dazu dienen, Informationen effektiver zu filtern und zielorientierter zu selektieren. Sobald ein Nutzer einen Stream nicht mehr benötigt, kann er das Abonnement kündigen.

In der Abbildung 6.1 wird die zum Zeitpunkt des Abrufs aktuelle Version des Windowsclients des Chattools dargestellt. Dabei ist bewusst auf eine Darstellung des unternehmensinternen Zulips verzichtet worden, da dieses nicht ohne eine Schwärzung wesentlicher Stellen möglich wäre. Die Darstellung und Verwendung gleicht der dargestellten Ansicht und ist somit für die exemplarische Beschreibung hinreichend geeignet. In der rechten Spalte sind die einzelnen Streams aufgelistet, was an den führenden Doppelkreuzen zu erkennen ist. In einem Stream kann es verschiedene Topics geben. In der

[8]Quelle: https://zulipchat.com

Abbildung sind dies beispielsweise der Stream „annual summit“ und das Topic „tuesday night catering“. Über die Selektierung der Topics können die Nachrichten entsprechend gefiltert werden. In der mittleren Spalte ist das Nachrichtenfenster zu sehen, wie es von einem Chattool erwartet wird. Die rechte Spalte stellt die Teammitglieder und ihren aktuellen Status dar.

6.2.2.1. REST-API

Das Kommunikationstool **Zulip** bietet eine REST-API[9] als Schnittstelle an. Diese kann über die Python-Library einfach angesprochen werden. Mittels der Schnittstelle ist ein automatisierter Export der Daten möglich. Für die Verwendung der Schnittstelle ist es lediglich notwendig, über die grafische Benutzeroberfläche einen API-Key zu generieren. Dieser kann auch als Datei exportiert und direkt in Python integriert werden. Die REST-API gliedert sich, wie in der nachstehenden Liste zu sehen ist, in mehrere Bereiche, von denen für diese Arbeit nur *Messages* relevant ist.

- *Messages*
- *Streams*
- *Users*
- *Server & organizations*
- *Real-time events*

Über folgende Anfrage an den **Zulip**-Server werden die letzten 1000 Nachrichten seit dem letzten Abrufen und die nächsten 1000 zurückgegeben:

Listing 6.1: Request zum Abrufen der Nachrichten eines Streams

```
{
        'use_first_unread_anchor': True,
        'num_before': 1000,
        'num_after': 1000,
        'narrow': [
                {
                'operator': 'stream',
                'operand': stream
                }
        ],
        'client_gravatar': False,
        'apply_markdown': False
}
```

Als Rückgabe liefert die REST-API ein JSON-Objekt, welches die Attribute aus der Tabelle 6.1 enthält.

Das entscheidendste Attribut in der Rückgabe ist *messages*. In diesem sind die Nachrichten gekapselt. Jede einzelne Nachricht wird in dem *messages* Array als JSON-Objekt repräsentiert und enthält die in Tabelle 6.2 dargestellten relevanten Attribute.

In dem untersuchten Entwicklungsprojekt wurden für die einzelnen Entwicklungskomponenten eigene

[9] Eine ausführliche Dokumentation der REST-API von **Zulip** ist unter https://zulip.com/api/rest verfügbar.

Tabelle 6.1.: Datenfelder der Antwort für das Abrufen der Nachrichten

Attributname	Bedeutung
anchor	Nachrichten ID
found_newest	Gibt an, ob die neuste gefilterte Nachricht mit übergeben wurde.
found_oldest	Gibt an, ob die älteste gefilterte Nachricht mit übergeben wurde.
found_anchor	Gibt an, ob die Anchor Nachricht mit übergeben wurde.
messages	Ein Array mit allen gefilterten Nachrichten.

Tabelle 6.2.: Relevante Attribute einer Zulipnachricht

Attributname	Bedeutung
content	Inhalt der Nachricht
stream_id	Der Stream in dem die Nachricht geschrieben wurde.
timestamp	Zeitpunkt, zu dem die Nachricht gesendet wurde.

Streams in **Zulip** eingerichtet. Neben diesen Entwicklungskomponenten wurden aber auch Streams für die private Kommunikation, wie beispielsweise die Planung der Mittagspause, für die Mitarbeiter erstellt. **Zulip** bietet eine Rest-API an, über die Kommunikationsdaten exportiert werden können.

6.2.2.2. Datenexport

Über die REST-API lassen sich die Nachrichten eines Streams exportieren. Die wohl wichtigsten Schritte in der Verarbeitung der privaten und geschäftlichen Korrespondenz stehen jedoch erst nach dem Export an. Dabei handelt es sich um die Wahrung des Datenschutzes und Geheimnisschutzes sowie des Schutzes der Privatsphäre. Besonders in einem vom Wettbewerb geprägten Markt sind Geschäftsgeheimnisse zu schützen. Dazu wurden im Rahmen dieser Arbeit umfängliche Abstimmungsarbeiten mit Interessenvertretern, Datenschutzbeauftragten und Projektverantwortlichen angestoßen. Diese führten zu dem Ergebnis, dass neben der bereits erwähnten umfänglichen Aufklärung der Betroffenen über das Transparenzschreiben eine Anonymisierung der Kommunikationsdaten erfolgen muss. Außerdem ist die Möglichkeit zu schaffen, Ergebnisse der Auswertung nur über Streams innerhalb eines Zeitraumes zu erzielen. Darüber hinaus sollte die Verarbeitung der Kommunikationsdaten ausschließlich in den Räumen des Unternehmens stattfinden und der Anonymisierungsprozess sowie der Export von Daten aus dem Unternehmen für die Publizierung in der Arbeit durch das Unternehmen selbst moderiert werden. Nach Festlegen dieser Bedingungen wurde direkt mit dem Export der Daten begonnen. Somit wurden die Kommunikationsdaten, auf denen die weitere Analyse fußt, am 24.07.2019 um 13 Uhr, über das Python-Export-Script[10] exportiert. Es wurden Nachrichten aus fünf Streams exportiert, sodass insgesamt 1947 Nachrichten aus der **Zulipinstanz** des Entwicklerteams

[10]Das Skript für den Export der Daten wie auch alle anderen Programmstücke werden der Arbeit auf einer CD beigefügt.

extrahiert wurden. Die 1947 Nachrichten wurden automatisch von einem Python-Script in 7070 Sätze aufgeteilt. An den exportierten Konversationen waren 65 Mitglieder des Projekts aktiv beteiligt. Mit aktiver Beteiligung sind diejenigen Personen gemeint, die in dem genannten Zeitrum mindestens eine Nachricht geschrieben haben. Es wurde an 107 Tagen im Zeitraum vom 11.02.2019 bis zum 24.07.2019 kommuniziert.

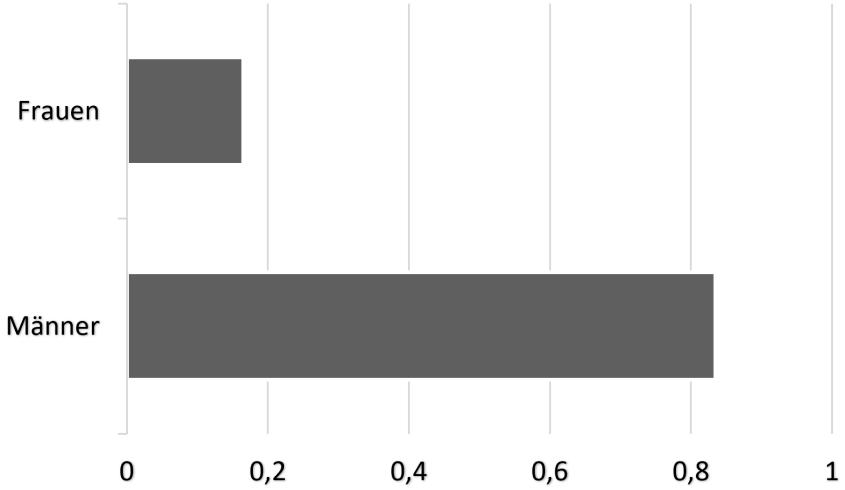

Abbildung 6.2.: Die Geschlechterverteilung in dem untersuchten Softwareentwicklungsteam

Wie in der Abbildung 6.2 zu sehen, ist die Geschlechterverteilung unausgeglichen. Wie in der Domäne der Softwareentwicklung häufig anzutreffen, ist das feminine Geschlecht im Vergleich zum maskuline Geschlecht unterrepräsentiert. Die Kenntnisse über die Zusammensetzung der Entwicklerteams könnten eine Verzerrung in der Kommunikation erklären.

Um die Rechte der Personen und des Unternehmens zu wahren, werden die Kommunikationsdaten nicht veröffentlicht. Die Rohdaten haben im Zuge der Analyse nie die Systeme des Unternehmens verlassen. Erst nachträglich wurden die Daten für die Abgabe der Arbeit exportiert.

Die nachfolgenden Verarbeitungsschritte bauen auf den exportierten Kommunikationsdaten auf.

6.3. Datenaufbereitung

Nach dem Export der Kommunikationsdaten ist deren Aufbereitung für die Analyse notwendig. Dies schließt die Bereinigung der Kommunikation von unrelevanten Daten sowie den Prozess des Aufteilens der Nachrichten in einzelne Sätze und die Anonymisierung der Sätze ein. Abschließend ist noch das Labeln der getrennten Nachrichten erforderlich. Nach dem Abschluss der Aufbereitung der Daten befinden sich diese damit in einem Format, das es möglich macht, das in der Umsetzung beschriebene Verfahren anzuwenden und damit die eigentliche Validierung des Analyseverfahrens durchzuführen. Hierfür werden die Nachrichten zunächst bereinigt, exploriert und sprachlich korrigiert.

6.3.1. Bereinigung der Daten

Nachrichten, die aus dem Datenexport exportiert werden, liegen in einem Format vor, das noch viele unerwünschte und unnötige Zeichenketten enthält. Durch Senden des Parameters *'apply_markdown': False* in der Anfrage an den **Zulip**-Server werden die Nachrichten nicht im HTML-Format übertragen, sondern in der vereinfachten Auszeichnungssprache **Markdown**[11]. **Markdown** ist eine beliebte Auszeichnungssprache, die in vielen Programmen Anwendung findet, darunter Gitlab, Confluence sowie auch bei vielen Entwicklerdokumentationen. Der Vorteil von **Markdown** besteht darin, dass ein in **Markdown** geschriebener Text leicht zu lesen ist. Dies liegt im Vergleich zur Sprache HTML an der einfachen Struktur von **Markdown**. Dennoch enthält eine in **Markdown** geschriebene Nachricht noch einige Zeichen, die für das entwickelte Analyseverfahren nicht relevant sind. Die folgende Nachricht wurde aus den Originalnachrichten aus dem Zulipexport abgeleitet, um keine der zuvor genannten Interessen zu verletzen, und soll diese Formatierungen beispielhaft verdeutlichen.

> Hallo @**Max Mustermann**,
> ich habe für **dich** die Berscheibung des Fehlers in der Dokumentation im [Confluence][https://confluence.musterunternehmen.de] ergänzt.
> Du hättest statt "private String SEHR_WICHTIG = ÄBC;" folgendes schreiben müssen: "private static String SEHR_WICHTIG = ÄBC;"
> Für das nächste Mal kannst du in der Doku nachlesen ;)

Wie schon in der Grußformel der Nachricht zu sehen ist, wird der Name über eine Mention-Funktion aufgerufen. Die Mention-Funktion dient dazu, Nutzer in einer Nachricht direkt anzusprechen oder zu verlinken. Der Pre- und Postfix sind störend für die Analyse des Namens und müssen somit entfernt werden. Zusätzlich muss der erwähnte Name noch anonymisiert werden. Die Verfahren der Anonymisierung wird im Abschnitt zur Anonymisierung näher erläutert.

Neben der Notation für die Mention-Funktion sind in der beispielhaften Nachricht noch Formatierungen enthalten. Durch die Eingrenzung eines Wortes mit den „**" wird dem Interpreten der **Markdown**-Sprache mitgeteilt, dass dieses Wort fettgedruckt werden soll. Neben dem doppelten Stern gibt es noch weitere Formatierungsanweisungen in **Markdown**. Diese müssen für die Anwendungen der Metriken aus den Nachrichten entfernt werden. Selbiges muss ebenfalls für die Erwähnung von Code geschehen und konsequenterweise auch für die Anonymisierung von URLs. Sollte ein Satz nach der Bereinigung der Sätze leer sein oder nur noch einen Namen enthalten, so wird dieser aus der Liste der zu verarbeitenden Sätze entfernt. Die übrigen Sätze können den weiteren Verarbeitungsschritten zugeführt werden.

[11] Eine Beschreibung der Sprache ist unter https://markdown.de/ zu finden

6.3.2. Datenexploration

Um die Kommunikationsdaten zu explorieren, werden neben dem offensichtlichen Lesen der Nachrichten und dem Versuch, manuell Auffälligkeiten zu erkennen, auch noch andere Darstellungsweisen gewählt. Zur Einordnung der Inhalte der Nachrichten eigenen sich Wordclouds besonders. Mit Wordclouds ist die Darstellung von Wörtern in einer Grafik gemeint, wobei die angezeigten Wörter und deren Größe und Farbe abhängig von deren Vorkommen ist. So werden sehr häufig vorkommende Wörter größer dargestellt. Auf diese Weise ist es möglich, zentrale Begriffe leicht zu erkennen und auch ohne sämtliche Nachrichten gelesen zu haben, einen Überblick über den Inhalt zu bekommen.

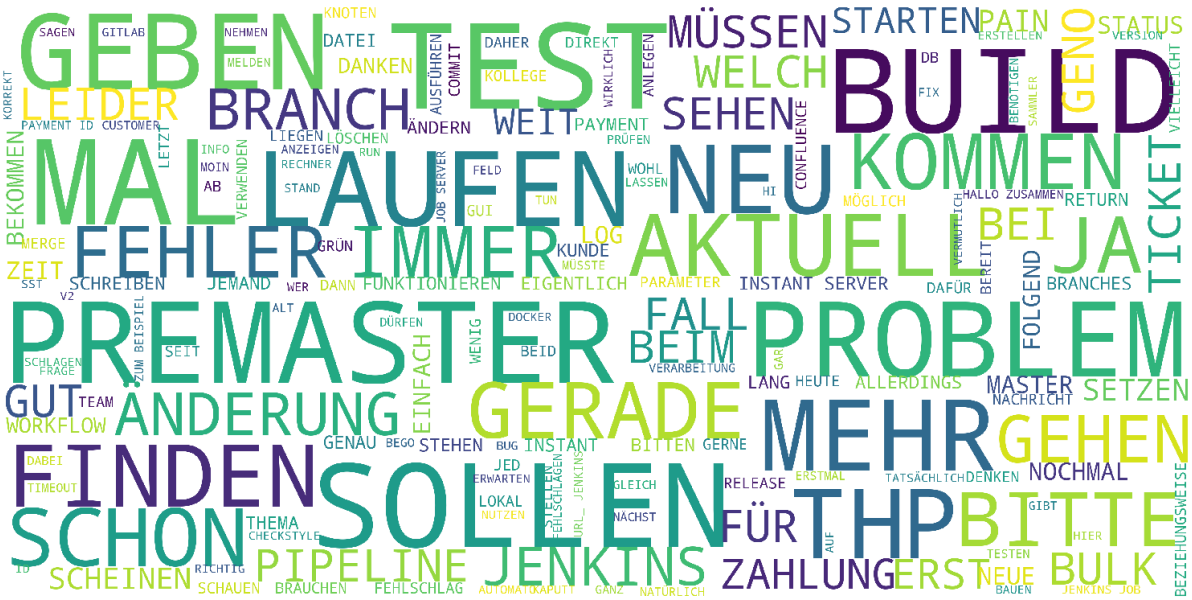

Abbildung 6.3.: Wordcloud der Kommunikation im Entwicklerteam

Wie in der Abbildung 6.3 zu sehen ist, stehen die Begriffe **TEST**, **BUILD**, **LAUFEN**, **PREMASTER** und **PROBLEM** im Vordergrund. In einem modernen Softwareentwicklungsprojekt sind diese Begriffe aus dem Entwicklungsalltag nicht wegzudenken. Besonders das Testen und das Lösen von Problemen sind alltäglich in der Arbeit in einem Softwareentwicklungsprojekt. Im Hintergrund treten auch viele spezielle Ausdrücke aus der Softwareentwicklung wie **JENKINS**, **CONFLUENCE**, **MASTER** und **SERVER** auf. Zweifelsfrei wird sehr themenbezogen kommuniziert, wie an der Verwendung der Fachbegriffe zu sehen ist. Trotz des häufigen Auftretens von Begriffen wie **FEHLER** und **PROBLEM** sind Begriffe wie **DANKE** und **BITTE** auch hervorgehoben.

6.3.3. Sätze bilden

Wie bereits angesprochen, ist das Aufteilen der Nachrichten, welche als String vorliegen, in die einzelnen Sätze für eine genauere Analyse notwendig. Denn in einer Nachricht können die einzelnen

Sätze durchaus verschiedene Aussagen und Stimmungen haben. Der Anfang und das Ende eines Satzes gehen nicht immer konsequenterweise aus der Interpunktion hervor, da eine Nachricht auch Emoticons und ähnliches enthalten kann. **Spacy** ermöglicht neben der Analyse von Sätzen auch die Untersuchung ganzer Textdokumente. Für diese Funktion hat **Spacy** die Möglichkeit integriert, Sätze entsprechend der sprachlichen Struktur zu trennen. Da **Spacy** mit einem trainierten Modell im Hintergrund arbeitet, ist die Erkennung der **Part-Of-Speech** und die Teilung der Sätze nicht fehlerfrei. Dennoch wurde der überwiegende Teil korrekt getrennt. Die Sätze, bei denen die automatische Trennung nicht funktioniert hat, werden ebenfalls als vollständige Sätze angesehen und weiterverarbeitet, um dem Anspruch der Automation gerecht zu werden. Wie an den Beispielsätzen in Tabelle 6.3 und Tabelle 6.4 zu sehen ist, gibt es insbesondere Probleme mit der Trennung der Sätze, wenn Sätze als Zitat eingeschoben oder Wörter durch Satzzeichen hervorgehoben werden. Weitere Probleme bei der Trennung der Sätze ergeben sich bei der Verwendung von Links oder Dateipfaden.

Tabelle 6.3.: Beispiel für eine erfolgreiche Nachrichtentrennung

Nr.	getrennter Satz
1.	Bitte habt auch einen Augenmerk auf IZV/AZV.
2.	Für die AHCA werden auch PAINs über den Job-Server eingelesen.
3.	Es gibt da auch schon das Ticket indem es um die verschiedenen Konfigmöglichkeiten bzgl. Dopplererkennung an der Schnittstelle geht.

Wie das obige Beispiel zeigt, wurde die Nachricht entsprechend der syntaktischen Trennung richtig getrennt. Im unteren Beispiel wurde hingegen die Nachricht nicht korrekt getrennt, da die Interpunktion anspruchsvoller ist.

Tabelle 6.4.: Beispiel für eine schlechte Nachrichtentrennung

Nr.	getrennter Satz
1.	Das Dokument BulkObject Mapping
2.	2.3.xlsx findet ihr unter:
3.	W:/Datenmodell UTO, UMO, UBO

Um dem Anspruch eines automatischen Exports gerecht zu werden, wurden die falsch getrennten Sätze ebenfalls in die Analyse integriert, da eine automatische Detektion bezüglich einer korrekten Satztrennung nicht zuverlässig vorgenommen werden kann. Für die weitere Verarbeitung wird nun nicht mehr auf der Nachrichtenebene, sondern auf der Satzebene analysiert.

6.3.4. Sprachliche Autokorrektur

Für die korrekte Anwendung der Metriken ist es relevant, dass die Wörter in den Sätzen sprachlich gesehen korrekt geschrieben wurden. Denn für die Lemmatisierung der einzelnen Wörter ist es notwendig, die angesprochene sprachliche Richtigkeit zu wahren. Die Wahrung der sprachlichen Korrektheit gelingt über die Bibliothek **pyspellchecker**, die mit der Levenshtein-Distanz zum deutschen und englischen Wörterbuch arbeitet und damit Vorschläge für die falsch geschriebenen Wörter macht. Außerdem kann die Bibliothek entscheiden, ob ein Wort richtig geschrieben wurde. Insbesondere für die Erkennung der Emotionalität über die Textkorpora von Waltinger [40] und Remus et al. [28] ist die Prüfung auf die sprachliche Richtigkeit und die darauf aufbauende Lemmatisierung entscheidend. In den Textkorpora von Waltinger [40] und Remus et al. [28] stehen die Wörter in der lemmatisierten Form.

6.3.5. Anonymisierung

Ein wesentlicher und sehr entscheidender Punkt bei der Vorverarbeitung der Rohdaten war die Anonymisierung der Kommunikationsdaten. Dabei wurden sämtliche Namen und Adressen durch Pseudonyme ersetzt. Hierfür mussten die Nachrichten händisch verändert werden. Ein Mapping von Pseudonym zu dem eigentlichen Namen wurde nicht gespeichert. Zwischenergebnisse und Erzeugnisse wurden keinem Mitglied des Managements vorgelegt und nach Abschluss der Arbeit vernichtet. Somit soll sichergestellt werden, dass die Interessen der Mitarbeiter gewahrt werden. Wie in dem Abschnitt Bereinigung der Daten zu sehen ist, können Namen von Personen nicht nur als Absender und Empfänger in den Metadaten auftreten, sondern auch direkt im Text. Diese zu entfernen erfordert einen hohen manuellen Aufwand. In der praktischen Anwendung des Analyseverfahrens würden die Kommunikationsdaten ohne menschliches Zutun verarbeitet werden, weshalb dieser händische Schritt des Anonymisierens entfallen könnte. Der Algorithmus würde auch bei nicht anonymisierten Kommunikationsdaten niemanden diskriminieren oder ein Teammitglied verurteilen. Für die Verwendung solcher schützenswerter Daten ist, insbesondere wie in diesem Fall der wissenschaftlichen Nutzung, die Anonymisierung nicht wegzudenken.

6.3.6. Labeln von Daten

Nach der Vorverarbeitung der Nachrichten zu Sätzen, der Bereinigung und der Anonymisierung ist es zwingend erforderlich, die Sätze für die Klassifizierung einer Klasse zuzuordnen. Dazu wurden die Daten in fünf Kategorien unterteilt **sehr positiv**, **positiv**, **neutral**, **negativ**, **sehr negativ**. Um ein besseres Verständnis dieser Klassen zu bekommen, sind im Folgenden einige Beispiele aufgeführt, welche Gefühle der jeweiligen Klasse zuzuordnen sind:

Tabelle 6.5.: Emotionen den Labelklassen zugeordnet

sehr positiv	positiv	neutral	negativ	sehr negativ
Glück	Freude	Ausdrucksloses	Trauer	Angst
Liebe	Anerkennung	Tatsachen	Scham	Wut
Überraschung	Neugier	Fakten	Reue	Hass
Faszination	Stolz		Enttäuscht	Ärger
Euphorie	Sympathie		Entmutigt	Ekel

Bewusst wurde eine sehr grobe Klassifizierung der Daten vorgenommen, da der Vorgang der händischen Klassifizierung ein sehr subjektiver Vorgang ist. Die Empfindung der Gefühle ist stark abhängig von dem Betrachter und seinem Gemütszustand. Um diese Einflussfaktoren bei der Klassifizierung möglichst gering zu halten, erfolgte eine Beschränkung auf diese fünf Klassen. Allerdings wurde nach

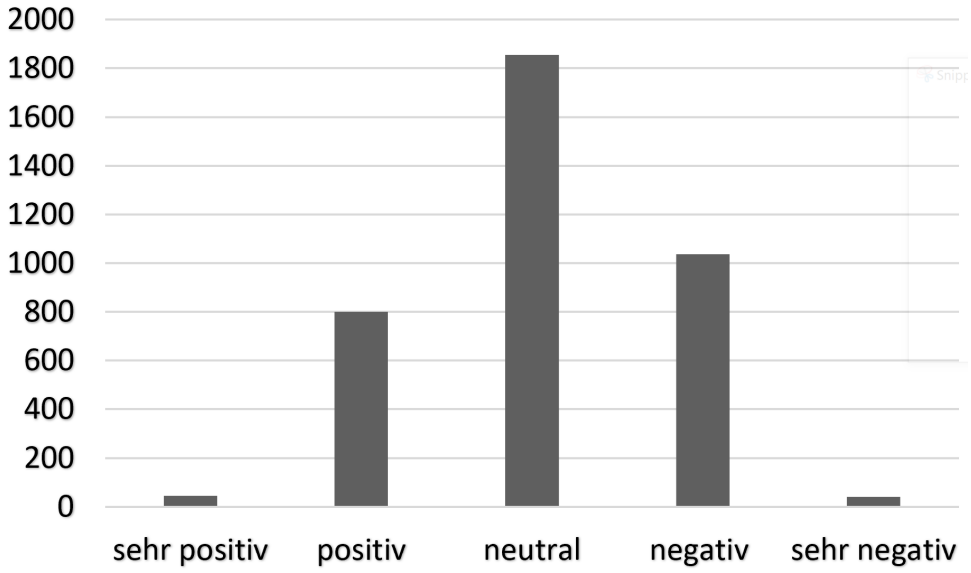

Abbildung 6.4.: Verteilung der fünf Klassen in den Kommunikationsdaten

dem Labeln der Daten eine Verteilung der Klassen erreicht, die einer Normalverteilung ähnelt, wie in der Abbildung 6.4 zu sehen ist. Festzustellen ist, dass die Klasse **neutral** mit 1856 Sätzen wesentlich häufiger vorkommt, als die anderen Klassen. Besonders auffällig ist, dass die Klassen **sehr positiv** (45) und **sehr negativ** (41) deutlich unterrepräsentiert sind. Diese Unterrepräsentation führt dazu, dass es für die Lernverfahren nicht möglich ist, das Modell ausgewogen zu trainieren, wodurch die Erkennung von **sehr positiv** und **sehr negativ** nicht akkurat durchgeführt werden konnte. Das weniger häufige Auftreten der Extrema ist jedoch in diesem professionellen Umfeld nicht anders zu erwarten gewesen. Daraus ergibt sich die Konsequenz, dass die Klassen **sehr positiv** und **sehr negativ** in die Klassen **positiv** und **negativ** eingegliedert werden.

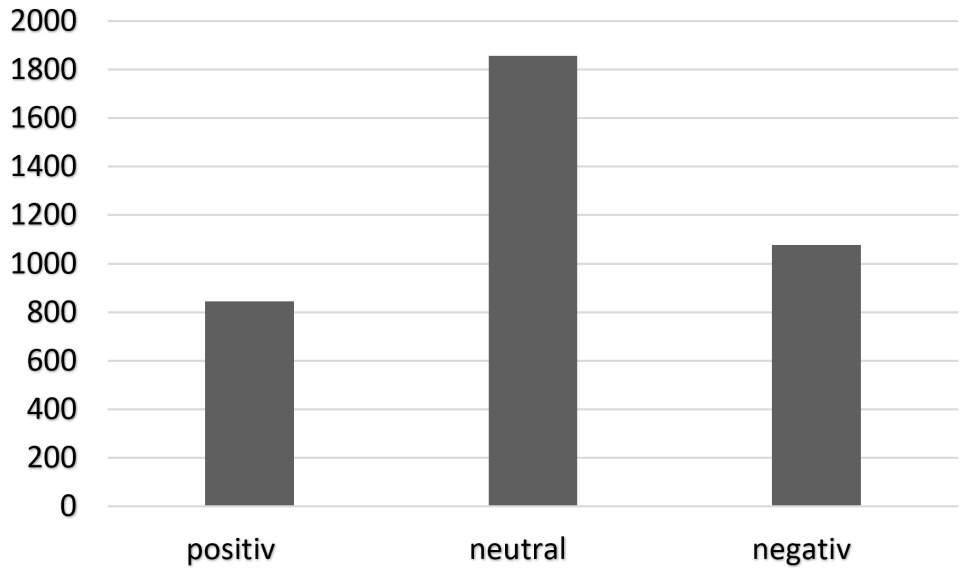

Abbildung 6.5.: Verteilung der drei Klassen in den Kommunikationsdaten

Dies hat, wie in Abbildung 6.5 zu sehen ist, den Effekt, dass die Verteilung der Klassen sich wesentlich geändert hat. Nun sind die drei Klassen ausbalancierter. Die Klasse **neutral** kommt 1856 mal vor, während **positiv** (845) und **negativ** (1077) etwas seltener vorkommen. Ein ausgeglichenerer Datensatz wäre eine bessere Voraussetzung für das Training der Lernverfahren, da das Lernverfahren die die Klasse **neutral** durch deren höhere Präsenz bevorzugen könnte. Unabhängig von der Verteilung der Klassen taucht ein weiteres Problem auf, das durch das Labeln der Kommunikationsdaten entstanden ist. Denn, wie aus der Vereinbarung mit dem Unternehmen hervorgeht, ist die Verwendung und Verarbeitung der Daten ausschließlich mit Vorbehalten auf dem System des Unternehmens zulässig. Somit war es nicht möglich, die manuelle Klassifizierung durch eine weitere Person validieren zu lassen. Wie eingangs erwähnt, ist das Labeln der Daten sehr von der durchführenden Person und dem Kontext abhängig. Aus diesem Grund wurde ein zufällig zusammengestellter Datensatz mit 200 Sätzen mit einem zeitlichen Abstand von einem Monat ein weiteres Mal gelabelt und mit den vorherigen Ergebnissen verglichen. Im Allgemeinen geschehen Fehler bei der Klassifizierung. Das Ausmaß der gemachten Fehler gibt die Güte der Klassifizierung an. Auch das händische Zuordnen eines Satzes zu einem Label stellt eine Klassifizierung dar und birgt dementsprechend die Gefahr von Fehlern. Um die Güte einer Klassifizierung darzustellen, werden die Ergebnisse in einer Konfusions-Matrix dargestellt. Bei dieser Darstellungsform sind die tatsächlichen Label auf der Vertikalen aufgetragen und die vorhergesagten Label durch die Klassifizierung auf der Horizontalen. Die händische Validierung der ursprünglich festgelegten Label ist in der Tabelle 6.6 zu sehen. Die Label der Konfusions-Matrix sind dabei stets symmetrisch. Daher sind auf der Diagonalen die richtig zugeordneten Klassen abzulesen.

Tabelle 6.6.: Konfusions-Matrix auf dem Validierungsdatensatz mit fünf Klassen

	sehr positiv	**positiv**	**neutral**	**negativ**	**sehr negativ**
sehr positiv	0	3	0	0	0
positiv	3	24	14	4	0
neutral	0	12	71	12	0
negativ	0	1	21	32	1
sehr negativ	0	0	1	0	1

Die Zeilen und Spalten folgen dabei in dieser Arbeit stets der folgenden Beschriftung: **sehr positiv**, **positiv**, **neutral**, **negativ** und **sehr negativ**. Vergleicht man die Zeilensumme (die Diagonale ausgenommen), so ist diese in den ersten drei Klassen (**negativ**, **neutral** und **positiv**) stets kleiner als die der Diagonalen. Somit wurde die händische Klassifizierung überwiegend korrekt durchgeführt. Lediglich bei den Labeln **sehr negativ** und **sehr positiv** war diese Klassifikation nicht erfolgreich. Diese Klassen sind allerdings auf Grund des seltenen Auftretens nicht repräsentativ. So kommen unter den 200 Sätzen lediglich zwei **sehr negativ** und drei **sehr positiv** Label vor. Die Klasse **negativ** war 55 mal repräsentiert, **neutral** 95 mal und **positiv** 45 mal. Trotz der zuvor selbst vorgenommenen Klassifizierung der Sätze sind die Abweichungen bei der Validierung beachtlich. So wurden 21 **negativ** Sätze dem Label **neutral** und einer dem Label **positiv** zugeordnet. Dies ist ein Indiz für die Herausforderung, die das Labeln der Sätze mit sich bringt. Es scheint eine hohe Verwechslung zwischen den Klassen **negativ** und **positiv** mit der Klasse **neutral** zu geben. Ein weiteres Maß für die Güte einer Klassifizierung ist die Accuracy (Genauigkeit). Die Accuracy ist ein Maß für die im Durchschnitt richtigen Klassifikationen. Bei der Vorlage der Konfusions-Matrix ist die Accuracy die Summe der Diagonalen geteilt durch die Anzahl der Sätze: $\frac{0+24+71+32+1}{200} = 64\%$. Wie bereits angesprochen, wurde auf die Verwendung der Klassen **sehr negativ** und **sehr positiv** verzichtet, da diese sehr unterrepräsentiert sind. In der Tabelle 6.7 ist die Konfusions-Matrix unter Berücksichtigung der Zusammenfassung der Label **sehr negativ** und **sehr positiv** dargestellt.

Durch die Anpassung und Zusammenführung der Label ergibt sich eine Genauigkeit von 66%, die damit zwei Prozentpunkte besser ist als ohne die Zusammenführung. Die Ungenauigkeiten beim Labeln können durch mehrere Aspekte erklärt werden. Zum einen ist das Labeln sehr individuell von einer Person abhängig, sodass die Stimmung und der Kontext sehr großen Einfluss auf die Bewertung der Kommunikation haben können. Neben diesen Faktoren spielt insbesondere die Ähnlichkeit einer neutralen, positiven oder negativen Nachricht eine entscheidene Rolle. Zusätzlich wurde im Gegensatz zum initialen Vorgang des Labelns ein zufälliges Sample aus dem gesamten Datensatz verwendet, sodass die Kontextsensitivtät der Nachrichten bei dem Labeln des Validierungsdatensatzes nicht

zum Tragen kam. Insbesondere die Personenabhängigkeit könnte durch die Verwendung mehrerer Personen, die dieselben Daten labeln, aufgelöst werden, um so eine Crossvalidation zu ermöglichen. Leider war dies, wie bereits erwähnt, in dieser Arbeit nicht möglich.

Tabelle 6.7.: Konfusions-Matrix auf dem Validierungsdatensatz mit drei Klassen

	sehr positiv	**positiv**	**neutral**	**negativ**	**sehr negativ**
sehr positiv	0	0	0	0	0
positiv	0	30	14	4	0
neutral	0	12	71	12	0
negativ	0	1	22	34	0
sehr negativ	0	0	0	0	0

6.4. Lernen des Modells

Nach dem Labeln der Nachrichtenteile folgt der Schritt, die Metriken aus Abschnitt 5.4 auf die Nachrichtenteile anzuwenden. Durch die Anwendung der Metriken werden Merkmale aus den Sätzen generiert, die einem Lernverfahren helfen sollen, die Sätze zu klassifizieren. Insgesamt wurden 5378 Merkmale aus jedem Satz extrahiert. Eine Vielzahl der Merkmale beruht auf den Metriken aus dem Abschnitt Bag of Words. Diese *Bag of Words*-Merkmale repräsentieren die Wörter in den Sätzen. Ohne die Zunahme der *Bag of Words*-Merkmale bleiben lediglich noch 28 Merkmale, die extrahiert wurden. Durch die Limitierung der Hardware benötigte das Extrahieren der Merkmale durch die Metriken mehr als zwölf Stunden. Für das Lernen des Modells sind Metriken erforderlich, die besonders aussagekräftig sind und eine geringe Überschneidung besitzen. Dies ermöglicht es dem Modell, besser zu differenzieren und mehr Entscheidungsgrundlagen zu haben. Somit sind Merkmale mit einer geringen Korrelation von Vorteil. Um die Korrelationen in den Merkmalen besser darstellen zu können, wurde eine Kovarianzmatrix gebildet. Diese ist in der Abbildung 6.6 zu sehen. Um die Kovarianzmatrix zu erzeugen, ist es notwendig, dass die Varianz jedes Merkmals paarweise mit jedem anderem Merkmal berechnet wird. Hierfür ist es unerlässlich, dass jeder Satz betrachtet werden muss. Die Varianz wird mittels der Berechnung von **Pearson** durchgeführt. Dabei gibt das Ergebnis an, wie stark die zwei Merkmale korrelieren. Die Berechnung der **Pearson**-Korrelation, wird nach folgender Formel durchgeführt:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Für jedes Merkmal (x,y) wird jeder Wert mit dem Mittelwert der Merkmale ($\bar{x}$,$\bar{y}$) verschoben und multipliziert und abschließend durch die Varianz der Summe geteilt. Das daraus resultierende r gibt

ein Maß für die Korrelation an, wobei der Wert zwischen -1 und 1 liegt. Je näher das Maß gegen 1 oder -1 strebt, desto so stärker ist die Korrelation. Ein Wert um 0 würde auf keine Korrelation hindeuten.

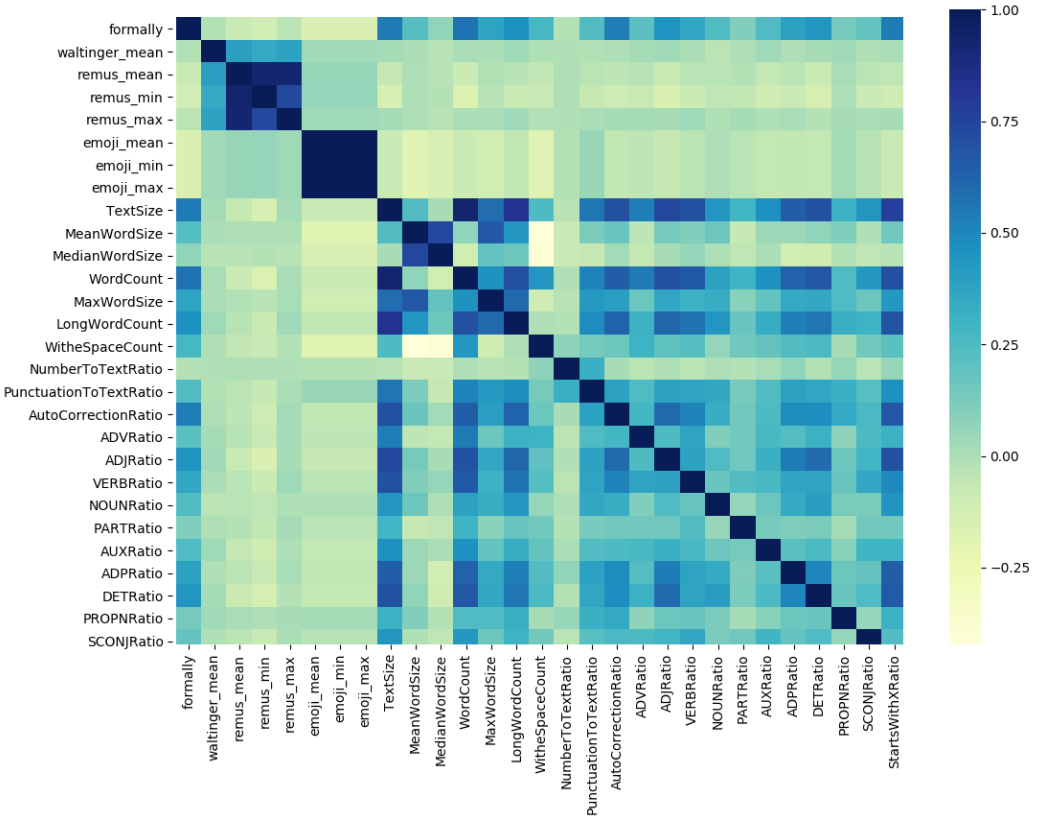

Abbildung 6.6.: Kovarianzmatrix der Merkmale ohne die *Bag of Words* Merkmale

Aufgrund der paarweisen Bestimmung der Korrelation sind die Metriken jeweils auf den Spalten und den Reihen aufgetragen. Durch diese Vergleichsmethode sind auf der Diagonalen der Matrix die gleichen Metriken miteinander verglichen, sodass die Felder dunkelblau sind und ein r von 1 besitzen. Der Korrelationswert lässt sich aus der nebenstehenden Legende ablesen. Starke Korrelationen lassen sich zwischen den Metriken Waltinger [40] und Remus et al. [28] feststellen. Dies kann mit Überschneidungen in den Korpora zusammenhängen. Nicht wenig verwunderlich ist, dass die Emoji-Metriken sehr stark miteinander korrelieren. Dies hat den Grund, dass in jedem Satz meist nur ein einziger Emoticon vorkommt, womit das Minimum, das Maximum und der Durchschnitt häufig identisch sind. Wegen der Korrelation zwischen der automatischen Sprachkorrektur und der Formalität in der Kommunikation wird angenommen, dass in einer formalen Kommunikation weniger Rechtschreibfehler gemacht werden. Ebenfalls auffällig ist, dass die Metriken **Textlänge** und **Anzahl an Wörtern** sowie **Anzahl**

an langen Wörtern stark miteinander korrelieren. Der überwiegende Teil der Metriken korreliert wenig. Das ist an der Häufung von Türkis und einem grünlichen Gelb zu erkennen. Somit bieten die Metriken bezüglich der geringen Korrelation Potenzial für eine gute Differenzierung. Letztendlich entscheidet allerdings nicht die Korrelation, sondern die Aussagekraft der Metrik und die Möglichkeit, den Merkmalsraum gut zu trennen. Bezogen auf die Verarbeitungspipeline aus der Abbildung 5.1 beginnt nun die Klassifikation über die Lernverfahren. Wie bereits im Vorfeld angesprochen, gibt es viele Hyperparameter, die das Lernen der Klassifikation beeinflussen können. Aus der Fülle an Parametern entsteht ein Optimierungsproblem, welches durch einen Evolutionären Algorithmus annähernd gelöst werden kann. In dieser Arbeit werden lediglich die Eingangsdaten als zu optimierende Hyperparameter betrachtet. Im Falle des Kommunikationsdatensatzes aus dem Softwareentwicklungsprojekt sind das 32 Metriken, deren Verwendung es zu optimieren gilt. Die Optimierung hat den Zweck, ein möglichst schlankes Modell zu erzeugen, welches leichtgewichtig trainiert und evaluiert werden kann. Hierfür hat der genetisch arbeitende Algorithmus das Training des Modells mit einem stets mutierten Parameterset durchgeführt. Die Einzelheiten zur Implementierung können unter dem Abschnitt Optimierung des Lernverfahrens nachgelesen werden. In der Abbildung 6.7 ist zu sehen, wie das Ergebnis der Fitness-Funktion zu jeder Generation aufgetragen wurde. Die Entwicklung der besten Generationen ist in der Pareto-Front eingezeichnet.

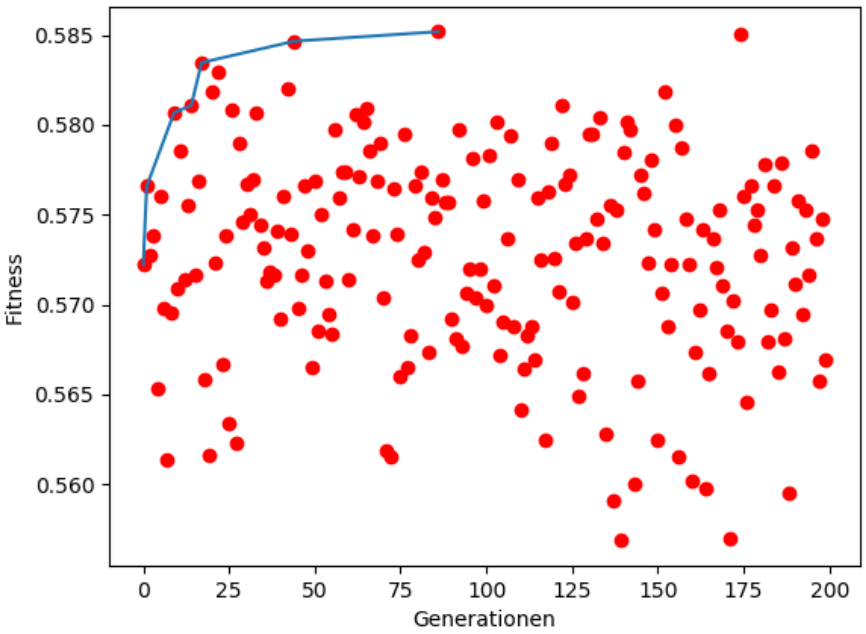

Abbildung 6.7.: Optimierung durch den EA

Wie zu erkennen ist, werden am Anfang des Optimierungsprozesses noch leicht bessere Lösungen

gefunden. So wurde die beste Lösung nach ungefähr 80 Generationen gefunden. Das Optimierungsverfahren arbeitet aber konsequent auf einem Nievau zwischen 55% und 58,5% durchschnittlicher Genauigkeit. Nach Abschluss des Optimierungsprozesses liefert der Evolutionäre Algorithmus folgendes Ergebnis:

[1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1].

Jeder Wert in dem Array steht für eine Metrik, die bei einer 1 für das Modell verwendet wurde und bei einer 0 nicht. Als Startwert wurden alle Metriken mit einer 1 in dem Array belegt. Durch die Bitflip-Mutation wurden im Laufe des Optimierungsprozesses folgende Metriken entfernt: Die durchschnittliche und maximale Remus et al. [28] Metrik, das Verhältnis von Leerzeichen zu Text, das Verhältnis von Zahlen zu Text, die sprachliche Autokorrektur, das Verhältnis von Adjektiven, Verben, AUX und ADP zu Text. Die Entscheidungen des Algorithmus fußen auf dem Ausprobieren der Kombinationen. Da der Merkmalsraum aufgrund der hohen Anzahl von Dimensionen sehr komplex ist, lässt sich nur mutmaßen, warum die acht Metriken für das Modell nicht relevant zu seien scheinen.

6.5. Evaluierung des Models

Wie in dem Abschnitt 6.4 beschrieben, wurde das Klassifizierungsmodell gelernt und über einen Evolutionären Algorithmus optimiert. Dabei erreichte der Evolutionäre Algorithmus bei der Erkennung der richtigen Emotionalität eines Satzes, bezogen auf die drei Klassen **negativ**, **neutral** und **positiv** in seiner besten Generation eine durchschnittliche Genauigkeit von 58,5%. Aus dieser Generation wurde das beste Modell gewählt, welches eine Genauigkeit von 62,96% erreicht. Angesichts einer Klassifikation sind üblicherweise knapp 63% als Erkennungsgenauigkeit nicht besonders gut. Betrachtet man aber den konkreten Anwendungsfall, bei dem selbst bei der manuellen Validierung des Datensatzes lediglich eine Genauigkeit von 66% erreicht wurde, scheinen die Ergebnisse des Modells nicht schlecht. So ist die menschliche Erkennung der Emotion nur 4,8% besser als das Computermodell. Betrachtet man nun die Baseline eines sehr naiven Klassifizierungsalgorithmus, welcher mit dem Wissen über die Verteilung der Daten arbeitet, würde dieser stets die häufigste Klasse wählen. Dies ist die Klasse **neutral** mit 49,13%. Das Baseline-Verfahren liegt somit in knapp 50% der Fälle richtig, womit es allerdings 22% schlechter ist als das gelernte Computermodell.

Tabelle 6.8.: Konfusions-Matrix des Modells auf dem Testdatensatz

	positiv	**neutral**	**negativ**
positiv	36	46	7
neutral	10	173	11
negativ	2	64	29

Wie in der Tabelle 6.8 gezeigt, ist die Konfusions-Matrix für das Training auf dem Testset zu sehen. Ähnlich der Konfusionsmatrix auf dem Validierungsdatensatz in der Tabelle 6.7, ist die Verwechslung zwischen **negativ** und **positiv** sehr gering. Allerdings ist die Verwechslung von **negativ** und **positiv** zu **neutral** sehr hoch, sodass häufiger **negativ** und **positiv** für **neutral** gehalten werden als für eine andere Klasse.

Tabelle 6.9.: Klassifizierungsreport für das Testset

Klasse	Precision	Recall	F1-Score	Häufigkeit
negativ	0.62	0.31	0.41	95
neutral	0.61	0.89	0.73	194
positiv	0.75	0.40	0.53	89

Betrachtet man die in der Tabelle 6.9 abgebildeten Klassifizierungsreports, so wird deutlich, dass die Klasse **positiv** eine höhere **Precision** hat als die anderen Klassen und somit sicher erkannt wird. Im Klassifizierungsreport sind die gängigen Metriken angegeben, um eine Klassifizierung zu beurteilen. Um die Metriken in dem Klassifikationsreport zu berechnen, ist es notwendig, die folgenden Werte für jede Klasse zu berechnen:

Die **True Positives** (TP) ist die Anzahl aller richtig erkannten Sätze für die Klasse. **True Negatives** (TN) ist die Anzahl aller Sätze, die nicht zu der Klasse gehören und auch als solche klassifiziert wurden. **False Positives** (FP) ist die Anzahl aller Sätze, die nicht zu der Klasse gehören, aber als solche klassifiziert wurden und **False Negatives** (FN) ist die Anzahl aller Sätze, die als falsch klassifiziert wurden, obwohl sie eigentlich zu der Klasse gehören.

Die **Precision** (p) ist dabei wie folgt definiert $p = \frac{TP}{TP+FP}$. Sie gibt an, wie viel Prozent der für die Klasse vorhergesagten Sätze tatsächlich zu der Klasse gehören. Der **Recall** (r) ist definiert durch $r = \frac{TP}{TP+FN}$. Er gibt an, wie viel Prozent der Sätze aus der Klasse zurückgegeben werden, wenn sie als solche klassifiziert wurden. Wie aus der Konfusions-Matrix zu erahnen ist, hat die Klasse **neutral** im Gegensatz zu den anderen Klassen einen besonders hohen **Recall**. Der **F1-Score** nimmt den **Recall** und die **Precision** gewichtet auf und ist daher neben der Genauigkeit ein geeignetes Maß zur Beurteilung einer Klassifikation. Der **F1-Score** definiert sich über F1-Score $= \frac{2 \cdot (r \cdot p)}{(r+p)}$. Besonders heraus sticht die Klasse **neutral**. Diese hat einen wesentlich höheren **F1-Score** als die anderen beiden Klassen, womit die Erkennung dieser Klasse wesentlich häufiger erfolgreich sein wird als bei den anderen. Im Gegensatz dazu zeigt der **F1-Score**, dass die Klasse **negativ** am schlechtesten erkannt wird. In der letzten Spalte des Klassifikationsreports ist die Häufigkeit der Klassen in dem Testset zu sehen, welche eine sehr ähnliche Verteilung zu dem Gesamtdatensatz darstellt.

Beachtet werden muss bei der Einordnung der Ergebnisse, dass der Algorithmus zum Lernen der Klassen lediglich die Vorgaben des Menschen hatte und damit nicht besser sein kann als die Datenlage.

In der Tabelle 6.10 sind einige kurze Beispiele für die Klassifizierung angegeben, die aus dem Testset gelabelt wurden. Das Testset wurde zuvor nicht für das Training verwendet, sodass das Modell die Erkennung bei unbekannten Daten zeigt.

Tabelle 6.10.: Beispielsätze für die Vorhersagen des Klassifizierungsmodells

Nr.	Satz	Manuell	Vorhergesagt
1	Ja, war mein Fehler.	**negativ**	**neutral**
2	herzlich willkommen in Bayern :-) ein kluge Entscheidung	**positiv**	**positiv**
3	Falls ihr das nicht wart, muss der sio wieder gefixt werden:	**neutral**	**neutral**
4	Hatte Dich heute morgen anders verstanden.	**negativ**	**neutral**
5	Dann sind wir uns einig.	**positiv**	**neutral**
6	Sehr gut!	**positiv**	**positiv**

Mensch und Algorithmus haben gleichsam Probleme beim Finden der richtigen Emotionalität, wie im Abschnitt zum Labeln von Daten nachzulesen ist. Dies liegt zum einen am großen Verwechslungspotentzial, welches durch die Emotionen in die Sätze kommt, aber auch an der allgemeinen Herausforderung überhaupt, die Emotionen aus einem Text zu extrahieren. Wang und Castanon [41] erklärten: „In fast der Hälfte der Fälle waren Emoticons die einzige Komponente im Text, durch die eine Erkennung einer positiven oder negativen Emotionalität möglich war." (Aus dem Englischen: *„In nearly a half of the cases, emoticons were the only component in the text that expressed some positive or negative sentiment."* [41] Seite 4) Es ist festzustellen, dass, ohne kontextuelle Informationen zu haben, eindeutige Tendenzen fast nur über die Verwendung von Emoticons möglich waren. In dem professionellen Softwareteam wurden in nur wenigen Fällen Emoticons verwendet, sodass die Erkennung der Tendenzen über diese sehr schwierig war. Hinzu kommt, wie in Tabelle 6.10 zu sehen ist, dass die Eindeutigkeit der gelabelten Klassen nicht ganz klar war und von der Kontextsensivität abhängen kann. So wirkt ein einzeln stehender Satz wie *„Dann sind wir uns einig."* eher **neutral** oder **negativ**, aber in einem anderen Zusammenhang **positiv**. Der Kontext eines Satzes oder einer Nachricht wird in diesem Analyseverfahren nicht betrachtet. Dieser kann aber durchaus Potenzial für die Erkennung der Emotionalität einer Nachricht bieten.

6.6. Ergebnisse für die Konversationen

Wird nun die Analysepipeline des Analyseverfahrens aus der Abbildung 5.1 betrachtet, wurden alle Analyseschritte bis auf den der Auswertung durchgeführt. Dieser wird im Folgenden durchgeführt. Hierbei werden die Ergebnisse der Klassifizierung auf die Stimmung im Softwareentwicklungsteam übertragen. Dazu wurden die Nachrichten im Entwicklerteam durch die zugeteilten Klassen gewichtet, wie im Abschnitt 5.6 beschrieben. Durch diese Gewichtung ist es möglich, mit den Kommunikations-

daten zu rechnen. So bieten sich diverse Anwendungsfälle um Rückschlüsse auf die Zusammenarbeit und die Stimmung im Team zu ziehen. Es wäre beispielsweise möglich, einzelne Mitarbeiter durch auffällig **positive** oder **negative** Kommunikationweisen zu identifizieren und entsprechend zu belobigen oder nachzusteuern. Ebenfalls wäre es möglich, die Kommunikation in einem Graphen zu visualisieren und Kanten mit dem Score der Kommunikation zwischen den Teammitgliedern zu belegen. Durch eine entsprechende Färbung der Kanten könnten so teaminterne Konflikte dargestellt oder gut kommunizierende Mitarbeiter identifiziert werden. In einem Softwareteam können auf diese Weise nicht nur Aufgaben und Teilprojekte nach Kompetenzen der Mitarbeiter, sondern auch nach der Zusammenarbeit gebildet werden. Ein weiterer Punkt könnte das gezielte Einstreuen von Teambuildingmaßnahmen für schlecht kommunizierende Mitarbeiter sein, um die Kommunikation zu fördern. Wie in Kapitel 1 dargelegt, ist gute Kommunikation der Schlüssel zur erfolgreichen Zusammenarbeit. Im Kapitel 7 werden weitere Anwendungsfälle für die aufbereiteten Kommunikationsdaten diskutiert.

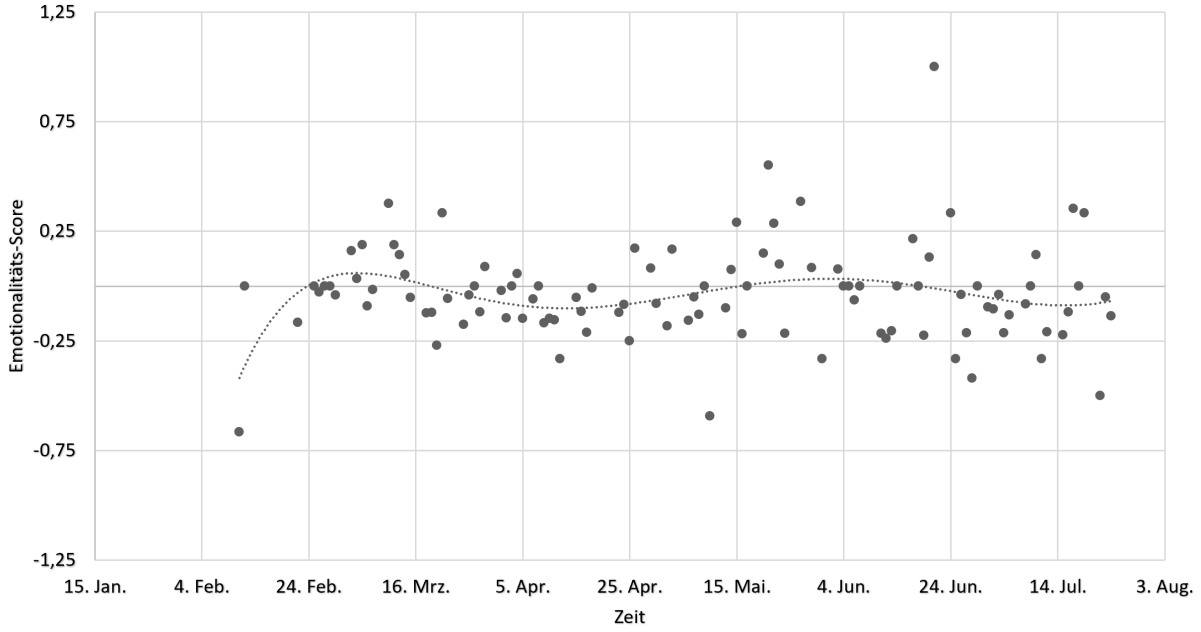

Abbildung 6.8.: Darstellung des Gefühlsniveaus nach den manuell gelabelten Daten

Im Rahmen dieser Arbeit sind diese Anwendungsfälle aufgrund der getroffenen Absprachen mit dem Unternehmen des untersuchten Softwareentwicklerteams nicht möglich, da sie Rückschlüsse auf einzelne Personen zulassen. Für diese Arbeit wurden explizit nur Anwendungsfälle in Betracht gezogen, welche die Kommunikationsdaten auf Teamebene zusammenfassen. Daher wurde für die Arbeit der Anwendungsfall verfolgt, die Kommunikation über das gesamte Entwicklerteam zusammenzufassen und nach Tagen zu gruppieren. Anschließend wurde die durchschnittliche Stimmung pro Tag gebildet und in einem Diagramm visualisiert. Die daraus entstehende Punktewolke ist in der Abbildung 6.8 zu sehen.

Zusätzlich wurde in der Darstellung noch eine Trendlinie (gepunktet) ergänzt, die anzeigt, wie die Stimmung sich im Softwareentwicklungsteam entwickelt. Besonders an der Darstellung in der Abbildung 6.8 ist zu sehen, dass dort der letzte Auswertungsschritt der Gewichtung der Kommunikationsdaten aus Abschnitt 5.6 auf die manuell gelabelten Sätze angewendet wurde. Somit eignen sich die Anwendungsfälle auch für die manuell Klassifizierung der Kommunikationsdaten eines Softwareentwicklerteams. Klar erkennbar ist in der Abbildung 6.8 eine Wellenbewegung in der Stimmung des Entwicklerteams. Diese Wellenbewegung deutet auf verschiedene Stressphasen im Team hin. Wie es von einem professionellen Softwareentwicklungsteam zu erwarten war, bewegt sich die Kommunikation überwiegend um die **neutrale** Stimmung herum. Es gibt nur wenige einzelne Ausreißer, die in die Extreme schlagen, überwiegend jedoch ins positive Extrem. Bei der Interpolation der Trendline ist zu beachten, dass stets die Daten von acht Tagen der Vergangenheit und Zukunft in die Berechnung einfließen, sodass besonders im ersten und letzten Punkt die Trendline mit Vorsicht zu genießen ist.

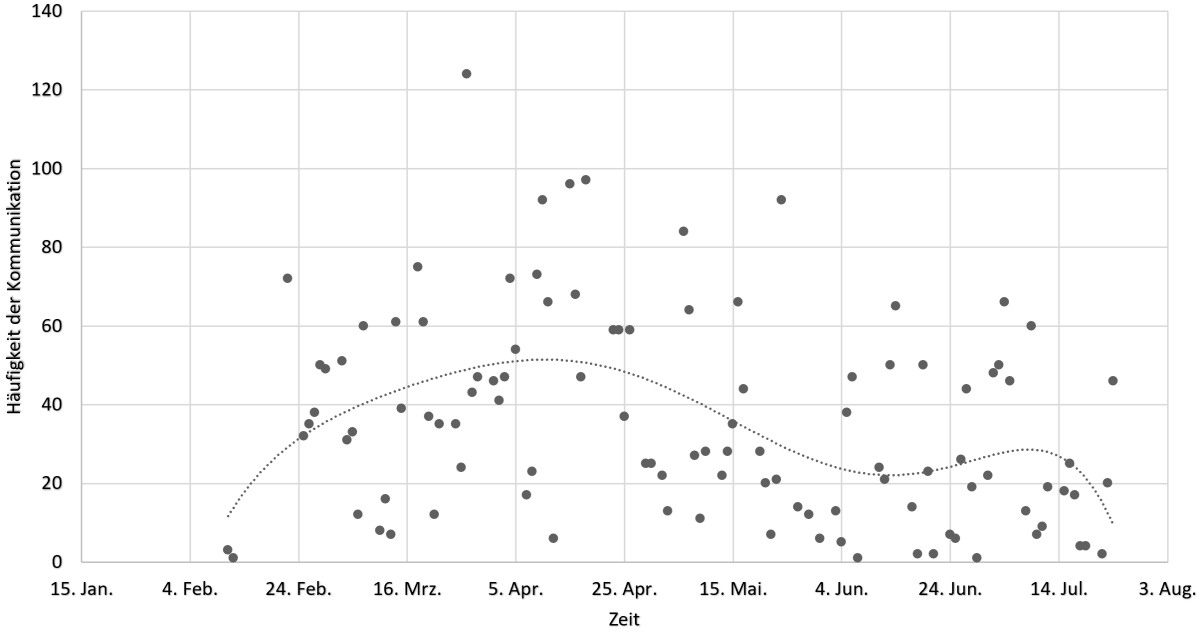

Abbildung 6.9.: Häufigkeit der Kommunikation in der Trend-Darstellung

Da an den Tagen häufig mit unterschiedlicher Frequenz kommuniziert wurde, sind in der Abbildung 6.9 die Häufigkeiten der Sätze aufgetragen, um einordnen zu können, wie die Ausreißer in der Abbildung 6.8 zu bewerten sind. Anzumerken ist, dass aus fünf Streams aus dem **Zulip** die Kommunikationsdaten mit unterschiedlichen Zeiträumen extrahiert wurden, da lediglich eine Limitierung der Anzahl der Nachrichten angegeben wurde und in einigen Streams diese Anzahl überschritten wurde. Limitierend kommt hinzu, dass das Tool **Zulip** erst Ende Februar in dem Softwareentwicklungsteam eingeführt wurde. Wie in der Abbildung 6.9 zu sehen ist, nahm die Häufigkeit in der Kommunikation zunächst bis Mitte April zu und danach gegen Mitte des Jahres ab. Das könnte

mit der Verlagerung von Teammitgliedern, der anstehenden Urlaubszeit oder mit den verwendeten Kommunikationskanälen zusammenhängen.

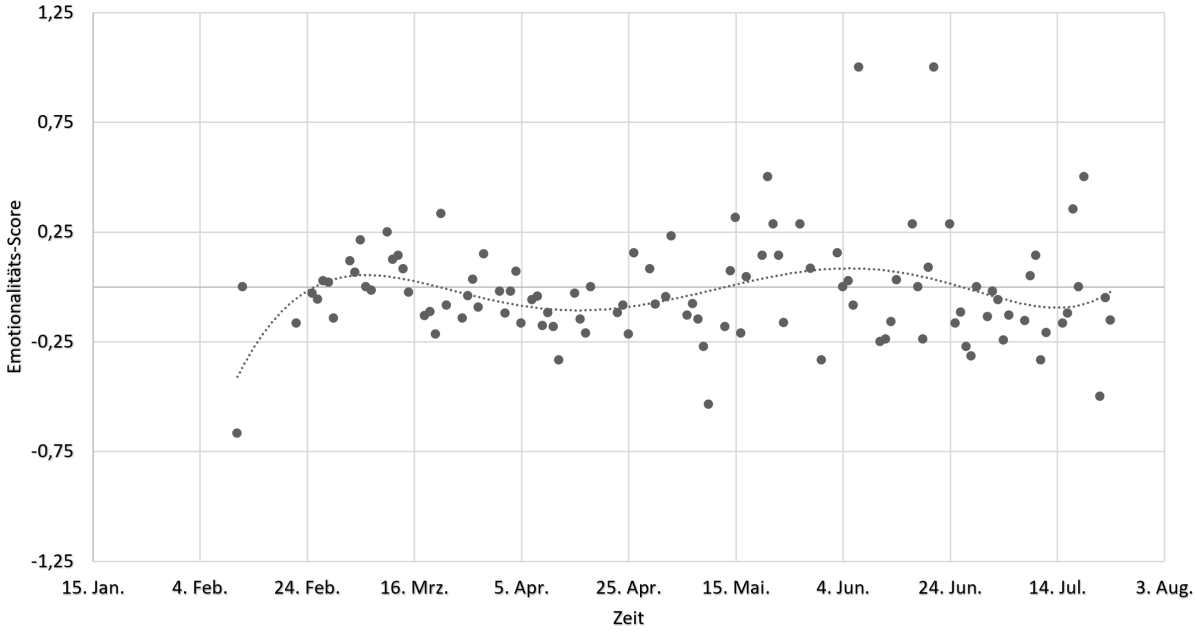

Abbildung 6.10.: Darstellung des Gefühlsnievaus nach der Vorhersage durch das Modell

Die Abbildung 6.10 wurde identisch zu der Abbildung 6.8 erzeugt. Lediglich die gezeigten Daten unterscheiden sich. So wurden in der Abbildung 6.8 die Label verwendet, die zum Training des Klassifizierungsmodells benötigt wurden. In der Abbildung 6.10 wurde hingegen für die vorhergesagten Label das Klassifizierungsmodell verwendet. Die Trendlinien ähneln sich in ihrem Verlauf sehr. Jedoch ist die Trendline des Klassifierungsmodells in den Ausschlägen deutlicher als die des händischen Labels. Einschränkend gilt selbstverständlich, dass bei der Darstellung in der Abbildung 6.10 der Klassifierer Sätze gelabelt hat, die zuvor für das Training verwendet wurden. Dennoch scheint das Klassifierungsmodell gelernt zu haben, die Sätze zu klassifizieren, da die Kurven sehr ähnlich sind. Wie bereits angedeutet, kann der Verlauf der Trendlinie von mehreren Faktoren abhängen. So scheint die Trendlinie der Abbildung 6.10 wie eine in den Mittelpunkt verschobene Ableitung der Häufigkeit der Kommunikation in der Abbildung 6.9. Hintergrund könnte sein, dass für die Analyse Streams im Zulip ausgewählt wurden, in denen viel kommuniziert wurde. Dies waren zumeist die Streams, in denen technische und infrastrukurelle Probleme genannt wurden, weshalb die Trendlinien der Emotionalität in Abbildung 6.8 und Abbildung 6.10 tendenziell eher **negativer** als **positiver** sind. In diesen Streams wurden die auftretenden Probleme besprochen, wenn ein Build fehlschlug. Wie in der Wordcloud aus Abbildung 6.3 zu sehen, ist das Wort **BUILD** häufig aufgetreten. Ein denkbarer Zusammenhang zu der Häufigkeit der Kommunikation ist, dass mehr kommuniziert und die Stimmung im Entwicklerteam angespannter wird, sobald es Probleme gibt. Daraus kann gefolgert werden, dass mehr kommuniziert wird, um durch Teamarbeit das Problem zu lösen, sobald Probleme

im Entwicklerteam auftreten. Die Trendlinien wurden einem projektverantwortlichen Mitarbeiter des Kooperationspartners vorgelegt, welcher die Stimmungsentwicklung aus dem Gedächtnis ebenfalls so eingeordnet hat. Für eine genauere Betrachtung der Stimmung im Softwareentwicklungsteam kann das Intervall, in dem die Kommunikationsdaten gruppiert werden, verkleinert werden. Um bei einer feineren Betrachtung noch verlässlichere Informationen zu bekommen, ist es erforderlich, dass viel kommuniziert wird. Im Schnitt wurden im betrachteten Projekt 35 Sätze am Tag geschrieben.

Um die Klassifizierung der Nachrichten besser einordnen zu können, wurden in der Abbildung 6.11 Boxplots über den Emotionalitäts-Score gebildet.

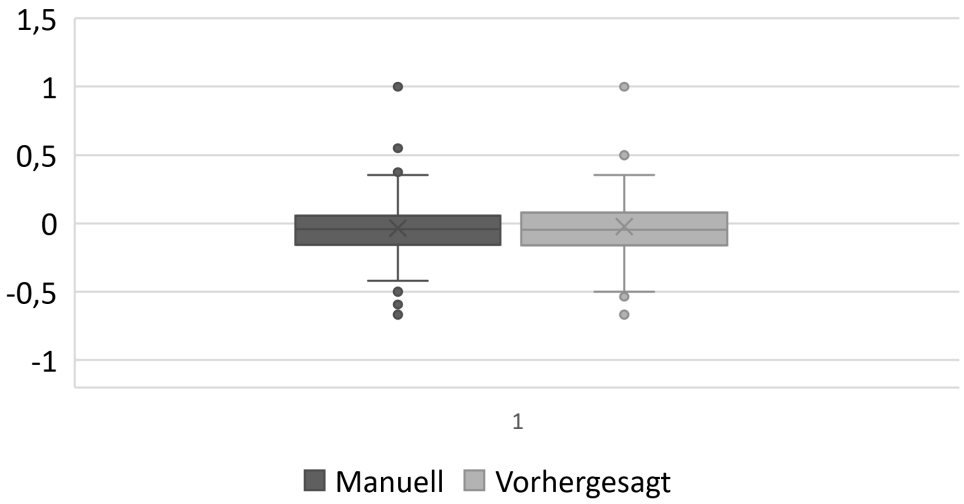

Abbildung 6.11.: Darstellung der Schwankungen in der Emotionalität im Softwareentwicklungsprojekt

Zu sehen ist, dass die Ausreißer im Extremen identisch scheinen. Allerdings haben die manuell gelabelten Daten mehr Ausreißer in der Grafik als die Klassifizierung durch das Modell. Der obere Whisker beider Boxplots scheint auf gleicher Höhe zu liegen. Der untere Whisker liegt bei dem Boxplot mit vorhergesagten Labeln leicht unter dem des manuell gelabelten. Der Eindruck der stärkeren Ausprägung der Trendlinie in der Abbildung 6.10 wird unterstützt durch den größeren Abstand zwischen dem unteren und oberen Quartil des rechten Boxplots. Die Ähnlichkeit zwischen den Trendlinien zeigt sich in der beinahe identischen Lage des Medians. Durch die Boxplots wird zusätzlich bestätigt, dass die Kommunikation überwiegend leicht **negativ** ist, aber im Extremen eher in das Positive ausschlägt. Ebenfalls wird deutlich, dass es sich um professionelle Kommunikation handelt, da sich 50% der Sätze zwischen dem unteren und oberen Quartil befinden, die Nulllinie eingeschlossen. Generell bestätigt die Ähnlichkeit zwischen den Boxplots, dass das Lernen des Modells erfolgreich war. Die leicht **negative** Tendenz in der Kommunikation kann in der bereits angesprochen Wahl der Streams mit dem Kommunikationskanal **Zulip** erklärt werden. Denn diese umfassten die technischen Probleme, die beim Build auftreten.

7

Diskussion

In dieser Arbeit wurde ein Konzept erarbeitet und umgesetzt, welches es ermöglicht, automatisiert aus textueller digitaler Kommunikation eines Softwareentwicklerteams die Stimmung im Team zu ermitteln. Das Verfahren wurde zusammen mit einem konkreten Anwendungsfall in einer Studie evaluiert. In der Studie wurde die Kommunikation eines Softwareentwicklerteams mit 65 im betrachteten Zeitraum aktiv an der Kommunikation beteiligten Teammitgliedern aus der privaten Wirtschaft untersucht. Das Verfahren klassifiziert die Nachrichten eines Kommunikationskanals in drei Klassen (**positiv**, **neutral** und **negativ**). Es stellte sich heraus, dass die Klassifizierung der Sätze selbst Menschen Schwierigkeiten bereitet. So wurde bei der Validierung der Label, der zum Training verwendeten Sätze, lediglich eine Genauigkeit von 66% erreicht. Das computergestützte Lernverfahren erreichte auf Basis der vom Menschen gelabelten Kommunikationsdaten eine Genauigkeit von 63%. Damit ist das Lernverfahren nur 4,8% schlechter als der Mensch in der Klassifikation.

Zentrale Forschungsfrage

Wie lässt sich das Kommunikationsverhalten in Entwicklerteams analysieren, um Rückschlüsse über die Stimmung abzuleiten?

Im Kontext dieser Arbeit wurde die oben genannte Forschungsfrage gestellt und durch ein Konzept eine Möglichkeit entwickelt, diese Frage zu beantworten. In der Evaluation wurde an einem Entwicklerteam die entwickelte Methode erprobt. Die Stimmung des Teams wurde anhand der Klassifizierung des gelernten Modells für Tage zusammengefasst und auf eine Zeitlinie aufgetragen. Durch eine Trendanalyse, wie in Abbildung 6.10 zu sehen ist, ist es dem Projektleiter möglich, die vergangene Entwicklung der Stimmung und den momentanen Stand zu visualisieren. Dadurch kann er Maßnahmen einleiten, um die Stimmung im Falle eines negativen Trends gezielt zu heben. Eine konkrete Maßnahme könnte sein, dass er ein Meeting einberuft, um die Probleme über eine direkte Face-To-Face Kommunikation zu lösen.

Beantwortung der Zentrale Forschungsfrage

Durch die Anwendung des Konzepts bei diesem konkreten Anwendungsfall lässt sich die oben genannte Forschungsfrage beantworten. Denn mit dem vorgestellten Konzept ist es tatsächlich möglich, das Kommunikationsverhalten im Entwicklerteam zu analysieren, um konkrete Rückschlüsse auf die Stimmung im Team abzuleiten.

7.1. Threats to Validity

Wie in der Studie aufgezeigt, gibt es Grenzen, die die Möglichkeiten zur Verallgemeinerung des Ansatzes und der getroffenen Ergebnisse limitieren. So wurden die Label der Trainingsdaten nur von einer Person validiert (*Conclusion Validity*). Dies geschah aufgrund der Vereinbarung mit dem kooperierenden Unternehmen, wonach nur der Autor alleine die Daten verarbeiten durfte. Wie bereits erwähnt, soll diese Restriktion die Interessen des Unternehmens und insbesondere die der Mitarbeiter wahren. Die gelabelten Kommunikationsdaten stellten allerdings die Grundlage der Studie dar und aus den genannten Gründen war eine Crossvalidation nicht möglich. Wie sich bei der Validierung eines repräsentativen Teils des gesamten Datensatzes herausgestellt hat, besteht eine gewisse Uneindeutigkeit in der Emotionalität der Sätze einer Nachricht. Bei der Erstellung dieses Validierungsdatensatzes wurde keine Rücksicht auf den Kontext der Nachrichten genommen, welcher beim ersten Labeln der Sätze durch die chronologische Abarbeitung gegeben war. Wie in der Tabelle 6.10 zu sehen ist, ist es ohne Betrachtung des Kontextes schwieriger, die tatsächliche Emotionalität einer Nachricht zu bestimmen.

Diese Kontextsensitivtät einer Nachricht wurde bei der Entwicklung des Konzepts und der Umsetzung nicht beachtet, weshalb neben der Problematik beim Labeln der Nachrichten der Algorithmus selbst mit der Berücksichtigung des Kontextes beim Labeln des Validierungsdatenssatzes keine Möglichkeit hat, über den Kontext zu lernen. Diese Limitierung schränkt das Verfahren auf eine satzweise Betrachtung ein (*Construct Validity*).

Neben der Limitierung durch die fehlende Betrachtung des Kontextes und der nicht durchführbaren Crossvalidation der Trainingsdaten besteht bei der Analyse der Sprache stets ein großes Problem bezüglich des kulturellen Verständnisses von Sprache. Dies kann sich nicht nur durch Dialekte von Region zu Region unterscheiden, sondern es können auch verschiedene Auffassungen der Bedeutungen einzelner Wörter oder Redewendungen großen Einfluss auf die Wirkung der Sprache ausüben (*External Validity*). Das in der Studie untersuchte Entwicklerteam operierte deutschlandweit, weshalb der Faktor von lokal begrenzten kulturellen Einflüssen und die Gefahr durch unterschiedliche Dialekte einen Einfluss auf die Kommunikation im Entwicklerteam haben könnten. In einem professionellen Umfeld sollten diese Unterschiede allerdings in der Einflussnahme auf die Kommunikation zu vernachlässigen sein.

Für die Einordnung der Nachrichten in die Klassen **positiv**, **neutral** und **negativ** ist eben diese Professionalität der Kommunikation ein Problem. So sind die Unterschiede zwischen den Klassen teilweise nur in feinen Tendenzen erkennbar, weshalb ein großes Potenzial besteht, dass die Klassen verwechselt werden (*Internal Validity*). Dieses Phänomen ist in der Konfusions-Matrix in der Tabelle 6.8 klar erkennbar.

In der Arbeit werden Textkorpora verwendet, welche die emotionale Färbung von Wörtern enthalten, um die emotionale Färbung eines Satzes zu ermitteln. Allerdings sind die Textkorpora von Waltinger [40] und Remus et al. [28] aus dem Jahr 2010. Sprache entwickelt sich jedoch ständig weiter. So verlieren einige Wörter an Bedeutung, andere werden aus anderen Sprachen eingedeutscht oder sind Wortneuschöpfungen. Dieser Neologismus führt unter anderem dazu, dass die verwendeten Textkorpora an Aussagekraft verlieren. Neben der Veränderung der Sprache verändern sich auch die Werte einer Gesellschaft, sodass die Gewichte der Wörter in den Textkorpora sich verändern können (*Conclusion Validity*).

Um die Emotionalität von Nachrichten noch genauer festzustellen, wurde eine Sensitivität für die auftretenden Emoticons in das Analyseverfahren integriert. Dazu wurden die in Wang und Castanon [41] vorgestellten Emoticons entsprechend der festgesetzten emotionalen Färbung für die Klassifizierung verwendet. Die in dem Paper vorgestellten Emoticons erheben keinen Anspruch auf Vollständigkeit und neben den vorgestellten gibt es noch weitere Emoticons, die nicht nur aus Interpunktion bestehen. Diese fanden in der Arbeit allerdings keine Berücksichtigung. Zusätzlich bieten die Emoticons Doppeldeutigkeiten und die Interpretation der Emoticons ist unter anderem von kulturellen Faktoren und der Persönlichkeit abhängig (*Internal Validity*). Einschränkend kommt hinzu, dass in einer professionellen Kommunikation auf Emoticons je nach Grad der Formalität verzichtet wird.

Die Studie, mit der das Konzept evaluiert wurde, beschränkt sich auf ein Softwareentwicklungsteam aus einem Unternehmen und nur einer Branche, weshalb das gelernte Modell und die erlangten Erkenntnisse sich nicht uneingeschränkt generalisieren lassen (*External Validity*). Um ein aussagekräftigeres Modell lernen zu können, sind annotierte Kommunikationsdaten aus mehreren Softwareentwicklungsprojekten erforderlich. Das Trainieren mit einer größeren Menge an Trainingsdaten könnte der Aussagekraft ebenfalls zuträglich sein. Für diese Arbeit wurden 3778 Sätze verwendet. Ziel der Studie war es lediglich, die Anwendbarkeit des Konzeptes zu beweisen und nicht das Treffen verallgemeinerbarer Ergebnisse für verschiedene Unternehmen.

Dessen ungeachtet sprechen die Ergebnisse der Studie für sich selbst. Die automatische Detektion der Emotionalität einer Nachricht wurde, soweit es die Daten zuließen, erfolgreich und in dem konkreten Anwendungsfall auch Nutzen stiftend angewendet. Außerdem konnte sie durch ein Teammitglied aus dem Gedächtnis validiert werden. Im Folgenden werden weitere konkrete Anwendungsfälle beschrieben, welche auf dem in dieser Arbeit vorgestellten Analysekonzept aufbauen können.

7.2. Weitere Anwendungsfälle

Wie bereits erwähnt, gibt es eine Vielzahl von möglichen Szenarien, in denen eine Analyse und abschließende Bewertung der Kommunikation eines Softwareentwicklungsprojekts nützlich sein kann. Neben der durchgeführten Trendanalyse der Stimmung im Entwicklerteam wurden in der Evaluation der Studie noch andere Ansätze vorgestellt. Das ist beispielsweise die Möglichkeit, ein Entwicklernetzwerk abzubilden, in dem jedes Teammitglied als Knoten und die Emotionalität in der Konversation über Kanten dargestellt werden. Das ist prinzipiell ähnlich der Masterarbeit von Kriegel [19], nur dass die Kanten anders gewichtet werden. So kann über Zentralitätsmaße nicht nur die Zusammenarbeit analysiert werden, sondern auch die Qualität der Kommunikation unter den zusammenarbeitenden Teammitgliedern. Auf diese Weise kann es möglich sein, das Team und die Aufgaben optimal zu verteilen und Einzelkämpfer besser zu identifizieren. Bei größeren Softwareentwicklerteams wäre eine Betrachtung von Teilprojekten als Knoten denkbar. Wallat [39] stellte in seiner Bachelorarbeit eine Anwendung vor, mit der die Modellierung von FLOW-Netzwerken möglich war. Auch in diesem Fall könnten die Kanten zwischen den Entitäten mittels einer emotionalen Analyse gewichtet werden, um so die Reibungsverluste besser abzuschätzen. Denn wie in der Einleitung festgestellt, ist eine erfolgreiche Kommunikation der Schlüssel zum Erfolg eines Projekts. Als erfolgreiche Kommunikation könnte eine emotional betrachtete **neutrale** oder **positive** Kommunikation angesehen werden.

Eine Integration einer automatischen Hintergrundanalyse des Kommunikationsverhaltens zur Erstellung von Reports könnte als PlugIn in JIRA integriert werden und somit dem Teamleiter zusammen mit anderen Berichten über das Entwicklerteam leicht zugänglich gemacht werden.

Das Betrachten der Emotionalität lässt noch weitere Anwendungen als Gedankenspiel zu. So könnten Projektphasen anhand der Trendkurven abgelesen werden. In den Entwicklernetzwerken können die Rollen der einzelnen Teammitglieder über das Verhältnis zu den anderen Teammitgliedern validiert werden. Auch ein Abgleich des Workloads einer Person zusammen mit der Untersuchung, wie emotional kommuniziert wird, könnte auf die Robustheit gegenüber Stress hindeuten. Ein Profiling der Mitarbeiter wäre beim Zusammenfließen aller digitalen Informationen denkbar, um die Aufgabenverteilung und Teamzusammensetzung zu optimieren.

Bei allen Anwendung muss der Datenschutz, die Datensicherheit, der Geheimnisschutz und insbesondere die Privatsphäre eines jeden Teammitglieds gewahrt werden. Besonders auf den letzten Punkt wurde sehr viel Wert bei dem für diese Arbeit gewählten Anwendungsfall gelegt. Besonders zu beachten gilt es, dass nicht alles, was im Bereich des Möglichen liegt, auch angewendet werden sollte. So sind bei der Untersuchung der Korrespondenz stets der Betriebsrat und die betroffenen Personen zu informieren. Im Fall der durchgeführten Studie wurden diese Faktoren beachtet und entsprechend mit allen Betroffenen transparent geteilt.

8

Zusammenfassung und Ausblick

In diesem Kapitel wird eine abschließende Übersicht über die Ergebnisse dieser Arbeit und die zukünftigen Anwendungen, die der Ansatz ermöglicht, gegeben. Im vorherigen Kapitel wurden bereits einige denkbare Anwendungsfälle vorgestellt, die einen konkreten Nutzen für die Analyse der digitalen textuellen Kommunikation ermöglichen. Ebenfalls wurde festgestellt, dass die zu Anfang gestellte Forschungsfrage bis auf einzelne Einschränkungen bezüglich der Verallgemeinbarkeit der Ergebnisse beantwortet werden konnte.

8.1. Zusammenfassung

Ziel dieser Arbeit war es, eine computergestützte Analyse des Kommunikationsverhaltens in Entwicklerteams unter Berücksichtigung digitaler Medien durchzuführen. Im Mittelpunkt dieser Arbeit wurde die Forschungsfrage untersucht, wie sich das Kommunikationsverhalten in Entwicklerteams analysieren lässt, um Rückschlüsse über deren Stimmung abzuleiten. Hierfür wurden verschiedenste Kommunikationskanäle in Betracht gezogen, die sich unter Entwicklern etabliert haben. Im weiteren Verlauf der Arbeit wurde ein Konzept entwickelt, welches mittels *natural language processing* und darauf aufbauenden Metriken die emotionale Färbung einer Nachricht extrahiert. Die entwickelten Metriken haben neben statistischen Merkmalen auch die Betrachtung der Wortfärbung einer Nachricht, die Betrachtung von Emoticons und der enthaltenen Wörter in die Analyse einfließen lassen. Die über die Metriken extrahierten Merkmale wurden verwendet, um das maschinelle Lernverfahren zu trainieren. Ziel des Trainings war es, einen Klassifizierer zu lernen, welcher die Nachrichten in drei Klassen (**positiv**, **neutral** und **negativ**) einteilt. Abschließend wurde die Kommunikation des Entwicklerteams aufbereitet und visualisiert, sodass ein Teamleiter die Stimmung im Softwareentwicklungsteam leicht ableiten kann. Das Konzept wurde in einer Studie validiert. Hierzu wurde eine Kooperation mit einem Unternehmen aus der Privatwirtschaft angestrengt. Im Zuge dieser Kooperation wurde zu Forschungszwecken und unter strengen Auflagen, die den Umgang mit den Kommunikationsdaten betrafen, die gesamte Kommunikation eines Softwareentwicklerteams mit bis zu 80

Mitgliedern untersucht. Während der Durchführung der Studie stellte sich heraus, dass die Klassifikation der Sätze selbst Menschen Schwierigkeiten bereitet. So wurde bei der Validierung der Label der zum Training verwendeten Sätze lediglich eine Genauigkeit von 66% erreicht. Dieser Umstand ergibt sich mutmaßlich aus mehreren Faktoren. Zum einen ist die Wirkung der Sprache stark von der interpretierenden Person und deren Gefühlslage abhängig. Zum anderem ist der Kontext eines Satzes sehr entscheidend. Da die einzelnen Nachrichten in Sätze aufgeteilt wurden, um die Emotion feiner zu bestimmen, ist eine Kontextsensitivität nicht gegeben. Zusätzlich konnten die manuell gelabelten Kommunikationsdaten nicht durch eine Crossvalidation validiert werden. Das computergestützte Lernverfahren erreichte auf Basis der vom Menschen gelabelten Kommunikationsdaten eine Genauigkeit von 63%. Damit ist das Lernverfahren nur 4,8% schlechter als der Mensch in der Klassifizierung. Die Nachrichten konnten automatisch aus dem Kommunikationskanal extrahiert und durch das Lernverfahren klassifiziert werden. Anhand der Klassifizierung war es möglich, eine Trendanalyse durchzuführen und so die aktuelle und vergangene Stimmung des Entwicklerteams abzuleiten.

Der Ansatz konnte, soweit es die Kommunikationsdaten zuließen, erfolgreich angewendet werden und einen echten Nutzen beweisen. Weiterführende Arbeiten könnten die Erkennungsgenauigkeit weiter verbessern. Insbesondere das Labeln der Kommunikationsdaten bietet Verbesserungspotenzial.

8.2. Ausblick

Wie in der Zusammenfassung angedeutet, birgt das Analyseverfahren noch Herausforderungen, die in Zukunft weiter erforscht und entwickelt werden können. So können die vorgestellten Anwendungsmöglichkeiten umgesetzt werden, um einen höheren Nutzen für das Softwareentwicklungsteam zu erzielen. Voraussetzung für die zuverlässigere Erkennung von Emotionen in der Kommunikation eines Softwareentwicklerteams ist das Erlangen einer höheren Genauigkeit in der Klassifizierung der Nachrichten. Neben dem Hinzufügen neuer Metriken, die andere Faktoren in die Analyse einbringen, besteht die Möglichkeit, das Aufbereiten der Testdaten zu optimieren. Im Laufe der Arbeit hat sich insbesondere herausgestellt, dass die Klassifizierung der Nachrichten durch den Menschen ein großes Verbesserungspotenzial bietet. Um dieses Verbesserungspotenzial auszuschöpfen, kann eine Crossvalidation der gelabelten Nachrichten durchgeführt werden. Dabei werden die Nachrichten von verschiedenen Personen unabhängig voneinander gelabelt, um eine zu individuelle Einschätzung einzelner Personen auszuschließen. Eine Homogenität der Personen, die die Kommunikationsdaten labeln, kann sich negativ auf die Generalisierbarkeit auswirken. Bei der Crossvalidation können die Abweichungen in den Labeln der Nachrichten aufgelöst werden, indem sie entsprechend dem häufigsten Label gewählt werden. Durch diese Herangehensweise sollten die Label der Nachrichten eindeutiger sein. Die hätte zur Folge, dass das Klassifizierungsverfahren ein generelleres Modell der Emotionalität der Kommunikation in einem Softwareentwicklungsprojekt lernen kann. Die Anwendung dieses Verfahrens war in dieser Arbeit nicht möglich, da an die Kooperation mit dem Unternehmen geknüpft

war, dass den Mitarbeitern des Unternehmens zum Beispiel durch das Labeln der Kommunikationsdaten keine zusätzliche Arbeitslast entsteht und dass die Daten keiner dritten Person zugänglich gemacht werden durften. Wie diese Arbeit zeigt besteht allerdings das Problem, dass es selbst für eine Person schwierig ist, die Label wieder zu reproduzieren, wenn der Kontext verloren geht.

Aus diesem Grund wurde im Verlauf der Arbeit vorgeschlagen, den Kontext einer Nachricht in der Analyse zu berücksichtigen. Um diese Kontextsensitivität zu erreichen, wäre folgender Ansatz denkbar: Eine weitere Metrik könnte zu den vorhandenen Metriken hinzugefügt werden, welche die Wertung der vorherigen Nachrichten mit in die Berechnung einfließen lässt. Der Nachteil dieser Metrik ist, dass das Aufbereiten der Nachrichten aufwändiger wird. Zu dem erhöhten technischen Aufwand kommt hinzu, dass der Kontext einer Nachricht klar sein muss. Nachrichten in einem Nachrichtenkanal müssen nicht zwingend zur selben Konversation gehören und damit kann das Herstellen eines Kontextes zu diesen Nachrichten die Klassifizierung negativ beeinflussen. Neben der Tatsache, dass Empfänger und Sender einer Nachricht für die Bildung eines Kontextes mit einbezogen werden müssen, darf der zeitliche Zusammenhang nicht außer Acht gelassen werden. So können in einem Nachrichtenaustausch die Nachrichten Tage oder länger auseinanderliegen und sich damit nicht im selben Kontext befinden, da zwischenzeitlich auf anderen Wegen kommuniziert wurde. In dieser Arbeit wurde in der Studie die Datenquelle **Zulip** verwendet. In **Zulip** gibt es keine einzelnen Chats, sodass es schwierig ist, den konkreten Kontext einer Nachricht festzustellen. Dennoch könnte die Betrachtung des Kontextes einer Nachricht die Erkennung verbessern.

Neben dem Hinzufügen der Kontextsensitivität wäre es noch möglich, die Nachrichten genauer zu klassifizieren, indem ein Maß für den Inhaltsbezug hergestellt werden kann. Dieses Maß würde feststellen, ob die Nachricht einen Bezug zur aktuell bearbeiteten Aufgabe hat oder eher privater Natur ist. Hierfür könnte eine Wissensdatendank eingesetzt werden, die über den Abgleich von Stichworten versucht, nicht aufgabenbezogene Nachrichten zu erkennen. Das Wissen um Nicht-Aufgabenbezogenheit kann Rückschlüsse auf die Beziehung zwischen den Teammitgliedern zulassen. So könnten vertraute und befreundete Teammitglieder auch vermehrt über nicht Relevantes kommunizieren.

Bei der aktuell verwendeten Darstellung für den in der Arbeit vorgestellten Anwendungsfall könnte noch die Häufigkeit der Kommunikation in Verbindung mit der Trendlinie visualisiert werden. Die Darstellungsweise könnte in einer Studie mit Projektleitern von Softwareentwicklerteams evaluiert werden. Somit könnte der Nutzen dieser Darstellungsweise in einer weiteren Studie mit Teamleitern evaluiert werden.

Neben der Optimierung des Verfahrens können auch andere Quellen für die Analyse herangezogen werden. So wird in der Studie nur ein Softwareentwicklungsteam betrachtet, weshalb das gelernte Modell sehr auf die Domäne und das Entwicklerteam spezialisiert ist. Denkbar wären andere Unternehmen oder Textkorpora der Universität Leipzig[12], die mit über eine Millionen Datensätzen

[12]Auf die Textkorpora kann über folgenden Link zugriffen werden: http://wortschatz.uni-leipzig.de/en

noch mehr Daten zum Lernen bieten. Durch ein Training mit einem größeren Datensatz könnte das Verfahren besser generalisiert werden. Allerdings sind die Textkorpora der Universität Leipzig nicht festgelegt auf die Kommunikation in Entwicklerteams, sondern greifen auf die Daten aus Millionen Zeitungsartikeln zurück.

Generell ist allerdings festzuhalten, dass das Verfahren trotz Einschränkungen erfolgreich angewendet werden konnte. Auch wenn es viele Möglichkeiten gibt, den Ansatz noch zu verbessern, so sind der Nutzen in der praktischen Anwendung und die Erkenntnisse für die Forschung, die das vorgestellte Analyseverfahren bietet, nicht nur für die Softwareentwicklung sehr interessant.

A

Literaturverzeichnis

[1] Alghalibi, M. ; Al-Azzawi, A. ; Lawonn, K. : Deep Tweets Analyzer Model for Twitter Mood Visualization and Prediction Based Deep Learning Approach. In: *International Journal of Computer and Communication Engineering* 8 (2019), Nr. 1, S. 1–17

[2] Calefato, F. ; Damian, D. ; Lanubile, F. : An Empirical Investigation on Text-Based Communication in Distributed Requirements Workshops, IEEE, S. 3–11

[3] Daft, R. L. ; Lengel, R. H. ; TEXAS A ; M UNIV COLLEGE STATION COLL OF BUSINESS ADMINISTRATION.: *Information Richness: A New Approach to Managerial Behavior and Organization Design*. Texas A & M University, 1983 (Office of Naval Research Technical Report series)

[4] Darwin, C. : *Die Entstehung der Arten*. Hamburg : Nikol, 2008

[5] Dennis, A. R. ; Valacich, J. S.: Rethinking media richness: Towards a theory of media synchronicity, IEEE Computer Society Press, S. 10

[6] Döring-Seipel, E. ; Lantermann, E.-D. : Komplexität – Eine Herausforderung für Unternehmen und Führungskräfte. In: Grote, S. (Hrsg.): *Die Zukunft der Führung* Bd. 23. Berlin and Heidelberg : Springer, 2012, S. 153–171

[7] Ebert, C. ; Neve, P. de: Surviving global software development. In: *IEEE Software* 18 (2001), Nr. 2, S. 62–69

[8] Eiben, A. E. ; Smith, J. E.: *Introduction to evolutionary computing*. 2nd ed. Berlin : Springer, 2015 (Natural Computing Series)

[9] Evaristo, J. ; Scudder, R. ; Desouza, K. C. ; Sato, O. : A dimensional analysis of geographically distributed project teams: a case study. In: *Journal of Engineering and Technology Management* 21 (2004), Nr. 3, S. 175–189

[10] Felbo, B. ; Mislove, A. ; Søgaard, A. ; Rahwan, I. ; Lehmann, S. : Using millions

of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In: Palmer, M. (Hrsg.) ; Hwa, R. (Hrsg.) ; Riedel, S. (Hrsg.): *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA : Association for Computational Linguistics, S. 1615–1625

[11] Grewe, L. L. ; Hu, C. : ULearn: Understanding and reacting to student frustration using deep learning, mobile vision and NLP. In: Grewe, L. L. (Hrsg.) ; Blasch, E. P. (Hrsg.) ; Kadar, I. (Hrsg.): *Signal Processing, Sensor/Information Fusion, and Target Recognition XXVIII*, SPIE, 30

[12] Gumm, D. C.: Distribution Dimensions in Software Development Projects: A Taxonomy. In: *IEEE Software* 23 (2006), Nr. 5, S. 45–51

[13] Herbsleb, J. D. ; Mockus, A. : An empirical study of speed and communication in globally distributed software development. In: *IEEE Transactions on Software Engineering* 29 (2003), Nr. 6, S. 481–494

[14] Iqbal, F. ; Fung, B. C. M. ; Debbabi, M. ; Batool, R. ; Marrington, A. : Wordnet-Based Criminal Networks Mining for Cybercrime Investigation. In: *IEEE Access* 7 (2019), S. 22740–22755

[15] Klünder, J. : *Analyse der Zusammenarbeit in Softwareprojekten mittels Informationsflüssen und Interaktionen in Meetings*. LOGOS VERLAG BERLIN, 2019

[16] Klünder, J. ; Handke, L. ; Gfesser, T. ; Schneider, K. ; Kauffeld, S. : Soziale Aspekte hybrider und traditioneller Softwareentwicklung. (2017)

[17] Klünder, J. ; Schneider, K. ; Kortum, F. ; Straube, J. ; Handke, L. ; Kauffeld, S. : Communication in teams – An expression of social conflicts. In: *Human-Centered and Error-Resilient Systems Development* (2016)

[18] Kraut, R. E. ; Streeter, L. A.: Coordination in software development. In: *Communications of the ACM* 38 (1995), Nr. 3, S. 69–81

[19] Kriegel, J. : *Konstruktion und Analyse von Kollaborationsnetzwerken in Entwicklerteams basierend auf Metadaten aus JIRA und Confluence*. 2019. – Leibniz Universität Hannover, Fachgebiet Software Engineering, Masterarbeit

[20] Liddy, E. d.: Enhanced Text Retrieval Using Natural Language Processing. In: *Bulletin of the American Society for Information Science and Technology* 24 (1998), Nr. 4, S. 14–16

[21] Liddy, E. d.: Natural Language Processing. In: *In Encyclopedia of Library and Information Science* 2nd Ed (2001)

[22] Lilac, A. E. A.-S. : Natural Language Processing for Conceptual Modeling. In: *International*

Journal of Digital Content Technology and its Applications 3 (2009), Nr. 3

[23] Lückehe, D. : *Evolutionary Wind Turbine Placement Optimization with Geographical Constraints*. Wiesbaden : Springer Fachmedien Wiesbaden, 2017

[24] Marjaie, S.-A. ; Rathod, U. : Communication in Agile Software Projects: Qualitative Analysis using Grounded Theory in System Dynamics. In: *Proceedings of the International Conference of the System Dynamics Society 2011* (2011)

[25] Niinimäki, T. ; Piri, A. ; Lassenius, C. : Factors Affecting Audio and Text-Based Communication Media Choice in Global Software Development Projects. In: *Proceedings of the Fourth IEEE International Conference on Global Software Engineering, 2009*. Piscataway, NJ : IEEE, 2009, S. 153–162

[26] Noll, S. : *Aufwand und Nutzen verschiedener Kommunikationswege in der Software-Entwicklung*. 2018. – Leibniz Universität Hannover, Fachgebiet Software Engineering, Bachelorarbeit

[27] Prenner, N. ; Klünder, J. ; Schneider, K. : Making meeting success measurable by participants' feedback. In: Begel, A. (Hrsg.) ; Serebrenik, A. (Hrsg.) ; Graziotin, D. (Hrsg.): *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering - SEmotion '18*. New York, New York, USA : ACM Press, 2018, S. 25–31

[28] Remus, R. ; Quasthoff, U. ; Heyer, G. : SentiWS - A Publicly Available German-language Resource for Sentiment Analysis. In: *Citeseer*. 2010

[29] Rothlauf, F. : *Design of Modern Heuristics: Principles and Application*. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2011 (Natural Computing Series). – 2011

[30] Sawyer, S. : Effects of intra-group conflict on packaged software development team performance. In: *Information Systems Journal* 11 (2001), Nr. 2, S. 155–178

[31] Schneider, K. ; Klünder, J. ; Kortum, F. ; Handke, L. ; Straube, J. ; Kauffeld, S. : Positive affect through interactions in meetings: The role of proactive and supportive statements. In: *The Journal of Systems & Software* 2018 (2018)

[32] Scholze-Stubenrecht, W. (Hrsg.) ; Wermke, M. (Hrsg.): *Der Duden*. Bd. in zwölf Bänden; das Standardwerk zur deutschen Sprache / hrsg. vom Wiss. Rat der Dudenred.: Matthias Wermke ... ; Bd. 1: *Duden - die deutsche Rechtschreibung: Das umfassende Standardwerk auf der Grundlage der aktuellen amtlichen Rechtschreibregeln;*. 25., völlig neu bearb. und erw. Aufl., [Nachdr.]. Mannheim : Dudenverl., 2010

[33] Schuh, T. ; Dreiseitl, S. : Evaluating Novel Features for Aggressive Language Detection. In: Karpov, A. (Hrsg.) ; Jokisch, O. (Hrsg.) ; Potapova, R. (Hrsg.): *Speech and Computer* Bd. 11096. Cham : Springer International Publishing, 2018, S. 585–595

[34] Schulz von Thun, F. : *Rororo*. Bd. 17489: *Miteinander reden 1: Störungen und Klärungen. Allgemeine Psychologie der Kommunikation*. 47. Aufl. Reinbek bei Hamburg : Rowohlt-Taschenbuch-Verl., 2009

[35] Seedall, M. ; Macfarlane, K. ; Holmes, V. : SafeChat System with Natural Language Processing and Deep Neural Networks. In: *Emerging Technology Conference 2019* (2019)

[36] Shakeri Hossein Abad, Z. ; Gervasi, V. ; Zowghi, D. ; Barker, K. : ELICA: An Automated Tool for Dynamic Extraction of Requirements Relevant Information. In: Groen, E. C. (Hrsg.) ; Harrison, R. (Hrsg.) ; Murukannaiah, P. K. (Hrsg.) ; Vogelsang, A. (Hrsg.): *2018 5th International Workshop on Artificial Intelligence for Requirements Engineering*. Piscataway, NJ : IEEE, 2018, S. 8–14

[37] Stapel, K. ; Schneider, K. : FLOW-Methode-Methodenbeschreibung zur Anwendung von FLOW. (2012). – Technischer Bericht, Leibniz Universität Hannover, Fachgebiet Software Engineering

[38] Suvethan, N. ; Avenash, K. ; A Q Huzaim, M. ; Mathusagar, R. ; P A W Gamage, M ; Imbulpitiya, A. : Virtual Student Advisor using NLP and Automatic Appointment Scheduler and Feedback Analyser. In: *International Journal of Scientific and Engineering Research* 7 (2019)

[39] Wallat, J. : *Vergleich von Algorithmen zur Informationsflussanalyse in der Software-Entwicklung*. 2017. – Leibniz Universität Hannover, Fachgebiet Software Engineering, Bachelorarbeit

[40] Waltinger, U. : Sentiment Analysis Reloaded: A Comparative Study On Sentiment Polarity Identification Combining Machine Learning And Subjectivity Features. In: *Proceedings of the 6th International Conference on Web Information Systems and Technologies (WEBIST '10)*. Valencia, Spain, 2010

[41] Wang, H. ; Castanon, J. A.: Sentiment expression via emoticons on social media. In: Ho, H. (Hrsg.): *2015 IEEE International Conference on Big Data*. Piscataway, NJ : IEEE, 2015, S. 2404–2408

[42] WOHLIN, C. ; RUNESON, P. ; HÖST, M. ; OHLSSON, M. C. ; REGNELL, B. ; WESSLEN, A. : *Experimentation in software engineering*. Springer Science & Business Media, 2012

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig 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 08.10.2019

Julian Horstmann

Anhang

1. Transparenzschreiben an die Mitarbeiter des Softwareunternehmens (geschwärzt)

Liebes Teammitglied,

mein Name ist Julian Horstmann. Ich bin Informatik-Masterstudent an der Gottfried Wilhelm Leibniz Universität Hannover. Im Rahmen meiner Masterarbeit zum Thema „Analyse digitaler Kommunikation in Entwicklerteams“ im Fachgebiet für Software Engineering, bin ich mit der Bitte an [redacted] herangetreten, mir Daten für meine Analyse zur Verfügung zu stellen. Konkret beschäftigt mich bei meiner Masterarbeit folgende Fragestellung: Wie lässt sich das Kommunikationsverhalten in Entwicklerteams analysieren, um Rückschlüsse auf die Zusammenarbeit abzuleiten? Um computergestützt Antworten auf diese Frage zu finden, habe ich mir einige Metriken überlegt, welche ich auf die digitale textuelle Kommunikation von Entwicklerteams anwenden möchte. Dabei sind von mir folgende Metriken angestrebt:

- Reaktionszeit (Wie schnell wird geantwortet?)
- Frequenz der Kommunikation (Wie häufig wird kommuniziert?)
- Nachrichtenlängen (Wie lang sind die Nachrichten?)
- Verwendung von Emoticons (Werden Emoticons verwendet? Welche Wirkung haben die verwendeten Emoticons?)
- Förmlichkeit der Konversation (Wird mit einer förmlichen Anrede gearbeitet? Wird viel Wert auf Rechtschreibung gelegt?)
- Inhaltliche Analyse (Welche emotionale Färbung hat die Kommunikation?)

Nach einigen Gesprächen mit der [redacted] wurde klar, dass eine Kooperation in diesem Zusammenhang unter strengen Bedingungen bezüglich Daten- und Geheimnisschutz denkbar wäre. Dazu habe ich mich mit der Interessenvertretung und der Datenschutzbeauftragten der [redacted] zusammengesetzt, um den gesetzlichen Vorgaben und dem hohen Anspruch Ihrer Kollegen gerecht zu werden. Sowohl eine umfängliche Anonymisierung der gesamten Kommunikation vor dem Beginn der Analyse als auch das Unterzeichnen umfänglicher Geheimhaltungs- und Datenschutzverpflichtungen waren selbstverständlich. Außerdem sollte die Verarbeitung der Daten ausschließlich auf [redacted]-Systemen durchgeführt werden. Ein Export von Ergebnissen und Datenauszügen erfolgte nur in Abstimmung mit [redacted] (dem Ansprechpartner für die Studenten bei der [redacted]).
Für die Analyse war es lediglich notwendig, einzelne Konversationen (Streams) über einen begrenzten Zeitraum zu verarbeiten. Bei der Anonymisierung der Daten wurden Absender und Empfänger sowie etwaige Erwähnungen von Namen oder Verlinkungen in den Nachrichten pseudonymisiert. Um eine Rückführung auf einzelne Pseudonyme oder Personen zu vermeiden, wurden die Ergebnisse der Analyse nur über Gruppen von Pseudonymen gebildet.

Was hat die Verarbeitung mit Ihnen zu tun?
Nach umfänglicher Beratung und Prüfung, welches Projekt sich für meine Masterarbeit eignen würde, fiel die Wahl auf „[redacted]“ ([redacted]). In diesem Projekt wird seit Monaten das Kommunikationstool Zulip verwendet, welches die Möglichkeit bietet, Daten über eine RestAPI abzufragen. Mir und den mit meinem Anliegen betrauten Personen bei der [redacted] war es sehr wichtig, alle betroffenen Personen umfänglich über mein Vorhaben zu informieren, insbesondere bevor Daten verarbeitet werden. Da Sie Mitglied dieses Projektteams sind, erreicht Sie dieses Schreiben.

Dieses Schreiben wurde nicht nur der Form halber geschrieben und soll auch niemanden vor vollendete Tatsachen stellen. Sollten Sie nicht einverstanden sein, oder ernsthafte Bedenken diesem Vorhaben gegenüber haben, so werden die Streams, an denen Sie aktiv beteiligt sind, nicht verwendet.

Ich hoffe, dass Sie mich bei meiner Masterarbeit unterstützen und bedanke mich hierfür bereits im Voraus.

Sollten noch Fragen, Anregungen oder Bedenken bestehen, so bin ich jederzeit unter folgender Mail zu erreichen: j.horstmann@stud.uni-hannover.de

Julian Horstmann