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

Verbesserung von evolutionären Algorithmen zur Klassifikation von schriftlicher Kommunikation in Entwicklungsteams

Improving evolutionary algorithms for classifying text-based communication in development teams

Bachelorarbeit

im Studiengang Informatik

von

Christian Meyer

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

Hannover, 23. November 2020

Erklärung der Selbstständigkeit

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

Hannover, den 23. November 2020

_____

Christian Meyer

Zusammenfassung

Kommunikation stellt einen wichtigen Faktor im Prozess der Softwareentwicklung dar. Diesen Faktor adressierend, entwarf Horstmann [14] ein Konzept für die Analyse textueller Kommunikation in Entwicklerteams. Dieses Konzept, das ein Ensemble von Klassifizierungsalgorithmen in Kombination mit einem evolutionären Algorithmus für die Merkmalsselektion verwendet, erzielte im Rahmen einer Auswertung durch Horstmann [14] eine Klassifizierungsgenauigkeit von 63%.

In dieser Arbeit wird das Konzept von Horstmann [14] genauer betrachtet und auf Möglichkeiten der Verbesserung der Klassifizierungsgenauigkeit untersucht. Im Rahmen einer initialen Auswertung des Konzeptes unter Verwendung eines Goldstandard-Datensatzes erzielt dieses eine Genauigkeit von 82%. Aufbauend auf diesen Ergebnissen werden sowohl die Klassifizierungsalgorithmen als auch die Merkmalsselektion und die Vorverarbeitungsschritte des Konzeptes analysiert. Basierend auf den erzielten Einblicken in das Konzept von Horstmann [14] werden Verbesserungen für einen Großteil der ermittelten Probleme identifiziert und implementiert. Die Verbesserungen umfassen dabei Ergänzungen an der Merkmalsmenge, die Implementierung eines domänenspezifischen Lexikons und eine Methode für die Verarbeitung von Negationen. Hinzu kommen Anpassungen an der Auswahl der Klassifizierungsalgorithmen und ihrer Hyperparameter sowie eine Erweiterung des evolutionären Algorithmus'. Diese Anpassungen evaluierend, wird der Einfluss der einzelnen Änderungen auf die Klassifizierungsgenauigkeit untersucht und eine Verbesserung der Genauigkeit um 5,1%-Punkte festgestellt, wobei nicht alle Änderungen positiv zu der Genauigkeit beitragen. Abschließend werden Schlussfolgerungen aus der Evaluation gezogen und auf weitere Möglichkeiten der Verbesserung der Klassifizierungsperformanz des Konzeptes von Horstmann [14] eingegangen.

Abstract

Improving evolutionary algorithms for classifying text-based communication in development teams

Communication plays an important role in the process of software engineering. Addressing the importance of this factor, Horstmann [14] developed a concept for analyzing textual communication in teams of software developers. This concept, using an ensemble of supervised learning algorithms in combination with an evolutionary algorithm for feature selection, achieved a classification accuracy of 63% as shown by Horstmann [14].

Addressing this issue, this paper uses an analytic approach to exploring the concept of Horstmann [14]. Following an initial evaluation of the concept on a gold standard dataset, an accuracy of 82% is observed. Using quantitative as well as qualitative aspects of analysis, issues regarding the classification algorithms as well as the feature selection and the preprocessing are identified. Following this process, appropriate changes are defined and implemented. These changes include an extension of the list of extracted features, the implementation of a domain-specific sentiment lexicon and a method for handling negations. Additionally, changes to the selection of classification algorithms and their parameters are made and the evolutionary algorithm used for feature selection is extended. Subsequently, an evaluation into the impact of the changes is conducted, and an increase in classification accuracy of 5.1 percentage points is observed. The paper ends with a summary of the lessons learned, as well as a discussion of possible further improvements to the classification accuracy of the concept defined by Horstmann [14].

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Für den Prozess der Softwareentwicklung sind Aspekte sozialer Interaktion, sei es zwischen Entwicklern und Kunden oder Entwicklern untereinander ähnlich wichtig wie die technischen Aspekte dieses Prozesses [19]. So nehmen diese sozialen Faktoren wie „Kommunikation, Kooperation und Koordination“ [19, S. 1] einen hohen Stellenwert in der Domäne der Softwareentwicklung ein. Besonders im Kontext global verteilter Entwicklerteams steht dabei die Kommunikation häufig in direktem Verhältnis zum Erfolg komplexer Softwareprojekte [9].

Aufgrund dieser Tatsache ist eine Analyse der Kommunikation in Entwicklerteams eine Möglichkeit, die Qualität der Kommunikation zu erfassen und falls notwendig Maßnahmen einzuleiten, um potenzielle Fehlstände auszugleichen [25].

Dabei findet Kommunikation über eine Vielzahl von Kommunikationskanälen statt [14]. Besonders digitale textuelle Kommunikation stellt, aufbauend auf dem Trend global verteilter Entwicklerteams, eine wichtige Form dieser Kommunikation dar [13]. Häufig arbeiten die Teams dabei über große Entfernungen und somit auch über globale Zeitzonen hinweg, wodurch herkömmliche Formen der Kommunikation (z.B. Meetings) häufig erschwert werden [13].

Eine Möglichkeit der Untersuchung textueller Kommunikation ist die „Sentimentanalyse“. Im Rahmen dieser Methode werden die subjektiven Aspekte der Kommunikation wie Emotion oder Sentiment betrachtet [31]. Diese Methodik wurde dabei bereits erfolgreich angewandt [14], weil Entwickler nicht nur Emotionen empfinden, die eine direkte Auswirkung auf Faktoren wie Produktivität oder Zufriedenheit haben [38, 12], sondern diese Emotionen auch in textuellen Artefakten der Softwareentwicklung hinterlassen [18, 29]. Eine Untersuchung dieser subjektiven Gesichtspunkte erlaubt dann nicht nur einen Rückschluss auf die Qualität der Kommunikation, aber auch auf die Stimmung innerhalb eines Teams [14].

1.1. Problemstellung

Im Rahmen seiner Masterarbeit erstellte Horstmann [14] ein Konzept für die Analyse deutschsprachiger textueller Kommunikation in Softwareentwicklerteams. Dieses Konzept befasst sich dabei mit der Analyse des Sentiments dieser Kommunikationsmethode. Hierzu verwendet das Konzept Methoden des maschinellen Lernens, um Textdokumente bezüglich der in ihnen beobachtbaren „Polarität“ in die Klassen „positiv“, „neutral“

und „negativ“ einzuordnen [14]. Auf Basis eingegebener Texte werden verschiedene Klassen von Merkmalen aus den Texten extrahiert und für die Klassifizierung über ein Ensemble, bestehend aus drei Klassifizierungsalgorithmen, verwendet [14]. Auf Basis dieser Algorithmen erfolgt außerdem eine Selektion der extrahierten Merkmale im Hinblick auf eine Optimierung der Klassifizierungsgenauigkeit [14]. Für diesen Schritt verwendet Horstmann [14] einen evolutionären Algorithmus.

Im Kontext einer Auswertung der Klassifizierung auf einem Datensatz von 3788 Textdaten eines Entwicklerteams aus der Privatwirtschaft stellte Horstmann [14] fest, dass die Klassifizierungsalgorithmen lediglich eine Genauigkeit von 63% erzielten. Da diese geringe Genauigkeit keine zuverlässige Anwendung des Konzeptes erlaubt, werden in dieser Arbeit die Methoden von Horstmann [14] aufgegriffen und auf Möglichkeiten der Verbesserung der Klassifizierungsgenauigkeit untersucht.

1.2. Lösungsansatz

Im Kontext der Verbesserung der Klassifizierungsgenauigkeit sollen die folgenden Forschungsfragen (RQ) gestellt und im Verlauf dieser Arbeit beantwortet werden:

RQ1 Welche Ansätze bieten die Vorverarbeitung und die Klassifizierung für eine Verbesserung der Klassifizierungsperformanz?

RQ2 Welchen Einfluss haben ausgewählte Verbesserungen auf die Klassifizierungsperformanz?

Die erste Forschungsfrage (RQ1) zielt darauf ab, zunächst mögliche Probleme in den für die Klassifizierung relevanten Schritten der Vorverarbeitung und der Klassifizierung des Konzeptes zu ermitteln, um anhand dieser Einblicke Verbesserungen zu definieren. Die im Rahmen dieser Arbeit betrachteten Verbesserungen werden dann, um die zweite Forschungsfrage (RQ2) zu beantworten, auf ihren Einfluss auf die Klassifizierungsperformanz untersucht. Hierdurch wird überprüft, ob die Änderungen tatsächlich Verbesserungen hervorbringen oder ob bei den Änderungen selbst Probleme bestehen, die in zukünftigen Iterationen des Konzepts betrachtet werden müssten und die gegen eine Implementierung besagter Änderungen sprechen.

Der Begriff der Performanz sei im Kontext dieser Arbeit als Überbegriff für die in Abschnitt 2.3.4 formulierten Metriken des F_1 und der Genauigkeit definiert.

Um mögliche Probleme der betrachteten Schritte zu ermitteln, wird das Konzept von Horstmann [14] zunächst auf einem englischen Goldstandard-Datensatz aus der Domäne der Softwareentwicklung trainiert und ausgewertet. Diese Entscheidung ist auf die Tatsache zurückzuführen, dass der von Horstmann [14] im Rahmen seiner Auswertung verwendete deutschsprachige Datensatz nicht mehr verfügbar ist und nach aktueller Kenntnis kein deutschsprachiger Datensatz für die Sentimentanalyse im Bereich der Softwareentwicklung existiert. Auf den Ergebnissen dieser Auswertung

aufbauend, werden die verwendeten Klassifizierungsalgorithmen sowie die Fehlklassifizierungen der Algorithmen genauer betrachtet. Darauf folgend wird der für die Optimierung verwendete evolutionäre Algorithmus bezüglich der Auswirkung auf die Klassifizierungsperformanz und seine Qualität als Suchalgorithmus untersucht. Auf den Ergebnissen dieser Analyse aufbauend werden Verbesserungsmöglichkeiten identifiziert und implementiert. Zuletzt findet eine Auswertung der einzelnen Änderungen im Hinblick auf die Klassifizierungsperformanz statt, um eine Antwort auf die zweite Forschungsfrage ableiten zu können.

1.3. Ergebnisse der Arbeit

In dieser Arbeit werden Probleme des Konzeptes von Horstmann [14] identifiziert. Diese umfassen generelle Fehler in der Vorverarbeitung und die fehlende Verarbeitung von Kontext, Negationen und domänenspezifischem Vokabular. Hinzu kommen Aspekte der Klassifizierer, die sich in unterdurchschnittlicher Performanz einzelner Klassifizierer und einer Überanpassung der restlichen Klassifizierer äußern. Der von Horstmann [14] formulierte evolutionäre Algorithmus wertet nur eine geringe Menge ähnlich strukturierter Lösungen aus. Die identifizierten Verbesserungen bezüglich dieser Probleme umfassen Ergänzungen und Anpassungen der Vorverarbeitung, eine Auswahl genauerer Klassifizierungsalgorithmen und eine Optimierung der Hyperparameter dieser. Der evolutionäre Algorithmus wird um Operatoren der „Selektion“ und „Rekombination“ erweitert. Im Kontext der Auswertung des Konzeptes von Horstmann [14] auf einem Goldstandard-Datensatz erzielt dieses eine Genauigkeit von ca. 82%. Die Implementierung der definierten Änderungen verbessert die Genauigkeit um 5,1%-Punkte und führt somit zu einer Genauigkeit von ca. 87%.

1.4. Struktur der Arbeit

Zunächst werden für das weitere Verständnis erforderliche Grundlagen formuliert und verwandte Arbeiten beschrieben. Darauf folgt eine ausführliche Beschreibung des Konzeptes von Horstmann [14] und der notwendigen initialen Anpassungen des Konzeptes im Kontext der Auswertung dieses in der vorliegenden Arbeit. Basierend auf den initialen Anpassungen des Konzepts werden die Klassifizierungsalgorithmen und der evolutionäre Algorithmus analysiert und Schlussfolgerungen aus der Analyse gezogen. Eingehend auf diese Schlussfolgerungen werden in Kapitel 6 die identifizierten Verbesserungen formuliert und die konkrete Implementierung dieser besprochen. In Kapitel 7 erfolgt die Evaluation der implementierten Verbesserungen sowie die Interpretation und Diskussion der Ergebnisse dieser Arbeit. Zuletzt erfolgt eine Zusammenfassung der Arbeit sowie ein Ausblick auf weitere Verbesserungs- und Untersuchungsmöglichkeiten des Konzeptes von Horstmann [14].

2. Grundlagen

Diese Arbeit befasst sich hauptsächlich mit dem Konzept der Sentimentanalyse bzw. der Sentimentklassifizierung. Dementsprechend werden in diesem Kapitel die Grundlagen der Sentimentanalyse und der Klassifizierung vermittelt. Ergänzend wird außerdem die generelle Funktionsweise evolutionärer Algorithmen beschrieben.

2.1. Sentimentanalyse

Natürlichsprachliche Texte enthalten häufig vom Verfasser eingebaute subjektive Aspekte wie Emotionen und Sentiment (definiert als: „Empfindung“ oder „Gefühl“ [3]) [31]. Das Verfahren der Sentimentanalyse untersucht eben dieses Sentiment und versucht „[...] die Stimmung von Personen anhand ihres Sprachausdrucks abzuleiten.“ [20, Kapitel 26, S. 633]. Horstmann [14] betrachtet im Rahmen des von ihm definierten Konzeptes eine Dimension dieses Sentiments, nämlich die Polarität eines Textes. Im Kontext der Klassifizierung dieser Polarität werden Textdokumente, basierend auf ihrer „Färbung“, in eine der drei Polaritätsklassen „positiv“, „neutral“ und „negativ“ eingeordnet [14]. Tabelle 2.1 zeigt jeweils ein Beispiel für die Polaritätsklassen, entnommen aus dem Datensatz von Novielli et al. [27]. Der Übersicht halber werden die Begriffe „Sentiment“ und „Polarität“ in dieser Arbeit für denselben Aspekt natürlicher Sprache verwendet.

Kommentar	**Polaritätsklasse**
„Great addition, thanks @user.“	positiv
„Would you mind porting this back to evaluate?“	neutral
„So we gave up testing the version? So sad.“	negativ

Tabelle 2.1.: Beispiele für die Polaritätsklassen.

Die Sentimentanalyse stellt einen Teilbereich des *Natural Language Processing* (NLP) dar und baut dementsprechend auf Methoden dieses Bereiches auf [31]. Für das weitere Verständnis der Verarbeitung von Texten im Rahmen der Analyse sind einzelne Definitionen aus dem Bereich des NLP erforderlich.

Ein natürlichsprachlicher Satz beziehungsweise eine Gruppierung von Sätzen, im weiteren Verlauf der Arbeit auch als „Dokument“ bezeichnet, besteht dabei aus sogenannten „Token“. Ein Token kann ein Wort, aber auch eine URL oder eine Gruppierung von Interpunktion z.B. in Form eines Emoticons, darstellen. Ein wichtiger Schritt der Verarbeitung von Dokumenten ist die Zuweisung von *Part-of-Speech*-Tags für jedes

Token eines Dokuments. Dieser Prozess beschreibt das Zuordnen von Wortarten (z.B. Substantiv, Verb) zu den betrachteten Token. Auf diesen Schritt der Verarbeitung aufbauend, ist im Konzept von Horstmann [14] eine Lemmatisierung ermittelter Token vorgesehen. Das Lemmatisieren eines Tokens steht dabei für das Rückführen eines Wortes auf die Grundform des Wortes (z.B.„he/she/it went“ zu „he/she/it go“). Dieser Schritt vereinfacht die Verwendung der später aufgeführten Sentimentlexika (siehe Abschnitt 4.1.2). Zuletzt ist das Konzept der „Stoppwörter“ aufzuführen, welche im Kontext dieser Arbeit aus betrachteten Texten entfernt werden. Stoppwörter sind dabei Wörter, die in natürlichsprachlichen Texten sehr häufig auftauchen und keinerlei Informationsgewinn im Rahmen einer Betrachtung des Textes darstellen (z.B. „the“ oder „a“ im Englischen).

2.2. Evolutionäre Algorithmen

Evolutionäre Algorithmen (EA) stellen eine Kategorie von Suchalgorithmen dar, die Ideen der Darwinschen Evolutionstheorie für das Lösen von Such- bzw. Optimierungsproblemen verwenden [39]. Mögliche Lösungen des betrachteten Problems, die als „Individuen“ bezeichnet werden, werden unter Verwendung von Mechanismen der Selektion, Rekombination und Mutation verarbeitet [39]. Auf diese Art werden Änderungen an existierenden Individuen vorgenommen bzw. neue Individuen generiert. Das Ziel dieser Änderungen ist das Generieren globaler Optima des betrachteten Optimierungsproblems [39]. Die Qualität einer Lösung bzw. eines Individuums ergibt sich dabei aus einer ausgewählten „Fitnessfunktion“ und stellt einen wichtigen Faktor für die Methodik der Selektion dar [39].

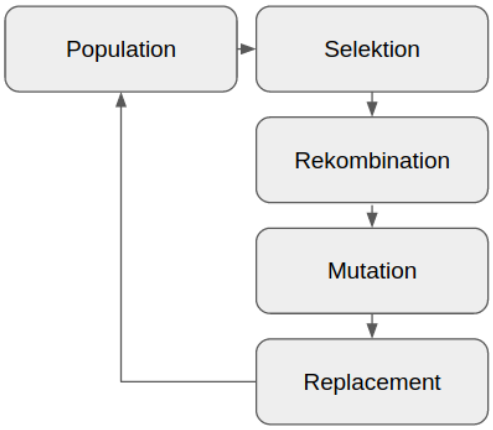

Abbildung 2.1.: Beschreibung des Durchlaufs einer Generation eines evolutionären Algorithmus’.

Abbildung 2.1 zeigt den Ablauf einer Generation eines EA. Basierend auf einer initialen Population werden solange Individuen aus der Population ausgewählt, rekombiniert,

mutiert und ersetzt, bis ein vordefiniertes Abbruchkriterium, wie z.B. eine bestimmte Anzahl an Generationen, erreicht ist [39].

2.3. Klassifizierung

In dieser Arbeit wird ein Teilproblem des überwachten maschinellen Lernens betrachtet, die Klassifizierung. Die Klassifizierung beschreibt das Einordnen von Eingabedaten in zwei oder mehr Klassen. Ein Klassifizierungsalgorithmus ist dann ein Algorithmus, der nach einem Trainingsprozess dazu in der Lage ist, eine Klassifizierung in diese vordefinierten Klassen durchzuführen. Im Kontext einer Klassifizierung im Bereich des Natural Language Processing werden dabei basierend auf den textuellen Eingabedaten „Merkmale“ dieser Eingaben erhoben. Diese Merkmale reichen von einfachen Textcharakteristika wie z.B. der durchschnittlichen Wortlänge, bis hin zu Vektorrepräsentationen der Eingabedaten. Diese ausgewählten Merkmale werden für jedes Eingabedokument ermittelt und stellen die Eingabe der Klassifizierer dar.

2.3.1. Klassifizierungsalgorithmen

Im Kontext des Konzeptes von Horstmann [14] werden die folgenden drei Klassifizierungsalgorithmen verwendet:

- **Support Vector Machine (SVM):** Eine SVM betrachtet die Eingabedaten als Vektoren innerhalb eines hochdimensionalen Vektorraumes und versucht eine Hyperebene zu ermitteln, welche die Eingabedaten innerhalb dieses Vektorraumes möglichst fehlerfrei in die jeweiligen Klassen trennt [11].
- **Random Forest (RF):** Ein RF stellt eine Menge zufällig generierter Entscheidungsbäume dar [11]. Diese Bäume werden während des Trainingsprozesses des Klassifizierers basierend auf den Merkmalen der Eingabedaten aufgebaut. Die Knoten dieser Entscheidungsbäume stellen ausgewählte Merkmale und die Blätter der Bäume die betrachteten Klassenbezeichnungen dar [11]. Soll nun ein Dokument klassifiziert werden, geben alle Bäume des RF eine Klassifikation ab. Das Ergebnis des RF für ein Dokument ist die Klasse, die von den Bäumen des RF am häufigsten gewählt wurde [11].
- **Gaussian Naive Bayes (GNB):** Ein GNB nimmt eine Normalverteilung aller Merkmale der Eingabedaten an [11]. Basierend auf den Erwartungswerten und Standardabweichungen aller in den Trainingsdaten betrachteten Merkmale bestimmt der GNB die Wahrscheinlichkeit der Zugehörigkeit eines Dokuments zu einer Klasse [11]. Die Klasse mit der höchsten Wahrscheinlichkeit entspricht dann der Klassifikation des GNB [11].

2.3.2. Ensemble der Algorithmen

Eine Möglichkeit, Kombinationen verschiedener Klassifizierer zu bilden besteht, darin, die Klassifizierer bezüglich des finalen Ergebnisses einer Eingabe abstimmen zu lassen. Diese Methode wird im Folgenden als VotingClassifier (VC) bezeichnet, basierend auf der Bezeichnung der Methode im von Horstmann [14] verwendeten scikit-learn[1] Python-Paket. Innerhalb eines VC werden die enthaltenen Klassifizierer unabhängig voneinander, aber anhand derselben Trainingsdaten trainiert. Auf gleiche Art werden die Algorithmen auch getestet. Dabei wird für jedes Dokument, von jedem Klassifizierer eine Klassifikation abgeben. Der VC kombiniert dann die jeweiligen Ausgaben durch Mehrheitsentscheidung, um ein Gesamtergebnis für das Ensemble zu bestimmen.

Ein Unentschieden z.B. bei jeweils unterschiedlichen Ausgaben für die drei Klassen durch die drei Klassifizierer, entscheidet der intern zuletzt aufgeführte Klassifizierer[2].

2.3.3. Kreuzvalidierungsverfahren

Die Kreuzvalidierung (KV) ist eine Methode zur Auswertung von Modellen des maschinellen Lernens [11]. In dieser Arbeit wird die stratifizierte k-fache KV verwendet. In dieser Form der KV wird der verwendete Datensatz in k möglichst gleich große und voneinander disjunkte Mengen (*Folds*) unterteilt [11]. Das Modell wird dann auf $(k-1)$ Folds trainiert und auf dem verbleibenden Fold ausgewertet [11]. Dieser Prozess wird k mal wiederholt, sodass jeder Fold genau ein mal zum Testen der Modellperformanz verwendet wird [11]. Betrachtete Metriken werden für jede Durchführung ermittelt und anschließend gemittelt [11]. Bei der *stratifizierten* KV wird außerdem darauf geachtet, dass die Verteilung der betrachteten Klassen innerhalb aller Folds der Klassenverteilung des gesamten Datensatzes möglichst ähnelt [11]. Diese Methode der Auswertung erlaubt es, die tatsächliche Performanz eines Modells genauer zu approximieren und durch die Wahl des Datensatzes bedingte Störfaktoren zu minimieren [11]. KV erlaubt außerdem eine bessere Auswertung kleiner Datensätze, da jedes Element des Datensatzes entweder zum Training, oder zum Testen verwendet wird [11].

2.3.4. Metriken

Um die Modelle des maschinellen Lernens in dieser Arbeit auszuwerten, werden die Metriken *Precision*, *Recall*, F_1 und *Accuracy* ermittelt. Diese Metriken basieren dabei auf den folgenden Gruppierungen der Ergebnisse [21]: *True Positives* (TP) sind die Beispiele einer betrachteten Klasse, die von einem Modell korrekt in diese eingeordnet werden. *True Negatives* (TN) sind die Beispiele einer fremden Klasse, die korrekt in diese fremde Klasse eingeordnet werden. *False Positives* (FP) sind Beispiele einer anderen Klasse, die fälschlicherweise der betrachteten Klasse zugeordnet werden. *False Negatives* (FN)

[1] https://scikit-learn.org/stable/
[2] Siehe https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

sind Beispiele der betrachteten Klasse, die vom Modell einer anderen Klasse zugeordnet wurden. Die Metriken sind dann wie folgt definiert [21]:

Precision (P): Precision beschreibt den Anteil der korrekt klassifizierten Elemente der betrachteten Klasse in Relation zur Gesamtzahl der in diese Klasse eingeordneten Beispiele.

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Recall (R): Recall beschreibt den Anteil der korrekt klassifizierten Elemente einer Klasse im Bezug auf die Gesamtzahl der Elemente der Klasse.

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

F-Score (F): Der F_1-Score ($\beta = 1$) fasst Precision und Recall gewichtet zusammen.

$$F_\beta = (1 + \beta^2)\frac{P * R}{P + R}$$

Accuracy (A): Accuracy, oder auch Genauigkeit beschreibt den Anteil korrekt klassifizierter Elemente eines Datensatzes in Relation zur Gesamtanzahl der Dokumente eines Datensatzes.

$$\mathrm{A} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Die Metriken Precision, Recall und F_1 werden in dieser Arbeit für die Beschreibung der klassenspezifischen Performanz eines Klassifizierers betrachtet, während die Genauigkeit die klassenübergreifende Performanz beschreibt.

3. Verwandte Arbeiten

Diese Arbeit behandelt die Verbesserung eines Konzeptes, das für die Sentimentanalyse in der Domäne der Softwareentwicklung konzipiert wurde. Zunächst werden Anwendungsfälle der Sentimentanalyse im Bereich der Softwareentwicklung aufgeführt. Darauf folgt eine kurze Beschreibung der Arbeit von Horstmann [14] sowie ähnlicher Anwendungen und Konzepte.

3.1. Anwendbarkeit und Domänenspezifität

Folgende Arbeiten vermitteln einen Eindruck über die Anwendbarkeit der Sentimentanalyse und die Relevanz domänenspezifischer Anwendungen für die Betrachtung der Rolle von Emotionen in der Softwareentwicklung.

Jurado et al. [18] untersuchten im Rahmen einer explorativen Studie den Ansatz der Sentimentanalyse für die Anwendung und Forschung im Bereich der Softwareentwicklung. Hierzu überprüften sie über 10.000 GitHub-Kommentare auf ihr Sentiment unter Verwendung mehrerer Emotions- und Sentimentlexika. Ihre Untersuchungen ergaben, dass die betrachteten Kommentare tatsächlich Sentiment enthalten. Eine Anwendung komplexerer Ansätze des NLP stellt somit nach Aussage der Autoren eine sinnvolle Möglichkeit der Untersuchung dieses Sentiments und der Rolle von Emotionen in der Softwareentwicklung dar.

Ortu et al. [29] untersuchten den Einfluss von Affekt (Emotionen, Sentiment und Höflichkeit) auf die Produktivität in der Softwareentwicklung. Im Rahmen dieser Untersuchungen wurden Kommentare aktiver Jira-Projekte auf Affekt untersucht und ein Zusammenhang zu der Bearbeitungszeit von Problemen hergestellt. Dabei stellten sie fest, dass das Auftreten verschiedener Emotionen und Grade von Höflichkeit in Relation zu der beobachtbaren Problembearbeitungszeit stehen. So steht z.B. das Auftreten von Emotionen wie „Freude“ in Relation mit einer Verringerung der beobachtbaren Bearbeitungszeit.

Jongeling et al. [17] werteten in ihrer Untersuchung vier herkömmliche Sentimentanalysetools auf einem Datensatz von 392 Textdokumenten aus der Softwareentwicklung aus. Sie stellten fest, dass die Ergebnisse dieser Anwendungen weder untereinander noch mit den Klassenzuweisungen des Datensatzes übereinstimmen. Aufbauend auf dieser Beobachtung weisen sie auf die Notwendigkeit domänenspezifischer Analysetools für weitere Forschung im Bereich der Softwareentwicklung hin.

Auch Novielli et al. [28] weisen als Ergebnis ihrer Untersuchungen auf die Notwendigkeit domänenspezifischer Sentimentanalyse für die Forschung in der Softwareentwicklung hin. Im Rahmen einer qualitativen Betrachtung des Sentiments von StackOverflow-Daten stellten die Autoren fest, dass Sentiment bzw. Polarität nur „ein“ wichtiger Faktor der von Entwicklern verfassten Emotionen darstellt. In diesen Untersuchungen sind den Autoren außerdem die Häufigkeit der Verwendung domänenspezifischen Vokabulars und die schlechten Ergebnisse „herkömmlicher“ Sentimentanalysetools im Kontext dieser aufgefallen.

3.2. Konzepte und Anwendungen

In diesem Abschnitt wird das Konzept von Horstmann [14] vorgestellt sowie ähnliche Anwendungen und Konzepte aufgegriffen.

Horstmann [14] stellte im Kontext seiner Masterarbeit ein Konzept für die Analyse des Kommunikationsverhaltens in Softwareentwicklerteams vor. Das Konzept beschreibt dabei sämtliche Anwendungsschritte, die dafür notwendig sind, die Sentimentanalyse in einem tatsächlichen Softwareprojekt anzuwenden. Für die Klassifizierung textueller Daten sieht Horstmann die Einordnung besagter Daten bezüglich ihrer Polarität im Rahmen einer Sentimentanalyse vor. Der Aspekt der Sentimentanalyse beruht dabei auf Methoden des überwachten maschinellen Lernens und baut auf eine Extraktion mehrerer Merkmalsklassen auf. So werden neben Emotions- und Emoticonmerkmalen auch Formalitätsmerkmale, statistische Merkmale, sowie *Bag-of-Words*-Merkmale extrahiert (siehe Abschnitt 4.1.2). Diese stellen die Eingabe für ein Klassifiziererensemble, bestehend aus einer *Support Vector Machine*, einem *Random Forest* und einem *Gaussian Naive Bayes*, dar. Ergänzend verwendet das Konzept einen evolutionären Algorithmus, um im Rahmen einer Merkmalsselektion eine optimale Merkmalsmenge zu bestimmen. Im Kontext einer Auswertung dieses Konzeptes erhob Horstmann [14] 3788 Dokumente anhand textueller Kommunikationsdaten eines Entwicklerteams aus der Privatwirtschaft und stellte fest, dass sich die Stimmung des betrachteten Entwicklerteams tatsächlich anhand des Sentiments ableiten ließ.

Als Reaktion auf das Problem der fehlenden Domänenspezifität in gängigen Sentimentanalysetools und um die Rolle von Emotionen in der Entwicklung genauer erforschen zu können, entwickelten unter anderem Calefato et al. [6] die Anwendung Senti4SD. Senti4SD verwendet dabei Methoden des maschinellen Lernens und baut auf Merkmalen basierend auf sowohl Sentiment- und Emoticonlexika, der Extraktion von N-Grammen und deren Häufigkeiten sowie Semantik auf, um die Polarität textueller Eingabedaten zu klassifizieren. Hinzu kommen Merkmale, die sogenannte „Microblogging-Aspekte“ wie die durchgehende Großschreibweise von Wörtern, erfassen. Für die Klassifizierung verwendet Senti4SD eine *Support Vector Machine* mit optimierten Hyperparametern, um State-of-the-Art-Ergebnisse auf einem Datensatz von 4800 StackOverflow-Kommentaren, in der domänenspezifischen Sentimentanalyse zu erzielen.

Ähnlich begründet entwickelten Islam et al. [16] das Tool „SentiStrength-SE“. Basierend auf einer qualitativen Analyse des lexikonbasierten und domänen-unabhängigen Analysetools „SentiStrength“ [34] deckten Islam et al. [16] häufige Fehlerquellen der Anwendung auf. Diese Fehlerquellen adressierend, entwarfen sie SentiStrength-SE als domänenspezifische Erweiterung von SentiStrength. SentiStrength-SE wies in einer qualitativen Analyse und einer quantitativen Analyse der Anwendung durch die Autoren eine deutlich bessere Klassifizierungsperformanz als im Bereich der Softwareentwicklung häufig verwendetete Sentimentanalysetools im Kontext einer Auswertung auf einem Datensatz von 5600 Jira-Kommentaren, auf.

Calefato et al. [7] entwickelten die Anwendung „EmoTXT“. EmoTXT erlaubt dabei die Einteilung textueller Daten in ein ausgewähltes Spektrum von Emotionen wie *Freude* oder *Zorn*. Für diese Aufgabe extrahiert die Anwendung eine Reihe an Merkmalen: Mono- und Bigramme, ein Emotionslexikon für die Zuweisung von Wörtern zu Emotionen, anhand von SentiStrength [34] berechnete Sentimentwerte sowie Merkmale für die Höflichkeit und Stimmung der Texte. Ähnlich wie Senti4SD verwendet EmoTXT eine *Support Vector Machine* für die Klassifizierung betrachteter Dokumente. Im Kontext einer Auswertung auf einem StackOverflow-Datensatz (4800 Dokumente) und einem Jira-Datensatz (4000 Dokumente) erzielt die Anwendung akzeptable und vor allem vergleichbare Ergebnisse auf den Datensätzen.

Ahmed et al. [4] stellen in ihrer Arbeit das Sentimentanalysetool „SentiCR“ vor. Der Fokus der Anwendung von SentiCR liegt dabei auf der Analyse von „Code Review“-Kommentaren. Hierzu verwendet SentiCR Methoden des überwachten maschinellen Lernens und setzt dabei auf eine umfassende Vorverarbeitung der zu verarbeitenden Daten. So implementiert SentiCR z.B. eine umfassende explizite Verarbeitung von Negationen, extrahiert aber außer TF-IDF-Merkmalen keine weiteren Merkmale. Um die Entscheidung für den gewählten Klassifikationsalgorithmus zu treffen, verglichen die Autoren acht verschiedene Algorithmen bezüglich Precision, Recall, F_1 und Accuracy. Die beste Performanz erzielte ein „Gradient-Boosting“-Klassifizierer, der als finaler Algorithmus für die Anwendung festgelegt wurde.

Islam et al. [15] entwickelten außerdem das Analysetool „MarValous“ für die Anwendung im Kontext der Softwareentwicklung. Ähnlich EmoTXT erfolgt im Rahmen der Anwendung von MarValous keine Einteilung in die drei Klassen positiv, neutral und negativ, sondern eine Einordnung von Eingabedaten in ein Spektrum verschiedener Arten von Emotionen. Die Anwendung setzt dabei auf eine umfassende Vorverarbeitung sowie explizites Negation Handling. Extrahierte Merkmale sind dabei neben TFIDF-Merkmalen (Mono- und Bigramme) auch händisch entworfene Emoticon- und Microbloggingmerkmale. Ähnlich Ahmed et al. [4] wird der verwendete Klassifikationsalgorithmus basierend auf einem Vergleich neun unterschiedlicher Algorithmen hinsichtlich ihrer Performanz auf einem Benchmarkdatensatz von 5122 StackOverflow- und Jira-Kommentaren ausgewählt.

4. Konzept

In diesem Kapitel wird das von Horstmann [14] definierte Konzept für die Analyse der Kommunikation in Entwicklerteams vorgestellt. Darauf folgt eine Auflistung initialer Änderungen, die für eine Auswertung des Konzeptes auf einem Goldstandard-Datensatz der Sentimentanalyse in der Softwareentwicklung im Kontext dieser Arbeit notwendig sind.

4.1. Konzept von Horstmann

Wie in Kapitel 3 beschrieben, umfasst das Konzept von Horstmann [14] sämtliche Verarbeitungsschritte, die dazu notwendig sind, um die Kommunikation eines tatsächlichen Softwareprojektes zu erheben, zu verarbeiten und auszuwerten. Für diese Aufgaben beschreibt Horstmann [14] eine Pipeline, die sich in die folgenden Bearbeitungsschritte gliedert.

4.1.1. Erheben von Quelldaten

Der erste Schritt dieser Pipeline befasst sich mit dem Erstellen eines Datensatzes, basierend auf Textdaten ausgewählter Kommunikationskanäle eines Entwicklerteams [14]. Die anhand dieser Kommunikationskanäle (z.B. Zulip oder Slack) ermittelten Textdaten enthalten dabei Metadaten (als Ergebnis des API-Calls an die jeweilige API) und Textartefakte wie URLs, Programmcode und Verweise, welche vor einer weiteren Verarbeitung entfernt werden [14]. Auf die Bereinigung folgt eine Aufteilung der zusammenhängenden Texte in einzelne Sätze und eine Anonymisierung aufgeführter Kommunikationsteilnehmer und Verweise [14]. Zuletzt werden die ermittelten Sätze manuell in eine der drei Polaritätsklasen, positiv, neutral oder negativ, eingeordnet [14].

Wie bereits angesprochen wird für die Auswertung des Horstmann-Konzeptes [14] im Kontext dieser Arbeit ein Goldstandard-Datensatz verwendet. Im Rahmen dieser Arbeit wird kein „eigener“ Datensatz generiert, der erste Schritt des Konzeptes wird dementsprechend nicht weiter betrachtet.

4.1.2. Vorverarbeitung und Extraktion von Merkmalen

Aufbauend auf den bereinigten Daten erfolgt die weitere Vorverarbeitung der ermittelten Sätze sowie die Extraktion von Merkmalen im Hinblick auf die Klassifizierung. Die

Vorverarbeitung umfasst dabei die folgenden Verarbeitungsschritte [14]:

1. **Rechtschreibprüfung:** Der Text wird mit einem deutschen und einem englischen Vokabular abgeglichen und Rechtschreibfehler werden korrigiert.
2. **Kleinschreibung:** Großbuchstaben werden in Kleinbuchstaben umgewandelt.
3. **Tokenisierung:** Der Text wird wortweise in seine Token aufgeteilt und das Part-of-Speech-Tagging wird durchgeführt.
4. **Interpunktion:** Nach dem Durchsuchen des Textes nach Emoticons werden sämtliche Token, die Interpunktion enthalten, entfernt.
5. **Zahlen:** Token, die eine oder mehrere Zahlen enthalten werden entfernt.
6. **Stoppwörter:** Stoppwörter werden aus dem Text entfernt.

Auf Basis der vorverarbeiteten Dokumente werden die folgenden Klassen von Merkmalen extrahiert [14]. Eine Auflistung aller extrahierten Merkmale des Konzeptes von Horstmann [14] ist in Tabelle 4.2 zu sehen (siehe S. 21). Um ein korrektes Arbeiten des distanzbasierten Klassifikationsalgorithmus „Support Vector Machine“ zu ermöglichen, werden alle numerischen Merkmale auf das Intervall $[-1, 1]$ transformiert [14].

Statistische Merkmale

Um die Struktur eines betrachteten Textes in die Klassifizierung einfließen zu lassen, werden statistische Merkmale extrahiert [14]. Beispiele für Merkmale dieser Klasse sind die Anzahl der Token eines Dokuments, die mittlere Länge der betrachteten Token und das Verhältnis eines spezifischen *Part-of-Speech*-Tags zur ermittelten Satzlänge [14].

Emotionalität und Emoticons

Um die Polarität eines Dokuments zu ermitteln verwendet das Konzept von Horstmann [14] zwei „Sentimentlexika“. Sentimentlexika umfassen sogenannte *prior polarities* eines ausgewählten Vokabulars [31]. *Prior polarities* sind in der Regel numerische Werte, welche die Polarität eines Wortes, im Hinblick auf eine negative bzw. positive Polarität beschreiben [31].

Anhand dieser Lexika wird das Sentiment eines Textes bestimmt und unter Verwendung mehrerer Merkmale modelliert [14]. Das erste Lexikon (Waltinger) beschreibt Einteilungen eines Vokabulars sentimentbehafteter Wörter in eine von drei Polaritätsklassen (positiv, neutral, negativ) [14]. Für die Extraktion der hiermit verbundenen Merkmale werden den Klassen Zahlenwerte zugewiesen [14]: Ein Element der neutralen Klasse entspricht einem Wert von 0, ein Element der negativen Klasse von -1 und ein Element der positiven Klasse von $+1$ [14]. Das zweite Lexikon (Remus) beschreibt numerisches, reellwertiges Sentiment aus dem Intervall $[-1, 1]$. Ein Wert von 1 entspricht stark positivem Sentiment, ein Wert von -1 einem stark negativen Sentiment [14]. Token,

die in den verwendeten Sentimentlexika nicht enthalten sind, werden für das Bilden der Emotionsmerkmale nicht beachtet.

Ergänzend zu den Sentimentmerkmalen werden außerdem „Emoticonmerkmale“ bestimmt [14]. Aufbauend auf einem analog zu dem zuletzt genannten Sentimentlexikon funktionierenden Emoticonlexikon erhebt Horstmann [14] Merkmale, welche die Polarität von Emoticons modellieren (siehe Tabelle 4.1.2).

Formalität

Um eine Aussage über den Grad an Formalität eines Textes zu treffen, wird das Merkmal *formally* extrahiert [14]. *Formally* kombiniert das Verhältnis von Personalpronomen der 3. Person Plural (Sie, Ihrer, Ihnen) zu den verbleibenden Personalpronomen, das Verhältnis korrekt geschriebener Nomen und Eigennamen zu inkorrekt geschriebenen Nomen und Eigennamen sowie das Verhältnis korrekter Rechtschreibung zu fehlerhafter Rechtschreibung in einem reellen Zahlenwert [14].

Bag-of-Words

Um alle Elemente eines natürlichsprachlichen Textes für die Algorithmen aufzubereiten ist es wichtig diesen Text zu repräsentieren. Eine Möglichkeit dafür, ist die Extraktion von Bag-of-Words-Merkmalen. Auf Basis betrachteter Token der behandelten Dokumente werden Vektorrepräsentationen der Dokumente erzeugt. Ein Eintrag eines solchen Vektors steht dabei für ein Token eines betrachteten Vokabulars.

Horstmann [14] verwendet zwei verschiedene Ansätze für das Generieren von Vektorrepräsentationen von Dokumenten: Termpräsenzvektoren und TF-IDF-Vektoren. Bevor ein Klassifizierungsalgorithmus trainiert wird, wird aus dem vorab ermittelten Trainingsdatensatz ein Vokabular V extrahiert [14]. Dieses Vokabular besteht aus den einzigartigen Token der in dem Datensatz enthaltenen Dokumente. Der Präsenzvektor beschreibt dann jedes Dokument d als Binärvektor der Länge $|V|$. Ein Eintrag von 1 für ein Token t aus dem Vokabular V zeigt an, dass Term t in Dokument d vorkommt. Die Vektorrepräsentation der *term frequency–inverse document frequency* (TF-IDF) ähnelt in seiner Form dem Termpräsenzvektor. Die Einträge des TF-IDF-Vektors werden basierend auf folgendem Produkt der *term frequency* $tf_{t,d}$ und der *inverse document frequency* idf_t errechnet [21]:

$$\text{tf-idf}(t, d) = tf_{t,d} * idf_t$$

Die $tf_{t,d}$ beschreibt dabei die Häufigkeit des Vorkommens des Terms t in Dokument d [21]. Die *inverse document frequency* für einen Term t ist wie folgt definiert [21]:

$$idf_t = log\frac{N}{df_t},$$

wobei N die Menge aller betrachteten Dokumente und df_t die Häufigkeit des Terms t in den N Dokumenten beschreibt [21]. Seltener auftretende Terme, welche in der Regel

einen höheren Informationsgehalt als häufiger auftretende Terme aufweisen, erzielen dementsprechend höhere TF-IDF-Werte [21].

4.1.3. Merkmalsselektion und Klassifizierung

Anhand der ermittelten Merkmale folgt der Schritt der Klassizierung [14]. Das Konzept von Horstmann [14] verwendet dabei die folgenden drei Klassifizierungsalgorithmen: Eine *Support Vector Machine*, einen *Random Forest* und einen *Gaussian Naive Bayes*. Die Ausgaben der Klassifizierer werden anhand eines *VotingClassifiers* kombiniert [14].

Bevor eine Klassifizierung unter Verwendung des ermittelten Datensatzes stattfindet, führt Horstmann [14] den Schritt der Merkmalsselektion auf. Es wird versucht, eine optimale (echte) Teilmenge der Gesamtmenge der Merkmale (Tabelle 4.1.2) zu bestimmen [14]. Hierzu wird das Ensemble von Algorithmen auf sich ändernden Merkmalsmengen trainiert und der so ermittelte Klassifizierer auf einem Testdatensatz ausgewertet [14]. Das Endergebnis dieses Prozesses ist die Merkmalsmenge, welche die beste Genauigkeit auf dem Testdatensatz erzielt [14]. Um die Merkmalsmenge zu permutieren und den vom Merkmalsvektor aufgespannten Suchraum nach optimalen Lösungen zu durchsuchen, verwendet Horstmann [14] einen evolutionären Algorithmus. Dieser führt in jeder Generation eine zufällige Anzahl an Bitflips auf dem Merkmalsvektor durch [14]. Unter Verwendung der ermittelten Merkmalsmenge wird das Ensemble trainiert und ausgewertet [14].

4.1.4. Auswertung

Im letzten Schritt seiner Pipeline sieht Horstmann [14] die Analyse bzw. die Auswertung der anhand des trainierten Modells generierten Daten und deren Sentiment vor. So stellt Horstmann [14] in seiner Arbeit die Möglichkeit einer Trendanalyse der Kommunikation eines Entwicklerteams vor, um auf diese Art den zeitlichen Verlauf der Stimmung des besagten Teams analysieren zu können.

Dieser Schritt wird im Folgenden nicht weiter betrachtet, da sich die vorliegende Arbeit mit den Schritten der Vorverarbeitung und Klassifizierung beschäftigt und die Analyse der Kommunikation eines tatsächlichen Entwicklerteams nicht Teil dieser Arbeit ist. Für den Rest dieser Arbeit bezieht sich die Auswertung der Algorithmen nicht auf den Schritt der Pipeline von Horstmann [14], sondern die Betrachtung der aufgeführten Metriken (siehe Abschnitt 2.3.4) und anderer Aspekte der Algorithmen.

4.2. Erforderliche Anpassungen

Um das Konzept von Horstmann [14] im Hinblick auf den Vergleich mit ähnlichen Anwendungen der Sentimentanalyse in der Softwareentwicklung auszuwerten, wird das Konzept in dieser Arbeit anhand eines GoldStandard-Datensatzes trainiert und

ausgewertet. Derartige Goldstandards für die Sentimentanalyse in der Domäne der Softwareentwicklung sind nach aktueller Kenntnis nur in englischer Sprache verfügbar. Aufgrund dieser Tatsache werden initiale Anpassungen am Konzept von Horstmann [14] vorgenommen, um eine korrekte Verarbeitung englischer Texte zu erlauben.

Für die Bearbeitung der NLP-Operationen (Lemmatisierung, Tokenisierung und Part-of-Speech-Tagging) im Kontext der Vorverarbeitung verwendet das Konzept von Horstmann [14] das python-Paket *spacy*. *Spacy* stellt dabei trainierte Modelle für die Durchführung der NLP-Operationen zur Verfügung. Diese Modelle sind sprachspezifisch, weshalb das deutsche Modell durch das englische Modell ersetzt wird. Auch die Liste der verwendeten deutschen Stoppwörter und die Methode der Rechtschreibkorrektur deutscher Texte werden durch ihre englischen Äquivalente unter Verwendung derselben python-Pakete [14], ersetzt.

Das bezüglich der Formalität extrahierte Merkmal wird vorerst entfernt, da dieses im Kontext der englischen Sprache nicht korrekt anwendbar ist und somit einen Störfaktor für die spätere Klassifizierung darstellt.

Die von Horstmann [14] verwendeten Sentimentlexika werden durch das von SentiStrength [34] verwendete Sentimentlexikon[1] ersetzt. Es wird dieses Lexikon verwendet, da es im Kontext der Sentimentanalyse in der Softwareentwicklung häufig zum Einsatz kommt [6]. Dieses Lexikon enthält 2546 englischsprachige Wörter, welchen ein ganzzahliger Wert zwischen -5 und 5 zugewiesen wird. Die Interpretation dieser Werte ergibt sich aus Tabelle 4.1.

Positives Sentiment	**Neutrales Sentiment**	**Negatives Sentiment**
$5 \geq x > 1$	$1 \geq x \geq -1$	$-1 > x \geq -5$

Tabelle 4.1.: Interpretation eines Sentimentwerts x für das SentiStrength-Lexikon [34].

Das Lexikon enthält 395 Wörter mit positivem, 1918 Wörter mit negativem und 233 Wörter mit neutralem Sentiment [34]. Im Rahmen der Integration des Lexikons wird das Waltinger-Merkmal entfernt und die Remus-Merkmale werden durch die drei SentiStrength-Merkmale *senti_mean*, *senti_min* und *senti_max* ersetzt. Die Definitionen der SentiStrength-Merkmale sind dabei analog zu den Definitionen der Remus-Merkmale (siehe Tabelle 4.2). Insgesamt werden in der Vorverarbeitung dieser „Baseline“ 27 Merkmale ohne Betrachtung der *Bag-of-Words*-Merkmale extrahiert.

[1] Entnommen aus der SentiStrength-Anwendung, http://sentistrength.wlv.ac.uk/

Kategorie	Name	Beschreibung
Textstatistik	Textsize	Anzahl aller Zeichen eines Dokuments
	Wordcount	Anzahl der Wörter eines Dokuments
	MaxWordSize	Maximale Wortlänge
	LongWordCount	Anzahl Wörter mit Länge > 6
	WhiteSpaceCount	Anzahl Leerzeichen
	NumberToTextRatio	Verhältnis der Häufigkeit von Zahlen zu Text
	PunctuationToTextRatio	Verhältnis der Häufigkeit von Interpunktion zu Text
	AutocorrectionRatio	Anzahl korrigierter Wörter
	StartsWithXRatio	Anzahl Token, die mit dem Buchstaben X beginnen
	Part-Of-Speech-Features	Verhältnis der Häufigkeit ausgewählter PoS-Tags zur Textlänge. Betrachtet werden die Tags ADV, ADJ, VERB, NOUN, PART, AUX, ADP, DET, PNOUN und SCONJ. Es werden insgesamt zehn Merkmale extrahiert.
Emotion	mean_waltinger	Durchschnitt der sentimentalen Wörter eines Dokumentes nach Waltinger
	mean_remus	Durchschnittliches Sentiment nach Remus
	min_remus	Minimales Sentiment eines Dokumentes nach Remus
	max_remus	Maximales Sentiment eines Dokumentes nach Remus
Emoticon	emoticon_mean	Durchschnittliches Emoticonsentiment
	emoticon_min	Minimum des Emoticonsentimentes
	emoticon_max	Maximum des Emoticonsentimentes
Formalität	formally	Grad der Formalität eines Textdokuments
Bag-of-Words	CountVectorizer	Termpräsenzmerkmale
	TF-IDF	TF-IDF-Merkmale

Tabelle 4.2.: Liste der extrahierten Merkmale nach Kategorie. Die Namen, Beschreibungen und Kategorien der einzelnen Merkmale sind übernommen aus der Arbeit von Horstmann [14].

5. Analyse

In diesem Kapitel wird eine Auswertung des Konzeptes von Horstmann [14] durchgeführt. Hierzu werden die in Kapitel 4 formulierten initialen Anpassungen an dem Konzept vorgenommen und die Klassifizierungssalgorithmen auf einem Datensatz von 7122 Github-Kommentaren [27] trainiert und getestet. Die Ergebnisse werden dann im Hinblick auf das Ermitteln potenzieller Verbesserungsmöglichkeiten analysiert.

5.1. Auswertung

Die initiale Auswertung des Konzeptes von Horstmann [14] mit dem Ziel, einen Vergleichswert für spätere Vergleiche zu ermitteln, wird unter Verwendung des GitHub-Goldstandards, erstellt von Novielli et al. [27], durchgeführt. Dieser Datensatz enthält dabei 7122 GitHub-Kommentare die in eine der drei Polaritätsklassen positiv, neutral und negativ eingeordnet wurden. Von diesen sind 2087 der negativen Klasse, 3022 der neutralen Klasse und 2013 Dokumente der positiven Klasse zuzuordnen (siehe Tabelle 5.1). Die im Rahmen des Konzeptes verwendeten Klassifizierer werden auf diesem Datensatz unter Verwendung einer vom evolutionären Algorithmus als optimal ermittelten Merkmalsmenge trainiert und im Kontext einer 5-fachen Kreuzvalidierung ausgewertet. Die Ergebnisse der Auswertung sind in Tabelle 5.2 aufgeführt.

Klasse	**Häufigkeit**
Negativ	2087
Neutral	3022
Positiv	2013
Insgesamt	7122

Tabelle 5.1.: Klassenverteilung des Github-Goldstandards von Novielli et al. [27]

Klasse	**Precision**	**Recall**	**F**	**Accuracy**
Negativ	78,3	81,6	79,9	
Neutral	82,4	81,9	82,1	82,5
Positiv	87,4	84,2	85,8	

Tabelle 5.2.: Auswertung der Baseline auf einer optimierten Merkmalsmenge.

Betrachtet man nun die Ergebnisse in Tabelle 5.2 so fällt auf, dass das Ergebnis dieser

Auswertung von der durch Horstmann [14] durchgeführten Auswertung stark abweicht. So erzielen die Klassifizierungsalgorithmen eine Genauigkeit von 82%, verglichen mit den von Horstmann [14] ermittelten 63%. Dieser Differenz ist auf die Unterschiede der verwendeten Datensätze im Bezug auf Umfang, Verteilung der Klassen und den Prozess der Klassenzuweisung zurückzuführen. Auch die in Abschnitt 4.2 vorgenommenen Änderungen bzw. insgesamt die Anpassung an die englische Sprache, können einen Einfluss auf dieses Ergebnis gehabt haben.

5.1.1. Betrachtung der einzelnen Klassifizierer

Auf dem Ergebnis der Auswertung aufbauend und um die Performanz der Klassifizierer des Ensembles zu ermitteln, werden die Klassifizierer ohne Kombination durch den *VotingClassifier* trainiert und ausgewertet. Die Auswertung erfolgt dabei auf der gleichen Aufteilung des Datensatzes wie zuvor. Es ist anzumerken, dass der evolutionäre Algorithmus im Rahmen dieser Betrachtung nicht angewendet wird, um die Performanz der Klassifizierer vorerst vom Einfluss des EA zu trennen. Entsprechend wird die Performanz der Algorithmen auf der vollständigen Merkmalsmenge ermittelt. Das Ergebnis der Auswertung der Klassifizierer ist in Tabelle 5.3 zu sehen.

	Negativ			**Neutral**			**Positiv**			
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**A**
VC	77,9	**79,3**	78,5	**81,6**	83,4	**82,5**	87,9	83,2	**85,5**	**82,1**
SVM	80,4	78,5	**79,4**	80,9	83,2	82,0	85,7	**84,0**	**84,8**	82,0
RF	**84,1**	66,6	74,3	72,9	**90,6**	80,8	**91,1**	77,5	83,8	79,9
NB	48,2	43,2	45,5	66,0	36,1	46,6	41,4	74,0	53,1	48,9

Tabelle 5.3.: Precision, Recall und F_1 für die Klassen *negativ, neutral* und *positiv* für den *Voting Classifier* (VC), die *Support Vector Machine* (SVM), den *Random Forest* (RF) und den *Naive Bayes* (NB), sowie die entsprechende Accuracy. Validiert durch eine 5-fache Kreuzvalidierung.

Bei Betrachtung der Werte in der Tabelle fällt vor allem die schwache Performanz des *Naive-Bayes*-Klassifizierers (NB) auf. So weist dieser eine um 31,0%-Punkte geringere Genauigkeit als der *Random-Forest*-Klassifizierer (RF) als nächstbester Klassifizierer, auf. Nur in der Precision der neutralen Klasse sowie dem Recall der positiven Klasse sind die Werte des NB vergleichbar mit dem des RF. Der RF erzielt in einigen der Metriken, wie der Precision der positiven Klasse (+5,4%-Punkte) sehr viel bessere Ergebnisse als der nächstbeste Klassifizierer, die *Support Vector Machine* (SVM). Insgesamt fällt der RF jedoch sowohl bezüglich des erzielten F_1-Wertes, als auch der generellen Genauigkeit hinter der SVM zurück. Es fällt außerdem die marginal bessere Genauigkeit unter Verwendung des *VotingClassifiers* (VC), verglichen mit dem besten einzelnen Klassifizierer, der SVM, auf. Sowohl in der neutralen als auch der positiven Klasse weist der VC einen um bis zu 0,7 besseren F_1-Wert auf. Lediglich in der negativen Klasse erzielt die SVM einen höheren F_1-Wert.

Des Weiteren ist zu bemerken, dass alle Algorithmen auf Dokumenten der positiven Klasse die besten Werte in den Metriken erzielen. Hierauf folgt die Performanz in der neutralen Klasse, zuletzt gefolgt von der negativen Klasse. Dies scheint darauf hinzudeuten, dass die existierenden Merkmale und Mechanismen, die Aspekte und Strukturen der verschiedenen Klassen unterschiedlich gut erfassen und modellieren können. Insgesamt erscheint es sinnvoll, den *Naive-Bayes*-Klassifizierer und potenziell auch den RF und die SVM durch Algorithmen zu ersetzen, die eine bessere Genauigkeit unter Verwendung der extrahierten Merkmale erzielen.

5.1.2. Lernkurven

Exemplarisch[1] werden in diesem Abschnitt die Lernkurven des *Random-Forest*-Klassifizierers (RF) betrachtet. Die Lernkurven eines Klassifizierers stellen dabei sowohl die Trainingsperformanz als auch die Testperformanz des Algorithmus' in Abhängigkeit der Größe des Trainingsdatensatzes dar. Lernkurven erlauben dadurch Einblicke in die Anpassung an die Trainingsdaten und die Generalisierbarkeit eines Klassifizierers und können dazu verwendet werden Probleme wie z.B. eine Überanpassung zu diagnostizieren. Die Lernkurven des RF sind in Abbildung 5.1 zu sehen. Die grünen Datenpunkte zeigen die erzielte Trainingsgenauigkeit, die roten Datenpunkte, die erzielte Testgenauigkeit. Der letzte Datenpunkt beschreibt die in der Auswertung betrachtete Aufteilung des Datensatzes.

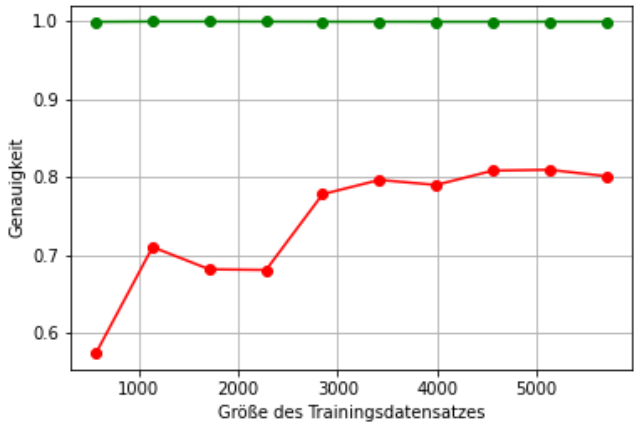

Abbildung 5.1.: Lernkurven des *Random-Forest*-Klassifizierers, 5-fach kreuzvalidiert.

Bei Betrachtung der Kurven fällt zunächst auf, dass der Klassifizierer, unabhängig von der Größe des Trainingsdatensatzes eine perfekte Trainingsgenauigkeit erzielt. Allerdings besteht eine vergleichsweise große Diskrepanz zwischen dieser Genauigkeit und der auf den Testdaten erzielten Genauigkeit. Im letzten Datenpunkt entspricht dieser Unter-

[1] Die Lernkurven der SVM und des NB sind inklusive Beschreibung im Anhang zu finden.

schied einer Verringerung der Testgenauigkeit um ca. 0,2. Der Klassifizierer ist demnach nicht dazu in der Lage ausgehend von den Trainingsdaten, korrekt zu generalisieren. Verglichen mit der Trainingsgenauigkeit weist die Kurve der Testgenauigkeit einen eigentlich normalen Verlauf auf. Je mehr Trainingsdaten zugeführt werden, desto besser kann der RF die Strukturen der Daten lernen und desto besser die Genauigkeit auf den Testdaten. Ab ca. 3300 Trainingsdaten nimmt die Genauigkeit nur noch geringfügig zu und scheint im letzten Datenpunkt sogar abzunehmen. Weitere Trainingsdaten könnten bei der Interpretation der Modellperformanz helfen.

Insgesamt zeigen die Lernkurven des RF Anzeichen einer Überanpassung an den Trainingsdatensatz und darin enthaltene Störfaktoren auf [11]. Die perfekte Trainingsgenauigkeit, zusammen mit der um 0,2 geringeren Testgenauigkeit sind Indikatoren hierfür. Eine Möglichkeit, die Diskrepanz zwischen Trainings- und Testgenauigkeit zu verringern, ist eine Anpassung der Hyperparameter des *Random-Forest*-Klassifizierers bzw. insgesamt eine Anpassung der Hyperparameter der verwendeten Algorithmen (siehe Abschnitt 6.2.2).

5.2. Fehleranalyse

In diesem Abschnitt wird im Rahmen einer qualitativen Fehleranalyse eine zufällige Stichprobe der Fehlklassifizierungen des Ensembles auf einem Testdatensatz ausgewertet. Es wird versucht, unter Betrachtung der Eingaben und deren Merkmale, Rückschlüsse auf den Grund der Fehlklassifizierung zu ziehen und so fehlerhafte Aspekte der Vorverarbeitung und der Klassifizierung aufzudecken. Für die Fehleranalyse in dieser Arbeit werden 100 falsch klassifizierte Dokumente der insgesamt 258 Fehlklassifizierungen untersucht und jeweils in mindestens eine der im Kontext diese Arbeit definierten Fehlerklassen $F_1, \ldots, F_6$ eingeordnet.

Für die Definitionen der Fehlerklassen wurde sich, neben in den Eingabedaten gefundenen Artefakten, auch an den Fehlerklassen der Analysen von sowohl Islam et al. [16], als auch Novielli et al. [27], orientiert. Die genauen Definitionen der in der vorliegenden Arbeit verwendeten Fehlerklassen werden im Folgenden aufgelistet. Eine Übersicht über die Häufigkeiten der Fehlerklassen ist in Tabelle 5.4 zu sehen.

F_1: NLP und Umfang

Die Fehlerklasse F_1 umfasst Fehlklassifizierungen, die auf Fehler in der Vorverarbeitung, generelle Fehler der Komponenten des NLP sowie auf fehlenden Umfang des verwendeten Sentiment- bzw. Emoticonlexikons zurückzuführen sind.

Ein häufig im Kontext der Vorverarbeitung auftretendes Problem ist, dass in den Daten enthaltene URLs und Programmcodeabschnitte auf Sentiment und Emoticons untersucht werden. Dieser Fehler basiert auf der Reihenfolge der Verarbeitungsschritte der Pipeline von Horstmann [14], die während der Vorverarbeitung der Texte keine Bereinigung von

Index	Fehlerklasse	Alle Features (%)
F_1	NLP und Umfang	46
F_2	Kontextfehler	33
F_3	Fehlende Funktionalität	32
F_4	Domänenspezifität	22
F_5	Nicht definierbar	16
F_6	Höflichkeit	6

Tabelle 5.4.: Ergebnisse der Fehleranalyse des Ensembles der Klassifizierer. Mehrfachzuweisungen sind während des Zuweisungsprozesses erlaubt.

URLs und ähnlichen Token vorsieht. Das Konzept von Horstmann [14] enthält zwar einen Schritt für die Bereinigung der Daten, diese wird jedoch im Kontext der Erstellung eines eigens für die Anwendung gefertigten Datensatzes implementiert. Da für die Auswertung in dieser Arbeit ein fremder Datensatz verwendet wird, bei dessen Erstellung URLs etc. nicht entfernt worden sind, findet der Algorithmus in jedem Dokument, welches eine URL enthält, die Zeichenfolge „:/“, die als negativ annotiertes Emoticon interpretiert wird. Diese Tatsache stellt eine Beeinflussung der Klassifizierung in Richtung der negativen Klasse dar.

Diese Fehlerklasse enthält außerdem Fehlklassifizierungen, die auf vereinzelte Fehler der NLP-Bibliotheken zurückzuführen sind. So existieren Fälle, in denen die Rechtschreibkorrektur fehlerhaft ist, sowie Fälle in denen Eigennamen nicht korrekt erkannt werden. Ein Beispiel hierfür lässt sich im folgenden Satz finden: Im Kontext der Verarbeitung des Satzes „Shauren is right, revert!“ wird der Eigenname „Shauren“ fälschlicherweise zu dem Wort „share“ korrigiert, welches laut dem Sentimentlexikon ein positives Sentiment aufweist.

Zuletzt enthält diese Klasse Fehler, die auf zu geringen Umfang der verwendeten Lexika basieren. So sind z.B. Emoticons wie „<3“, sentimentale Wörter, wie „superb“ (ausgezeichnet/vorzüglich) oder auch Beschreibungen der Zustimmung wie „+1“ im Umfang der Lexika nicht enthalten. Hinzu kommt, dass in der Vorverarbeitung Emojis, die in der Form „:rage:“ in den Dokumenten auftreten, nicht erkannt werden können.

F_2: Kontextfehler

Fehlerklasse F_2 gruppiert Fälle in welchen eine auf den Verwendungskontext eines Wortes zurückführbare Veränderung des Sentiments nicht korrekt erfasst wird. Ein häufig auftretendes Beispiel ist das Wort „like“. „Like“ weist, verwendet als Verb (z.B. „to like“) ein positives Sentiment auf. Häufig wird „like“ jedoch im Kontext eines Vergleiches z.B. in der Form „A behaves like B“ verwendet. In letzterem Kontext trägt das Wort dabei kein Sentiment. Das Auftreten kontextabhängiger Mehrdeutigkeiten von Wörtern wird dementsprechend nicht korrekt verarbeitet.

F_2 umfasst außerdem Fälle sogenannter, von Wilson [37] beschriebener, *Objective Polar Utterances* (OPU). Diese OPUs sind dabei Beschreibungen inhärent begehrenswerter bzw. nicht begehrenswerter Fakten, die mit neutralem Sentiment formuliert werden [37]. Ein Beispiel hierfür ist folgender Auszug aus den Testdaten: „[...] trying to find the spaces among the underscores slows down code reading [...]“. Die Tatsache, dass das Lesen von Programmcode durch eine Änderung verlangsamt wird, ist inhärent negativ annotiert, wird jedoch ohne Verwendung sentimentaler Wörter formuliert.

F_3: Fehlende Funktionalität

„Fehlende Funktionalität“ gruppiert Dokumente, in denen Fehler auftreten, welche auf fehlende Funktionalität der Vorverarbeitung zurückzuführen sind. So werden Negationen (z.B. *not* happy) und „Verstärkungswörter“, definiert von Islam et al. [34], (z.B. *really* happy) nicht korrekt interpretiert. Dies kann sich relativ stark auf das geschätzte Sentiment auswirken, da diese entweder vorhandenes Sentiment verstärken oder im Falle von Negationen das Sentiment des negierten Wortes oder Textabschnittes umkehren [31]. Ein Beispiel für den Fall der Negation lässt sich in folgendem Satz finden: „ [...], because Travis isn't happy“. Das Sentiment des Tokens „happy“ wird unverändert als positiv interpretiert, obwohl eine Negation durch das Token „isn't“ erfolgt.

Diese Klasse enthält außerdem Fehler, die auf das inkorrekte Handhaben von Ironie und Sarkasmus zurückzuführen sind, da diese sprachlichen Phänomene einen Sonderfall der Negation darstellen [31].

F_4: Domänenspezifität

Die Fehlerklasse „Domänenspezifität“ umfasst die Dokumente, in denen Fälle einer domänenspezifischen Fehlinterpretation von Wörtern vorzufinden sind. Domänenspezifische Fehlinterpretationen sind dabei Fälle, in denen ein Token bei Verwendung innerhalb einer Domäne oder eines spezifischen Themas ein verändertes Sentiment aufweist [16]. Bezogen auf die Domäne der Softwareentwicklung sind dies Wörter wie „value“, „error“ oder „bug“. Diese Wörter weisen laut dem SentiStrength-Lexikon [34] ein Sentiment auf, sind innerhalb eines technischen Kontexts allerdings häufig als neutral zu betrachten [16].

F_5: Nicht definierbar

Diese Fehlerklasse enthält Dokumente, bei denen auch nach mehrmaliger Betrachtung die Ursache der Falschklassifikation nicht erkenntlich geworden ist und sich dadurch nicht in eine der anderen Klassen einordnen ließen. Ursachen hierfür können, aus menschlicher Sicht von den Klassifizierern falsch bzw. anders erlernte Aspekte natürlicher Sprache oder fehlerhafte Interpretation während der Zuweisung der Fehlerklassen sein. Eingaben und extrahierte Metriken lassen sich nur begrenzt interpretieren und eine kausale Wirkung kann schwer festzustellen bzw. abzuschätzen sein.

F_6: Höflichkeit

Während der Klassenzuweisung bei Erstellung des Github-Goldstandards, wurden höfliche Phrasen und Wörter (z.B. „thanks“, „please“) als neutral betrachtet [27]. Die innerhalb dieses Kontexts gebräuchlichen Wörter weisen laut dem SentiStrength-Lexikon [16] ein positives Sentiment auf, wodurch die Klassifikation in Richtung der positiven Klasse beeinflusst wird.

5.3. Evolutionärer Algorithmus

Im Hinblick auf die Beantwortung der ersten Forschungsfrage wird auch der evolutionäre Algorithmus (EA) von Horstmann [14] auf Ansätze für eine Verbesserung der Klassifizierungsgenauigkeit überprüft. Hierbei sollen die folgenden Aspekte des EA bzw. der Merkmalsselektion angeschnitten werden: Welche Auswirkung hat die Merkmalsselektion auf die Klassifizierungsperformanz? Wie gut funktioniert der EA als Methode des Lösens des vorliegenden Optimierungsproblems? Zunächst wird dabei auf die Funktionsweise von Horstmanns [14] EA eingegangen. Danach wird die Methodik der Auswertung angesprochen und die Ergebnisse der Auswertung betrachtet.

5.3.1. Funktionsweise

Im Hinblick auf die Optimierung der Klassifizierungsgenauigkeit wird die Menge der extrahierten Merkmale (Kapitel 4.1.2) als Binärvektor betrachtet [14]. Ein Eintrag von 1 bedeutet, dass das Merkmal verwendet wird, ein Eintrag von 0, dass ein Merkmal nicht verwendet wird. Anhand dieser Beschreibung für die Merkmalsmenge lässt sich folgendes Optimierungsproblem definieren [14]: Welche (echte) Teilmenge der Merkmalsmenge maximiert die Klassifikationsgenauigkeit? Dementsprechend verwendet der EA von Horstmann [14] die erzielte Klassifizierungsgenauigkeit unter Verwendung einer bestimmten Merkmalsmenge als Fitnessfunktion. Der von Horstmann [14] verwendete EA arbeitet dabei wie folgt:

Initial betrachtet der EA den voll besetzten Merkmalsvektor. Die Bits dieses Vektors werden dann zufälliger Bitflip-Mutationen unterzogen und darauf folgend die Fitness des auf diese Art erzeugten Vektors bestimmt. Ist die ermittelte Fitness besser als die des bisher „besten“ Vektors, so wird der ausgewertete Vektor als Startpopulation in die Folgegeneration übernommen. Ist die ermittelte Fitness schlechter als die des bisher besten Vektors, so wird der betrachtete Vektor verworfen und die Mutation erneut durchgeführt. Das Ergebnis des EA ist der Vektor, der während der Ausführung die beste Fitness auf dem verwendeten Datensatz erzielt. Es ist anzumerken, dass der EA die im Kontext der Anwendung der Vectorizer extrahierten Merkmale (siehe Abschnitt 4.1.2) unter zwei Einträgen des Bitvektors zusammenfasst. Der EA betrachtet dementsprechend die Genauigkeit mit allen Merkmalen einer Vektorrepräsentation oder keinen Merkmalen einer Vektorrepräsentation.

5.3.2. Methodik

Um den Einfluss der Anwendung des evolutionären Algorithmus' auf die Klassifizierungsperformanz zu ermitteln, wird zunächst die Performanz der Klassifizierungsalgorithmen auf zwei verschiedenen Merkmalsmengen betrachtet. In Tabelle 5.5 sind diese Ergebnisse für die beiden Merkmalsmengen S_{A} (alle Merkmale) und S_{EA} (eine vom EA als optimal ermittelte Merkmalsmenge) nach 5-facher Kreuzvalidierung aufgeführt.

Um die Eignung des EA im Bezug auf das Durchsuchen des betrachteten Suchraumes bzw. das Lösen des gegebenen Optimierungsproblemes zu untersuchen, wird der EA im Hinblick auf die in Tabelle 5.6 gezeigten Metriken betrachtet. Es soll eine Aussage darüber getroffen werden, wie effektiv die Methodik des EA zum Bestimmen einer optimalen Merkmalsmenge ist oder ob sich Schwächen, welche einen negativen Einfluss auf die Lösungsfindung haben, aufdecken lassen. Ermittelt werden die Metriken basierend auf 25 Durchläufen des EA, in denen dieser für jeweils 200 Generation betrieben wird. Die Ergebnisse der Metriken sind in Tabelle 5.6 aufgeführt. Hierbei ist zu beachten, dass die Genauigkeiten auf einer einzelnen 70%/30%-Aufteilung (Training/Test) des Datensatzes basieren und aufgrund von Beschränkungen der Laufzeit keine Kreuzvalidierung durchgeführt wird.

Die in Tabelle 5.6 aufgeführten Metriken sollen dabei die Aspekte der *Exploration* und *Exploitation* sowie die Performanz der durchschnittlich generierten Merkmalsmengen untersuchen. Die Auswahl der Metriken bezüglich der Qualität des EA als Suchalgorithmus basiert dabei auf initialen Beobachtungen der Ergebnisse des EA. Exploration und Exploitation sind zwei wichtige Qualitäten eines Suchalgorithmus' (SA) [36]. Exploration beschreibt die Fähigkeit eines SA, den Suchraum global auf „Areale“ guter Lösungen zu untersuchen, während Exploitation die „feine“ Suche nach optimalen Lösungen innerhalb dieser Areale beschreibt [39].

	Negativ			**Neutral**			**Positiv**			
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**A**
S_{A}	77,9	79,3	78,5	81,6	**83,4**	**82,5**	**87,9**	83,2	85,5	82,1
S_{EA}	**78,3**	**81,6**	**79,9**	**82,4**	81,9	82,1	87,4	**84,2**	**85,8**	**82,5**

Tabelle 5.5.: Metriken für verschiedene Merkmalsmengen. S_{A} ist die Menge aller Merkmale (siehe Abschnitt 4.1.2), S_{EA} die vom evolutionären Algorithmus bestimmte Merkmalsmenge. Der beste Wert einer Metrik ist in fett hervorgehoben.

5.3.3. Auswertung

Die in Tabelle 5.5 für S_{EA} beschriebene Performanz basiert auf dem folgenden, vom evolutionären Algorithmus nach 200 Generationen erzeugten, Merkmalsvektor:

$$[1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0]$$

Metrik	Ergebnis
Alle Vektoren	
Anzahl besuchter einzigartiger Vektoren	2177 / 5000 (43,5%)
Durchs. Abweichung vom Startvektor (Hamming-Distanz)	5 Bit
25 Beste Vektoren	
Beste Genauigkeit (insgesamt)	83,6%
Durchs. Genauigkeit der besten Ergebnisse	83,0%
Durchs. Verbesserung gegenüber der Baseline	+1,8%

Tabelle 5.6.: Ergebnisse der Untersuchung des evolutionären Algorithmus unterschieden in Metriken, die über sämtliche besuchte Vektoren erstellt werden und Metriken, die anhand der 25 besten Vektoren über die 25 Durchläufe erhoben werden.

Dementsprechend werden die Merkmale *sentistrength_max*, *TextSize*, *AutoCorrectionRatio*, *VERBRatio*, *NOUNRatio*, *AUXRatio*, *SCONJRatio*, *StartsWithXRatio* und sämtliche Merkmale des Termpräsenzvektors entfernt (siehe Abschnitt 4.1.2). Insgesamt werden im Durchschnitt 8424 Merkmale (davon 8418 Termpräsenz-Merkmale) entfernt, was einem Anteil von 84% der durchschnittlichen Gesamtmerkmalsanzahl von 10013 Merkmalen pro Eingabedokument entspricht.

Vergleicht man die Performanz unter Verwendung der vollständigen Merkmalsmenge und die Performanz der reduzierten Merkmalsmenge (siehe Tabelle 5.5), so fällt ein geringfügiger Anstieg der Klassifizierungsgenauigkeit unter Verwendung der reduzierten Menge auf. Betrachtet man nun die Performanz der einzelnen Klassen, fällt auf, dass besonders die negative Klasse von der gekürzten Merkmalsmenge profitiert. In der neutralen Klasse ist hingegen ein leichter Abstieg des F_1-Wertes festzustellen, was vor allem auf einen vergleichsweise großen Verlust des Recalls der Klasse zurückzuführen ist. Die positive Klasse weist einen geringen Anstieg des F_1-Wertes auf. Eine Anpassung der Merkmalsmenge unter Verwendung des vorgestellten EA hat insgesamt einen leicht positiven Einfluss auf die Klassifizierungsperformanz. Dies wird unterstrichen von den in Tabelle 5.6 aufgeführten Ergebnissen. So führt eine Anwendung des EA in 25 der 25 betrachteten Durchläufe zu einer Verbesserung der Genauigkeit um durchschnittliche 1,8%-Punkte. Dabei ist anzumerken, dass die 25 Durchläufe 23 unterschiedliche „optimale“ Merkmalsmengen erzeugen, die alle vom voll besetzten Vektor abweichen. Eine Reduktion der Merkmalsmenge führt in jedem Fall zu einer Verbesserung der Klassifizierungsgenauigkeit. Der in Tabelle 5.6 aufgeführte durchschnittliche Anstieg, verglichen mit der Genauigkeit aller Merkmale, ist jedoch mit Vorsicht zu genießen, da für diese Auswertung keine Kreuzvalidierung verwendet wurde. Die stattdessen verwendete einzelne Aufteilung des Datensatzes scheint für die betrachteten Klassifizierer leichter zu verarbeiten zu sein, was zu einem Überschätzen der Genauigkeit im Vergleich mit der in Tabelle 5.5 aufgeführten 5-fach kreuzvalidierten Genauigkeit führt.

Betrachtet man nun die für die Qualität des EA als Suchalgorithmus erhobenen Metriken (siehe Tabelle 5.6) fällt auf, dass der EA in lediglich 43,5% der Auswertungen einen einzigartigen Vektor auswertet. Diese vergleichsweise häufige Wiederholung in der Auswertung, zusammen mit der geringen durchschnittlichen Abweichung vom voll besetzten Startvektor, weisen darauf hin, dass der EA häufig dem Startvektor ähnliche Vektoren auswertet.

Anzumerken ist außerdem die „alles-oder-nichts“-Haltung des EA bezüglich der Bag-of-Words-Merkmale. Diese steht im Widerspruch zu der in Tabelle 5.7 dargestellten *Mutual Information* (MI) [8]. Die Idee dieser Auflistung in Abbildung 5.7 ist übernommen aus der Arbeit von Calefato et al. [6]. MI steht dabei für den Informationsgewinn, den die Verwendung eines Merkmales darstellen kann [8].

Merkmal	**Mutual Information**
emoji_mean	0.260958
emoji_min	0.24948
emoji_max	0.246013
sentistrength_mean	0.2018
sentistrength_min	0.192538
sentistrength_max	0.147811
word_thank	0.0741146
MaxWordSize	0.0627642
cv_thank	0.0537568
LongWordCount	0.0526101
WhiteSpaceCount	0.0461244
MeanWordSize	0.0447634
PunctuationToTextRatio	0.0434211
word_sorry	0.0405856
TextSize	0.0400377

Tabelle 5.7.: Top 15 Merkmale bezüglich ihrer Mutual Information. Erzeugt auf dem gleichen Datensatz wie in der Auswertung des EA. Es werden alle Merkmalsklassen (Kapitel 4.1.2) betrachtet. Merkmale mit dem Präfix "cv_" entstammen dem Termpräsenzvektor, Merkmale mit Präfix "word_" dem TF-IDF-Vektor. Je größer der Wert der Mutual Information, desto besser.

Es fällt auf, dass einige Bag-of-Words-Merkmale - gekennzeichnet durch die Präfixe „word_“ und „cv_“ - in den 15 „besten“ Merkmalen nach MI enthalten sind. So stellen die Token „thank“ und „sorry“ einen ähnlich hohen Informationsgewinn wie vordefinierte Merkmale wie „MeanWordSize“ dar. Eine „feinere“ Herangehensweise an die Merkmalsselektion könnte für eine Verbesserung der Klassifizierungsperformanz von Nutzen sein, damit auch innerhalb der Menge der Bag-of-Words-Merkmale eine Unterscheidung in die einzelnen Merkmale stattfindet.

5.4. Zusammenfassung

Im Rahmen der Analyse des Horstmann-Konzeptes [14] wurden die Klassifizierungsalgorithmen anhand des Github-Datensatzes von Novielli et al. [27] ausgewertet und die Performanz der Algorithmen genauer betrachtet. Auch die Methodik und der Einfluss der Anwendung des evolutionären Algorithmus' wurden untersucht. Basierend auf den Erkenntnissen dieser Analyseschritte soll nun die erste Forschungsfrage beantwortet werden. Dies erlaubt ein Ableiten bzw. Identifizieren von Verbesserungsmöglichkeiten, die in Kapitel 6 ausgeführt werden. Die Forschungsfrage wurde dabei eingangs wie folgt formuliert:

RQ1 Welche Ansätze bieten die Vorverarbeitung und die Klassifizierung für eine Verbesserung der Klassifizierungsperformanz?

Folgende Ansätze wurden aufgrund der durchgeführten Analyseschritte ermittelt, die einen potenziell negativen Einfluss auf die Klassifizierungsperformanz haben:

- Das Konzept geht nicht auf wichtige Aspekte der Eingabetexte wie Kontext, Negation oder Domänenspezifität ein. Hinzu kommt, dass Eingabedaten „fremder" Datensätze nicht korrekt verarbeitet werden und der Umfang der verwendeten Lexika besonders im Hinblick auf die Erkennung von Emojis zu gering ist.
- Die vergleichsweise schwache Performanz des verwendeten *Naive-Bayes*-Klassifizierers und die Überanpassung des *Random-Forest*-Klassifizierers sowie des *Support-Vector-Machine*-Klassifizierers und die daraus resultierende verbesserungsbedürftige Generalisierbarkeit der Algorithmen.
- Der Faktor der Exploration des evolutionären Algorithmus', da die Funktionsweise des EA keine größeren Abweichungen vom Startvektor ermöglicht und somit potenziell bessere Kombinationen von Merkmalen nicht erkundet. Hinzu kommt der Faktor der Granularität des EA im Umgang mit den *Bag-of-Words*-Merkmalen, welcher potenziell informative Merkmale verwirft.

Aufbauend auf diesen Aspekten, werden nun die folgenden Gruppen von Anpassungen am Konzept von Horstmann identifiziert: Im Hinblick auf ein Ersetzen des *Naive-Bayes*-Klassifizierers, soll ein Vergleich ausgewählter Klassifizierungsalgorithmen durchgeführt werden, um Algorithmen zu ermitteln die potenziell besser mit den Merkmalen des Konzeptes von Horstmann [14] arbeiten. Für die so ausgewählten Algorithmen soll außerdem eine Optimierung der Hyperparameter, im Hinblick auf das potenzielle Problem der Generalisierbarkeit durchgeführt werden. Ein Justieren der Hyperparameter führt in der Regel dazu, dass das Modellverhalten z.B. anhand einer Einschränkung der erlaubten Modellkomplexität angepasst wird [11]. Dies kann wiederum Problemen in Modellen des maschinellen Lernens wie einer Überanpassung entgegenwirken [11]. Die konkreten Änderungen und Auswertungen werden in Abschnitt 6.2 genauer beschrieben.

Für die in der Fehleranalyse ermittelten Probleme werden Anpassungen sowie Ergänzungen an der Vorverarbeitung vorgenommen (siehe Abschnitt 6.1).

Außerdem soll der von Horstmann [14] vorgestellte EA um Methoden der Selektion und Rekombination ergänzt werden und die initiale Population des Algorithmus' zufällig bestimmt werden. Diese Änderungen finden im Hinblick auf eine potenzielle Verbesserung der „Exploration“ statt, um so den Einfluss dieses Faktors auf das Lösen des Optimierungsproblemes zu untersuchen. Die konkreten Anpassungen werden in Abschnitt 6.3 beschrieben.

6. Identifizierte Verbesserungen

In diesem Kapitel werden die zuvor ermittelten Ansätze aufgegriffen und mögliche Lösungsansätze sowie deren Implementierung vorgestellt. Dabei wird in diesem Kapitel in die Anpassungen und Ergänzungen der Vorverarbeitung, der Klassifizierungsalgorithmen sowie ihrer Parameter und die Methodik der Merkmalsselektion unterschieden.

6.1. Vorverarbeitung

In diesem Abschnitt werden die implementierten Anpassungen der Vorverarbeitungsschritte, basierend auf den Ergebnissen der Fehleranalyse beschrieben. Die Aufteilung der Anpassungen orientiert sich an den in Abschnitt 5.2 definierten Fehlerklassen. Es ist anzumerken, dass alle hinzugefügten numerischen Merkmale auf das Intervall $[-1, 1]$ transformiert werden.

NLP und Umfang

Im Bezug auf die Behebung der Fehler in der Vorverarbeitung wird eine Anpassung am Konzept von Horstmann [14] vorgenommen, sodass weder URLs noch Auszüge von Programmcode für die Extraktion der Emoticon- und Emotionsmerkmale betrachtet werden. Entsprechend können innerhalb von URLs oder Programmcode kein Sentiment und keine Emoticons mehr gefunden werden.

Bezüglich des fehlenden Umfangs der verwendeten Lexika (Emotion und Emoticon), werden keine Änderungen vorgenommen. Nach aktueller Kenntnis existiert kein Sentimentlexikon für die auf Github verfügbaren Emojis. Hier ließen sich potenziell Parallelen aus plattformübergreifenden Emojilexika ziehen, dies lag jedoch nicht im Fokus dieser Arbeit und sollte daher Bestandteil zukünftiger Forschung sein. Auch ein Vergleich der verwendeten Lexika mit anderen Lexika des Sentiments bzw. der Emoticons wäre für weitere Iterationen des Konzepts denkbar. Eine Untersuchung verschiedener NLP-Bibliotheken im Hinblick auf das Beheben des Problems der fehlerhaften Autokorrektur und Erkennung von Eigennamen stellt eine weitere Möglichkeit der Erweiterung dar.

Kontextfehler

Die Integration eines *Distributional Semantic Models* (DSM) und das Ableiten von Merkmalen anhand dieses Modells stellt eine Möglichkeit dar, Kontextinformationen in

den Klassifizierungsprozess zu integrieren [23]. Ein DSM stellt dabei maschinell erlernte Vektorrepräsentationen eines ausgewählten Vokabulars dar [6]. Im Prozess der Erstellung dieser Repräsentation fließen dabei Informationen bezüglich des Verwendungskontextes betrachteter Token in die jeweilige Vektorrepräsentation eines Token ein [23]. Ähnliche Repräsentationen zweier Token lassen dann auf einen ähnlichen Nutzungskontext schließen [24].

Da das Generieren einer eigenen DSM nicht Ziel dieser Arbeit war, wird stattdessen die von Calefato et al. [6] im Kontext der Erstellung von Senti4SD vorgestellte DSM[1] verwendet. Die DSM von Calefato et al. [6], trainiert auf Textdaten des Frage-Antwort-Forums „Stack Overflow“ umfasst ein Vokabular von mehr als 340.000 Wörtern, für die jeweils eine 600-elementige Vektorrepräsentation abrufbar ist. Die Architektur der DSM von Calefato et al. [6] basiert dabei auf dem CBOW-Ansatz (Continuous-Bag-of-Words) der word2vec-Architektur [23]. Da die DSM auf StackOverflow-Daten basiert, findet ein Einbinden domänenspezifischen Kontextes statt. Für die Integration dieses DSM in das Konzept von Horstmann [14] wird das python-Paket *gensim*[2] verwendet.

Anhand der DSM werden dann analog zu der Idee und Implementierung von Calefato et al. [6] die in Tabelle 6.1 aufgeführten Merkmale extrahiert. Im Rahmen der Ermittlung dieser Merkmale wird für jedes Wort eines Dokuments der jeweilige Vektorwert, falls im Vokabular der DSM enthalten, ermittelt und aufsummiert. Es entsteht für jedes Dokument ein einzelner Vektorwert, welcher anhand der Kosinus-Ähnlichkeit auf Ähnlichkeit mit einem von drei von Calefato et al. [6] als „Prototypvektoren“ bezeichneten Vektoren geprüft wird [6]. Die Prototypvektoren p_{pos}, p_{neg} und p_{neu} basieren dabei analog zum Ansatz von Calefato et al. [6] auf den Vektorsummen aller positiven, neutralen bzw. negativen Einträge des in dieser Arbeit verwendeten Sentimentlexikons. Auch die Bezeichnungen der Merkmale (siehe Tabelle 6.1) sind analog zu denen aus Senti4SD [6].

Name	**Beschreibung**
sim_pos	Kosinusähnlichkeit zu p_{pos}
sim_neu	Kosinusähnlichkeit zu p_{neu}
sim_neg	Kosinusähnlichkeit zu p_{neg}

Tabelle 6.1.: Namen und Beschreibungen der DSM-Merkmale.

Fehlende Funktionalität

Wie in Abschnitt 5.2 angesprochen ist das Handhaben von Negationen (im folgenden als *Negation Handling* bezeichnet) ein wichtiger Teil der Sentimentanalyse [31]. Im Kontext des überwachten Lernens existieren hier zwei Kategorien von Herangehensweisen für das Negation Handling [32]: Implizite Ansätze, die auf das Extrahieren höherdimensionaler

[1] Entnommen aus https://github.com/collab-uniba/pySenti4SD.
[2] https://radimrehurek.com/gensim/

N-Gramme setzen und explizite Ansätze, wie sie z.B. von Ahmed et al [4] in ihrer Anwendung „SentiCR“ verwendet werden. Im Rahmen der Vorverarbeitung in SentiCR werden dabei Token, die auf ein Negationswort folgen, vor der Verarbeitung durch einen TF-IDF-Vectorizer mit dem Präfix „not_“ versehen [4].

Im Kontext dieser Arbeit wird ein implizites Negation Handling implementiert. Um einfache Negationsstrukturen, die sich über maximal zwei Wörter erstrecken (z.B. „not happy“) in den Klassifizierungsprozess einzubeziehen, werden die beiden *Bag-of-Words-Vectorizer* (siehe Abschnitt 4.1.2) angepasst: Zusätzlich zu den bisher extrahierten *Monogrammen* werden auch *Bigramme* bestimmt. Ein Bigramm stellt dabei ähnlich einem Monogramm einen Eintrag des Vokabulars der Vectorizer dar. Bigramme fassen zwei im Text aufeinanderfolgende Token zu einem Eintrag zusammen. Der Termpräsenzvektor z.B. umfasst dann auch Einträge wie „*not* happy“. Die erweiterte Repräsentation erlaubt auch eine Integration einfacher „Verstärkungsstrukturen“ nach Definition von Islam et al. [16] wie „*really* good“. Damit diese Strukturen korrekt erhoben werden können, sind Anpassungen an der verwendeten Stoppwortliste notwendig. Diese enthält in ihrer unangepassten Form häufig verwendete Negationswörter, wie „no“ oder „not“, welche für die Betrachtung der besagten Strukturen notwendig sind. Damit die Liste der Stoppwörter zielgerichtet zu kürzen, wird sich an den Wortlisten für Negations- und Verstärkungswörter[3] der SentiStrength-Anwendung [34] orientiert. Dementsprechend werden Überschneidungen der Stoppwortliste und den Wortlisten aus der Stoppwortliste entfernt. Insgesamt werden die folgenden 11 Wörter aus der Stoppwortliste entfernt:

$$no, nor, not, could, did, ought, should, so, some, very, would$$

Die Extraktion höherdimesionaler N-Gramme ist außerdem eine Möglichkeit, weitere Kontextinformationen in den Klassifikationsprozess einzubinden [40]. Bigramme enthalten Informationen über die Reihenfolge von Token innerhalb eines Textes und bieten somit einfache Informationen über den Verwendungskontext dieser Token und die Struktur des Textes [40].

Domänenspezifität

Um das Problem des domänenabhängigen Sentiments des betrachteten Vokabulars zu beheben, wird das von Islam et al. [16] entwickelte Sentimentlexikon[4] für die Extraktion der Emotionsmerkmale verwendet. Dieses Lexikon wurde speziell auf die Domäne der Softwareentwicklung angepasst und umfasst, verglichen mit dem bisher verwendeten Lexikon, ein deutlich reduziertes Vokabular (siehe Abbildung 6.1). Trotz des geringen Umfangs führte die Integration dieses Lexikons in den Untersuchungen von Islam et al. [16] zu einem nennenswerten Anstieg der Klassifizierungsperformanz ihrer SentiStrength-SE Anwendung gegenüber einer SentiStrength-Baseline [34]. Die Interpretation der Sentimentwerte des neuen Lexikons ist dabei analog zu der in Abschnitt 4.2 formulierten

[3] Entnommen aus SentiStrength, http://sentistrength.wlv.ac.uk/
[4] Entnommen aus SentiStrength-SE, V1.5, https://laser.cs.uno.edu/Projects/Projects.html

Interpretation des bisher verwendeten Lexikons [16]. Es ist anzumerken, dass die Integration eines neuen Lexikons eine erneute Ermittlung der zuvor genannten und nach Idee von Calefato et al. [6] implementierten „Prototypvektoren“ voraussetzt.

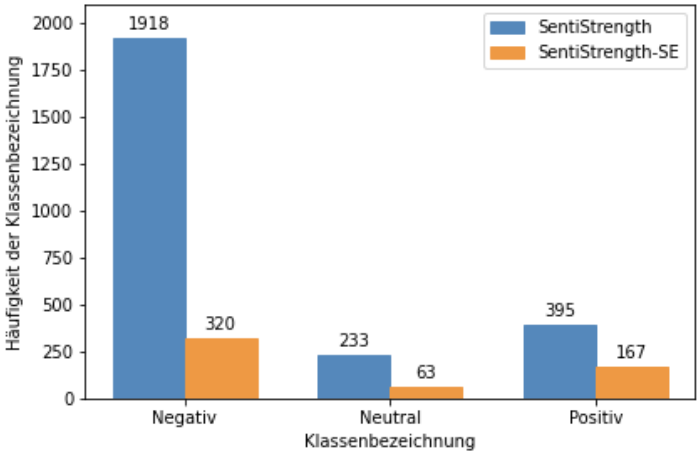

Abbildung 6.1.: Vergleich der Klassenhäufigkeiten des SentiStrength-Lexikons [34] und des SentiStrength-SE-Lexikons [16].

Hinzufügen aussagekräftigerer Merkmale

Bei Betrachtung der Fehlklassifizierungen im Rahmen der Fehleranalyse ist außerdem aufgefallen, dass die bisher extrahierten Sentimentmerkmale (*senti_mean, senti_min, senti_max*, siehe Abschnitt 4.2) nicht dazu geeignet sind, Häufigkeiten von Sentimentwörtern zu modellieren. Dokumente, die überdurchschnittlich viele Sentimentwörter enthalten, werden dabei ähnlich informativ „modelliert“ wie Dokumente mit wenigen Sentimentwörtern. Um dieses Problem aufzugreifen, werden die Merkmale der Klasse „Emotion“ und „Emoticon“ (siehe Abschnitt 4.1.2) um die in Tabelle 6.2 aufgeführten Merkmale ergänzt. Die Idee und Struktur der Merkmale ist dabei aus Teilen der von Calefato et al. [6] für Senti4SD definierten und den von Basile et al. [5] definierten Merkmalen übernommen. Im Rahmen dieser Änderungen wird die bisherige Merkmalsmenge um neun Merkmale erweitert.

Verbleibende Fehlerklassen

Sechs der 100 betrachteten Fehlklassifizierungen sind auf Höflichkeit zurückzuführen. Um das grundlegende Problem dieser Fehlerklasse zu beheben, ist eine Anpassung des SentiStrength- [34] bzw. SentiStrength-SE-Lexikons [16] für die oft in höflichen Formulierungen auftretenden Wörter (z.B. „thanks“ oder „please“) notwendig. Im Hinblick auf die geringe Häufigkeit dieser Fehlerklasse und die angestrebte Generalisierbarkeit der

Kategorie	Name	Beschreibung
Emotion	num_pos_emotion	Anzahl Tokens mit positivem Sentiment
	num_neg_emotion	Anzahl Tokens mit negativem Sentiment
	sum_pos_emotion	Summe positiver Sentimentwerte
	sum_neg_emotion	Summe negativer Sentimentwerte
	last_pos	Sentiment des letzten positiven Tokens
	last_neg	Sentiment des letzten negativen Tokens
Emoticon	emot_num_pos	Anzahl Emoticons mit positivem Sentiment
	emot_num_neg	Anzahl Emoticons mit negativem Sentiment
	emot_last	Sentiment des letzten Emoticons

Tabelle 6.2.: Namen und Beschreibungen der hinzugefügten Merkmale der Kategorien „Emotion“ und „Emoticon“.

Anwendung auf Daten außerhalb des GitHub-Datensatzes, wird für diese Fehlerklasse keine Änderung implementiert.

Für die Klasse der nicht definierbaren Fehler stellen die bereits angesprochenen Ansätze der Verbesserung der Genauigkeit, der aussagekräftigeren Merkmale sowie der Hyperparameteroptimierung eine Möglichkeit dar, die Häufigkeit dieser Klasse implizit zu verringern.

6.2. Klassifizierungsalgorithmen

In diesem Abschnitt werden die Verbesserungsansätze besprochen, die auf den Klassifizierungsalgorithmen und ihren Parametern aufbauen.

6.2.1. Vergleich ausgewählter Klassifizierungsalgorithmen

Wie in Kapitel 5 beschrieben, erzielt der *Naive-Bayes*-Klassifizierer im Kontext der verwendeten Merkmale keine guten Ergebnisse. Diese Tatsache lässt darauf schließen, dass für die Aufgabe der Sentimentanalyse verschiedene Arten von Klassifizierungsalgorithmen unterschiedlich gut mit den extrahierten Merkmalen arbeiten. Dementsprechend wird in diesem Abschnitt eine ausgewählte Menge an Klassifizierungsalgorithmen auf ihre Passform für die vorliegende Aufgabe der Klassifizierung untersucht. Dieser Ansatz ist analog zu der Auswertung der Klassifizierungsalgorithmen in der Arbeit von Ahmed et al. [4] und der Arbeit von Islam et al. [15]. Folgende Auswahl an Algorithmen wird analog zu der Auswahl in den aufgeführten Arbeiten betrachtet:

- Random-Forest-Classifier (RF)
- Lineare Support-Vector-Machine (SVC)
- Multi-Layer-Perceptron-Classifier (MLP)

- Adaboost-Classifier (AB)
- Gradient-Boosting-Classifier (GB)
- K-Nearest-Neighbors Classifier (KNN)
- Decision-Tree-Classifier (DT)
- SVC mit Stochastic Gradient Descent (SGD)

Für die Implementierung der Algorithmen wird, im Einklang mit der bisherigen Implementierung der Algorithmen, das *scikit-learn* python-Paket verwendet. Dabei werden alle Algorithmen anhand ihrer voreingestellten Parameter beurteilt; eine Anpassung der Parameter erfolgt nicht. Ausgewertet werden die Algorithmen auf dem bisherigen Datensatz und das Ergebnis wird 5-fach kreuzvalidiert. Der in Kapitel 5 besprochene *Naive-Bayes*-Klassifizierer wird dabei in dieser Untersuchung aufgrund seiner stark unterdurchschnittlichen Performanz nicht betrachtet. Es ist anzumerken, dass dieser Vergleich auf der Merkmalsmenge durchgeführt wird, die bereits die Änderungen aus Abschnitt 6.1 enthält. Dies garantiert, dass die Klassifizierer ausgewählt werden, die anhand der aufgeführten Änderungen und der erweiterten Funktionalität die beste Genauigkeit erzielen. Die Ergebnisse der Auswertung sind in Abbildung 6.2 zu sehen. Betrachtet wird für diese Auswertung nur die Genauigkeit der Algorithmen.

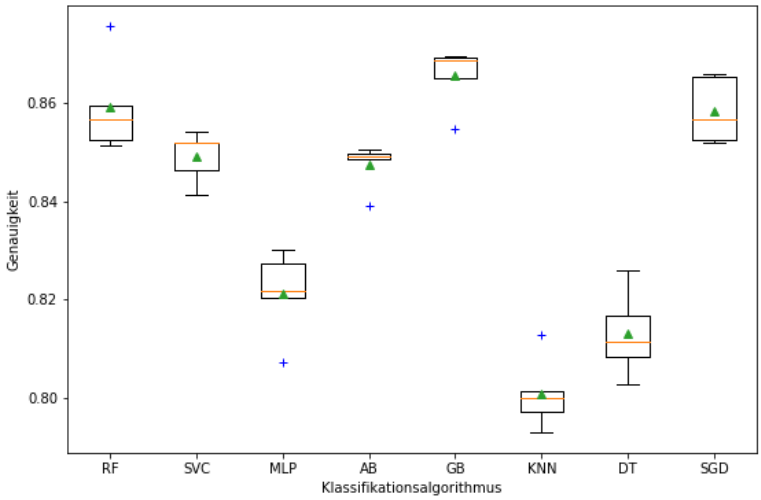

Abbildung 6.2.: Genauigkeit der acht Klassifizierungsalgorithmen nach 5-facher Kreuzvalidierung. Die grünen Dreiecke beschreiben die Mittelwerte. Ausreißer sind durch ein blaues Plus gekennzeichnet.

Es fällt zunächst auf, dass sämtliche Algorithmen eine durchschnittliche Genauigkeit von über 80% erzielen. Dennoch führt die Verwendung verschiedener Klassifizierer zu einem

Verlust an Genauigkeit. Auffällig sind hier besonders der MLP, der KNN und der DT, die allesamt negativ von der Genauigkeit der bisher verwendeten Algorithmen (SVC, RF) abweichen. Die bisher akzeptable Performanz der SVC und des RF wird vom GB und dem SGD noch übertroffen, wenn auch nur um wenige Prozentpunkte.

Basierend auf den durchschnittlichen Genauigkeiten werden die drei besten Klassifizierer ausgewählt und über die Ensemblemethode kombiniert. Das finale Ensemble besteht dementsprechend aus dem *Gradient-Boosting*-Klassifizierer, dem *Random-Forest*-Klassifizierer und dem *Stochastic-Gradient-Descent*-Klassifizierer.

6.2.2. Hyperparameteroptimierung

In Abschnitt 5.4 wurde das Problem der Generalisierbarkeit des *Random-Forest*-Klassifizierers angesprochen und die Methodik der Hyperparameteroptimierung als Verbesserungsmöglichkeit vorgestellt. Hyperparameter sind Parameter eines Klassifizierungsalgorithmus', die das Modellverhalten bzw. Teile des Modellverhaltens bestimmen [11]. Eine Anpassung dieser Parameter kann so z.B. die erlaubte Komplexität eines Algorithmus' begrenzen und Aspekten wie einer Überanpassung entgegenwirken [11].

Im Rahmen dieser Optimierung, werden für die drei verwendeten Klassifizierungsalgorithmen die Parameter mit dem potenziell größten Einfluss auf die Klassifizierungsperformanz betrachtet [1, 2]. Diese Parameter sind zusammen mit dem betrachteten Wertebereich der Parameter, in Tabelle 6.3 aufgeführt. Untersucht werden die Parameter bzw. die Kombinationen von Parametern unter Verwendung der Methodik der „Rastersuche". Im Kontext dieser werden alle Parameterwerte bzw. Kombinationen von Parameterwerten, im Hinblick auf die erzielte Klassifizierungsgenauigkeit ausgewertet und die Ergebnisse 5-fach kreuzvalidiert. Die Parameterwerte bzw. die Kombinationen von Parameterwerten, welche die Genauigkeit maximieren sind dabei in Tabelle 6.3 aufgeführt.

Algorithmus	**Parameter**	**Betrachtete Werte**	**Finaler Wert**
Gradient-Boosting-C.	n_estimators	$\{100, 200, 300\}$	300
	learning_rate	$\{0.1, 0.3, 0.6, 1.0\}$	0.1
Random-Forest-C.	n_estimators	$\{100, 300, 500, 700\}$	700
	max_features	{log2, sqrt, auto}	auto
Stochastic-Gradient-Desc.	α	$10^{-i}, i \in \{1, 2, \ldots, 7\}$	10^{-3}

Tabelle 6.3.: Betrachtete Parameter der Klassifizierer und die ausgewählten Werte.

6.3. Evolutionärer Algorithmus

In Kapitel 5 wurde die Annahme formuliert, dass eine Ergänzung des evolutionären Algorithmus um weitere grundlegende Operatoren evolutionärer Algorithmen, im Hinblick auf eine Verbesserung der „Exploration“ des Algorithmus', die Ergebnisse des EA verbessern könnte. Die Implementierung dieser Operatoren soll dazu führen, dass eine umfassendere Durchsuchung des Suchraumes erfolgt und somit Kombinationen von Merkmalen ausgewertet werden, die eine bessere Klassifizierungsgenauigkeit erzielen, als die dem Startvektor ähnlichen Mengen des EA von Horstmann [14]. Ob ein gewählter Operator den Faktor der Exploration tatsächlich verbessert, ist ohne umfassende Untersuchungen nicht festzustellen [36]. Dennoch sollten die implementierten Anpassungen dazu führen, dass eine größere Anzahl einzigartiger Vektoren und eine größere Vielfalt von Kombinationen der Merkmale untersucht und potenziell eine Merkmalsmenge ermittelt wird, die eine bessere Genauigkeit erzielt als eine von Horstmanns [14] EA generierte Menge. Um diese Annahme zu untersuchen, wird ein erweiterter EA formuliert sowie implementiert und mit dem bisher verwendeten EA verglichen.

6.3.1. Funktionsweise

Erweitert wird der EA dabei um Operatoren der Selektion und Rekombination. Außerdem wird statt des voll besetzten Merkmalsvektors, eine Menge zufällig generierter Startvektoren betrachtet. Der erweiterte EA (EEA) hat dementsprechend drei Eingabeparameter: pop_size, p_{mut} und num_generations. Pop_size bestimmt dabei die Populationsgröße, p_{mut} die Wahrscheinlichkeit der Mutation eines Bits und num_generations die Anzahl durchlaufener Generationen.

Ähnlich zu dem bisher verwendeten EA wird die Genauigkeit des Ensembles trainiert mit der betrachteten Merkmalsmenge als Fitnessfunktion verwendet. Der EEA arbeitet wie folgt: Initial wird die Menge der Startvektoren der Größe pop_size zufällig generiert. Zu Beginn jeder Generation werden $n \leq$ pop_size Vektoren für die Rekombination ausgewählt. Für den Prozess der Selektion wird die Methode des „Stochastic Universal Samplings“ [39] (SUS) verwendet. Hierzu wird die relative Fitness aller Individuen der Population $(p_1, \ldots, p_{\text{pop\_size}})$ bestimmt und basierend auf dieser relativen Fitness n Individuen zufällig ausgewählt. Je höher die relative Fitness eines Individuums, desto höher die Wahrscheinlichkeit ausgewählt zu werden. Abbildung 6.3 zeigt das Prinzip der SUS für $n =$ pop_size $= 4$. Die auf diese Art ausgewählten Individuen werden dann rekombiniert. Hierfür werden alle einzigartigen Paare der ermittelten Individuen dazu verwendet, über den Operator des „Uniform Crossovers“ [39] (UC) jeweils zwei Nachfolger zu generieren. Die Methode des UC arbeitet so, dass zufällig entweder ein Biteintrag des ersten Elternvektors oder des zweiten Elternvektors mit gleicher Wahrscheinlichkeit ausgewählt wird und den jeweiligen Biteintrag des Nachfolgevektors darstellt. Ein Beispiel für die Funktionsweise dieses Operators ist in Abbildung 6.4 zu sehen.

Auf diese Weise erzeugte Nachfolger werden dann einer wie bisher verwendeten zufälligen

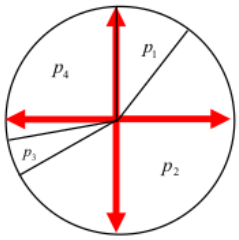

Abbildung 6.3.: Funktionsweise des „Stochastic Universal Sampling“. Abbildung entnommen aus „Introduction to evolutionary algorithms“ von Yu et al. [39, S. 68].

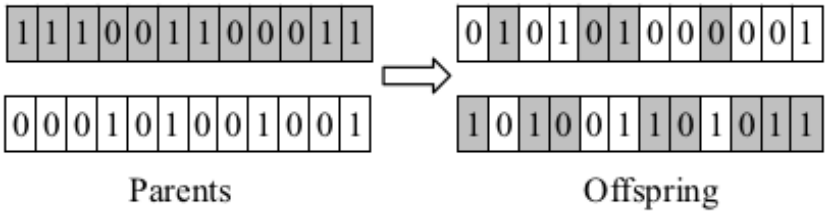

Abbildung 6.4.: Funktionsweise des ”Uniform Crossovers”. Abbildung entnommen aus „Introduction to evolutionary algorithms“ von Yu et al. [39, S. 44].

Bitflip-Mutation unterzogen. Die Wahrscheinlichkeit eines Bitflips ergibt sich aus dem Parameter p_{mut}. Abschließend werden die fünf besten Individuen aus der Menge der Eltern- und Nachfolgerindividuen, im Rahmen eines „Fitness-Based-Replacements“ [39] ermittelt und als initiale Population für die nächste Generation festgelegt. Das Ergebnis des EEA nach Durchlaufen der *num_generations* Generationen, ist die im letzten Durchlauf generierte Population.

6.3.2. Vergleich der Algorithmen

Um festzustellen welchen Einfluss das Generieren einer reduzierten Merkmalsmenge auf die Klassifikationsperformanz hat und um die in Abschnitt 5.4 formulierte Annahme zu untersuchen, werden im folgenden zwei Merkmalsmengen ausgewertet. Es ist anzumerken, dass die Anwendung beider EA auf den in Abschnitt 6.1 formulierten Änderungen sowie den neu ausgewählten Klassifikationsalgorithmen (siehe Abschnitt 6.2.1), aufbaut. Merkmalsmenge M_1 basiert auf einer Anwendung von Horstmanns [14] EA (HEA), während Merkmalsmenge M_2 die Merkmalsmenge des besten Individuums des Ergebnisses des EEA beschreibt. Der EEA wird dabei mit den in Tabelle 6.4 aufgelisteten Parametern betrieben. Die Ergebnisse beider EA basieren auf einer Durchführung der Algorithmen über 50 Generationen.

Tabelle 6.5 zeigt die Auswertung der beiden Merkmalsmengen M_1 und M_2, 5-fach kreuzvalidiert, verglichen mit der vor der Durchführung der Merkmalsselektion ermittelten

Parameter	Wert
pop_size	5
p_{mut}	0,02
num_generations	50

Tabelle 6.4.: Parameter des EEA für die Auswertung.

Performanz, B.

In der Tabelle fällt auf, dass der HEA und der EEA zwei Merkmalsmengen generieren, die sich in ihrer erzielten Genauigkeit nicht unterscheiden. Dies ist festzustellen, trotz großer Abweichungen im Aufbau der erzeugten Merkmalsvektoren der Länge 41. M_1 verwendet 37 Merkmale, M_2 lediglich 21 Merkmale. Beide Mengen erzielen eine Verbesserung der Genauigkeit gegenüber der Baseline um 0,02. Die vom HEA erzeugte Menge führt zu einem größeren Anstieg des F_1-Wertes der negativen Klasse, die vom EEA erzeugte Menge bewirkt einen größeren Anstieg in der neutralen Klasse. Für die weiteren Untersuchungen wird die vom EEA erzeugte Merkmalsmenge M_2 ausgewählt, da diese insgesamt weniger Merkmale enthält und eine bessere Laufzeit erzielt.

	Negativ			**Neutral**			**Positiv**			
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**A**
B	86,6	82,5	84,4	**84,7**	90,2	87,3	91,2	**86,5**	88,8	86,9
M_1	**86,7**	**83,1**	**84,9**	**84,7**	90,2	87,4	91,6	**86,5**	**88,9**	**87,1**
M_2	86,2	83,0	84,6	**84,7**	**90,8**	**87,6**	**92,2**	85,6	88,8	**87,1**

Tabelle 6.5.: Metriken für verschiedene Merkmalsmengen. Beide Algorithmen wurden für 50 Generationen ausgeführt und die Ergebnisse 5-fach kreuzvalidiert.

7. Evaluation

Die im Kontext von Kapitel 6 identifizierten Verbesserungsmöglichkeiten werden in diesem Kapitel auf dem Goldstandard-Datensatz von Novielli et al. [27] evaluiert. In der darauf folgenden Diskussion wird die zweite Forschungsfrage beantwortet und Schlussfolgerungen aus der Evaluation gezogen.

7.1. Methodik

Um eine Aussage darüber treffen zu können, welche Auswirkung die identifizierten Verbesserungen auf die Klassifizierungsperformanz aufweisen, werden die in Kapitel 6 formulierten Verbesserungen sukzessive der „Baseline“ von Horstmann [14] hinzugefügt (siehe Abschnitt 4.2). Hierzu werden im Voraus die folgenden Gruppen von Änderungen formuliert:

B Die Baseline von Horstmann [14]. Es werden alle Merkmale der Baseline betrachtet, der EA wird nicht angewendet.

S_1 Lösungsansätze für die Fehlerklasse „NLP und Umfang“.

S_2 Neue Emotions- und Emoticonmerkmale.

S_3 Lösungsansätze für die Fehlerklasse „Kontextfehler“.

S_4 Lösungsansätze für die Fehlerklasse „Domänenspezifität“.

S_5 Lösungsansätze für die Fehlerklasse „Fehlende Funktionalität“.

S_6 Verwenden der neuen Klassifikationsalgorithmen.

S_7 Anwenden der vom erweiterten EA ermittelten Merkmalsmenge.

S_8 Verwenden der als optimal ermittelten Hyperparameter.

Das Konzept von Horstmann [14] wird unter Verwendung dieser Gruppen trainiert und im Kontext einer 5-fachen Kreuzvalidierung ausgewertet. Es ist anzumerken, dass eine Merkmalsgruppe alle Änderungen der zuvor hinzugefügten Gruppen umfasst und so auf den vorangehenden Gruppen aufbaut. $S_1, \ldots, S_5$ beschreiben die Anpassungen bzw. Ergänzungen an den Schritten der Vorverarbeitung, $S_6, \ldots, S_8$ die Anpassungen an den Klassifizierungsalgorithmen und die Optimierung dieser.

Klasse	**Metrik**	**Änderungen**								
		B	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
Negativ	P	77,9	78,6	80,4	81,0	83,4	83,7	**86,6**	86,2	**86,6**
	R	78,9	80,3	83,5	**84,2**	82,8	80,8	82,5	83,0	83,4
	F_1	78,3	79,4	81,9	82,5	83,1	82,2	84,4	84,6	**84,9**
Neutral	P	81,3	81,5	83,6	**85,1**	84,3	82,5	84,7	84,7	84,5
	R	83,5	83,8	84,7	85,6	87,5	88,3	90,2	90,8	**90,9**
	F_1	82,4	82,6	84,1	85,3	85,9	85,3	87,3	**87,6**	**87,6**
Positiv	P	87,5	88,4	90,1	89,9	89,8	90,1	91,2	92,2	**92,4**
	R	82,6	82,7	84,6	85,3	85,2	83,6	**86,5**	85,6	85,5
	F_1	85,0	85,5	87,2	87,5	87,4	86,7	**88,8**	**88,8**	**88,8**
	A	81,9	82,4	84,3	85,1	85,5	84,8	86,9	**87,1**	**87,1**
	ΔA	-	+0,5	+1,9	+0,8	+0,4	-0,7	+2,1	+0,2	+0

Tabelle 7.1.: Auswertung der sukzessiv hinzugefügten Merkmale bzw. Methoden. Ergebnisse 5-fach kreuzvalidiert. Die Zeile „ΔA“ steht für die Änderung der Genauigkeit gegenüber der zuvor betrachteten Merkmalsmenge. Die besten Ergebnisse einer Metrik sind fett unterlegt.

7.2. Ergebnis

Das Ergebnis der Auswertung ist in Tabelle 7.1 zu sehen. Aufgeführt ist die Performanz der drei Polaritätsklassen für die Baseline sowie die definierten Merkmalsmengen. Die letzte Zeile der Tabelle beschreibt dabei die Änderung der Genauigkeit die durch Implementierung der Menge erreicht wird.

Hier ist zunächst die Verbesserung der Klassifizierungsgenauigkeit im Vergleich mit der Baseline anzumerken. Die im Kontext dieser Arbeit vorgenommenen Änderungen haben zu einer Verbesserung der Genauigkeit um insgesamt 5.1%-Punkte geführt. Dabei profitiert vor allem die Performanz der negativen Klasse von den Änderungen. Der F_1-Wert der negativen Klasse verbessert sich um insgesamt 6,6%-Punkte, verursacht durch einen starken Anstieg der Precision und eines moderaten Anstiegs des Recalls. Sowohl in der neutralen Klasse, als auch der positiven Klasse, sind positive Auswirkungen der Veränderungen festzustellen. So verbessert sich der Recall der neutralen Klasse um +7,4%-Punkte. Die positive Klasse, die bereits zu Beginn eine vergleichsweise gute Leistung erzielte, vermerkt den geringsten Anstieg des F_1-Wertes, um 3,8%-Punkte.

76% des Anstiegs der Genauigkeit sind dabei auf das Hinzufügen der neuen Emotions- und Emoticonmerkmale sowie das Ersetzen der bisher verwendeten Klassifizierer zurückzuführen. Sowohl die DSM-Merkmale als auch das domänenspezifische Lexikon, sowie das Filtern von URLs, führen zu moderaten Verbesserungen der Klassifizierungsperformanz. Es ist zu beobachten, dass die Implementierung des impliziten Negation Handlings zu einer Verschlechterung der Klassifizierungsperformanz um 0,7%-Punkte führt. Diese Tatsache ist wahrscheinlich auf den vergrößerten Umfang der Merkmalsmenge im

Rahmen der Extraktion von Bigrammen zurückzuführen. Überraschenderweise führen die beiden Schritte der Optimierung (S_7, S_8) zu einer sehr geringen Veränderung der Performanz. Die Optimierung der Merkmalsmenge führt dabei zu einer Verbesserung um 0,2%-Punkte, während die angepassten Hyperparameter keinen vermerkbaren Einfluss auf die Performanz haben.

	Negativ			**Neutral**			**Positiv**			
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**A**
GB	85,7	82,9	84,2	84,6	89,6	87,0	90,9	85,8	88,2	86,6
GB*	86,4	82,4	84,4	84,5	90,2	87,2	90,8	85,9	88,3	86,7
RF	86,1	81,5	83,8	83,5	90,4	86,8	91,8	85,2	88,4	86,4
RF*	86,0	82,0	84,0	83,7	90,5	87,0	92,1	85,3	88,6	86,5
SGD	85,5	81,7	83,5	82,9	88,1	85,4	89,6	85,0	87,2	85,4
SGD*	87,9	80,6	84,1	83,4	90,6	86,8	90,6	86,4	88,4	86,5
E	86,2	83,0	84,6	84,7	90,8	87,6	92,2	85,6	88,8	87,1
E*	86,6	83,4	84,9	84,5	90,9	87,6	92,4	85,5	88,8	87,1

Tabelle 7.2.: Performanz der einzelnen Klassifizierer und des Ensembles vor bzw. nach Anwendung der Hyperparameter (markiert mit *).

7.3. Diskussion

Wie zuvor dargelegt, führte die Implementierung der identifizierten Anpassungen zu einer Verbesserung der Performanz des Konzepts von Horstmann [14] um 5,1%-Punkte. Basierend auf diesem Ergebnis soll nun die zweite Forschungsfrage beantwortet werden, die eingangs wie folgt definiert wurde:

RQ2 Welchen Einfluss haben die Anpassungen auf die Klassifizierungsperformanz?

Sechs der acht identifizierten Verbesserungen haben einen positiven Einfluss auf die Klassifizierungsperformanz. So haben die Anpassungen an der Vorverarbeitung anhand eines Filterns von URLs und Programmcode, dem Hinzufügen weiterer Emotions- und Emoticonmerkmale, der Integration einfacher Kontextinformationen und die Verwendung eines domänenspezifischen Lexikons einen positiven Einfluss auf die Performanz. Lediglich das implizite Negation Handling verschlechtert die Genauigkeit der Klassifizierung. Hinzu kommen die Anpassungen an den Klassifizierungsalgorithmen, nämlich das Ersetzen der bisher verwendeten Algorithmen sowie eine Optimierung der Merkmalsmenge, die einen positiven Einfluss auf die Performanz haben. Eine Optimierung der Hyperparameter besagter Algorithmen führte zu keiner vermerkbaren Verbesserung der Genauigkeit.

7.3.1. Interpretation der Ergebnisse

Aufbauend auf der Antwort auf die zweite Forschungsfrage wird im Folgenden der Einfluss der Anpassungen genauer ausgeführt und Schlussfolgerungen aus den Einsichten gezogen.

Wie angemerkt, hat lediglich die Extraktion von Bigrammen im Kontext des impliziten Negation Handlings zu einer Verschlechterung der Genauigkeit geführt. Diese Tatsache ist wahrscheinlich auf die Zunahme der Merkmalsmenge zurückzuführen. Bigramme stellen dabei ca. 1500 weitere Merkmale pro Dokument dar. Die Zunahme der Merkmalsmenge verbunden mit der Anpassung der verwendeten Stoppwortliste scheint das Problem der Überanpassung durch das Hinzufügen von Störfaktoren zu verstärken. Die Implementierung einer Methode des Negation Handlings bleibt dabei dennoch relevant. Statt des verwendeten impliziten Ansatzes sollten dabei Ansätze verwendet werden, welche die Merkmalsmenge nicht ähnlich stark vergrößern wie der in dieser Arbeit implementierte Ansatz.

Das Filtern von URLs und Programmcode hat einen positiven Einfluss auf alle betrachteten Klassen, wobei der höchste Anstieg des F_1-Wertes in der negativen Klasse zu vermerken ist. Der Anstieg der Precision weist darauf hin, dass die Klassifizierung tatsächlich durch die in URLs enthaltenen Emoticons negativ beeinflusst wurde.

Der zweithöchste Anstieg der Klassifizierungsgenauigkeit ist auf das Einfügen weiterer Emotions- und Emoticonmerkmale (S_2) zurückzuführen. Eine umfassende Integration der Sentimentwerte erscheint im Kontext der Sentimentanalyse als essenziell.

Die DSM-Merkmale (S_3) haben einen positiven Einfluss auf alle Klassen, verbessern jedoch besonders die Performanz der negativen und neutralen Klasse. Besonders der F_1-Wert der neutralen Klasse steigt um 1,2%-Punkte. Dies kann auf die Tatsache zurückzuführen sein, dass die DSM-Merkmale das bisher einzige Merkmal beinhalten, das explizit die Neutralität eines Dokumentes beschreibt. Eine umfangreichere Modellierung der Aspekte der neutralen Klasse könnte die Performanz dieser weiter verbessern. Es ist jedoch anzumerken, dass der dargestellte Bezug zwischen DSM und Kontext eines Textes, auf einer Fehlannahme basiert. Bei Erstellung der DSM fließen zwar Kontextinformationen in die jeweilige Vektorrepräsentation eines Wortes ein, diese stellen jedoch keine direkte Lösung für die in Abschnitt 6.1 definierten Kontextfehler dar. Stattdessen könnte eine Integration der in der Vorverarbeitung ermittelten *Part-of-Speech*-Zuweisungen für das Auflösen des Problems der Mehrdeutigkeiten von Wörtern erfolgen.

Während das domänenspezifische Lexikon (S_4) die Performanz der positiven Klasse geringfügig verschlechtert, sind Anstiege der F-Werte in den beiden verbleibenden Klassen zu bemerken. Diese Tatsache deutet an, dass ein Großteil der Fälle domänenspezifischen Vokabulars von negativem Sentiment ausgehen, welches innerhalb der betrachteten Domäne jedoch als neutral zu annotieren ist. Das beobachtete Ergebnis ist wahrscheinlich auf den geringeren Umfang des SentiStrength-SE-Lexikons [16] und

den deutlich verringerten Umfang des negativen Vokabulars dieses Lexikons, gegenüber dem SentiStrength-Lexikon [34] zurückzuführen (siehe Abschnitt 6.1). Die Verwendung eines domänenspezifischen Lexikons ist insgesamt als positive Änderung des Konzepts zu vermerken.

Anzumerken sind zudem die geringen Änderungen durch die beiden Optimierungsschritte der Merkmalsselektion und Hyperparameteroptimierung. Im Rahmen der Aufwertung der Merkmalsselektion wurde ein erweiterter evolutionärer Algorithmus vorgestellt. Es wurde gezeigt, dass dieser Algorithmus eine Merkmalsmenge generiert, welche die gleiche Genauigkeit erzielt, wie eine von Horstmanns [14] EA generierte Merkmalsmenge (siehe Abschnitt 6.3.2). Dieses Ergebnis deutet an, dass der Faktor der Exploration im Kontext des vorliegenden Optimierungsproblemes nicht so relevant ist, wie zunächst vermutet. Große Abweichungen von der vollständigen Merkmalsmenge sind potenziell nicht erforderlich. Diese Annahme basiert jedoch auf einer einzelnen Beobachtung. Weitere Untersuchungen beider evolutionärer Algorithmen sowie ein quantitativer Vergleich der Algorithmen sind für das Treffen fundierter Aussagen erforderlich. Aufgrund des insgesamt geringen Anstiegs der Genauigkeit lässt es sich jedoch vermuten, dass bezüglich der Merkmalsselektion unabhängig von betrachteten Merkmalskombinationen wenig Potenzial für Verbesserungen der Genauigkeit besteht.

Um die geringe Auswirkung der Hyperparameteroptimierung genauer zu untersuchen, werden ergänzend die in Tabelle 7.2 aufgeführten Performanzwerte der Klassifizierer vor bzw. nach Optimierung (markiert durch den *) der Hyperparameter betrachtet. Es fällt auf, dass die untersuchten Parameter für den *Gradient-Boosting*-Klassifizierer (GB) und den *Random-Forest*-Klassifizierer (RF) eine sehr geringe Auswirkung auf die Klassifizierungsgenauigkeit haben (+0,1%-Punkt). Lediglich die Anpassung der Parameter des *Stochastic-Gradient-Descent*-Klassifizierers (SGD) führte zu einem bemerkbaren Anstieg in der Genauigkeit des Klassifizierers um 1,1%-Punkte. Es fällt jedoch auf, dass dieser Anstieg der Performanz des SGD keine Auswirkung auf die Ensembleperformanz (E^*) hat. Eine mögliche Erklärung ist, dass sowohl der GB als auch der RF häufig gleiche Ergebnisse erzeugen und das Ergebnis des SGD im Kontext des verwendeten Mehrheitsentscheides dominieren. Diese Auslegung steht jedoch im Widerspruch zu dem Anstieg der Ensembleperformanz gegenüber dem besten einzelnen Klassifizierer. Würden GB und RF tatsächlich immer gleiche Ergebnisse erzeugen, so wäre ein Anstieg der Ensembleperformanz um 0,4%-Punkte nicht möglich. Weitere Einblicke in das Ensembleverhalten sind notwendig, um die tatsächliche Dynamik des Ensembles abzuschätzen.

7.3.2. Threats to Validity

Folgende Einschränkungen der Validität der Ergebnisse und der Generalisierbarkeit der Ergebnisse sind im Rahmen dieser Arbeit anzumerken:

Initial wurden erforderliche Anpassungen am Konzept von Horstmann [14], die für eine Auswertung auf einem englischen Goldstandard-Datensatz notwendig sind, beschrieben.

Es wurde in dieser Arbeit darauf geachtet, dass die im Rahmen dieser Anpassung vorgenommenen Änderungen die Mechaniken und Ideen des Konzeptes weitestgehend erhalten. Dennoch ist nicht auszuschließen, dass diese Anpassungen einen Einfluss auf die ermittelte Performanz haben und nicht die eigentliche Performanz des Konzeptes von Horstmann [14] beschreiben (Conclusion Validity).

Um sowohl die Klassifizierungsalgorithmen als auch die Performanz verschiedener Merkmals- und Funktionalitätsmengen vergleichen zu können, wurden die Metriken Precision, Recall, Accuracy und F_1 ermittelt. Für die vollzogenen Vergleiche wurden lediglich die numerischen Werte gegenübergestellt, die besonders bei geringen Unterschieden der Zahlenwerte keinen eindeutigen Rückschluss auf tatsächliche Verbesserung oder Verschlechterung bzw. Änderung in der Ausgabe eines Algorithmus' erlauben. Ein Vergleich der Ausgaben bzw. die Prüfung der Änderungen auf statistische Signifikanz anhand eines statistischen Tests wie dem McNemar-Test [22], wurde in dieser Arbeit nicht durchgeführt (Conclusion Validity).

Im Rahmen der Auswertung wurden Klassifizierungsalgorithmen verglichen, welche eine Zufallskomponente enthalten. So basiert die zufällige Einteilung der Bäume innerhalb eines *Random Forest* auf einem veränderlichen *Seed Key*. Keys dieser Art wurden im Rahmen der Auswertungen nicht kontrolliert. Entsprechend kann dies dazu führen, dass trotz Ermittelns der Metriken anhand einer 5-fachen Kreuvalidierung und eine Auswertung auf gleichen Aufteilungen des Datensatzes die Ergebnisse zwischen Ausführungen geringfügig schwanken (Conclusion Validity).

Um den von Horstmann [14] verwendeten Ansatz der Merkmalsselektion zu optimieren, wurden in dieser Arbeit reduzierte Merkmalsmengen unter Verwendung zwei unterschiedlicher evolutionärer Algorithmen generiert. Es ist anzumerken, dass die ermittelten Fitnesswerte der Merkmalsmengen zum Zeitpunkt der Ausführung der EA nicht validiert werden. Die Fitnesswerte basieren auf einer einzelnen Messung der Genauigkeit bei einer gleichbleibenden Aufteilung von 70%/30% des Github-Datensatzes [27]. Dies kann dazu führen, dass Merkmalsmengen, die auf dem gegebenen Datensatz eine schlechtere Performanz erzielen als womöglich im Kontext einer 5-fachen Kreuzvalidierung vom EA verworfen werden (Conclusion Validity).

Im Rahmen der Auswertung der verwendeten Klassifizierungsalgorithmen wurde eine Fehleranalyse durchgeführt, um Schwachstellen in der Vorverarbeitung und Klassifizierung aufzudecken. Eine Fehleranalyse ist dabei eine inhärent qualitative Methode und kann subjektiven Einsichten des Durchführenden unterliegen. Dieser Aspekt der Analyse und die Tatsache, dass der Einordnungsprozess von nur einer Person durchgeführt wurde, kann einen Einfluss auf die Gültigkeit der Ergebnisse der Fehleranalyse haben. Es ist außerdem anzumerken, dass die Stichprobe potenziell nicht repräsentativ für die Gesamtmenge der Fehlklassifizierungen ist (Conclusion Validity).

Im Kontext der Evaluation wurden sukzessiv aufeinander aufbauende Merkmals- und Funktionalitätsgruppen implementiert. In der Evaluation kann es dazu kommen, dass die Verbesserungen bzw. Verschlechterungen von Metriken nicht nur auf das

hinzugefügte Merkmal oder die hinzugefügte Funktion zurückzuführen sind. Es besteht die Möglichkeit, dass die Änderungen in Kombination mit existierenden Mechanismen entstehen; es also zu Wechselwirkungen zwischen den Merkmalen kommen kann und ein Merkmal einzeln betrachtet, zu einer abweichenden Änderung der Metriken führt (Construct Validity).

Sämtliche Auswertungen von Methoden des Horstmann-Konzeptes [14] in dieser Arbeit wurden auf dem Github-Goldstandard-Datensatz von Novielli et al. [27] durchgeführt. Es besteht die Möglichkeit, dass sowohl Werte der Metriken als auch die ermittelten Parameter der Algorithmen und die Merkmalsmengen spezifisch für den verwendeten Datensatz sind. Es ist nicht auszuschließen, dass eine Überanpassung an den vorliegenden Datensatz erfolgt, welche die Generalisierbarkeit der Ergebnisse einschränkt (External Validity).

8. Zusammenfassung und Ausblick

Abschließend werden in diesem Kapitel die Untersuchungen und Ergebnisse dieser Arbeit zusammengefasst und darauf aufbauend ein Ausblick auf weitere Ansätze und Untersuchungsmöglichkeiten des Konzeptes von Horstmann [14] gegeben.

8.1. Zusammenfassung

Im Kontext dieser Arbeit wurde das von Horstmann [14] für die Analyse textueller Kommunikation in Entwicklerteams entwickelte Konzept im Hinblick auf eine Verbesserung der Performanz der verwendeten Klassifizierungsalgorithmen untersucht. Aufbauend auf einer Auswertung auf dem GitHub-Goldstandard von Novielli et al. [27] - in der das Konzept von Horstmann eine Genauigkeit von 82% erzielte - wurden die verwendeten Algorithmen, die Lernkurven der Algorithmen und die Fehlklassifizierungen der Algorithmen auf Möglichkeiten einer Verbesserung der Performanz untersucht. Zudem wurde der evolutionäre Algorithmus auf seinen Einfluss auf die Klassifizierungsperformanz und seiner Eignung als Suchalgorithmus analysiert. Anhand der Einblicke dieser Analyseschritte wurden Änderungen und Anpassungen an der Vorverarbeitung, den Klassifizierungsalgorithmen und ihren Parametern sowie der Merkmalsselektion vorgenommen. Die Performanz des auf diese Art ergänzten Konzeptes wurde ausgewertet und die Auswirkung der einzelnen Änderungen auf die Performanz ermittelt.

Durch die entsprechenden vorgenommenen Änderungen wurde eine Verbesserung der Genauigkeit um 5,1%-Punkte auf 87%, erreicht. Dabei wurde festgestellt, dass nicht jede Änderung einen positiven Einfluss auf die Genauigkeit erzielt. Die Vergrößerung der Merkmalsmenge im Rahmen einer Verarbeitung von Negationen führte zu einer Verschlechterung der Genauigkeit. Die beiden Optimierungsschritte der Merkmalsselektion und Hyperparameteroptimierung hatten zudem einen insgesamt geringen Einfluss auf die Gesamtgenauigkeit. Sowohl in dieser Hinsicht als auch im Erhalten genauerer Einblicke in die Methoden des Ensembles sowie der evolutionären Algorithmen besteht weiterhin Potenzial für eine Verbesserung. Es kann jedoch gesagt werden, dass das initiale Ziel dieser Arbeit, die Klassifizierungsgenauigkeit zu verbessern, erreicht und eine Richtung für weitere Anpassungen des Konzepts vorgegeben wurde.

8.2. Ausblick

Wie in der Zusammenfassung angesprochen, bestehen weiterhin Möglichkeiten, die Klassifizierungsperformanz des Konzeptes von Horstmann [14] zu verbessern.

So besteht weiterhin das offene Problem der fehlenden Verarbeitung von Emojis aus der Fehlerklasse „NLP und Umfang“. Die Implementierung eines entsprechenden Sentimentlexikons wie vorgestellt von Novak et al. [26] stellt eine sinnvolle Ergänzung des Konzeptes dar. Vor allem im Hinblick auf die Verfügbarkeit von Emojis in häufig verwendeten Anwendungen wie „Zulip“ oder „Slack“ erscheint eine derartige Ergänzung sinnvoll.

In dieser Arbeit wurden weitere Emotions- und Emoticonmerkmale in das Konzept von Horstmann [14] integriert. Aufbauend auf den vermerkten Erfolgen dieser Merkmale erscheint es sinnvoll, weitere manuell erstellte Merkmale in die Klassifikation einfließen zu lassen. So lassen sich Parallelen zu Arbeiten von z.B. Calefato et al. [6] ziehen, die neben Emotionsmerkmalen auch Merkmale für Aspekte des Microbloggings und anderer Aspekte natürlichsprachlicher Texte verwenden. Auch ein erneutes Hinzufügen des Formalitätsmerkmals von Horstmann [14], angepasst an die englische Sprache, erscheint interessant. So könnten z.B. die Häufigkeit der Verwendung von *Contractions* (z.B. "can't"/"cannot"), oder die Häufigkeit von Slangwörtern in ein solches Merkmal einfließen [33].

In der Diskussion wurde bereits aufgegriffen, dass die Aufgabe des Negation Handlings weiterhin einen wichtigen Aspekt der Verarbeitung von Textdaten im Hinblick auf die Analyse des Sentiments eines Textes darstellt. Im Kontext dieser Arbeit wurde gezeigt, dass ein implizites Negation Handling zu einer Verschlechterung der Klassifizierungsgenauigkeit führte. Stattdessen könnte ein explizites Negation Handling implementiert werden, wie es in der von Ahmed et al. [4] vorgestellten Anwendung „SentiCR“ der Fall ist. In SentiCR werden N-Gramme, die auf ein Negationswort folgen, mit dem Präfix "not_" versehen und so in das Vokabular der *Bag-of-Words*-Modelle übernommen [4].

Im Rahmen dieser Arbeit wurde eine ausgewählte Menge von Klassifizierungsalgorithmen im Hinblick auf ihre Performanz miteinander verglichen. Die Auswahl der Klassifizierer für das verwendete Ensemble basierte dabei lediglich auf den Klassifizierern, die die beste Genauigkeit im Kontext einer 5-fachen Kreuzvalidierung erzielten. Diese einfache Auswahl bester Klassifizierer beachtet jedoch einige wichtige Aspekte von Ensemblemethoden nicht. Denn die Klassifizierer einer Ensemblemethode sollten so gewählt werden, dass sie möglichst unterschiedliche Ausgaben erzeugen, dabei aber jeweils eine gute Performanz erzielen [35]. Dementsprechend ist eine Auswahl der Klassifizierer basierend auf einer Übereinstimmungsmetrik wie „Fleiss' Kappa“ [10], kombiniert mit einer Performanzmetrik wie „Genauigkeit“ womöglich eine bessere Option, um die Auswahl für das Ensemble zu treffen. Eine derartige Untersuchung des Ensembles würde auch tiefere Einblicke in die zuvor beschriebene geringe Veränderung der Ensembleperformanz bieten. Auch das Verwenden von mehr als drei Klassifizierern könnte z.B. zum vollständigen Vermeiden von Unentschieden in der Abstimmung

untersucht werden.

Wie bereits angesprochen, basiert der Vergleich der evolutionären Algorithmen auf einer einzelnen Beobachtung. Um zu ermitteln, wie wichtig der Faktor der Exploration für die Merkmalsselektion im Kontext des Konzeptes von Horstmann [14] ist, muss eine weitere Untersuchung der Algorithmen erfolgen. So kann der erweiterte EA anhand derselben Metriken wie der EA von Horstmann [14] in Kapitel 5 betrachtet werden. Auch ein quantitativer Vergleich der beiden Algorithmen würde das Treffen einer Aussage darüber erlauben, ob einer der beiden Algorithmen bessere Ergebnisse generiert. Darüber hinaus könnte auch der in Abschnitt 5.3.3 angesprochene Aspekt der „Granularität“ der Merkmalsselektion aufgegriffen werden.

Die Auswertung des Horstmann-Konzeptes erfolgte in dieser Arbeit anhand des GitHub-Datensatzes von Novielli et al. [27]. Wie in der Diskussion bereits erwähnt, sind Auswertungen des Konzeptes auf weiteren Datensätzen notwendig, um Rückschlüsse über die Generalisierbarkeit der Ergebnisse dieser Arbeit zu ziehen. Für weitere Auswertungen bieten sich andere Goldstandard-Datensätze, wie der Stackoverflow-Goldstandard von Calefato et al. [6], der Jira-Goldstandard von Orthu et al. [30] oder ein direktes Beispiel aus der Industrie, wie es von Horstmann [14] in seiner Arbeit betrachtet wurde, an.

Abschließend ist festzustellen, dass trotz weiterer Verbesserungsmöglichkeiten die in dieser Arbeit implementierten Ansätze erfolgreich zu einer Verbesserung der Klassifizierungsperformanz geführt haben, um somit die Anwendbarkeit des Konzeptes von Horstmann [14] für weitere Forschung und direkte Umsetzung in Entwicklerteams zu verbessern.

A. Anhang

A.1. Lernkurven

Support Vector Machine (SVM)

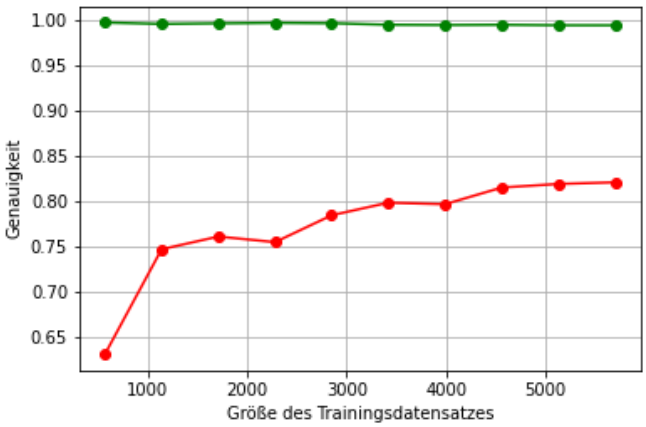

Abbildung A.1.: Lernkurven des *Support-Vector-Machine*-Klassifizierers, 5-fach kreuzvalidiert.

Die für die SVM ermittelten Lernkurven sind in Abbildung A.1 zu sehen. Ähnlich der Lernkurve des *Random-Forest*-Klassifizierers (RF) fällt hier die Trainingsgenauigkeit von 1,0 auf, die mit zunehmender Größe des Trainingsdatensatzes um 0,01 abnimmt. Auch fällt erneut die vergleichsweise große Diskrepanz zwischen Trainings- und Testgenauigkeit auf, die im letzten Datenpunkt um ca. 0,18 beträgt. Die Trainingsgenauigkeit weist dabei auf eine Überanpassung des Klassifizierers hin [11], während die Testgenauigkeit wahrscheinlich von weiteren Trainingsdaten profitieren könnte, was gegen den Aspekt der Überanpassung spricht. Weitere Trainingsdaten sind für das Treffen einer tatsächlichen Aussage notwendig.

Gaussian Naive Bayes (GNB)

Die Lernkurven des GNB sind in Abbildung A.2 zu sehen. Hier fällt auf, dass die Trainingsgenauigkeit bei zunehmender Größe des Trainingsdatensatzes abnimmt, während die Testgenauigkeit überwiegend gleich bleibt bzw. sich ab ca. 4000 Trainingsdaten

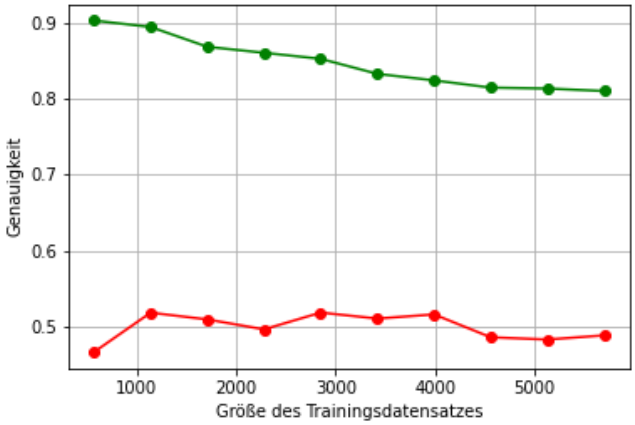

Abbildung A.2.: Lernkurven des *Gaussian-Naive-Bayes*-Klassifizierers, 5-fach kreuzvalidiert.

verschlechtert. Im Gegensatz zur SVM und dem RF, weist der GNB die höchste Trainingsgenauigkeit bei einer Datensatzgröße von ca. 600 Dokumenten auf. Auch im GNB fällt dabei die große Diskrepanz von Trainings- und Testgenauigkeit auf, die in diesem Klassifizierer im letzten Datenpunkt sogar ca. 0,32 beträgt. Der Algorithmus ist nicht dazu in der Lage, die mit der Größe der Trainingsdaten steigende Anzahl der Merkmale korrekt zu verarbeiten und weist ähnlich der SVM und dem RF eine schwache Generalisierbarkeit auf.

Literaturverzeichnis

[1] Dokumentation der scikit-learn Bibliothek: Ensemble Classifier. `https://scikit-learn.org/stable/modules/ensemble.html`. Letzter Zugriff am: 22.11.2020.

[2] Dokumentation der scikit-learn Bibliothek: Stochastic Gradient Descent. `https://scikit-learn.org/stable/modules/sgd.html#sgd`. Letzter Zugriff am: 22.11.2020.

[3] Dudenredaktion (o. J.): „Sentiment“ auf Duden online. `https://www.duden.de/node/164751/revision/164787`. Letzter Zugriff am: 19.11.2020.

[4] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi. Senticr: A customized sentiment analysis tool for code review interactions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 106–111, 2017.

[5] P. Basile and N. Novielli. Uniba: Sentiment analysis of english tweets combining micro-blogging, lexicon and semantic features. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 595–600, 2015.

[6] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli. Sentiment polarity detection for software development. *Empirical Software Engineering*, 23(3):1352–1382, Jun 2018.

[7] F. Calefato, F. Lanubile, and N. Novielli. Emotxt: A toolkit for emotion recognition from text. In *2017 Seventh International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 79–80, 2017.

[8] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers Electrical Engineering*, 40(1):16 – 28, 2014. 40th-year commemorative issue.

[9] J. F. DeFranco and P. A. Laplante. Review and analysis of software development team communication research. *IEEE Transactions on Professional Communication*, 60(2):165–182, 2017.

[10] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[11] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[12] D. Graziotin, X. Wang, and P. Abrahamsson. Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ*, 2:e289, 2014.

[13] E. Guzman and B. Bruegge. Towards emotional awareness in software development teams. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, page 671–674, New York, NY, USA, 2013. Association for Computing Machinery.

[14] J. Horstmann. Computer-gestützte analyse des kommunikationsverhaltens in entwicklerteams unter berücksichtigung digitaler medien. Master's thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2019.

[15] M. R. Islam, M. K. Ahmmed, and M. F. Zibran. Marvalous: Machine learning based detection of emotions in the valence-arousal space in software engineering text. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1786–1793, New York, NY, USA, 2019. Association for Computing Machinery.

[16] M. R. Islam and M. F. Zibran. Sentistrength-se: Exploiting domain specificity for improved sentiment analysis in software engineering text. *Journal of Systems and Software*, 145:125 – 146, 2018.

[17] R. Jongeling, P. Sarkar, S. Datta, and A. Serebrenik. On negative results when using sentiment analysis tools for software engineering research. *Empirical Software Engineering*, 22(5):2543–2584, Oct 2017.

[18] F. Jurado and P. Rodriguez. Sentiment analysis in monitoring software development processes: An exploratory case study on github's project issues. *Journal of Systems and Software*, 104:82 – 89, 2015.

[19] P. Layzell, O. P. Brereton, and A. French. Supporting collaboration in distributed software engineering teams. In *Proceedings Seventh Asia-Pacific Software Engeering Conference. APSEC 2000*, Proceedings Seventh Asia-Pacific Software Engeering Conference. APSEC 2000, pages 38–45, 2000.

[20] B. Liu. Sentiment analysis and subjectivity. In N. Indurkhya and F. J. Damerau, editors, *Handbook of Natural Language Processing*, chapter 26, pages 627–666. Chapman Hall/CRC, 2nd edition, 2010.

[21] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008.

[22] Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, Jun 1947.

[23] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[24] G. A. Miller and W. G. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.

[25] A. Murgia, P. Tourani, B. Adams, and M. Ortu. Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In *MSR 2014*, 2014.

[26] P. K. Novak, J. Smailovic, B. Sluban, and I. Mozetic. Sentiment of emojis. *CoRR*, abs/1509.07761, 2015.

[27] N. Novielli, F. Calefato, D. Dongiovanni, D. Girardi, and F. Lanubile. Can we use se-specific sentiment analysis tools in a cross-platform setting? *Proceedings of the 17th International Conference on Mining Software Repositories*, Jun 2020.

[28] N. Novielli, F. Calefato, and F. Lanubile. The challenges of sentiment detection in the social programmer ecosystem. In *Proceedings of the 7th International Workshop on Social Software Engineering*, pages 33–40, 2015.

[29] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli. Are bullies more productive? empirical study of affectiveness vs. issue fixing time. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, MSR '15, page 303–313. IEEE Press, 2015.

[30] M. Ortu, A. Murgia, G. Destefanis, P. Tourani, R. Tonelli, M. Marchesi, and B. Adams. The emotional side of software developers in jira. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, page 480–483, New York, NY, USA, 2016. Association for Computing Machinery.

[31] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1–2):1–135, Jan. 2008.

[32] R. Remus. Modeling and representing negation in data-driven machine learning-based sentiment analysis. In *ESSEM@ AI* IA*, pages 22–33, 2013.

[33] F. A. Sheikha and D. Inkpen. Generation of formal and informal sentences. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 187–193, 2011.

[34] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61:2544–2558, 12 2010.

[35] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection science*, 8(3-4):385–404, 1996.

[36] M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, 45(3), July 2013.

[37] T. Wilson. Annotating subjective content in meetings. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*,

Marrakech, Morocco, May 2008. European Language Resources Association (ELRA).

[38] M. R. Wrobel. Emotions in the software development process. In *2013 6th International Conference on Human System Interactions (HSI)*, pages 518–523, 2013.

[39] X. Yu and M. Gen. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

[40] N. N. Yusof, A. Mohamed, and S. Abdul-Rahman. A review of contextual information for context-based approach in sentiment analysis. *International Journal of Machine Learning and Computing*, 8(4):399–403, 2018.